

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Телеграм-бот для школи німецької мови
Dehto “Dehto-bot”»**

Виконав студент групи ІПЗ-21-5
ВОЛКІВСЬКИЙ Ярослав Сергійович

Керівник роботи:
ЧИЖМОТРА Олексій Володимирович

Рецензент:
ГРОМСЬКИЙ Олександр Олександрович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ВОЛКІВСЬКИЙ Ярослав Сергійович**

Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**

Тема роботи: **«Telegram-бот для школи німецької мови Dehto “Dehto-bot”»**,
затверджена Наказом закладу вищої освіти від «14» лютого 2025 р., №61/с

Вихідні дані для роботи: **Розробка Telegram-бота для автоматизації комунікацій у навчальному процесі. Об'єкт дослідження: Використання Node.js, TypeScript, MongoDB та Telegraf для створення ботів, що забезпечують автоматичні комунікації через Telegram.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В..	27.02.2025р.	27.02.2025р.
2	Чижмотря О.В..	27.02.2025р.	27.02.2025р.
3	Чижмотря О.В..	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з телеграм бота з вивчення німецької та пояснювальної записки. Пояснювальна записка до випускної роботи містить 165 сторінок, включає 55 ілюстрацій, 9 таблиць, 15 додатків та 30 використаних джерел.

Метою роботи є створення телеграм-боту з вивченням англійської мови з використанням сучасних технологій та інтеграцією системи оплати для її автоматизації та зберіганням тимчасових й постійних даних.

Архітектура телеграм-боту для вивчення німецької мови базується на фреймворці Telegraf, який присутній на мові JS/TS та надає можливість взаємодіяти з ботом через інтерфейс месенджера. Бот надає можливості адміністрування, як заняттями так і користувачами, переглядати розклад, планувати заняття і звісно їх проводити.

Оскільки вся взаємодія з ботом відбувається за допомогою телеграму, то користувачі мають швидкий та зручний доступ до нього, що підвищує ефективність та швидкість роботи. Також взаємодія відбувається за допомогою тексту та кнопок. Взаємодія валідується, що запобігає виникненню помилок в системі.

КЛЮЧОВІ СЛОВА: ТЕЛЕГРАМ, БОТ, ШКОЛА, НІМЕЦЬКА МОВА, ОПЛАТА

					ІПЗ.КР.Б-121-25-ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Волківський Я.С.			Телеграм-бот для школи німецької мови Dehto “Dehto-bot” Пояснювальна записка	Літ.	Арк.
Керівник		Чижмотря О.В.				3	165
Рецензент		Громський О.О.				Житомирська політехніка, група ІПЗ-21-5	
Зав. каф.		Вакалюк Т.А.					

ABSTRACT

The Bachelor's qualification thesis consists of a Telegram bot for learning German and an explanatory note. The explanatory note to the graduation project comprises 170 pages, including 55 illustrations, 9 tables, 15 appendices, and 30 references.

The aim of the project is to create a Telegram bot for learning English using modern technologies and integrating a payment system for its automation, as well as for storing temporary and permanent data.

The architecture of the Telegram bot for learning German is based on the Telegraf framework, which is built on the JavaScript/TypeScript language and provides the ability to interact with the bot through the messenger interface. The bot offers features for administering both lessons and users, viewing the schedule, planning lessons, and, of course, conducting them.

Since all interaction with the bot takes place via Telegram, users have fast and convenient access to it, which increases efficiency and speed of operation. Interaction is carried out through text and buttons. The interaction is validated, which prevents system errors.

KEYWORDS: TELEGRAM, BOT, SCHOOL, GERMAN LANGUAGE, PAYMENT

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ТЕЛЕГРАМ-БОТУ ШКОЛИ НІМЕЦЬКОЇ.....	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту.....	13
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	20
Висновки до першого розділу	23
РОЗДІЛ 2. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ.....	24
2.1. Визначення варіантів використання та об'єктно-орієнтованої системи.....	24
2.2. Розробка бази даних системи.....	32
2.3. Проектування та реалізація алгоритмів роботи системи.....	35
2.4. Реалізація функціоналу Телеграм бота «dehto bot».....	38
Висновки до другого розділу.....	45
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ТЕЛЕГРАМ-БОТОМ ШКОЛИ НІМЕЦЬКОЇ.....	46
3.1. Порядок встановлення та налаштування бота.....	46
3.2. Тестування бота та інтерфейсу.....	48
3.3. Тестування адмін панелі та інтерфейсу.....	74
Висновки до третього розділу.....	78
ВИСНОВКИ.....	79
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	80
ДОДАТКИ.....	83

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

MVC - Model-View-Controller

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

CRUD – основні операції, що використовуються для управління даними (Create, Read, Update, Delete)

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. Сучасні інформаційні технології значно спрощують процеси управління та адміністрування освітніх послуг. У сфері викладання іноземних мов, зокрема німецької, все більшої популярності набувають автоматизовані рішення, що дозволяють організовувати заняття, взаємодіяти зі студентами та оптимізувати адміністративні процеси. Одним із таких інструментів є чат-боти, які забезпечують швидкий та зручний доступ до інформації та послуг навчального закладу.

Розвиток телеграм-ботів відкриває нові можливості для автоматизації запису на заняття, управління розкладом та організації навчального процесу. Вони дозволяють як адміністраторам, так і студентам ефективно взаємодіяти з системою, спрощуючи процес реєстрації, редагування та видалення занять. Завдяки цьому навчальні заклади можуть підвищити якість освітніх послуг та знизити адміністративне навантаження.

Метою данної роботи є розробка телеграм-бота для школи німецької мови "dehto", який надає можливість адміністрування занять, покращувати досвід роботи з освітнім процесом, а саме записуванням та веденням записів щодо проведення занять та взагалом про внутрішні процеси школи.

Поставлена мета обумовлює необхідність виконання наступних завдань:

- аналіз функціональних потреб навчального закладу;
- вибір оптимальної архітектури та технологічного стеку для реалізації бота;
- розробка алгоритмів роботи телеграм-бота;
- реалізація основного функціоналу;
- інтеграція з базою даних для збереження інформації про студентів та розклад занять;
- тестування та оптимізація роботи бота.

Об'єктом дослідження є інформаційні технології для автоматизації освітніх процесів.

Предметом дослідження є використання телеграм-бота для адміністрування занять у школі німецької мови.

Для створення даного телеграм-бота було використано: мова програмування TypeScript для строгої типізації, щоб уникнути можливі помилки в ході розроблення

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

системи призначеної для обробки великої кількості даних; використані дві бази даних, а саме Redis для збереження проміжних або локально потрібних користувачу даних та MongoDB для збереження глобальних даних або даних не пов'язаних з користувачами напряму, Telegraf, як API яке дає доступ до можливості створення бота.

За тематикою випускної кваліфікаційної роботи бакалавра були опубліковані тези [1-2].

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ТЕЛЕГРАМ-БОТУ ШКОЛИ НІМЕЦЬКОЇ

1.1. Постановка задачі

В умовах цифрової трансформації та розвитку освітніх технологій дедалі більше навчальних процесів переходять в онлайн-середовище. Це стосується як індивідуальних занять, так і групових форматів, таких як розмовні клуби. Важливим аспектом ефективного навчання є зручність організації занять, управління розкладом, комунікація між учасниками та можливість швидкої та безпечної оплати.

З огляду на це, Dehto Bot розробляється як інтелектуальна автоматизована система на базі Telegram, що надає можливість планування, адміністрування та оплати навчальних занять безпосередньо в боті. Він дозволяє створювати та керувати індивідуальними та груповими заняттями, спрощуючи процес для учнів, викладачів та адміністраторів.

Одним із ключових завдань бота є забезпечення зручного та прозорого способу оплати занять через інтеграцію з Monobank API. Це рішення дозволяє здійснювати фінансові транзакції всередині бота без необхідності використання сторонніх платіжних сервісів [3].

Dehto Bot включає декілька основних модулів, що забезпечують різноманітні функції для користувачів:

Модуль планування занять є ключовим елементом системи, який забезпечує зручний та гнучкий доступ учнів до організації власного навчального процесу. Цей модуль надає можливість учням обирати між індивідуальними заняттями або груповими сесіями, такими як розмовні клуби, відповідно до власних уподобань та навчальних цілей.

Крім того, система дозволяє переглядати доступний розклад занять, який можна фільтрувати за різними параметрами: датою, викладачем або рівнем складності. Це дає змогу швидко знаходити відповідні варіанти, що відповідають потребам учня та його вільному часу, значно покращуючи користувацький досвід та доступність освітніх послуг.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

Важливою функцією модуля є можливість бронювання занять безпосередньо через систему. Після вибору та підтвердження уроку учнем, запис автоматично вноситься до загального розкладу, що забезпечує синхронізацію між студентом, викладачем та адміністративною системою. Це зменшує необхідність у ручній координації та мінімізує ризик перетинів чи помилок у розкладі.

У цілому, модуль планування занять сприяє підвищенню ефективності управління навчальним процесом, покращує тайм-менеджмент учасників освітнього процесу та забезпечує більш впорядковану й автоматизовану взаємодію в межах навчальної платформи.

Модуль адміністрування забезпечує повноцінне керування навчальним процесом з боку адміністраторів та викладачів, дозволяючи ефективно організовувати та контролювати розклад занять. Однією з основних функцій модуля є керування розкладом, що включає можливість редагування існуючих занять, зміну їх часу, тривалості або викладача, а також скасування бронювань у разі потреби. Це дає змогу оперативно реагувати на зміни та підтримувати актуальність розкладу.

Модуль також дозволяє створювати нові заняття з можливістю вказання їх тривалості, тематики, формату (індивідуальні чи групові), а також обмеження кількості учасників. Це особливо важливо для планування розмовних клубів або спеціалізованих сесій, де контроль над кількістю студентів має значення для якості навчання.

Ще однією важливою функцією є відстеження статистики щодо проведених занять, включно з кількістю учасників, відвідуваністю та ефективністю проведення. Крім того, модуль надає можливість обробки платежів, що спрощує фінансову взаємодію між школою та студентами, забезпечуючи прозорість і контроль за оплатами.

Модуль сповіщень виконує важливу функцію інформування учасників навчального процесу, забезпечуючи своєчасну комунікацію між системою, учнями та викладачами. Однією з ключових особливостей модуля є автоматичні нагадування

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

про заплановані заняття, які надсилаються як учням, так і викладачам. Це дозволяє уникнути пропусків та підвищує відповідальність за відвідування занять.

Окрім цього, модуль надсилає повідомлення про успішне бронювання, зміни в розкладі, зокрема зміну часу уроку або його скасування. Таким чином, усі сторони завжди залишаються поінформованими про актуальний стан свого розкладу, що значно знижує ймовірність непорозумінь або конфліктів.

Модуль оплати через Monobank API забезпечує швидкий, зручний і безпечний процес фінансових розрахунків між студентами та навчальним закладом. Завдяки інтеграції з Monobank API, користувачі мають можливість здійснювати оплату за заняття безпосередньо через Telegram-бот, не переходячи на сторонні сервіси чи додатки. Це значно підвищує зручність користування платформою та мінімізує час на проведення фінансових операцій [5].

Система реалізує автоматичне підтвердження платежів, що дозволяє боту в режимі реального часу фіксувати факт оплати та оновлювати статус занять чи балансу користувача без участі адміністратора. Це зменшує навантаження на адміністративний персонал та виключає людський фактор із процесу перевірки транзакцій.

Крім того, модуль забезпечує відображення актуальної інформації про баланс кожного користувача, а також надає доступ до історії оплат, що дозволяє студентам самостійно відстежувати свої фінансові операції, контролювати витрати та вчасно поповнювати рахунок у разі потреби.

Модуль безпеки та гарантій відіграє ключову роль у забезпеченні надійності та довіри до системи як з боку учнів, так і адміністрації. Одним із основних завдань модуля є захист фінансових даних користувачів, що реалізується шляхом шифрування транзакційної інформації та використання безпечних протоколів передачі даних. Також передбачено безпечну авторизацію, яка дозволяє уникнути несанкціонованого доступу до особистих кабінетів користувачів.

Важливою складовою є верифікація користувачів, яка запобігає створенню фейкових акаунтів та знижує ризики шахрайства. Це дозволяє підтримувати високу

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

якість користувацької бази та забезпечити прозорість у проведенні як освітніх, так і фінансових операцій.

У сукупності, модуль безпеки та гарантій створює надійне середовище для взаємодії всіх учасників освітнього процесу, забезпечуючи конфіденційність, прозорість та відповідальність у межах платформи.

Розробка Telegram-бота відбувається поетапно, із дотриманням принципів модульності та адаптивності.

Розробка Telegram-бота для школи німецької мови відбувалася поетапно, з дотриманням чіткої структури та послідовності дій.

Етап 1: Проєктування архітектури системи розпочався з аналізу аналогічних рішень, доступних на ринку, з метою визначення найкращих практик та особливостей реалізації. Було проведено детальне формулювання функціональних вимог до системи та створено технічну документацію, включаючи архітектурні схеми, що окреслюють компоненти майбутньої платформи, їх взаємозв'язки та принципи взаємодії.

Етап 2: Розробка клієнтської частини охоплював реалізацію безпосередньої взаємодії бота з користувачами за допомогою Telegram API. Було створено діалогові сценарії, які забезпечують інтуїтивно зрозумілий користувацький досвід (UX), а також реалізовано набір команд, що дозволяють студентам і адміністраторам працювати з розкладом, здійснювати бронювання занять та проводити оплату [4].

Етап 3: Реалізація серверної частини включала створення бази даних для збереження інформації про користувачів, розклад занять і платежі. На цьому етапі налаштовувалася логіка обробки запитів, здійснювалося підключення до Telegram API, а також розроблявся функціонал для адміністрування системи та контролю за платежами.

Етап 4: Інтеграція Monobank API полягала у впровадженні функціоналу створення платіжних запитів, обробки повідомлень про статус платежів та автоматичному оновленні статусу занять після підтвердження оплати. Це дало змогу забезпечити повноцінний платіжний модуль безпосередньо в Telegram-боті.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Етап 5: Тестування та оптимізація передбачав перевірку коректності реалізованих сценаріїв роботи бота, включаючи взаємодію з базою даних і Monobank API. Також було проведено оптимізацію коду та усунено потенційні вразливості з метою підвищення стабільності та безпеки роботи системи.

Етап 6: Запуск MVP та масштабування завершував цикл розробки. Була впроваджена перша робоча версія Telegram-бота, після чого зібрано зворотний зв'язок від користувачів. На основі отриманої інформації було додано нові функції, а сама система оптимізована для підтримки більшої кількості користувачів, що дозволило розпочати масштабування проєкту [6-7].

Очікуваним результатом роботи є повністю автоматизований Telegram-бот, який дозволяє учням легко записуватися на індивідуальні заняття або розмовні клуби, адміністраторам — зручно керувати розкладом та відстежувати статистику, а також реалізовано можливість здійснення оплат безпосередньо в боті через Monobank API. Користувачі отримують автоматичні нагадування про заплановані уроки та користуються зручним і безпечним інтерфейсом, доступним прямо в Telegram. Dehto Bot покликаний автоматизувати управління навчальними процесами та зробити навчання доступним, ефективним та технологічно зручним.

1.2. Аналіз аналогів Dehto Bot

Перш ніж розробляти Telegram-бот для управління заняттями та платежами через Monobank API, варто проаналізувати існуючі рішення, які мають схожий функціонал. Це допоможе зрозуміти, які можливості вже реалізовані на ринку, а також знайти слабкі місця конкурентів та переваги, які можна використати в Dehto Bot.

Telegram-боти для організації навчальних процесів:

Перший це SkyEng Bot (рис. 1.1)

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

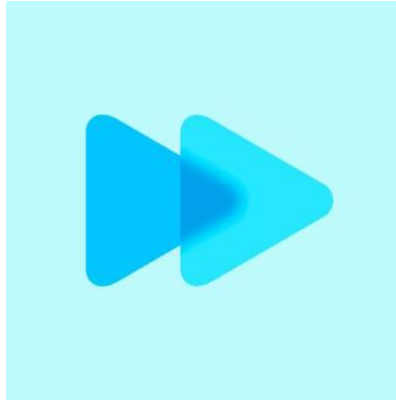


Рис.1.1. SkyEng Bot

Одним із прикладів аналогічного рішення в освітній сфері є Telegram-бот, розроблений для платформи SkyEng. Він забезпечує базовий функціонал для взаємодії учнів з освітньою системою, зокрема дозволяє записуватись на заняття з викладачем безпосередньо через Telegram. До переваг бота можна віднести інтеграцію з платформою SkyEng, що дозволяє синхронізувати розклад та дані про студентів, наявність системи нагадувань про заплановані уроки, а також можливість оплати занять через вбудовані платіжні системи.

Водночас рішення має і певні недоліки. Telegram-бот орієнтований виключно на користувачів платформи SkyEng, тому не може бути адаптований для інших навчальних закладів без значних змін. Крім того, у ньому відсутня підтримка розмовних клубів, що є популярною формою навчання серед студентів, а також бракує інтеграції з Monobank API, через що оплата здійснюється через сторонні сервіси, що знижує зручність для українських користувачів.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Наступний аналог це LinguaLeo Bot (рис. 1.2)



Рис.1.2. LinguaLeo Bot

Існуючий освітній Telegram-бот має низку корисних функцій, що сприяють навчанню користувачів. Зокрема, він дозволяє надсилати навчальні матеріали безпосередньо у чат, здійснювати перевірку слів та завдання на запам'ятовування для закріплення знань, а також надає можливість записуватись на індивідуальні уроки. Крім того, бот підтримує push-сповіщення, які інформують користувачів про появу нових навчальних матеріалів, що покращує їх залучення до процесу навчання.

Проте цей бот має й певні обмеження. Зокрема, відсутній функціонал оплати занять, що ускладнює організацію процесу навчання безпосередньо через бота. Також немає можливості бронювати заняття через Telegram, що обмежує зручність користування системою. Крім того, бот не підтримує групові заняття, які є важливою складовою колективного навчання і можуть підвищити мотивацію студентів.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Інший аналог - iTalki Bot (рис. 1.3)



Рис.1.3. iTalki Bot

Ще одним прикладом освітнього рішення є Telegram-бот, інтегрований з платформою iTalki. Його функціонал дозволяє користувачам зручно підключатися до сервісу, здійснювати пошук викладачів за заданими критеріями та записуватися на заняття. Оплата уроків відбувається через офіційний сайт iTalki, що забезпечує захищені транзакції, а також є можливість перепланування уроків, що додає гнучкості в організації навчального процесу.

Однак бот має низку обмежень. Зокрема, він не підтримує пряму оплату через Telegram, що знижує зручність користування ботом безпосередньо в месенджері. Відсутня інтеграція з локальними платіжними системами, такими як Monobank чи ПриватБанк, що робить його менш адаптованим до потреб українських користувачів. Крім того, бот орієнтований виключно на платформу iTalki і не має можливості кастомізації під власні навчальні курси або школи, що обмежує його використання як універсального інструменту для різних освітніх проєктів.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Ще можна знайти Open English Bot (рис. 1.4)



Рис.1.4. Open English Bot

Telegram-бот, розроблений у межах платформи Open English, надає низку ефективних інструментів для організації навчального процесу. Серед його основного функціоналу — надсилання навчальних матеріалів безпосередньо в чат, що дозволяє студентам отримувати контент у зручному форматі. Бот також підтримує перевірку знань за допомогою інтерактивних тестів, що сприяє закріпленню вивченого матеріалу. Крім того, він надає можливість запису на групові онлайн-заняття, а завдяки інтеграції з веб-платформою Open English синхронізується з розкладом і персональними даними користувачів.

Водночас цей бот має певні недоліки. Зокрема, він не підтримує оплату занять безпосередньо через Telegram, що ускладнює повноцінну автоматизацію процесу. Відсутність інтеграції з Monobank API робить його менш зручним для українських користувачів. Крім того, бот орієнтований виключно на клієнтів Open English і не може бути використаний поза межами цієї платформи без істотної адаптації.

На сучасному ринку цифрових освітніх рішень спостерігається активне впровадження Telegram-ботів як зручного інструменту взаємодії між навчальними закладами, викладачами та студентами. Проте аналіз існуючих продуктів свідчить, що більшість таких ботів виконують суто допоміжну функцію — зокрема, вони використовуються переважно для надсилання навчальних матеріалів, інформаційних

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

повідомлень або базового тестування знань. Їхній функціонал обмежується виконанням окремих завдань, не забезпечуючи комплексного підходу до організації навчального процесу.

Окрему категорію становлять Telegram-боти з можливістю здійснення платежів, однак ці рішення, як правило, орієнтовані на загальні фінансові операції — наприклад, перекази коштів, перевірку балансу або оплату товарів та послуг загального призначення. Вони не враховують специфіки освітньої сфери, не мають адаптованих механізмів бронювання занять, відстеження розкладу або прив'язки до конкретних учнів і викладачів.

Найбільш помітним недоліком поточних рішень є відсутність Telegram-бота, який би об'єднував у собі можливість гнучкого планування навчального процесу (зокрема, бронювання занять) із прямою інтеграцією з популярними локальними платіжними системами, такими як Monobank API. Така функціональна прогалина свідчить про актуальну потребу у створенні цілісного, багатофункціонального та зручного інструменту, який би відповідав запитам сучасного користувача та враховував реалії вітчизняного ринку освітніх послуг.

Dehto Bot вирізняється низкою переваг, які роблять його інноваційним інструментом для організації навчального процесу в школі іноземних мов. Однією з ключових переваг є повністю автоматизоване бронювання занять, що відбувається без потреби переходу на сторонні сайти чи заповнення додаткових форм — усе здійснюється в межах зручного та інтуїтивно зрозумілого інтерфейсу Telegram. Це суттєво спрощує процес для користувача, мінімізуючи часові витрати та знижуючи бар'єри у взаємодії зі школою.

Ще однією важливою перевагою є інтеграція з Monobank API, завдяки якій користувачі мають змогу здійснювати оплату занять безпосередньо через Telegram. Це не лише забезпечує високий рівень зручності, а й гарантує безпеку фінансових операцій у знайомому для українських користувачів середовищі. Завдяки такій інтеграції процес оплати стає максимально прозорим, швидким та надійним.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Також, бот надає користувачам гнучке керування розкладом: кожен учень може обрати найбільш зручний для себе час проведення заняття з урахуванням доступності викладачів, що особливо важливо в умовах інтенсивного ритму життя. Такий підхід сприяє підвищенню мотивації до навчання та забезпечує індивідуальний підхід до кожного студента.

Для адміністративного персоналу Dehto Bot відкриває доступ до розширеної аналітики. Адміністратори можуть у режимі реального часу отримувати детальну статистику щодо кількості оплачених занять, фактичної відвідуваності, а також інших важливих метрик, що дозволяє оперативно приймати управлінські рішення та вдосконалювати навчальний процес.

І, нарешті, бот підтримує як індивідуальні, так і групові заняття, включно з розмовними клубами. Це робить систему універсальною та придатною для широкого кола навчальних форматів, дозволяючи охопити як тих, хто віддає перевагу особистим зустрічам із викладачем, так і тих, хто прагне практикувати мову в колективі. Завдяки цьому Dehto Bot виступає не просто як інструмент автоматизації, а як повноцінна цифрова платформа для якісної організації освітнього процесу.

Таблиця 1.1

Порівняння аналогів

Назва бота	Запис на заняття	Розмовні клуби	Індивідуальні заняття	Нагадування	Оплата в Telegram	Інтеграція з банками	Автопідтвердження оплати
SkyEng bot	+	-	+	+	-	-	-
LinguaLeo bot	-	-	+	+	-	-	-
iTalki bot	+	-	+	+	-	-	-
Open English bot	+	+	+	+	-	-	-

Таким чином, проєкт Dehto Bot є актуальним і затребуваним рішенням у сфері онлайн-навчання, оскільки він поєднує автоматизований запис на заняття, адміністрування розкладу та інтегровану систему оплати через Monobank. Це

дозволяє користувачам легко реєструватися на індивідуальні чи групові сесії, отримувати нагадування та зручно здійснювати оплату безпосередньо в Telegram. Аналіз аналогів підтверджує, що на ринку існує багато подібних сервісів, однак вони не мають комплексної інтеграції з платіжними системами, що робить Dehto Bot унікальним рішенням. Завдяки сучасним технологіям, бот забезпечує ефективний навчальний процес, зменшує адміністративні витрати та покращує взаємодію між учасниками, сприяючи зручному та доступному освоєнню знань.

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Проект Telegram-бота розробляється за допомогою технологій, які забезпечують ефективну роботу як на серверній, так і на клієнтській стороні. Серверна частина бота буде побудована на основі Node.js з використанням фреймворку Telegraf для взаємодії з Telegram API. Бот буде реалізовувати основні функції, такі як взаємодія з користувачем через повідомлення, кнопки, інлайн-кнопки, а також обробку команд та автоматизацію різноманітних процесів.

У процесі розробки Telegram-бота Dehto використовувалися сучасні та ефективні інструментальні засоби, які забезпечують високу продуктивність, масштабованість і гнучкість системи. Одним із ключових технологічних компонентів виступає Node.js — потужне серверне середовище, що дозволяє запускати JavaScript-код на стороні сервера. Завдяки своїй подієво-орієнтованій та асинхронній природі Node.js ідеально підходить для обробки великої кількості одночасних запитів, що є критично важливим для стабільної роботи бота, особливо в періоди високого навантаження [8-9].

Ще одним важливим елементом є Telegraf — популярний фреймворк для створення Telegram-ботів на базі Node.js. Цей інструмент значно спрощує взаємодію з Telegram API, пропонуючи розробникам зручні методи для обробки команд, повідомлень, inline-кнопок, меню та інших інтерактивних елементів. Завдяки своїй простоті та розширюваності Telegraf дозволяє в найкоротші терміни реалізувати

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

навіть складні сценарії поведінки бота, включно з логікою обробки платежів, розкладу занять і персоналізованих повідомлень [10].

Для зберігання довготривалих даних використовується MongoDB — сучасна нереляційна база даних, яка зберігає інформацію у вигляді гнучких документів формату JSON. Такий підхід дозволяє ефективно оперувати складними структурами даних, що притаманні багатofункціональним ботам: від профілів користувачів до історії занять, платіжних записів і адміністративних налаштувань. MongoDB забезпечує швидкий доступ до інформації, легке масштабування системи та адаптацію до змін у структурі збережених даних [11].

Для оптимізації роботи з тимчасовими або проміжними даними, як-от сесії користувачів, стан виконання команд або тимчасові повідомлення, у системі застосовується Redis. Це високошвидкісний інструмент для кешування, який дозволяє значно знизити навантаження на основну базу даних, забезпечуючи миттєвий доступ до часто використовуваних даних. Redis особливо корисний у контексті Telegram-ботів, де важлива реакція в реальному часі та швидка обробка запитів користувача [12].

Загалом, поєднання цих інструментів дозволяє створити надійну, ефективну та масштабовану систему, яка відповідає сучасним вимогам до автоматизації освітніх процесів і забезпечує якісний досвід як для студентів, так і для адміністраторів школи.

Використання сучасних технологій у створенні Telegram-бота Dehto дозволяє досягти високого рівня ефективності, гнучкості та масштабованості системи. Кожна з обраних технологій відіграє ключову роль у загальній архітектурі проекту та має свої унікальні переваги. Наприклад, Node.js надає потужні інструменти для створення серверних додатків, які можуть обробляти тисячі одночасних з'єднань без втрати продуктивності. Завдяки асинхронному підходу до обробки запитів, ця технологія забезпечує стабільну та швидку роботу Telegram-бота навіть у періоди пікового навантаження, що особливо важливо для сервісів, які функціонують у режимі 24/7.

Фреймворк Telegraf, у свою чергу, значно полегшує процес розробки бота, дозволяючи легко взаємодіяти з Telegram API та створювати інтерактивний досвід для

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

користувача. Його зручний синтаксис і багатий набір інструментів забезпечують надійність і гнучкість у реалізації всіх необхідних функцій — від обробки повідомлень і команд до створення кастомізованих меню, кнопок та автоматизованих сценаріїв взаємодії.

Для зберігання постійних даних про користувачів, заняття, оплати й інші аспекти роботи системи використовується MongoDB — потужна документно-орієнтована база даних, яка забезпечує високу гнучкість у роботі з різними структурами даних. Це особливо корисно для динамічних проєктів, де вимоги до сховища можуть змінюватися з часом. Крім того, MongoDB чудово масштабується, що дає змогу легко розширювати проєкт у майбутньому без потреби у глобальних змінах архітектури.

Доповненням до основної бази даних є Redis — високошвидкісний інструмент для кешування та зберігання тимчасових даних. Його використання значно знижує навантаження на MongoDB, адже Redis чудово підходить для збереження короткотривалих сесій, проміжних результатів і тимчасових налаштувань. Завдяки цьому забезпечується швидкий доступ до часто використовуваної інформації, що позитивно впливає на загальну реактивність системи.

Для повноцінного функціонування таких технологій також висувуються певні вимоги до апаратного забезпечення, які гарантують стабільну й безперебійну роботу всієї системи. Серверне середовище, в якому розміщується Telegram-бот, повинно мати достатню обчислювальну потужність і ресурси для обробки великої кількості одночасних запитів. Це можуть бути як фізичні сервери, так і гнучкі хмарні рішення на кшталт AWS, Heroku або DigitalOcean, які дозволяють масштабувати потужності у разі зростання навантаження [13-15].

Окрему увагу слід приділити розміщенню баз даних. Для забезпечення високої продуктивності та відмовостійкості MongoDB та Redis доцільно розгортати на окремих серверах або використовувати відповідні керовані хмарні сервіси. Це дозволить не лише підвищити швидкість доступу до даних, а й забезпечити резервування на випадок збоїв.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

Нарешті, жодна сучасна система не може обійтися без надійних засобів моніторингу та безпеки. Важливо впровадити сервіси для постійного спостереження за станом системи, швидким виявленням помилок і загроз. Також критично важливо захищати передані дані за допомогою SSL-сертифікатів та інших засобів шифрування, особливо якщо йдеться про обробку платіжної інформації та персональних даних користувачів. Усі ці заходи в комплексі дозволяють створити надійну, безпечну та масштабовану інфраструктуру, що відповідає сучасним стандартам розробки програмного забезпечення.

Висновки до першого розділу

У першому розділі було проведено аналіз предметної області, що охоплює створення Telegram-бота для автоматизації певних бізнес-процесів або освітніх задач. Визначено основні вимоги до функціональності бота, серед яких — зручна взаємодія з користувачем, збереження даних та виконання асинхронних завдань.

У ході аналізу схожих рішень було розглянуто наявні підходи до побудови Telegram-ботів, їхні переваги та недоліки. Це дозволило визначити доцільність використання Node.js у поєднанні з бібліотекою Telegraf, а також обґрунтувати вибір MongoDB як основної бази даних для зберігання інформації користувачів та станів.

Крім цього, були сформовані основні технічні вимоги до системи та обрано технології, які відповідають вимогам до продуктивності, масштабованості та зручності розробки.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ

2.1 Визначення варіантів використання та об'єктно-орієнтованої системи

Telegram-бот школи німецької мови надає зручний інтерфейс взаємодії, через який користувачі можуть реєструватися на курси, отримувати навчальні матеріали та спілкуватися з адміністрацією школи. Функціональність бота залежить від активності користувачів. Користувачів можна розділити на дві основні групи: студенти, які навчаються або планують навчатися у школі, та адміністратори, які керують навчальним процесом і обробляють запити користувачів. Бот містить інформацію про доступні курси, розклад занять, рівні володіння мовою, а також іншу важливу інформацію, що може зацікавити потенційних студентів. Для наочної демонстрації взаємодії між елементами системи була створена Діаграма варіантів використання (рис. 2.1 та рис. 2.2).



Рис. 2.1. Діаграма варіантів використання (ч.1)

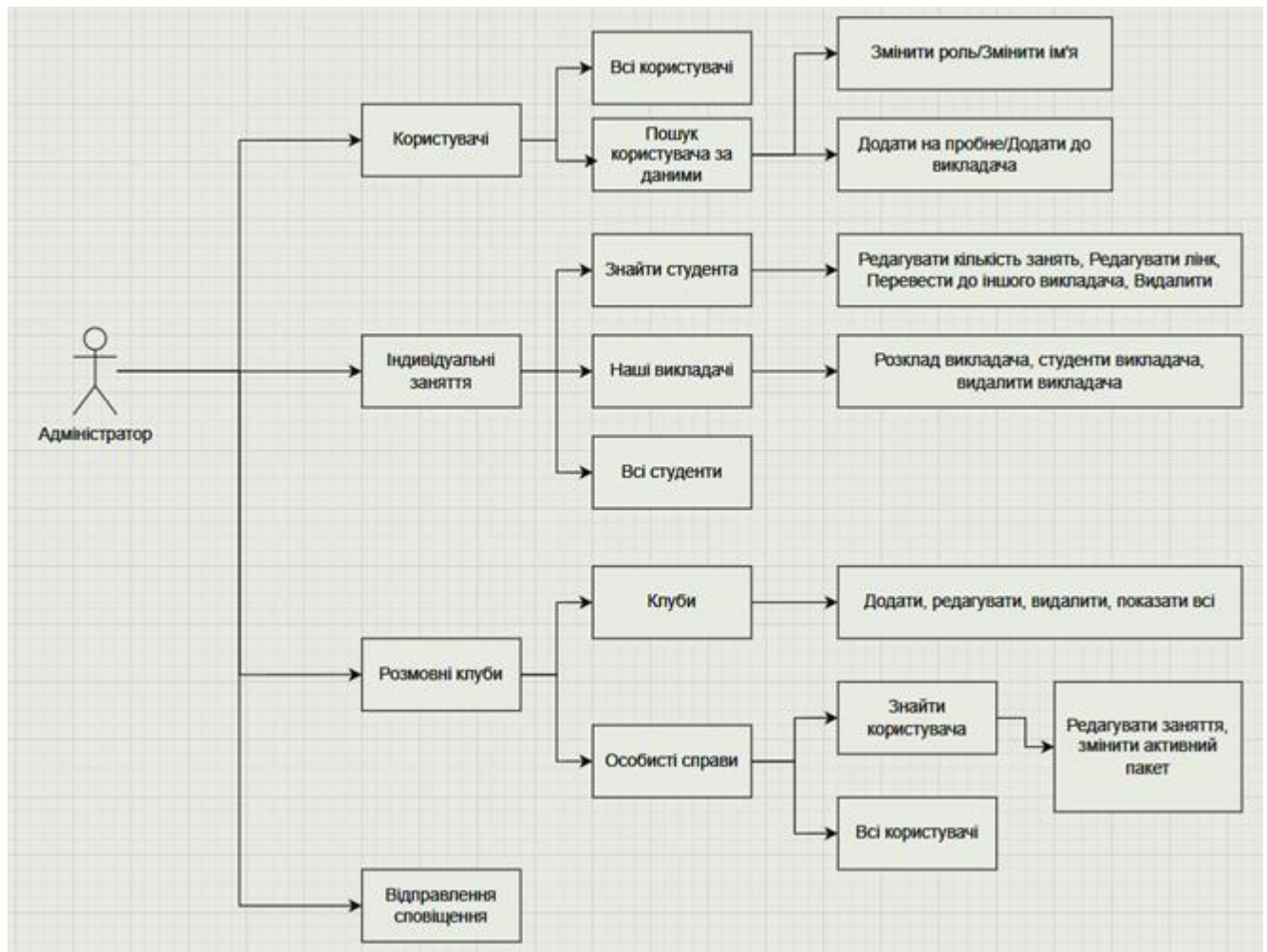


Рис. 2.2. Діаграма варіантів використання (ч.2)

Таблиця 2.1

Опис прецеденту «Вхід/Реєстрація»

Назва варіанту використання	Вхід/реєстрація
Основні актори	Адміністратор, сапорт, студенти, викладачі
Короткий Опис	Користувачі вже авторизовані за допомогою Телеграму, використовується їх актуальний аккаунт

Таблиця 2.2

Опис прецеденту «Створення та участь в Індивідуальних заняттях»

Назва варіанту використання	Створення та участь в Індивідуальних заняттях
Основні актори	Адміністратор, сапорт, студенти, викладачі
Короткий Опис	Користувачі можуть взаємодіяти з заняттями, а саме для викладача це можливість створювати заняття зі своїм студентом, для адміністратора це можливість переглядати та вести статистику, для студента – відвідувати

Таблиця 2.3

Опис прецеденту «Надання сапорту студентам»

Назва варіанту використання	Надання сапорту студентам
Основні актори	Сапорт, студенти
Короткий Опис	Студенти мають можливість створити запити на живу підтримку, в якій сапорти надають підтримку, підтверджуючи запит

Таблиця 2.4

Опис прецеденту «Оплата занять»

Назва варіанту використання	Оплата занять
Основні актори	Студенти
Короткий Опис	Студенти мають можливість оплачувати всі типи занять та мати баланс, який вони можуть використовувати, щоб приймати участь в заняттях

Таблиця 2.5

Опис прецеденту «деЗавдання»

Назва варіанту використання	деЗавдання
Основні актори	Викладачі, студенти
Короткий опис	Викладач має можливість давати студентам індивідуальні завдання, використовуючи всі можливості Телеграм. А студенту, очевидно, виконувати та здавати їх.

Основна ціль розробки Telegram-бота «Дехто» полягає в створенні онлайн-середовища, яке дозволяє ефективно взаємодіяти з користувачами для організації процесу вивчення німецької мови. Бот спрямований на забезпечення зручної комунікації між учасниками навчального процесу (студентами, викладачами, організаторами), а також на надання простого та надійного механізму для збору інформації, навчання та підтримки користувача.

Основна мета полягає у сприянні засвоєнню знань, розвитку мовних навичок та підтримці мотивації користувачів шляхом інтерактивної взаємодії через бот. «Дехто» надає можливість користувачам реєструватися, обирати свою роль у навчанні, проходити тести, отримувати навчальні матеріали та автоматичні нагадування. Завдяки цьому, основною метою є створення умов для ефективного самостійного або групового навчання, яке може мати позитивний вплив на особистий розвиток, освітній рівень та подальші можливості користувача.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

У процесі розробки Dehto Bot надзвичайно важливою є орієнтація на реальні потреби різних категорій внутрішніх користувачів, адже саме вони безпосередньо взаємодіють із системою, формуючи її ефективність і зручність. Кожна з груп має свої особливі вимоги та функціональні очікування, які повинні бути враховані при проєктуванні бота.

Адміністратор – це користувач, який відповідає за організаційні процеси, керування освітньою діяльністю та підтримку стабільного функціонування сервісу. Йому необхідно мати доступ до звітності у зручному форматі, наприклад, у вигляді сповіщень через окремого адміністративного бота, що інформуватиме про всі важливі події, зміни та івенти у системі. Адміністратор повинен мати інструменти для керування користувачами, а також можливість адмініструвати освітні процеси, включаючи створення та модерацію «Розмовних клубів». Крім того, йому важливо мати функціонал для надсилання повідомлень усім користувачам з використанням усіх можливостей Telegram (текст, фото, відео, кнопки тощо), а також отримувати персоналізовані сповіщення за 40 хвилин до початку будь-якого заняття, що дозволяє оперативно реагувати на можливі зміни або потребу втручання.

Студент, як основний користувач платформи, очікує зручного доступу до розкладу індивідуальних занять і розмовних клубів, можливості оперативно зв'язатися зі службою підтримки у випадку виникнення запитань або складнощів. Важливо також, щоб він міг оплачувати заняття безпосередньо через інтегроване платіжне API та контролювати свій баланс занять. Крім цього, студент повинен мати змогу записуватися на доступні розмовні клуби, отримувати автоматичні сповіщення про нові події, важливі новини та адміністративні оновлення. Окремим функціональним елементом є взаємодія з «деЗавданнями» – індивідуальними навчальними вправами, які студент отримує, виконує та надсилає викладачеві на перевірку.

Викладач – ще одна ключова категорія користувачів, якій потрібно мати широкі можливості для організації навчального процесу. Вони повинні мати змогу планувати та керувати індивідуальними заняттями зі своїми студентами, переглядати власний

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

розклад, отримувати автоматичні нагадування за 40 хвилин до початку кожного заняття, а також брати участь у розмовних клубах. Інформування про нові заняття, які передбачають їхню участь, має здійснюватися своєчасно, аби викладач міг якісно підготуватись. Також викладачі мають функціонал для створення та видачі індивідуальних «деЗавдань» кожному учневі з подальшим отриманням виконаних робіт, що дозволяє підтримувати ефективну та персоналізовану освітню взаємодію. Як і студенти, викладачі повинні отримувати важливі адміністративні повідомлення в рамках системи.

Останньою, але не менш важливою групою користувачів є гість – потенційний студент або просто відвідувач, який ще не є повноцінним користувачем платформи. Для цієї аудиторії важливо забезпечити відкритий доступ до перегляду актуальних розмовних клубів, щоб ознайомитися з пропозицією та зацікавитися участю. Після поповнення балансу гість може почати відвідувати заняття. Як і всі інші користувачі, гість повинен мати змогу звернутися до служби підтримки за допомогою у вирішенні будь-яких питань, пов'язаних з платформою або навчальним процесом.

Функціональні вимоги до системи Dehto Bot охоплюють ключові аспекти, необхідні для забезпечення ефективної взаємодії користувачів із ботом, підтримки освітнього процесу та налагодженої адміністративної роботи. Насамперед, у боті повинна бути реалізована можливість авторизації користувачів через Telegram ID, що дозволить однозначно ідентифікувати кожного користувача без потреби у створенні окремого облікового запису. Це значно спрощує доступ до функціоналу бота, підвищуючи зручність користування і водночас забезпечуючи базовий рівень безпеки та персоналізації.

Одним із важливих елементів функціональності є система сповіщень, яка має автоматично надсилати нагадування про початок занять як викладачам, так і студентам. Це забезпечить пунктуальність, зменшить кількість пропущених занять і загалом сприятиме кращій організації навчального процесу.

Крім того, у боті повинна бути передбачена можливість повноцінного адміністрування занять. Це означає, що адміністратори мають мати змогу

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

переглядати, створювати, редагувати або видаляти заняття, а також контролювати їхній статус та пов'язані з ними ресурси. Такий функціонал дозволить гнучко керувати розкладом, враховуючи потреби викладачів і студентів.

Окрему роль відіграє можливість надсилання повідомлень користувачам, зокрема новин, нагадувань або спеціальних оголошень. Така функція є важливою для підтримки постійного інформаційного зв'язку з користувачами, оперативного інформування про зміни, нові можливості або технічні оновлення.

Нефункціональні вимоги до Dehto Bot є важливою складовою загального проєкту, адже саме вони визначають якість, зручність і стабільність роботи системи, що безпосередньо впливає на користувацький досвід та ефективність використання бота в реальних умовах. Починаючи зі сприйняття, слід зазначити, що інтерфейс бота має бути максимально дружнім і простим у використанні, адже це суттєво впливає на мотивацію користувачів швидко освоїти всі функції і активно користуватись сервісом. В ідеалі, час, необхідний для повного ознайомлення з функціоналом, не повинен перевищувати п'яти хвилин, що робить бота доступним навіть для новачків, які раніше не працювали з подібними інструментами. Для більш досвідчених користувачів цей час має бути ще коротшим — не більше двох хвилин, що свідчить про інтуїтивність і логічну структуру інтерфейсу.

Ще одним важливим аспектом є швидкодія системи — адже затримки у відповіді можуть створювати дискомфорт і відштовхувати користувачів. Саме тому середній час відгуку бота повинен утримуватися в межах 3-5 секунд, що є цілком прийнятним для текстових запитів і команд, за винятком ситуацій, коли йдеться про завантаження великих файлів, де природні затримки неминучі. Такий рівень швидкодії гарантує, що взаємодія з ботом буде плавною і не викликатиме непотрібних затримок чи роздратування.

Щодо надійності, то тут вимоги максимально жорсткі, оскільки користувачі очікують, що бот буде доступним у будь-який час доби без перебоїв. Забезпечення роботи 24/7 є критичною умовою, адже навчальні процеси часто потребують гнучкості у часі, а нестабільна робота сервісу може спричинити втрату довіри і

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

зниження лояльності користувачів. Таким чином, система має бути побудована з урахуванням резервування, моніторингу і швидкого реагування на будь-які збої.

Ще одним важливим аспектом є продуктивність — адже бот має одночасно обробляти запити багатьох користувачів, зокрема мінімум сотні активних учасників. Це вимагає грамотно спроектованої архітектури, що дозволить масштабувати систему без втрати продуктивності чи виникнення “вузьких місць” під час пікових навантажень. Відповідність цим вимогам дозволить Dehto Bot працювати ефективно і підтримувати комфортну взаємодію для всіх користувачів без перебоїв.

Для кінцевого користувача, з погляду системних вимог, все зроблено максимально просто і зрозуміло: для користування ботом достатньо мати будь-який пристрій із доступом до інтернету, що підтримує Telegram або сучасний браузер із можливістю запуску Telegram Web. Не менш важливим є наявність власного Telegram-акаунту, що слугує базовою умовою для авторизації і подальшої роботи з ботом. Такий підхід робить сервіс доступним для широкої аудиторії, незалежно від технічного рівня користувача чи типу пристрою.

Що ж стосується вимог до серверного обладнання і середовища, тут передбачено комплексний набір програмного забезпечення, необхідного для стабільної і швидкої роботи бота. До нього входять Node.js як середовище виконання, npm — менеджер пакетів, pm2 — інструмент для керування процесами, Bun — сучасний runtime для JavaScript/TypeScript, а також Redis і MongoDB як бази даних і кешування. Рекомендується використовувати операційну систему Linux через її надійність і широке розповсюдження в серверах, хоча Windows також може бути варіантом за певних обставин. Щодо апаратних ресурсів, сервер має бути оснащений мінімум одним логічним ядром з тактовою частотою не менше 1.5 ГГц, з рекомендацією мати ядро частотою від 3.5 ГГц для забезпечення більшої продуктивності. Оперативної пам’яті слід мати не менше 1.5 ГБ (хоча система здатна працювати і з 1 ГБ, це підходить лише для невеликих навантажень), а оптимально — від 2 ГБ. Для зберігання даних потрібен SSD накопичувач об’ємом не менше 20 ГБ, з

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

рекомендованою верхньою межею близько 40 ГБ, що забезпечить достатньо швидкий доступ до необхідної інформації [16-19].

В цілому, ці нефункціональні вимоги є своєрідним фундаментом для стабільної, швидкої та зручної роботи Dehto Bot, що дозволяє гарантувати якісний користувацький досвід, зручність у використанні та надійність роботи сервісу навіть у випадках інтенсивного навантаження чи тривалого використання.

2.2. Розробка бази даних системи

Для збереження постійної та незалежної інформації використовується MongoDB, оскільки тип даних для котрих використовується БД є в вигляд екземплярів, котрі не потребують реляційної бази даних. Також для збереження проміжних даних зібраних в ході роботи з ботом потрібних для подальших операцій або ж для проміжних рішень використовується швидка база даних Redis. Переваги цієї БД в тому, що вона зберігає дані в оперативній пам'яті що значно прискорює швидкість читання та запису, а ніж MongoDB та інші класичні SQL/NoSQL, також при завершенні роботи вона створює дамб, який дозволяє при відновленні роботи системи відновити всі дані. Якщо казати про структуру Redis, то це Telegram ID : data, тобто основним ключом є айді користувача, а далі йдуть додаткові дані також по ключам, що дозволяє зберігати дані індивідуально сесії [20-21].

Для MongoDB було створено кілька колекцій для збереження даних, а саме: dataUsers, clubLessons, deTask, individualLessons, sentNotificationIndividualLessons та supportActiveChannels. Опис кожної з основних таблиць бази даних наведений у таблицях 2.6 – 2.8.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

Схема «dataUsers»

Поле	Тип	Призначення
_id	ObjectId	Автоматично знегероване значення базою даних, є унікальним ідентифікатором документа
available	String undefined	Це поле призначається сапортерам, надає інформацію про те чи вільний він наразі в взмозі прийняти новий запит
count	Number undefined	Призначено для зберігання балансу розмовних занять і присвоюється тільки студентам
detask	ObjectId undefined	Присвоюється ObjectId активного «деЗавдання» для студента
email	String undefined	Як видно з назви це поле призначене для зберігання електронної пошти користувача, але воно не обов'язкове
haveTrialLessonClub	String undefined	Поле призначене для відмітки чи користувач був на пробному по розмовних клубах
Id	Number	Це основне поле ідентифікації користувача, поле має Telegram ID користувача
individual_count	Number undefined	Таке ж саме поле, що й “count”, але призначено саме для індивідуальних занять і також призначається тільки для студентів
individual_lessons	ObjectId[] undefined	Зберігає ObjectId запланованих індивідуальних занять для студента
miro_link	String undefined	Містить посилання на дошку для індивідуальних занять на платформі Miro
name	String	Ім'я користувача, саме в базі це Ім'я та Прізвище
number	Number	Містить номер телефону користувача
recordClubs	ObjectId[] string	Містить розмовні клуби на які записаний користувач
registered_students	Number[] undefined	Містить записаних на викладача студентів для індивідуальних занять та «деЗадання»
role	String	Містить інформацію про роль користувача, а саме Гість, Студент, Викладач, Адміністратор

Схема «dataUsers»

Поле	Тип	Призначення
set_detasks	ObjectId[] undefined	Містить «деЗавдання» задані викладачем
set_individual_lessons	ObjectId[] undefined	Теж саме що і set_detasks, але зберігає індивідуальні заняття викладача
system_role	String undefined	Відмітка чи є користувач сапортером
teacher	Number undefined	Зберігає інформацію щодо актуального викладача для індивідуальних занять студента, а саме його айді
trial	ObjectId undefined	Якщо гість бере пробне індивідуальне заняття, то саме тут зберігається його ObjectId
trial_students	Number[] undefined	Містить в собі айді гостей з якими викладач має змогу запланувати пробне індивідуальне заняття
username	String	Зберігає ім'я користувача телеграма

Таблиця 2.7

Схема «individualLessons»

Поле	Тип	Призначення
_id	ObjectId	Автоматично згенерований айді для документу
date	String	Дата на котре заплановане індивідуальне заняття
idStudent	Number	Айді студента, котрий приймає участь в індивідуальному занятті
idTeacher	Number	Айді викладача, котрий приймає участь в індивідуальному занятті
time	String	Містить час на котре заплановане індивідуальне заняття
type	String	Тип індивідуального заняття, або пробне, або звичайне

Схема «sentNotificationIndividualLessons»

Поле	Тип	Призначення
_id	ObjectId	Автоматично згенерований айді базою даних
id	ObjectId	Айді заняття, сповіщення якого вже було надіслано

Якщо розглядати більш детально кожну з таблиць, а саме призначення, то можна визначити, що у системі використовується таблиця dataUsers, яка зберігає всі ключові дані щодо користувачів системи: від типу користувача до балансу занять, якщо це саме студент, або до списку активних студентів, якщо це викладач. Таблиця supportActiveChannels зберігає дані про активні сесії живої підтримки, включаючи айді сесії, айді сапорта, айді студента та дані самої переписки. Таблиця individualLessons зберігає заплановані індивідуальні заняття, а саме айді заняття, айді студента, айді викладача, а також тип (пробне для гостя або звичайне) і час з датою. Таблиця clubLessons зберігає дані щодо розмовних клубів, призначення таке ж саме, що й для individualLessons, але замість одного студента передбачено декілька, і також зберігається файл — додатковий контент до заняття. Таблиця deTask зберігає дані щодо «деЗавдань», а саме айді студента і айді викладача, котрий задав це завдання, також матеріал самого завдання і в подальшому відповіді до завдання. Таблиця sentNofiticationIndividualLessons зберігає айді запланованих занять, котрі вже були оповіщенні як ті, що розпочнуться за 40 хвилин.

2.3. Проєктування та реалізація алгоритмів роботи системи

Бот, який було реалізовано в межах даного проєкту, відзначається надзвичайною простотою у своєму функціонуванні та використанні, що є одним із ключових елементів, на які було звернуто особливу увагу в процесі його розробки. Варто наголосити, що головний акцент при побудові взаємодії з користувачем було зроблено саме на забезпеченні інтуїтивно зрозумілого та максимально зручного користувацького досвіду (UX). Це пов'язано з тим, що передбачувана цільова аудиторія охоплює досить широкий віковий діапазон — від молодших користувачів, віком приблизно від 8 років, до старших осіб, яким може бути до 50 років або навіть

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

більше. У зв'язку з цим виникла необхідність створення такого інтерфейсу, який не потребував би спеціальних технічних знань або навичок для початку роботи з ботом. Це дозволяє навіть тим користувачам, які вперше стикаються з подібними рішеннями або лише починають своє ознайомлення з цифровими технологіями, швидко адаптуватися до роботи з ботом, розуміти логіку його дій та успішно користуватися всіма доступними функціональними можливостями.

У контексті архітектурного та функціонального устрою бота, важливо зазначити, що в його основі лежить концепція розподілу користувачів за ролями. Такий підхід не лише сприяє більш точному налаштуванню функціоналу під конкретні потреби різних типів користувачів, але й дозволяє зберігати логічну цілісність системи. Зокрема, в боті було реалізовано кілька чітко визначених ролей, кожна з яких має власне призначення та набір прав доступу. До таких ролей відносяться: гість, студент, викладач, адміністратор, а також спеціальна роль розробника, яка введена з метою забезпечення повноцінного тестування функціоналу в усіх можливих сценаріях.

Варто підкреслити, що кожна з ролей не є простою умовною категорією — натомість, для кожної з них передбачено чітко визначений набір сценаріїв використання, що докладно розглядається на рис. 2.3 [22]. У цій таблиці подано деталізовану інформацію щодо доступних функцій, очікуваної взаємодії з ботом, а також обмежень і можливостей, притаманних кожній з ролей. Це дозволяє не лише систематизувати доступний функціонал, але й забезпечити більш гнучке та адаптивне управління правами доступу.

З метою глибшого розуміння логіки роботи бота з боку користувача та для наочного представлення структури взаємодії, в роботі також було створено схему, яка демонструє всі можливі етапи, які може пройти студент або гість для того, щоб здійснити запис на захід формату «Розмовні клуби». Дана схема є максимально детальною і враховує як стандартні сценарії взаємодії, так і можливі альтернативні розгалуження, що можуть виникати в залежності від поведінки користувача, обраних ним опцій, або навіть потенційних технічних збоїв. Це дозволяє не лише

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

краще зрозуміти внутрішню логіку системи, але й оцінити ступінь її адаптивності до дій користувача.

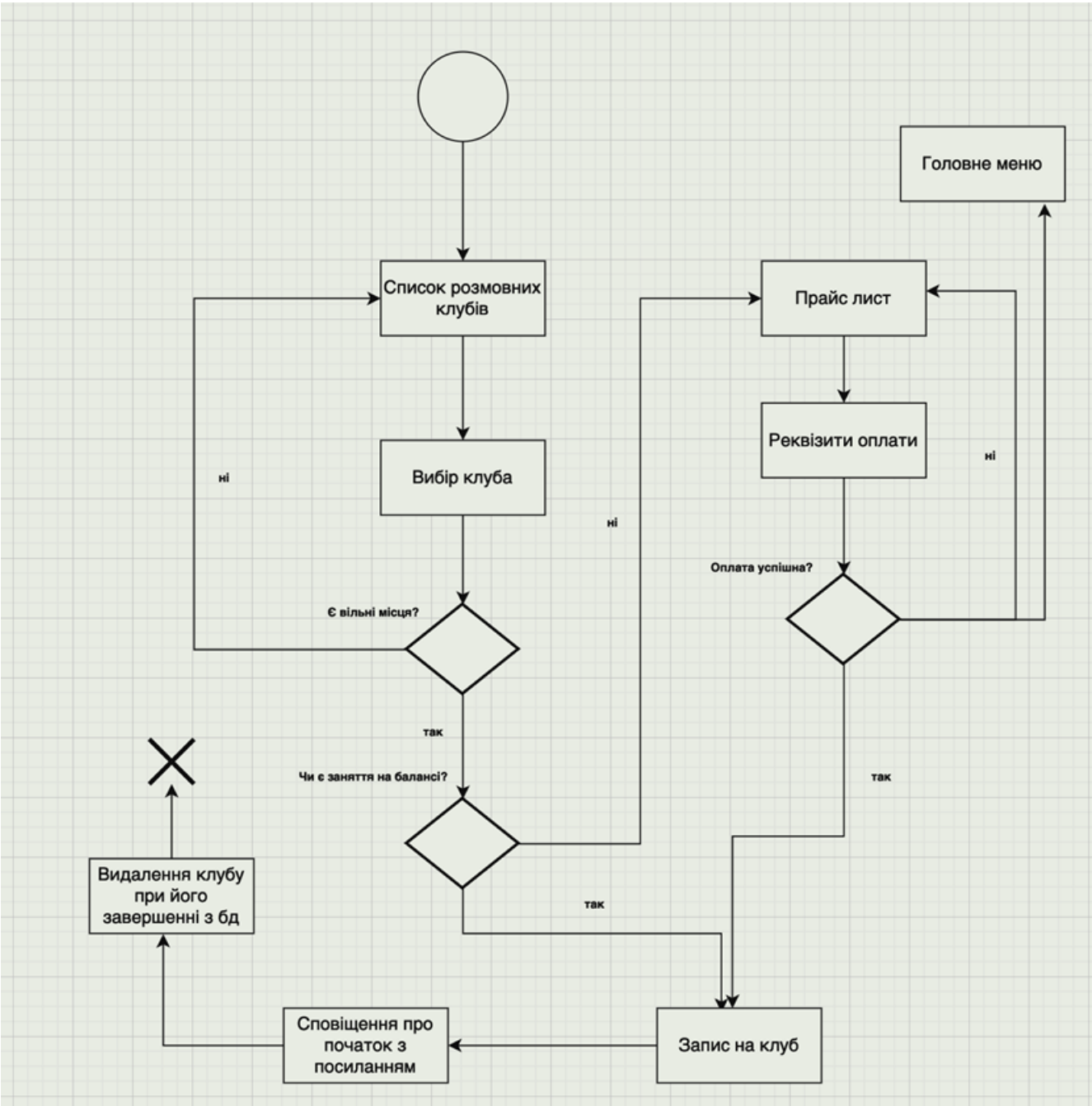


Рис 2.3 – Діаграма активності «Процес запису користувачем на «Розмовний клуб»»

2.4. Реалізація функціоналу Телеграм бота «dehto bot»

Розглянемо основні моменти з роботи Телеграм боту, а саме основні функції, які складають весь функціонал роботи системи (рис. 2.4).

Наприклад створення користувача з основними полями (інші поля додаються в ході роботи системи за потребою).

```
function createUser(  
  id: chat.id,  
  name: string,  
  username: string,  
  role: string  
){}  

```

Після створення нового користувача в нього є всі необхідні дані щоб продовжити користуватися ботом, наприклад перегляд наявних актуальних розмовних клубів на які кожен має змогу записатись та приймати участь (якщо є доступні місця), клуби, котрі створені одним із адміністратором та можуть, відповідно їй редагуватись та видалятись.

```
function createSpeakingClub(  
  title: string,  
  teacherId: number,  
  studentsIds: number[],  
  startDate: Date  
){}  

```

Окрім розмовних клубів в системі наявні індивідуальні на які робиться основний акцент. Створює їх викладач з одним із своїх учнів перед тим як узгодив час і інші моменти потрібні для створення (планування) індивідуального заняття. Також ці заняття також можуть переноситись, видалятись і всі інші операції, які присутні і в розмовних клубах, але в цьому випадку цим всім займається саме викладач.

```
function createIndividualLesson(  
  teacherId: number,  
  studentsIds: number[],  
  startDate: Date  
){}  

```

Якщо говорити про живу підтримку, як вже зазначалось вище, студент має змогу надіслати запит на підтримку, після підтвердження запиту вільним сапортером – отримати успішний сапорт щодо актуальних питань всіма можливостями Телеграм

```
function createIndividualLesson(  
  sessionId: number,  
  supporterId: number,  
  studentId: number,  
  studentMessage: string[],  
  supporterMessage: string[]  
){}  

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

І звісно ж «деЗавдання», для їх роботи викладачу потрібні записані на нього активні студенти, після чого він в змозі обрати одного зі своїх студентів та надіслати йому свіжі завдання для виконання. Після чого студент отримує сповіщення про нове «деЗавдання», яке йому слід виконати та надіслати рішення в Телеграм, знову ж таки, всіма можливостями Телеграма.

```
function createNewDeTask(  
  taskId: number,  
  teacherId: number,  
  studentId: number,  
  deTaskContent: string[]  
){}  

```

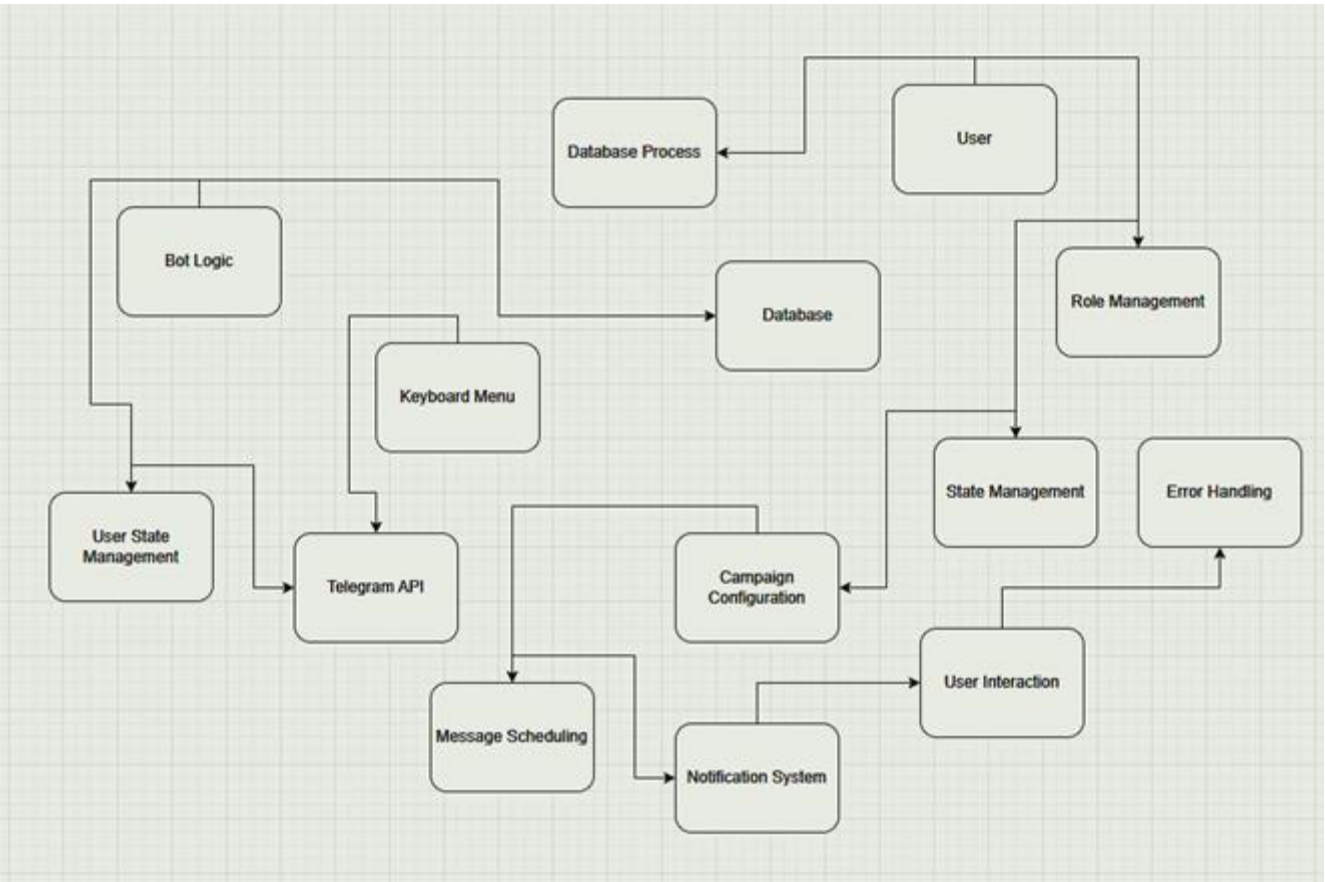


Рис 2.4. Діаграма компонентів основної логіки бота

Компонентна діаграма, що описує архітектуру Dehto Bot, є візуальним відображенням внутрішньої структури системи, яка дозволяє зрозуміти, як саме відбувається взаємодія між основними її частинами. У центрі всього — Telegram Bot, який виступає основним інтерфейсом між користувачем та всією системою. Саме через нього користувачі надсилають команди, отримують відповіді, переглядають

розклад занять, отримують нагадування та іншу важливу інформацію. Telegram Bot функціонує на основі Telegram API, що забезпечує стабільну, швидку й зручну комунікацію в межах популярного месенджера без потреби у встановленні сторонніх додатків.

Після отримання запиту від користувача, Telegram Bot передає його до наступного критично важливого компоненту — Bot Logic. Це своєрідний мозковий центр системи, який відповідає за обробку запитів, визначення поточного контексту взаємодії, запуск внутрішніх функцій і генерацію відповідей. Саме тут відбувається аналіз повідомлень, логічне маршрутизування команд і контроль за всіма процесами, пов'язаними з ботом. Bot Logic у тісній співпраці з іншими елементами, такими як база даних, системи станів та клавіатурне меню, забезпечує безперервну й логічно послідовну взаємодію з кожним користувачем.

Ще одним важливим компонентом є Database, яка виконує функцію довготривалого зберігання даних. Саме тут фіксується інформація про користувачів, історія їх дій, налаштування, стан проходження освітніх процесів, записи на заняття, повідомлення тощо. Цей компонент гарантує, що всі дії користувачів мають контекст, який можна швидко відновити або оновити при потребі. Завдяки використанню сучасних технологій зберігання даних, база може легко масштабуватись і обробляти значні обсяги інформації без зниження продуктивності системи.

Для організації послідовної взаємодії з ботом існує окремий логічний блок — User State Management. Його головна задача полягає у відстеженні, в якому саме етапі взаємодії перебуває кожен користувач. Це дозволяє забезпечити персоналізований підхід: наприклад, для нового користувача бот запропонує реєстрацію, тоді як для зареєстрованого — перелік функцій відповідно до його ролі. Компонент управління станами пов'язаний як з логікою бота, так і з базою даних, оскільки використовує збережену інформацію для прийняття подальших рішень [23-24].

Не менш важливим елементом є Keyboard Menu — зручне меню у вигляді кнопок, що інтегроване безпосередньо в Telegram-інтерфейс. Воно дозволяє користувачам обирати дії без потреби вводити команди вручну, що суттєво підвищує

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

зручність користування ботом. Це меню є динамічним і адаптується до контексту взаємодії з ботом, надаючи лише ті опції, які є актуальними на певному етапі. З технічної точки зору, Keyboard Menu є функціональним інтерфейсом, який тісно інтегрується з Bot Logic і забезпечує швидку реакцію системи на дії користувача.

Одним із ключових елементів у цьому ланцюгу є Telegram API — це інтерфейс, через який відбувається вся взаємодія між ботом і месенджером Telegram. Саме за допомогою цього API бот отримує повідомлення від користувачів, відправляє відповіді, передає файли, обробляє натискання кнопок, реєструє нових користувачів та виконує безліч інших операцій, що роблять взаємодію живою, швидкою і динамічною. Telegram API є надійною точкою доступу до Telegram-екосистеми, забезпечуючи стабільний канал обміну даними.

Для ефективної роботи з даними існує окремий внутрішній механізм — Database Process. Це не просто доступ до бази, а цілий набір процедур, які забезпечують правильне зчитування, оновлення, запис і видалення інформації. Усі запити до бази даних, незалежно від їхньої складності, проходять саме через цей компонент. Він гарантує, що інформація буде оброблена коректно, без втрати цілісності або помилок у логіці. Database Process працює тісно з самою базою, а також з логікою бота, і відіграє фундаментальну роль у підтриманні надійності системи.

У центрі всієї взаємодії, звичайно ж, знаходиться користувач. Саме User є кінцевим бенефіціаром усіх функціональних можливостей системи. Користувач взаємодіє з Telegram Bot, надсилаючи запити, обираючи кнопки, читаючи повідомлення, отримуючи сповіщення та виконуючи дії згідно своєї ролі в системі. Увесь шлях взаємодії користувача з ботом визначається логікою бота, яка враховує як поточний стан користувача, так і його роль у системі.

Система ролей представлена компонентом Role Management, який дозволяє розрізняти різні типи користувачів — адміністратора, викладача, студента чи гостя — та надавати кожному відповідний набір прав і доступів. Наприклад, адміністратор має повний контроль над ботом, у той час як студент може переглядати розклад і здавати завдання. Role Management безпосередньо пов'язаний з компонентом User State

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Management, оскільки багато функцій стають доступними або недоступними саме залежно від ролі.

Ще одним важливим функціональним блоком є State Management. Він відповідає за фіксацію поточного стану користувача — чи знаходиться він у процесі виконання дії, чи завершив реєстрацію, чи очікує відповіді. Стан впливає на поведінку бота, визначаючи, яку відповідь надати, яку опцію запропонувати або на яку кнопку зреагувати. Стани також часто залежать від ролі, тому тісно переплетені з управлінням ролями.

Крім того, до архітектури входить компонент Campaign Configuration. Його задача — надати адміністраторам можливість налаштовувати кампанії комунікацій: від типів повідомлень, які мають бути надіслані, до їх змісту, розкладу й аудиторії. Цей компонент є надзвичайно корисним для організації масових розсилок, інформування про події, запуску інформаційних кампаній, і працює у зв'язці з логікою бота, яка виконує задані сценарії.

З Campaign Configuration безпосередньо пов'язаний Message Scheduling — модуль, який дозволяє планувати розсилку повідомлень у майбутньому. Наприклад, це може бути нагадування за 40 хвилин до заняття, або повідомлення про нову подію, яке буде відправлено у визначений час. Завдяки цьому компоненту система стає більш динамічною і здатною до автоматизації великих обсягів інформаційної взаємодії.

Також в цьому переліку наявний — Notification System. Це спеціалізована підсистема для обробки й надсилання сповіщень користувачам. Вона може включати як термінові нагадування, так і заплановані повідомлення в рамках кампаній, чи персоналізовані відповіді в контексті користувацької активності. Notification System активно використовує дані з Message Scheduling для визначення правильного часу і формату сповіщень, що дозволяє забезпечити своєчасне інформування та підтримку користувача у його взаємодії з ботом.

Якщо мова про взаємодію з користувачем — саме це забезпечує компонент User Interaction. Його функція полягає у тому, щоб слугувати своєрідним містком між користувачем і системною логікою, перетворюючи кожен дію користувача на

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

зрозумілий для системи запит, а кожен відповідь — на зручне, доречне й логічно вплетене повідомлення у чат. Це можуть бути як стандартні текстові відповіді, так і кнопки, інлайн-меню, емодзі, підказки тощо. Його тісна інтеграція з Bot Logic дає змогу зробити комунікацію не просто функціональною, а по-справжньому дружньою до користувача — гнучкою, інтуїтивною й адаптивною під ситуацію. Саме цей компонент відповідає за те, щоб користувачі не почували себе розгубленими або “загубленими” у процесі — кожне натискання має викликати очікувану реакцію.

Однак, навіть за найідеальнішої архітектури система не застрахована від збоїв, непередбачених ситуацій чи просто людських помилок при введенні даних. І тут на авансцену виходить не менш важливий учасник архітектури — Error Handling. Цей компонент є системою безпеки, яка забезпечує стійкість у непередбачуваних обставинах: від банальної помилки у форматі введеного тексту до втрати з’єднання з базою даних або збою при збереженні інформації. Error Handling функціонує як “страхувальний трос”, що дозволяє уникнути краху або зависання системи, перенаправляючи користувача на безпечний шлях, надаючи пояснення й навіть альтернативні варіанти дії. Завдяки тому, що цей компонент пов’язаний буквально з усіма іншими частинами системи — від бази даних до планувальника повідомлень — він може оперативно реагувати на помилки незалежно від їхнього джерела.

Його наявність не лише підвищує загальну надійність, а й значно покращує довіру користувача до системи, створюючи враження цілісності, стабільності та продуманості всіх процесів. У тандемі з User Interaction це дозволяє зробити взаємодію із ботом не лише ефективною, а й приємною — без напруги, страху зробити щось не так чи втратити дані. В результаті система демонструє не лише технічну досконалість, а й емпатію до свого користувача.

У структурі взаємодії між компонентами Telegram-бота ключову роль відіграє Bot Logic — саме він є центральним вузлом, що об’єднує всю архітектуру системи в єдине ціле. Цей компонент виконує роль своєрідного мозкового центру, який координує та управляє всіма іншими елементами: від обробки запитів до контролю доступу, формування відповідей і запуску допоміжних процесів. Саме через Bot Logic

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

реалізується інтеграція з базою даних, системами управління станами, ролями користувачів, конфігурацією кампаній тощо.

База даних (Database) виконує не менш важливу функцію — вона є надійним сховищем усіх даних користувачів, включаючи інформацію про поточні стани, призначені ролі, історію дій, а також параметри активних та запланованих кампаній. Уся ця інформація надзвичайно цінна для прийняття рішень Bot Logic'ом і для забезпечення контекстної взаємодії з користувачем.

Для забезпечення зв'язку з зовнішнім світом, а саме з користувачами Telegram, використовується Telegram API. Завдяки цьому інтерфейсу бот має змогу надсилати повідомлення, реагувати на команди, отримувати дані від користувачів та забезпечувати миттєву інтерактивність. Це критично важливий канал комунікації, який забезпечує прозору, зручну й безперебійну взаємодію в реальному часі.

Важливу функціональну роль у системі відіграють User State Management і Role Management. Перший відповідає за відстеження поточного стану кожного користувача — наприклад, чи знаходиться він у процесі заповнення форми, вибору дати заняття або оплати. Другий же визначає доступ до різних функцій залежно від ролі — студент, викладач, адміністратор або гість. У сукупності ці два компоненти дозволяють формувати персоналізований досвід для кожного користувача, адаптуючи функціональність та поведінку системи до конкретної ситуації та прав доступу.

Щоб бот міг працювати як інструмент планування та сповіщення, необхідні компоненти Campaign Configuration і Message Scheduling. Перший дозволяє гнучко налаштовувати майбутні комунікаційні кампанії — наприклад, створення серій повідомлень або нагадувань. Другий — виконує роль «хронометра», визначаючи, коли і які саме повідомлення мають бути надіслані відповідно до встановлених правил і розкладу.

Доповнює цей механізм Notification System, яка реалізує безпосереднє надсилання сповіщень користувачам. Це можуть бути як важливі інформаційні повідомлення, так і звичайні нагадування про заплановані заняття. Завдяки цьому

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

користувач завжди залишається в курсі актуальних подій та не втрачає важливих деталей.

На завершення, не менш важливим є Error Handling, що виконує роль страховки для всієї системи. У разі виникнення будь-якої помилки — від помилкового введення до технічного збою — цей компонент забезпечує її коректну обробку. Він не тільки запобігає краху системи, але й гарантує, що користувач отримає зрозуміле пояснення та можливість повторити дію або скористатись альтернативним шляхом.

Висновки до другого розділу

В процесі проєктування Telegram-бота Dehto, який орієнтований на допомогу студентам у вивченні німецької мови, були проаналізовані всі можливі сценарії використання програмного продукту відповідно до вимог до програмного забезпечення. На основі зібраної інформації була спроектована архітектура бота.

Проектування включало в себе моделювання логіки взаємодії користувача з ботом, побудову структури діалогів та визначення сценаріїв для різних ролей (наприклад, студента чи викладача). Також було спроектовано базу даних для зберігання інформації про користувачів, їхній прогрес у вивченні мови, обрані теми та активність у розмовних клубах.

Було описано алгоритм роботи бота, включаючи послідовність кроків для реєстрації нового користувача, вибору навчального курсу, проходження вправ та отримання зворотного зв'язку. Окремо описано механізми запису на розмовні клуби — як платні, так і безкоштовні, — з урахуванням різних моделей доступу.

Також був детально розглянутий основний функціонал бота — це автоматизований супровід користувача у процесі навчання, інтерактивна взаємодія через кнопки, нагадування, зберігання даних у Redis та Telegram API для обробки повідомлень і подій.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ТЕЛЕГРАМ-БОТОМ ШКОЛИ НІМЕЦЬКОЇ

3.1. Порядок встановлення та налаштування бота

Встановлення та налаштування бота вимагає розгортання всіх його компонентів та налаштування як кожного елементу окремо, так і загальної системи в цілому. Для опису архітектури платформи було створено діаграму розгортання (рис. 3.1).

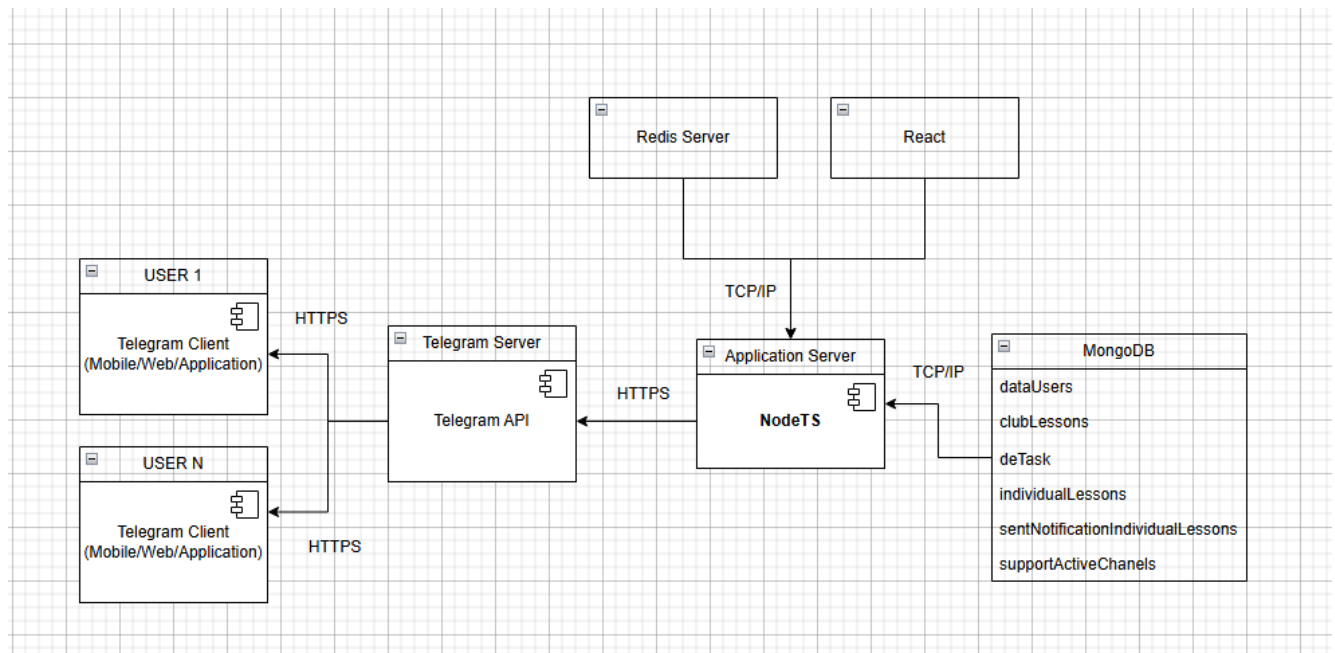


Рис.3.1. Діаграма розгортання Телеграм-бота dehto

Система Telegram-бота представлена як сучасна багатокомпонентна архітектура, основою якої є Application Server, реалізований на Node.js із використанням TypeScript (NodeTS). Саме цей сервер виступає як центральний вузол усієї взаємодії між клієнтами, базою даних, кешем, зовнішніми API та вебінтерфейсом для адміністрування.

Кінцеві користувачі (USER 1, USER N) взаємодіють з ботом через Telegram Client — мобільний, веб або десктопний застосунок. Усі запити, що надсилаються з цих клієнтів, проходять через Telegram Server, який забезпечує безпечну HTTPS-комунікацію між користувачем і Telegram API. Telegram API, у свою чергу, надсилає вхідні повідомлення користувачів до Application Server, де вони обробляються відповідно до заданої логіки [25-27].

Application Server на NodeTS є ядром проєкту. Саме тут реалізовано сценарії спілкування, обробку повідомлень, керування станами користувачів, бізнес-логіку, а також взаємодію з іншими компонентами. Цей сервер підключається до бази даних MongoDB, де зберігається вся інформація про користувачів, уроки (як клубні, так і індивідуальні), виконані завдання, розсилки повідомлень та активні канали підтримки. Кожна з цих таблиць/колекцій виконує конкретну функцію у підтримці роботи системи: наприклад, `dataUsers` — це профілі користувачів, а `individualLessons` — це розклад індивідуальних занять.

Ще одним критичним компонентом є Redis Server, який слугує для кешування даних, зберігання тимчасових станів, TTL для повідомлень чи розкладів. Redis забезпечує швидкий доступ до даних і мінімізує навантаження на основну базу. З'єднання з Redis реалізоване через TCP/IP і використовується з NodeTS для зберігання швидкоплинних даних, як-от сесії користувача, таймери, або проміжні відповіді [28].

Окремо варто згадати React — це клієнтський інтерфейс, що, найімовірніше, виконує роль адміністративної панелі. Він забезпечує зручну взаємодію для адміністраторів, дозволяючи їм переглядати та редагувати дані користувачів, керувати уроками, отримувати статистику. React підключається до NodeTS через API та обробляє запити відповідно до ролей і прав доступу [29-30].

Щодо розгортання системи, воно передбачає кілька ключових етапів. На першому кроці встановлюються всі необхідні сервіси: Node.js з TypeScript, MongoDB, Redis, а також допоміжні інструменти на кшталт PM2 для запуску і моніторингу процесів.

Наступним кроком є реєстрація webhook у Telegram, який вказує Telegram API, куди пересилати повідомлення користувачів. Це здійснюється за допомогою спеціального запиту до Telegram API.

Серверна частина отримує всі необхідні конфігурації через файл `.env`, у якому вказуються такі змінні, як токен Telegram, URI до MongoDB, адреса Redis та URL адміністративної панелі. NodeTS запускається або напряму через Node/PM2. При

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

цьому важливо переконатися, що порти відкриті, а всі сервіси можуть з'єднуватися між собою через внутрішню мережу.

React-частина збирається за допомогою ``npm run build`` і розміщується через вебсервер або спеціалізовані сервіси на кшталт Vercel чи Netlify. Для більшої безпеки рекомендується обмежити доступ до адміністративної панелі авторизацією.

Таким чином, система забезпечує ефективну роботу Telegram-бота з масштабованою та надійною інфраструктурою, де кожен компонент виконує свою чітку роль. Завдяки модульному підходу її легко підтримувати, доповнювати новими можливостями та адаптувати під різні сценарії використання.

3.2. Тестування боту та інтерфейсу

Перед тим як користувач зможе розпочати повноцінну взаємодію з Telegram-ботом, необхідно виконати його початковий запуск і пройти одноразову реєстраційну процедуру. Цей етап є обов'язковим і проводиться лише один раз, одразу після першого контакту користувача з ботом. Метою цієї реєстрації є збір основної ідентифікаційної інформації, яка дозволить системі точно визначати особу користувача та забезпечити персоналізований і безперебійний сервіс у подальшому.

Під час цього етапу бот автоматично зчитує такі ключові дані, як унікальний chat ID, який використовується для встановлення зв'язку з конкретним Telegram-акаунтом, а також нікнейм користувача, якщо він присутній у налаштуваннях Telegram. Крім того, фіксується ім'я, яке вказане в профілі, і найважливіше — актуальний номер телефону, прив'язаний до облікового запису Telegram. Саме номер телефону виступає додатковим засобом автентифікації та, у разі необхідності, дає змогу адміністрації або викладачам оперативно зв'язатися з учнем для уточнення деталей, вирішення організаційних питань або надсилання важливої інформації (рис. 3.2).

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

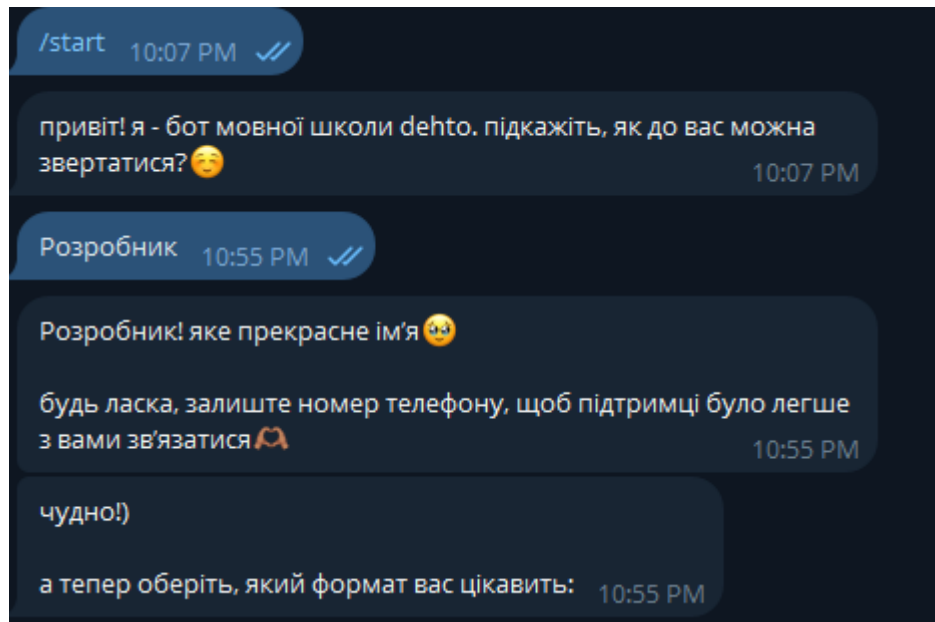


Рис.3.2. Етап реєстрації

Також на кожному етапі взаємодії з ботом передбачено використання інтерактивних клавіатур, які забезпечують зручну та інтуїтивно зрозумілу навігацію. Вони динамічно змінюються відповідно до поточного контексту діалогу, а також залежать від ролі користувача в системі. Кожна роль — наприклад, учень, викладач чи адміністратор — має власний набір опцій та функціональності, які відображаються через спеціально адаптовану клавіатуру (рис. 3.3).

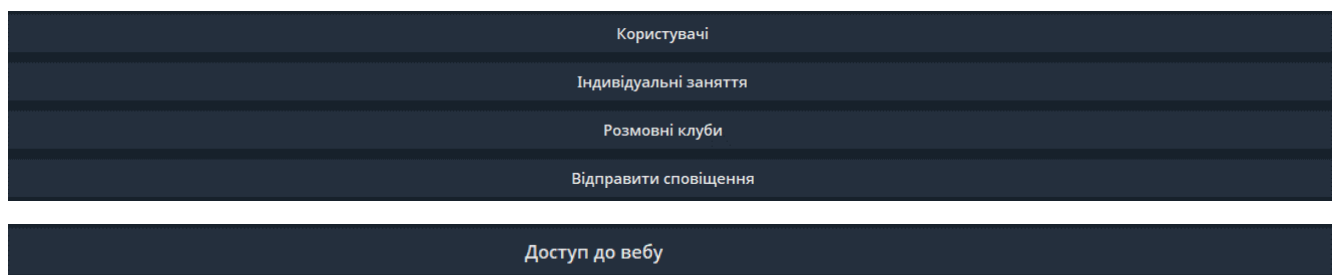


Рис.3.3. Клавіатура адміністратора

Це дозволяє спростити користувацький досвід, уникнути помилок при виборі дій і водночас обмежити доступ до функцій, не призначених для конкретного типу користувача. Такий підхід забезпечує як зручність для кожного користувача, так і контрольованість усієї логіки роботи бота. Наприклад, клавіатура головного меню для різних ролей (рис. 3.4-3.5).

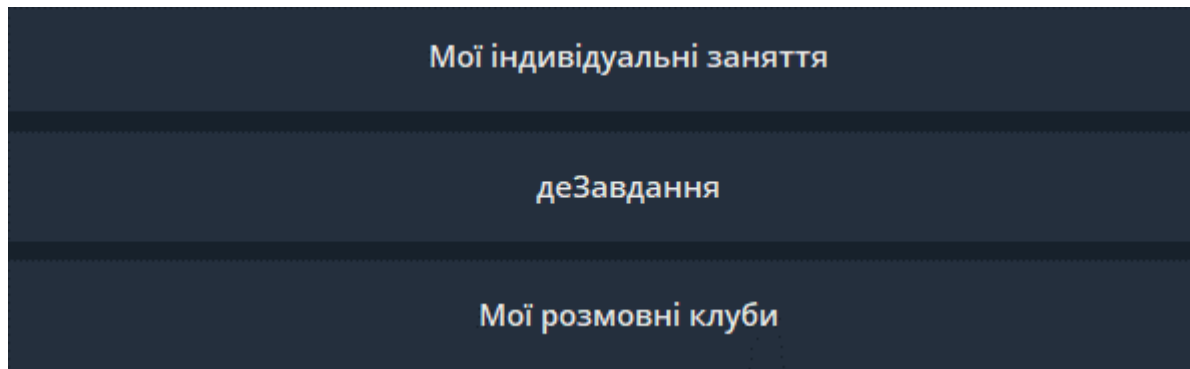


Рис.3.4. Клавіатура викладача

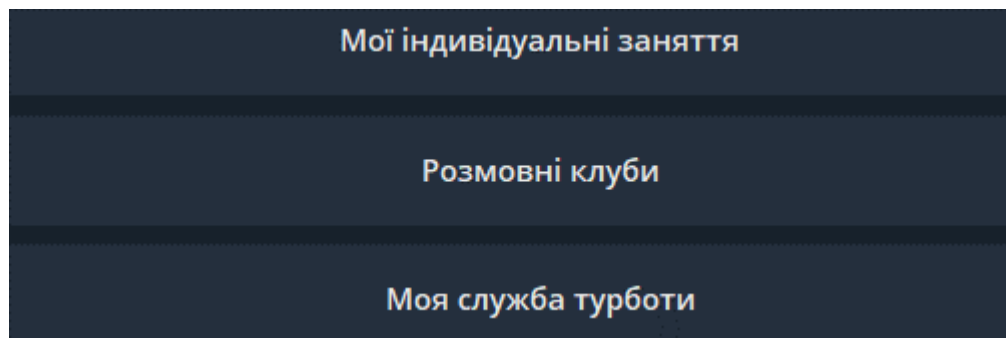


Рис.3.5. Клавіатура студента

Враховуючи мультирольність системи та різний рівень доступу для кожної категорії користувачів, доцільно послідовно розглядати функціональні можливості кожної ролі окремо. Такий підхід дозволяє структурувати опис і краще розуміти логіку взаємодії всередині системи. Почнемо з ролі адміністратора, яка володіє найширшими правами, і поступово перейдемо до викладачів та студентів.

Однією з ключових функцій адміністратора є розділ «Користувачі». Цей функціонал відповідає за повноцінне адміністрування всіх користувачів, зареєстрованих у системі (рис. 3.6).

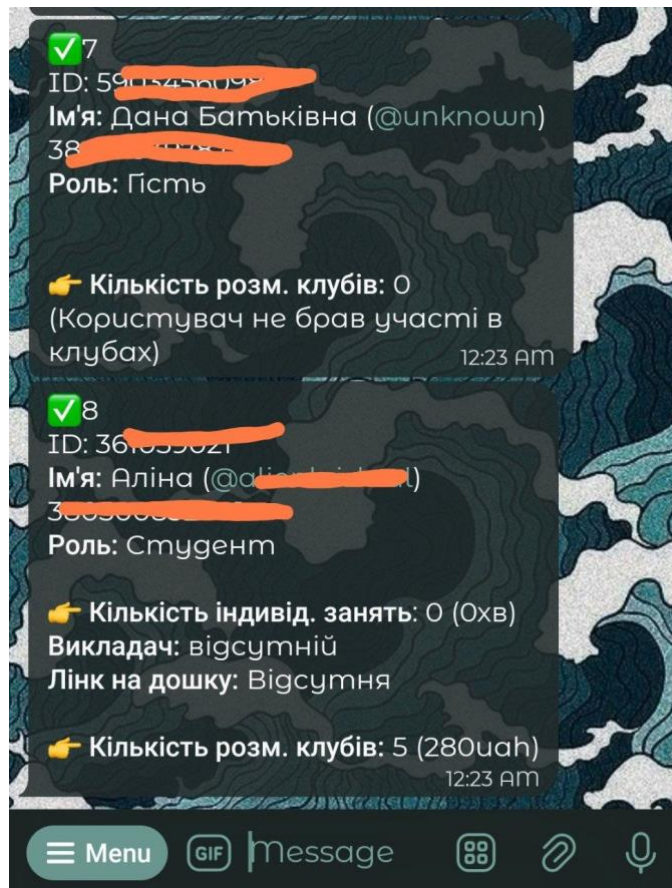


Рис.3.6. Перегляд всіх користувачів

Адміністратор має змогу переглядати, фільтрувати (рис. 3.7) та редагувати інформацію про кожного користувача — як студентів, так і викладачів. Зокрема, доступні дані такі як ім'я (рис. 3.8), Telegram-нікнейм, номер телефону, роль у системі, активність, статуси участі у клубних чи індивідуальних заняттях, а також загальна історія взаємодії з платформою. Через цей інтерфейс можна швидко змінювати ролі користувачів, блокувати або активувувати облікові записи, а також здійснювати ручне призначення певних дій — наприклад, додавання студента до групи чи закріплення за конкретним викладачем.

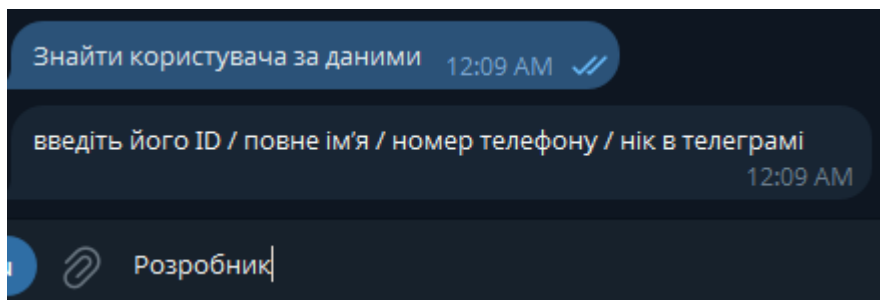


Рис.3.7. Пошук користувача за іменем

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

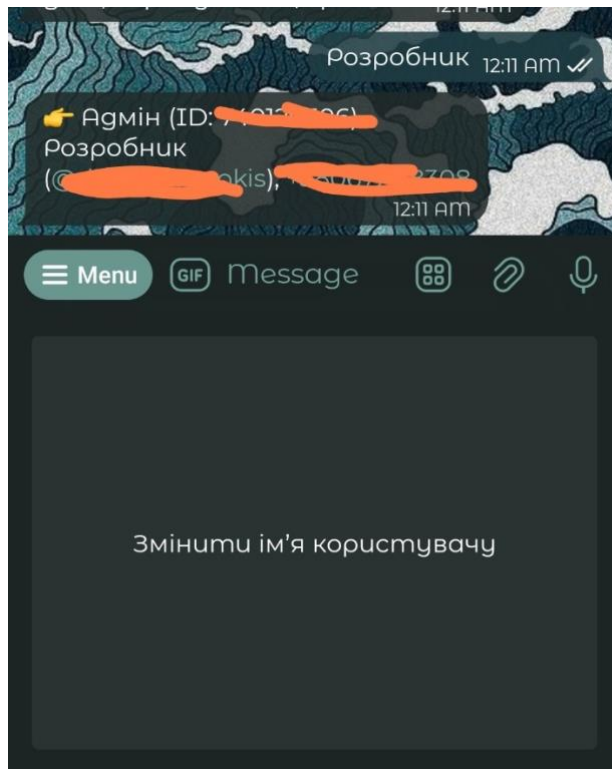


Рис.3.8. Користувача знайдено

Таким чином, функція «Користувачі» є базовим інструментом керування складом учасників навчального процесу та підтримки порядку в системі. Вона дозволяє адміністраторам оперативно реагувати на будь-які зміни, вести облік і забезпечувати контроль як за якістю сервісу, так і за структурною організацією взаємодії між усіма ролями (рис. 3.9).

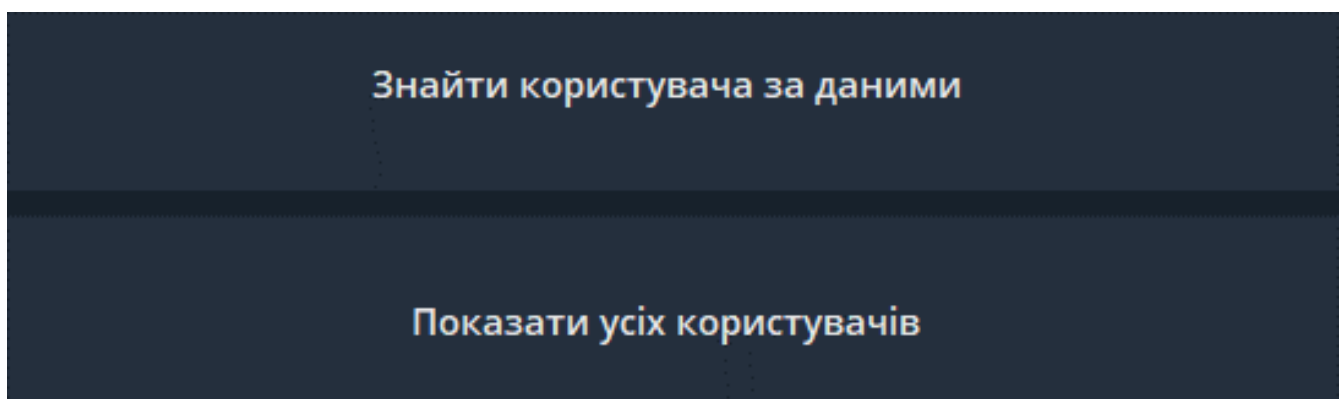


Рис.3.9. Клавіатура функції «Користувачі»

Функціонал від типу користувача може різнитися, як вже було зазначено вище, тому для гостей клавіатура щедріша (рис. 3.10)

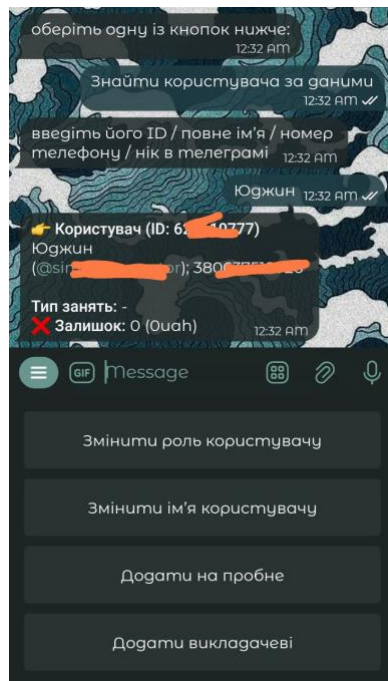


Рис.3.10. Пошук користувача (Гість)

Тепер розглянемо один із ключових інструментів адміністративного керування — функцію «Індивідуальні заняття». Цей розділ системи дозволяє адміністраторам ефективно організовувати навчальний процес у форматі персоналізованих занять, а також здійснювати гнучке управління як студентами, так і викладачами, що залучені до індивідуального формату (рис. 3.11).

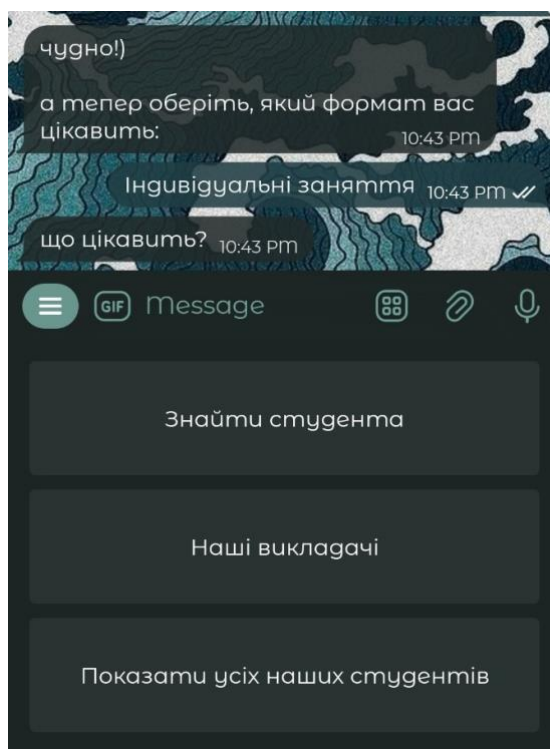


Рис.3.11. Головна клавіатура функції «Індивідуальні заняття»

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

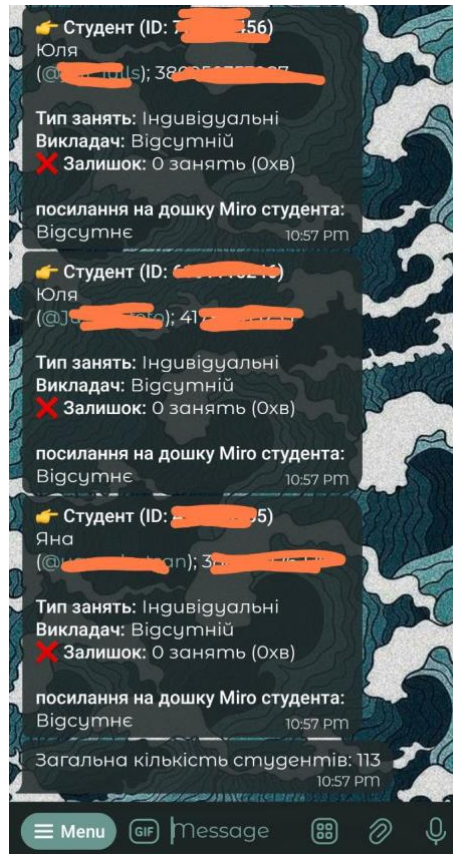


Рис.3.13. Перегляд всіх студентів Індивідуальних занять

Окрему увагу приділено управлінню викладачами (рис. 3.14). У цьому розділі адміністратор може переглядати актуальний розклад викладача, що дає змогу контролювати завантаженість і ефективність розподілу часу. Також можна ознайомитися зі списком усіх студентів, закріплених за конкретним викладачем, що сприяє контролю якості навчального процесу та можливості швидкого втручання у разі потреби. У випадках організаційних змін або з інших причин, викладача можна зняти з посади, понизивши його роль до рівня “гостя” — що формально відключає його від системи викладання, не видаляючи повністю обліковий запис (рис. 3.15).

Загалом, функція «Індивідуальні заняття» є потужним інструментом адміністрування, що дозволяє з високою точністю керувати всіма аспектами персоналізованого навчання, підтримуючи гнучкість, прозорість і ефективність освітнього процесу.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

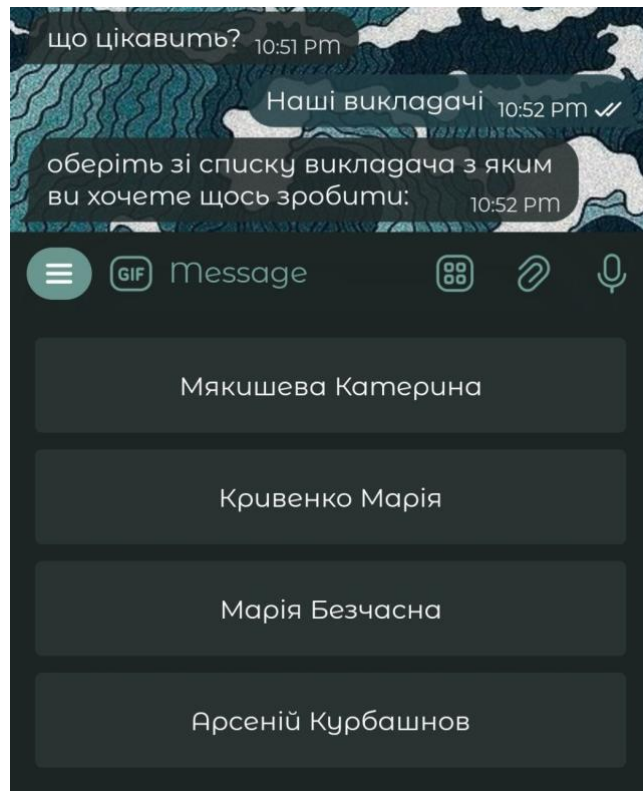


Рис.3.14. Вибір викладача

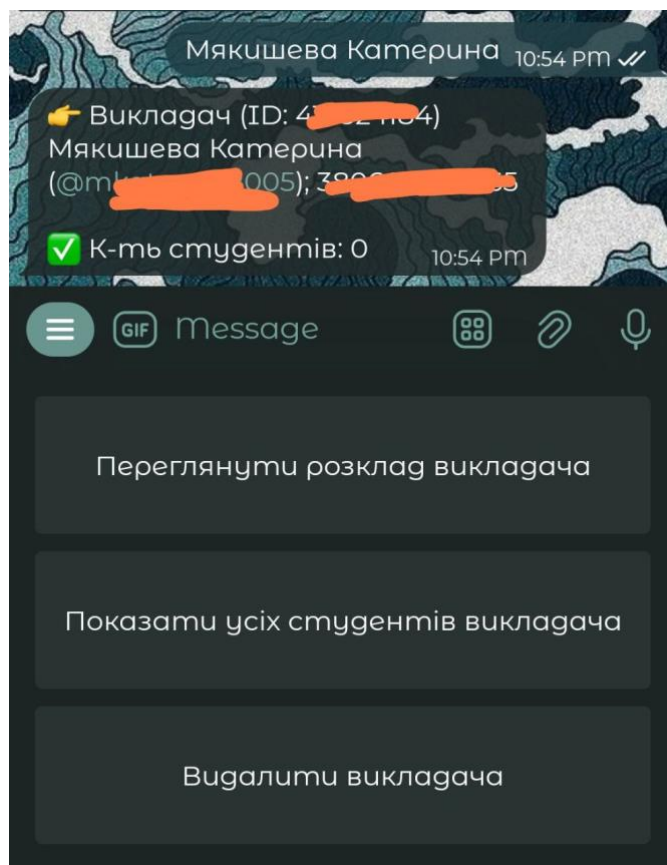


Рис.3.15. Функціонал для адміністрування користувачів ролі Викладач

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

Тепер перейдемо до розгляду наступної важливої функції — «Розмовні клуби». Цей розділ системи відповідає за організацію та адміністрування спеціальних групових занять, основна мета яких — розвиток та вдосконалення розмовних навичок у німецькій мові. Розмовні клуби створюють можливість для студентів практикувати живу мову в невимушеній атмосфері, покращуючи вимову, словниковий запас та впевненість у спілкуванні.

Функція «Розмовні клуби» поділена на два підрозділи: «Клуби» та «Особові справи», кожен з яких виконує свою окрему, але взаємодоповнюючу роль (рис. 3.16 та рис. 3.17).

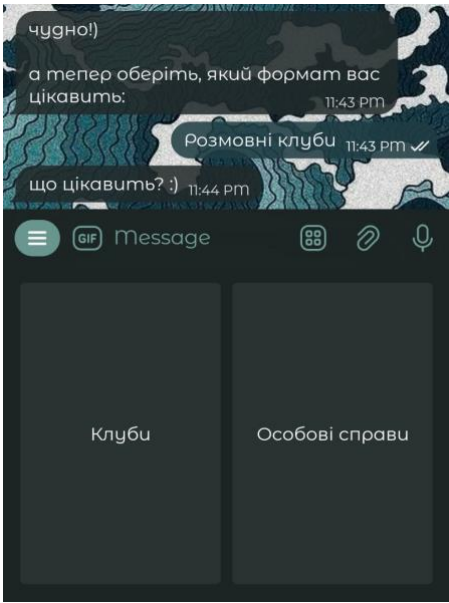


Рис.3.16. Головна клавіатура «Розмовні клуби»

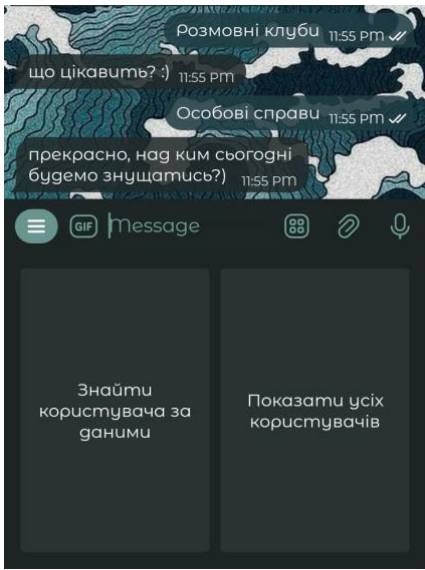


Рис.3.17. Клавіатура «Особові справи»

У розділі «Клуби» адміністратор має змогу створювати нові клуби (рис. 3.18), вказуючи основні параметри: назву клубу, викладача, який його вестиме, точний час проведення заняття, а також приєднувати додатковий файл із документацією — навчальні матеріали або інструкції, які будуть корисними студентам під час участі (рис. 3.19).

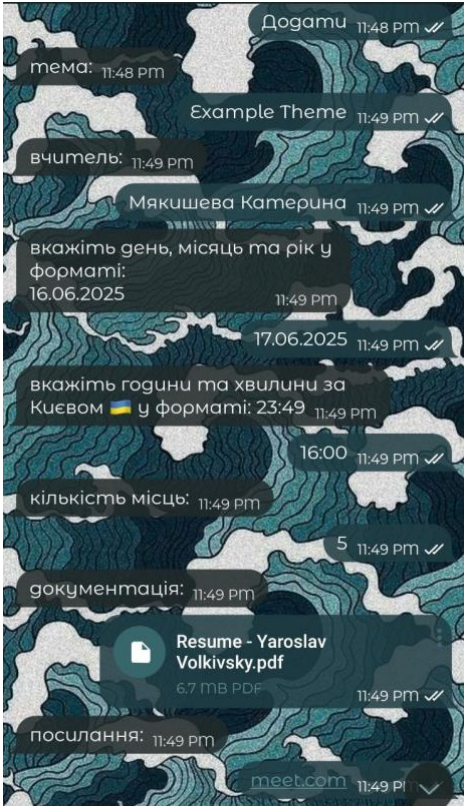


Рис.3.18. Процес додавання клубу

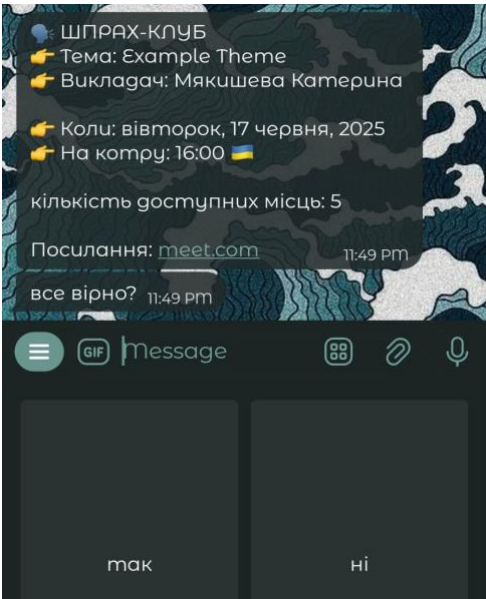


Рис.3.19. Перевірка та постинг клубу

Інтерфейс також дозволяє редагувати будь-який із вказаних параметрів у вже створених клубах, у разі якщо потрібно змінити час, викладача або оновити матеріали. За потреби, розмовний клуб можна видалити. Крім того, доступна функція перегляду усіх актуальних клубів — зручна опція для швидкого аналізу, нагадування (рис. 3.20) або перевірки розкладу (рис. 3.21).

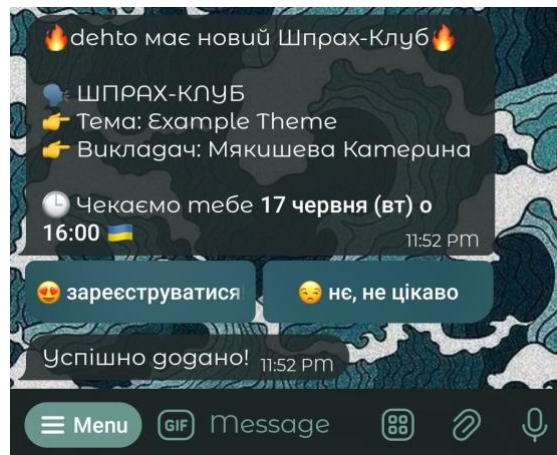


Рис.3.20. Сповіщення про новий клуб зі зручною можливістю запису

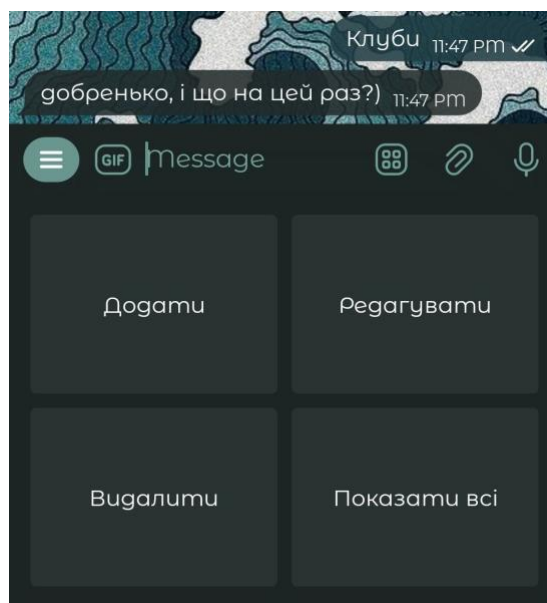


Рис.3.21. Клавіатура підфункції «Клуби»

Другий підрозділ — «Особові справи» — відповідає за адміністрування учасників розмовних клубів, тобто студентів та викладачів, які беруть участь у заняттях. Аналогічно до інших адміністративних розділів, тут можна керувати профілями користувачів, однак акцент зроблено саме на налаштування параметрів, пов'язаних із відвідуванням клубів. Наприклад, адміністратор може змінювати баланс

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

занять у студента (тобто кількість доступних відвідувань клубів), а також активний пакет участі — якщо студент переходить на інший тариф або оформлює нову підписку. У випадках, коли певні транзакції були проведені поза межами системи (наприклад, оплата відбулася вручну або через зовнішній канал), дані можна синхронізувати вручну, безпосередньо внісши зміни до профілю студента через відповідний інтерфейс (рис. 3.22 та рис. 3.23).

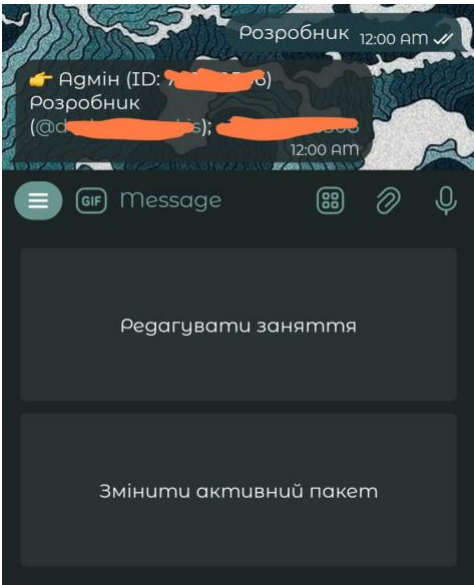


Рис.3.22. Підфункції з «Знайти користувача за даними»

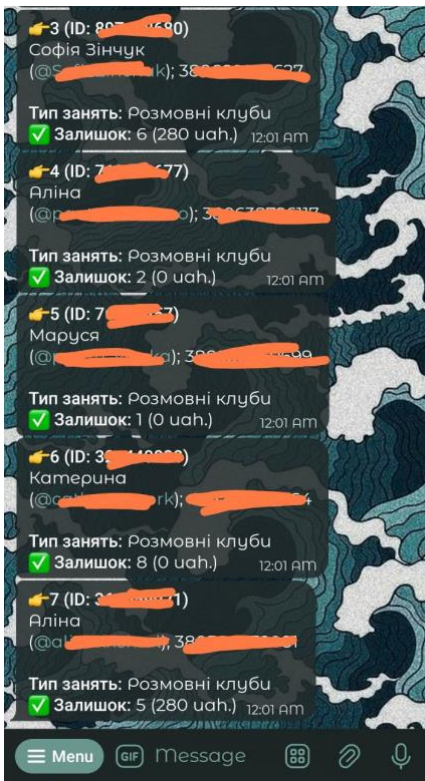


Рис.3.23. Всі користувачі з «Розмовні клуби»

Загалом, функція «Розмовні клуби» забезпечує повний цикл організації групової мовної практики — від створення та редагування клубів до управління правами доступу та ресурсами студентів, тим самим підтримуючи гнучкість і якість навчального процесу.

Далі розглянемо одну з ключових і водночас найпотужніших функцій, доступних адміністраторам системи, а саме «Відправити сповіщення». Цей інструмент є критично важливим для забезпечення ефективної комунікації між адміністрацією та користувачами, а також для оперативного інформування про будь-які зміни, оновлення або важливі події в рамках освітнього процесу.

Функція дозволяє надсилати повідомлення будь-якого змісту та формату — як текстові повідомлення, так і зображення, документи або інші мультимедійні вкладення. Адміністратор має змогу самостійно обирати, кому саме буде надіслане повідомлення: це може бути як глобальна розсилка для всіх користувачів платформи, так і таргетоване повідомлення для певної категорії — наприклад, лише студентам, лише викладачам або лише гостям. Більше того, існує можливість адресного надсилання повідомлення конкретному користувачу, що є особливо корисним у випадках, коли потрібно швидко передати важливу інформацію без залучення сторонніх.

Завдяки цій функції можна надсилати сповіщення про нові розмовні клуби, нагадування про заняття, повідомлення про зміни в розкладі, запрошення на заходи, акції чи знижки, а також вирішувати будь-які інші організаційні чи адміністративні питання. Гнучкість фільтрації одержувачів, зручний інтерфейс і висока швидкість доставки роблять цю функцію незамінним інструментом у щоденній роботі адміністратора.

Загалом, «Відправити сповіщення» це універсальний канал прямого зв'язку з аудиторією, який дозволяє не лише підтримувати стабільну комунікацію, а й створювати відчуття постійної присутності та турботи з боку адміністрації (рис. 3.24).

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

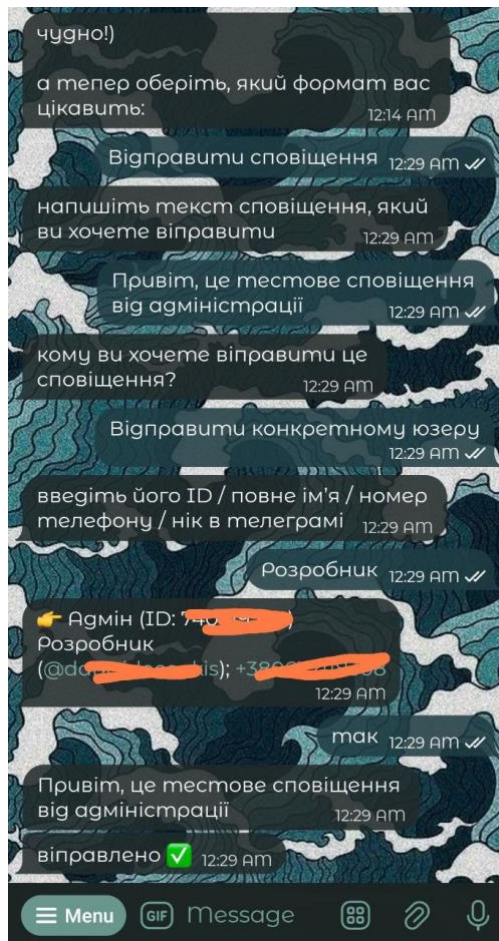


Рис.3.24. Демонстрація роботи функції «Віправити сповіщення» зі специфічним користувачем

Ще однією важливою та корисною функцією в арсеналі адміністратора є «Доступ до вебу». Вона виконує декілька критичних завдань, пов'язаних із авторизацією та безпекою доступу до веб-версії адміністративної панелі.

Перш за все, ця функція дозволяє адміністратору швидко дізнатися Chat ID поточного користувача. Це унікальний ідентифікатор, який використовується для ідентифікації кожного користувача Telegram у межах бота. Наявність цього ID є необхідною для налаштування персоналізованого доступу до веб-інтерфейсу, а також для точного контролю за правами доступу до тих чи інших розділів системи.

Крім того, у межах цієї функції доступна можливість змінити пароль для входу до адмін-панелі веб-версії. Це особливо корисно в тих випадках, коли є необхідність оновити облікові дані з міркувань безпеки або при зміні адміністративного складу. Варто зауважити, що з міркувань безпеки система не зберігає та не відображає сам пароль у відкритому вигляді. Усі паролі проходять процес хешування — це

стандартна практика в сфері кібербезпеки, яка гарантує, що навіть у разі несанкціонованого доступу до бази даних справжній пароль залишиться захищеним.

Таким чином, функція «Доступ до вебу» виконує роль інструменту технічного адміністрування, сприяючи безпечному та контрольованому доступу до веб-інтерфейсу системи, при цьому зберігаючи конфіденційність критичних даних (рис. 3.25).

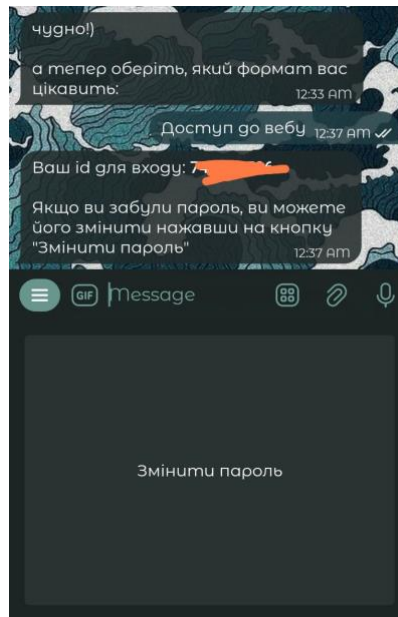


Рис.3.25. Демонстрація роботи функції «Доступ до вебу»

Після детального ознайомлення з функціоналом, доступним адміністратору, логічним наступним кроком буде перехід до розгляду можливостей, які система надає викладачам. Враховуючи роль викладача як ключового учасника освітнього процесу, інтерфейс та функціональність для цієї категорії користувачів були спеціально адаптовані під їхні щоденні потреби та обов'язки.

Перш за все, слід звернути увагу на клавіатуру головного меню викладача, яка представлена на рисунку 3.26. Вона є інтуїтивно зрозумілою, водночас функціональною — усі основні дії згруповані логічно та доступні з головного екрану, що дозволяє викладачу швидко орієнтуватися у системі навіть у межах щільного робочого графіка.

Серед основних можливостей, доступних викладачеві, — перегляд власного розкладу, що дозволяє в реальному часі бачити усі заплановані заняття, включаючи

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

як індивідуальні, так і розмовні клуби. Також викладач може переглядати профілі студентів, закріплених за ним, що дає змогу слідкувати за їхнім прогресом, перевіряти залишок занять, наявність доступу до матеріалів та інші ключові параметри.

Окрім цього, система дозволяє залишати нотатки або коментарі щодо кожного заняття — це корисно для відстеження динаміки навчання, фіксації успіхів чи труднощів студентів, а також для планування майбутніх сесій. У випадку розмовних клубів викладач також має доступ до документації, підготовленої адміністрацією або ним особисто, що полегшує проведення занять та забезпечує стабільну якість контенту.

Таким чином, функціонал викладача побудований так, щоб максимально спростити рутинні процеси та зосередити увагу саме на якісному проведенні навчання. Він дає змогу не лише контролювати власне навантаження та ефективність занять, а й взаємодіяти зі студентами у структурований, продуманий та зручний спосіб.

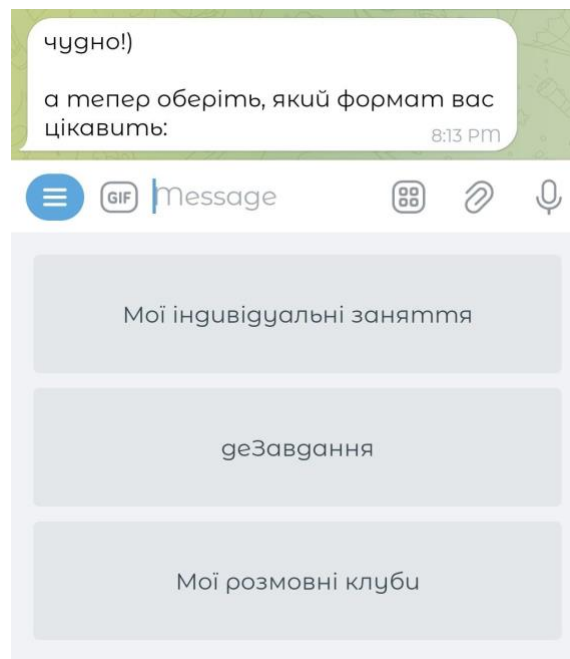


Рис.3.26. Головна клавіатура ролі Вчитель

Розглянемо одну з ключових функцій для викладача «Мої індивідуальні заняття» (рис. 3.27). Ця функція є основним інструментом для планування та управління навчальним процесом у рамках персоналізованого формату занять, і забезпечує повний контроль над розкладом та студентською активністю.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

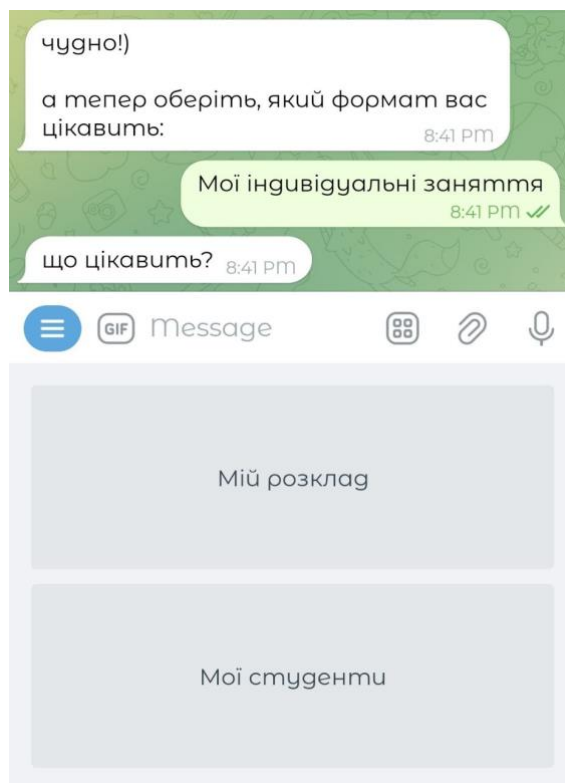


Рис.3.27. Функція «Індивідуальні заняття»

Насамперед, функція надає можливість переглядати актуальний розклад занять викладача. Це дозволяє завжди бути в курсі своїх навчальних сесій, уникати накладок у графіку та ефективно планувати свій час. Усі заплановані індивідуальні заняття відображаються у зручному форматі, що допомагає швидко орієнтуватися у своєму навантаженні.

Окрім перегляду, викладач має змогу самостійно запланувати нові заняття, обираючи зручний час і студента. Така гнучкість особливо важлива у випадках, коли студент і викладач узгоджують заняття в індивідуальному порядку.

У разі потреби передбачена функція перенесення занять — якщо з певних причин необхідно змінити час чи дату вже запланованої сесії, викладач може легко це зробити без залучення адміністрації. Так само доступна опція видалення занять, якщо вони були помилково заплановані або більше неактуальні.

Ще одна важлива особливість — можливість запланувати пробні заняття для гостей, які вже були попередньо закріплені за цим викладачем. Це дозволяє гнучко організовувати ознайомчі сесії з потенційними студентами, демонструвати якість навчання та допомагати їм приймати рішення про подальше навчання.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

Загалом, функція «Мої індивідуальні заняття» створена для того, щоб викладач мав повний контроль над своїм навчальним процесом — від перегляду розкладу до управління кожним окремим заняттям. Це підвищує ефективність роботи та створює комфортне, організоване середовище як для викладача, так і для студентів (рис. 3.28 та рис. 3.29).

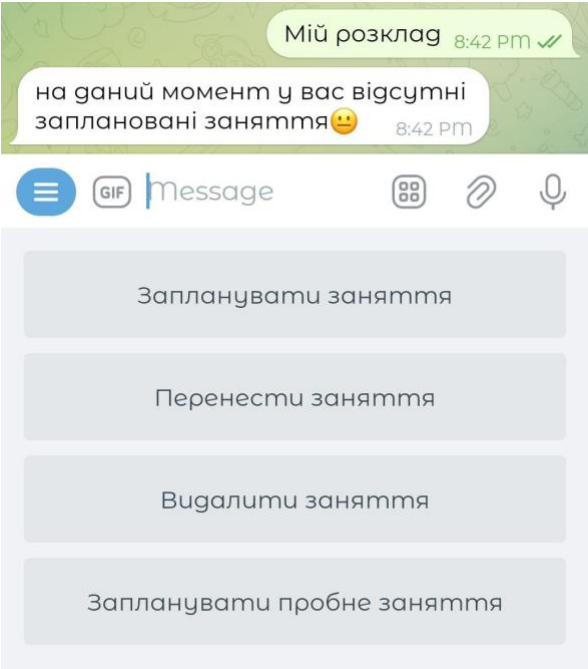


Рис.3.28. «Мій розклад» та підфункції



Рис.3.29. «Мої студенти», котрі записані у викладача

Наступною важливою функцією, яку слід розглянути у межах функціоналу викладача, є «деЗавдання». Вона відіграє ключову роль у забезпеченні асинхронної взаємодії між викладачем та студентом і дає змогу ефективно організувати домашню або додаткову роботу поза межами основних занять.

Функція «деЗавдання» дозволяє викладачеві надавати студентам завдання з використанням усього арсеналу можливостей, які пропонує Telegram. Це можуть бути текстові повідомлення, фото, відео, документи, голосові повідомлення, а також будь-які інші формати, які можуть бути корисними у навчальному процесі. Такий підхід робить завдання не лише функціональними, але й гнучкими — їх можна адаптувати до різних форматів подачі матеріалу, рівнів студентів та цілей заняття.

Після того, як викладач створює завдання, студент отримує автоматичне сповіщення в боті. Таким чином, жодне завдання не залишається непоміченим, а студент завжди має чітке уявлення про те, що і коли потрібно виконати. Це значно покращує організацію навчального процесу та дозволяє утримувати високу дисципліну.

Аналогічно, як тільки студент надсилає відповідь на завдання, система автоматично сповіщає викладача, що завдання надійшло і готове до перевірки. Це дає можливість своєчасно реагувати на роботу учня, надавати зворотний зв'язок та, у разі потреби, скеровувати його для доопрацювання або закріплення матеріалу.

Функція «деЗавдання» не просто інструмент для обміну навчальними матеріалами. Це повноцінна двостороння система взаємодії, яка сприяє розвитку відповідальності у студентів, підвищує якість навчання та дозволяє викладачам ефективно керувати навчальним процесом навіть поза межами основного заняття (рис. 3.30).

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

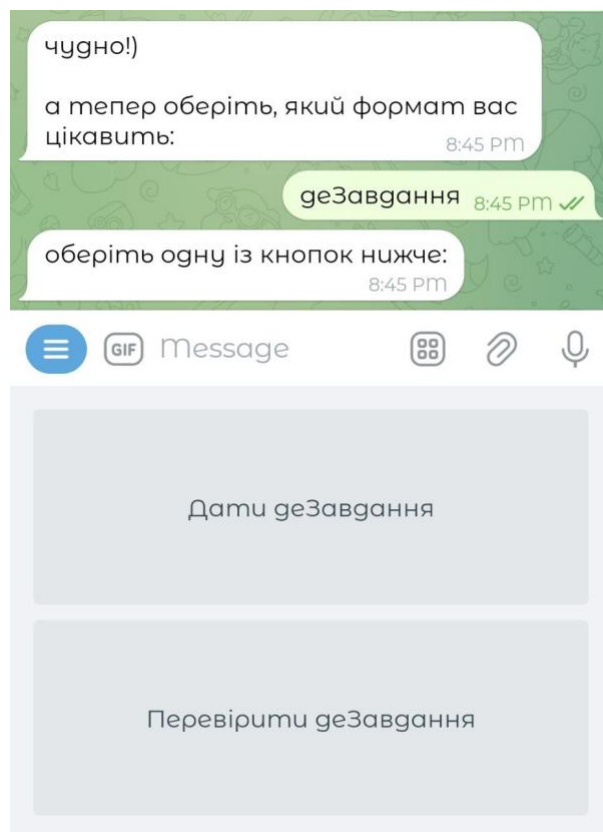


Рис.3.30. Функція «деЗавдання»

Функція «Мої розмовні клуби» створена спеціально для того, щоб надати викладачеві зручний і швидкий доступ до актуального розкладу розмовних клубів, у проведенні яких він бере участь або за які безпосередньо відповідає.

Завдяки цій функції викладач має змогу в будь-який момент переглянути всі заплановані зустрічі, що дозволяє не лише краще організовувати свій робочий час, а й завчасно готуватися до кожного клубу. Усі сесії відображаються у структурованому вигляді, що робить навігацію простою й інтуїтивно зрозумілою.

Функція особливо корисна для тих викладачів, які паралельно ведуть кілька груп або чергуються між різними клубами. Вона дозволяє контролювати навантаження, уникати накладок у розкладі та забезпечити стабільну якість підготовки до кожного заняття (рис. 3.31).

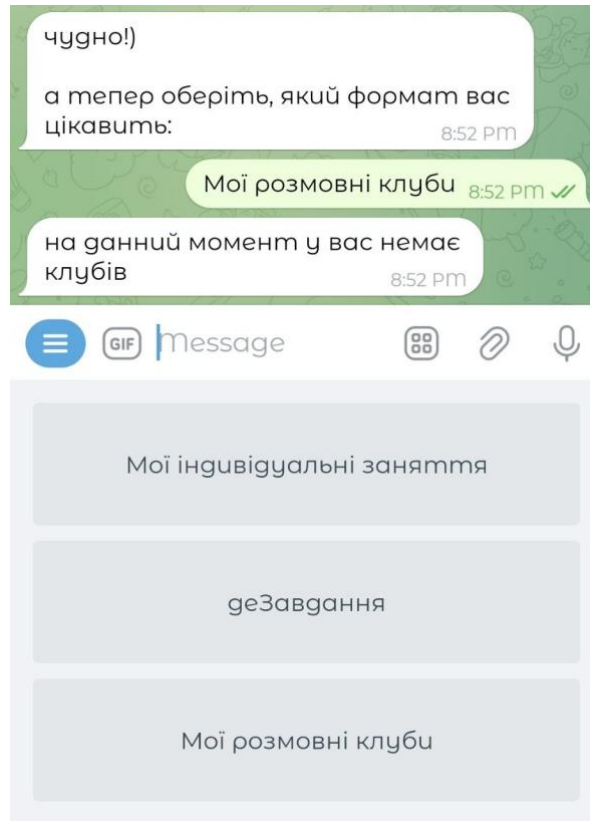


Рис.3.31. Функція «Мої розмовні клуби»

Переходячи до розгляду останньої, але не менш важливої ролі в системі, слід зупинитися на студентові — саме він є кінцевим користувачем усіх освітніх послуг (рис. 3.32 та рис. 3.33).

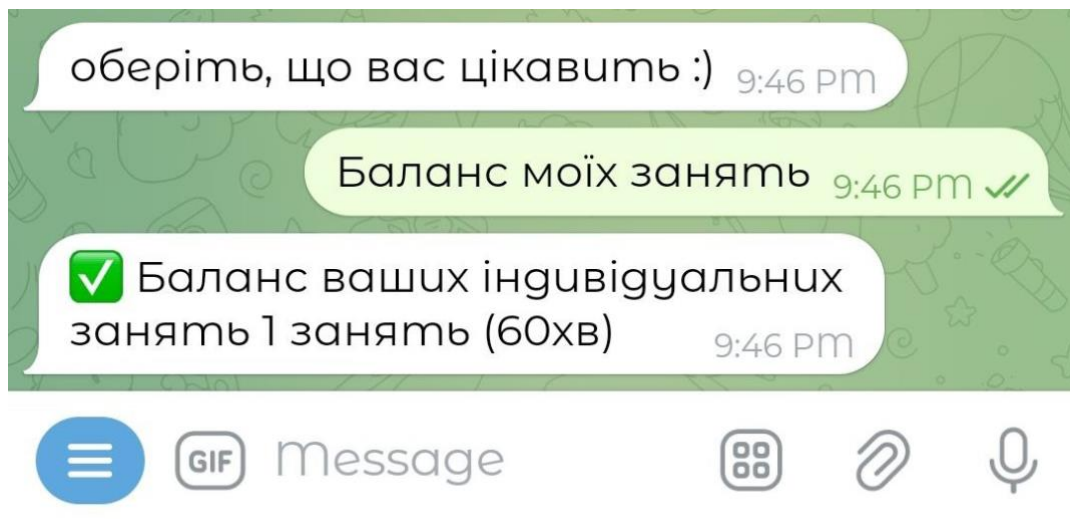


Рис.3.32. Демонстрація підфункції «Баланс моїх занять»

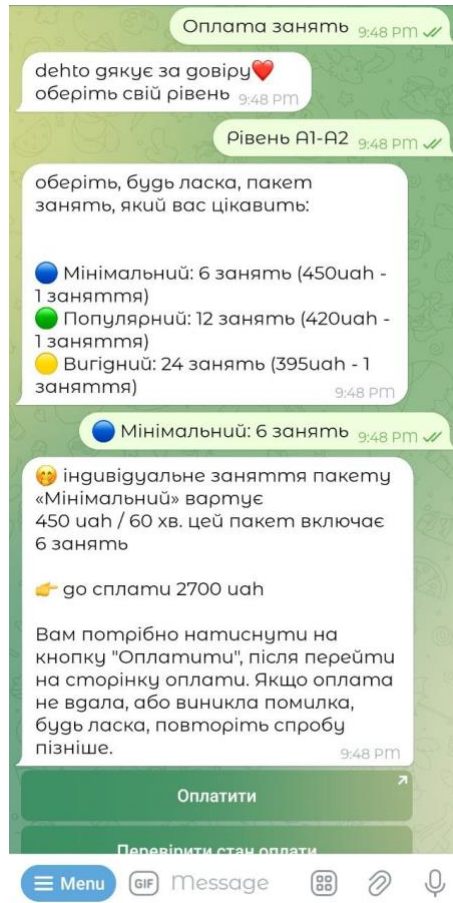


Рис.3.33. Оплата занять (API Monobank)

Тому функціонал, доступний для нього, має бути максимально зручним, інтуїтивним та корисним у повсякденному навчальному процесі (рис. 3.34).

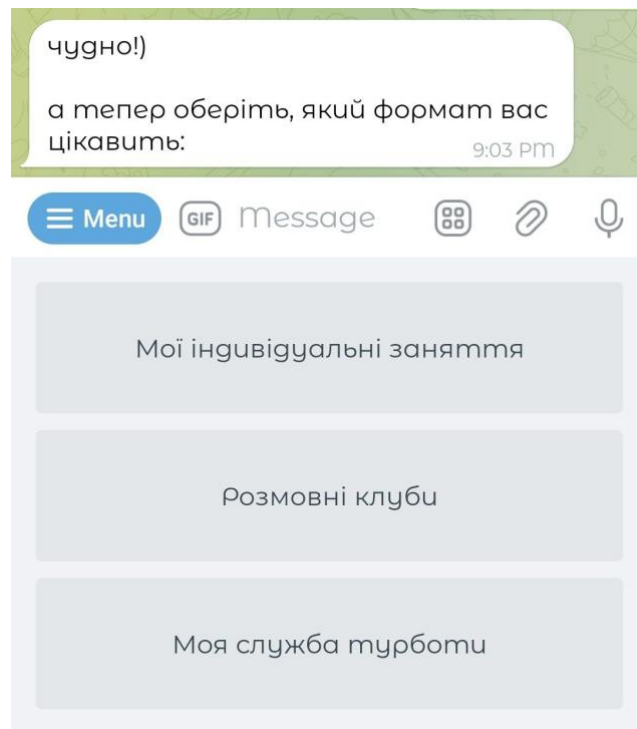


Рис.3.34. Клавіатура головного меню

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

Насамперед студент має змогу оплачувати навчання — як у форматі індивідуальних занять, так і для участі в розмовних клубах. Оплата проводиться через інтегровані інструменти, які забезпечують зручність, безпеку і швидкість фінансових операцій. Це дозволяє студентам самостійно керувати своїм освітнім балансом та уникати зайвої адміністративної бюрократії (рис. 3.35).

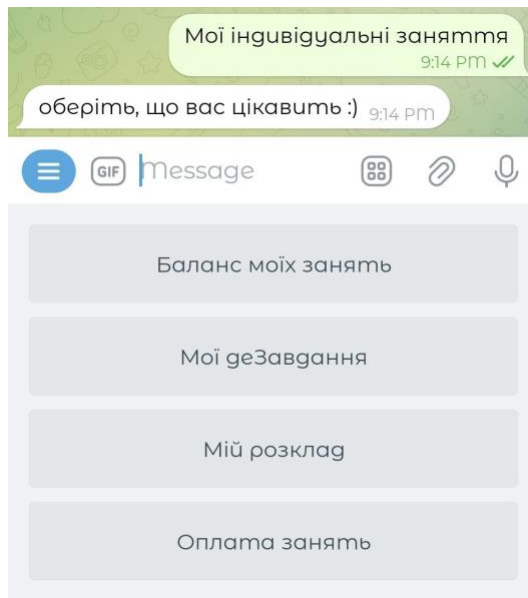


Рис.3.35. Підфункції «Мої індивідуальні заняття»

Крім того, студент отримує можливість працювати з «завданнями» — це завдання, які йому надсилає викладач. Студент не лише отримує їх у зручному форматі безпосередньо в Telegram, але й може надсилати виконані відповіді, які потім перевіряються викладачем. Це дозволяє зберігати постійний зворотний зв'язок між учнем і наставником навіть поза межами основних занять.

Також студент має доступ до розкладу занять, як індивідуальних, так і розмовних клубів. У будь-який момент він може переглянути свій графік, зорієнтуватися в часі, не забути про заплановані сесії і вчасно підготуватися до них. Це формує звичку до самодисципліни та покращує навчальну ефективність.

Функція реєстрації на розмовні клуби дозволяє студенту самостійно обирати ті зустрічі, які йому цікаві або відповідають його рівню підготовки, і долучатися до них без зайвих складнощів. Завдяки цьому студент сам стає учасником формування власної освітньої траєкторії (рис. 3.36).

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

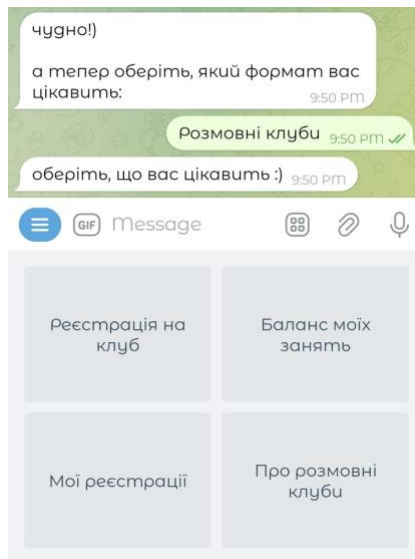


Рис.3.36. Підфункції «Розмовні клуби»

Окремої уваги заслуговує сервіс «Моя служба турботи» — це справжній центр підтримки, де студент (рис. 3.37) має можливість звернутися до сапортера (оператора підтримки) (рис. 3.38) в реальному часі, використовуючи всі зручні засоби Telegram: текст, голос, медіа.

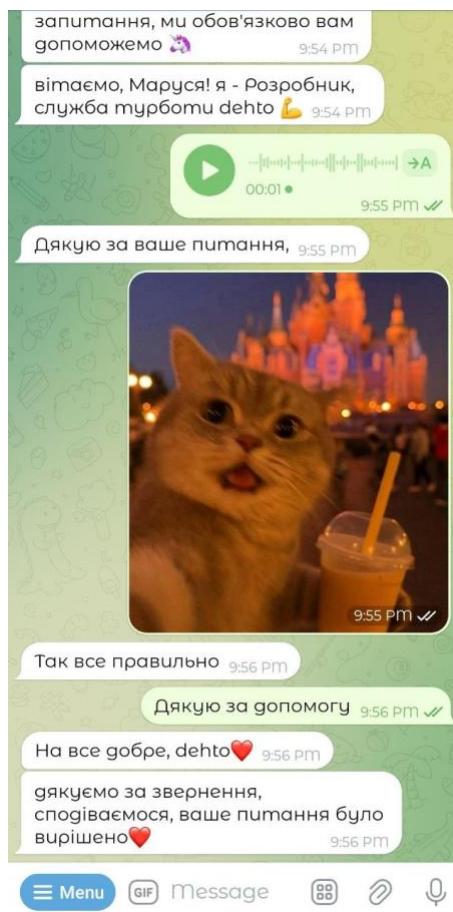


Рис.3.37. «Служба турботи» сторона студента

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				72
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис.3.38. «Служба турботи» сторона сапортера

У випадку будь-яких труднощів або запитань студент не залишається наодинці з проблемою, а миттєво отримує допомогу від реальної людини. Це створює відчуття турботи, довіри до платформи та значно підвищує рівень задоволеності навчанням (рис. 3.39).

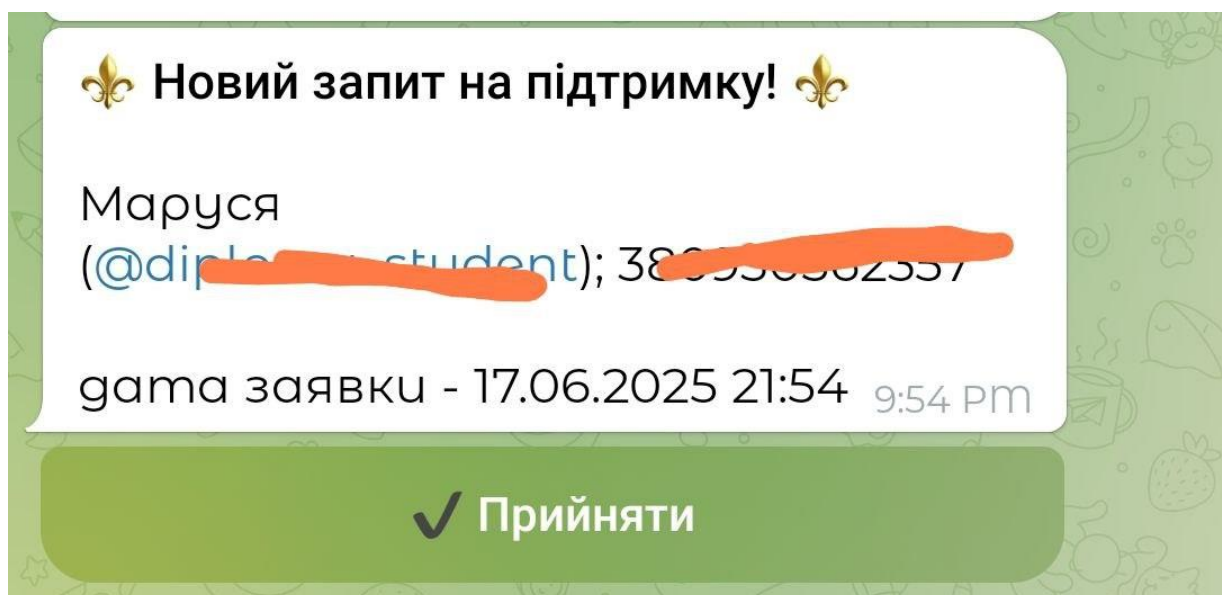


Рис.3.39. Запит на підтримку Сапортеру

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

3.3. Тестування адмін панелі та інтерфейсу

Працювати з Telegram-ботом дійсно зручно та швидко — він завжди «під рукою», знаходиться в межах кількох кліків, і не потребує жодних зайвих дій для доступу до функцій. Проте, варто визнати, що не для всіх користувачів такий формат є оптимальним. У випадках, коли потрібна ширша оглядовість, більше простору для аналітики або просто комфортна робота за комп'ютером, стає доречним використання повноцінного веб-інтерфейсу (рис. 3.40).

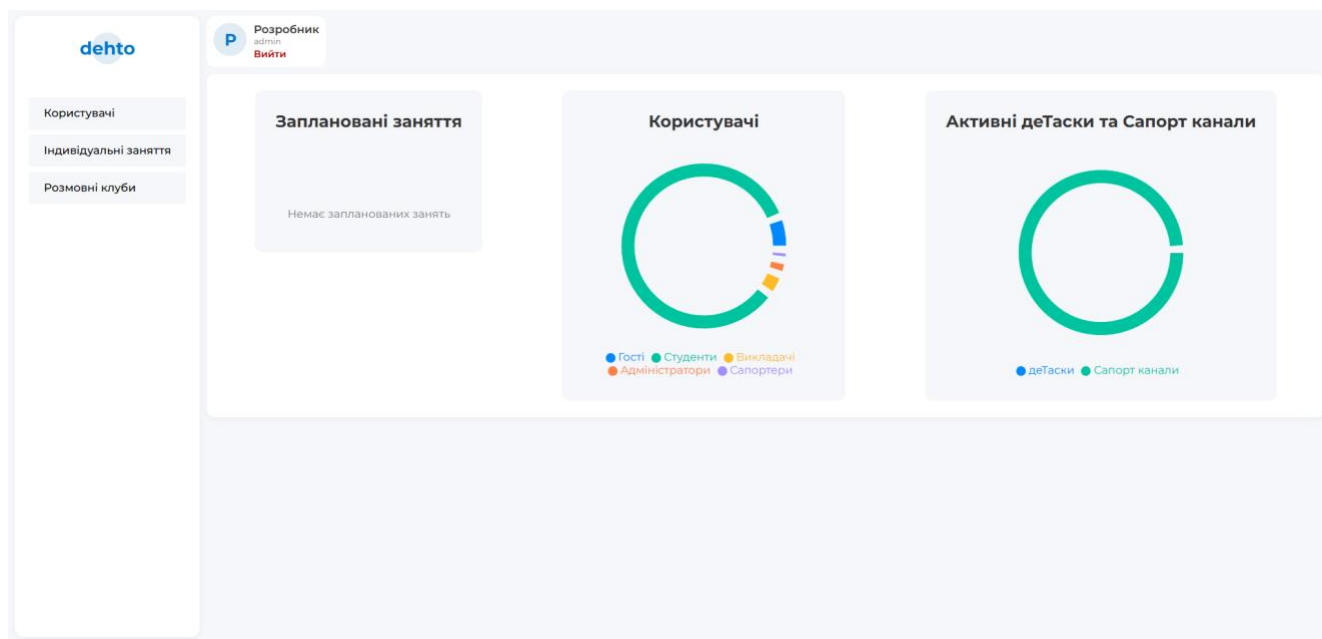


Рис.3.40. Дашборд адмін панелі

Саме тому Telegram-бот доповнений спеціальним вебзастосунком, створеним з використанням сучасної бібліотеки React, який виконує роль інтерфейсу для адміністрування та керування процесами. Цей вебдодаток взаємодіє з бекендом через REST API, реалізований на основі фреймворку Express, що забезпечує надійний та масштабований зв'язок між клієнтом і сервером.

Такий підхід дозволяє побачити повну картину всієї системи у зручному візуальному форматі, що особливо актуально для адміністраторів або співробітників, які займаються керуванням користувачів і навчального процесу. Адмін-панель вебверсії дає змогу працювати з основними функціями, які також реалізовані в боті. Зокрема, це розділи «Користувачі» та «Індивідуальні заняття» (рис. 3.41).

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

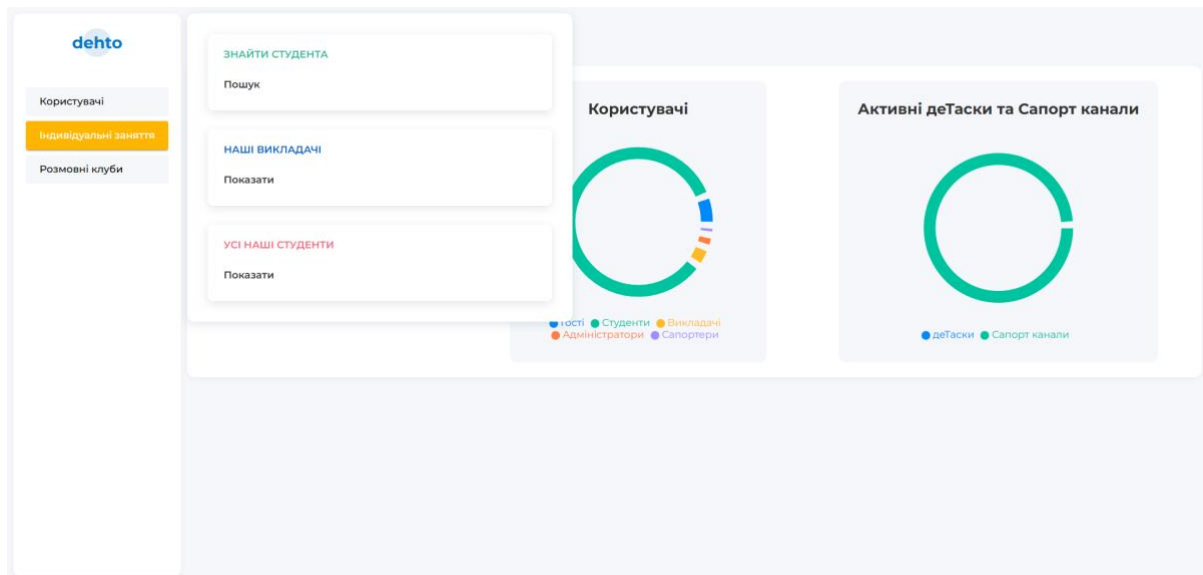


Рис.3.41. Відкрита функція «Індивідуальні заняття»

Інтерфейс представлений у більшому, зручнішому форматі. Він підтримує масові дії, зокрема сортування (рис. 3.42) та швидкий пошук (рис. 3.43) і інші.

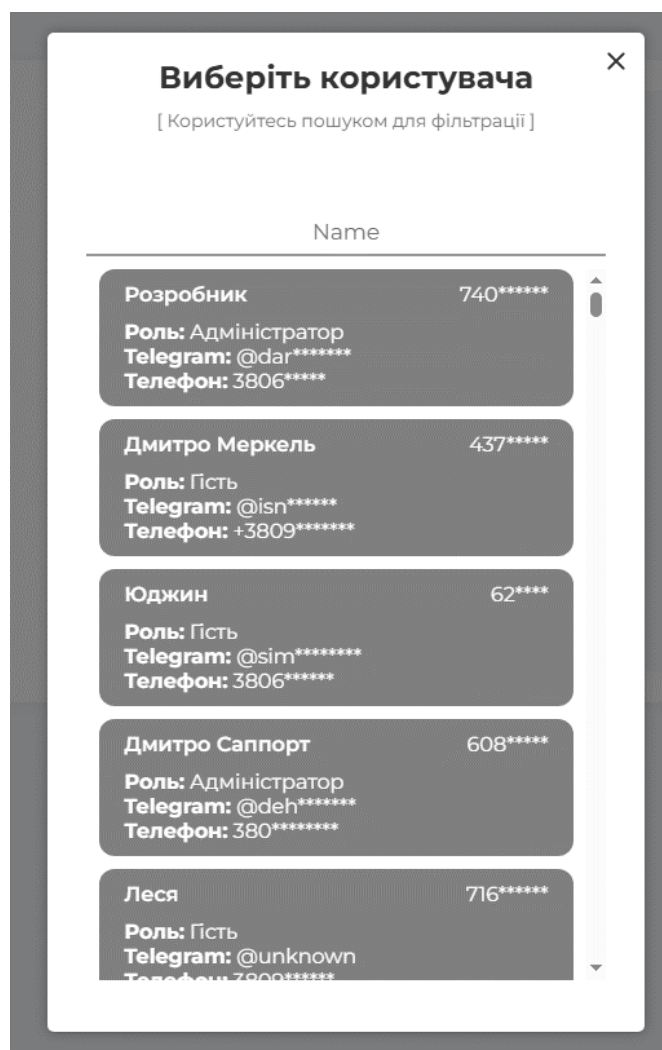


Рис.3.42. Класичний список вибору, в даному випадку користувачів

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

Рис.3.43. Класичне поле пошуку користувачів

З міркувань безпеки, доступ до адмін-панелі захищено. Вхід відбувається за допомогою унікального Chat ID користувача Telegram, у поєднанні з паролем, який, для додаткової безпеки, зберігається у вигляді хешу й не відображається в жодному вигляді. Оновлення паролю можливе лише через Telegram-бота й доступне тільки для користувачів із відповідною роллю (наприклад, адміністратора) (рис. 3.44 та рис. 3.45).

Рис.3.44. Класичне повідомлення про помилку

Рис.3.45. Вхід у систему

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

Таким чином, поєднання Telegram-бота та React-вебінтерфейсу формує універсальне та гнучке середовище для роботи з освітньою платформою. Незалежно від того, чи користувач працює з телефону в дорозі, чи з комп'ютера в офісі — він завжди має зручний інструмент під рукою (рис. 3.46).

Користувачі

< **Розробник** ×

[Виберіть операцію та виконайте її]

Telegram Id: 7401*****

Повне ім'я: Розробник

Роль: Адміністратор

Номер телефону: 38067*****

Нік в Telegram: @dark*****

Змінити ім'я

Name

Рис.3.46. Інформація про користувача з можливістю зміни імені

Висновки до третього розділу

У цьому розділі було детально розглянуто архітектуру та процес розгортання Telegram-бота, який є центральною ланкою системи. Було надано чітке пояснення кожного структурного компонента, його ролі в загальній системі, а також показано, як окремі елементи взаємодіють між собою. Такий підхід дозволив отримати цілісне уявлення про логіку побудови та функціонування системи, розкрити принципи, на яких вона базується, та зрозуміти взаємозв'язки між її складовими.

Окрему увагу було приділено демонстрації механіки роботи самого Telegram-бота — зокрема, було показано, як відбувається взаємодія з користувачем через інтерфейс месенджера Telegram. Було розглянуто логіку розмежування функціоналу для різних типів ролей (адміністратор, викладач, студент), що дозволило оцінити гнучкість та адаптивність бота до конкретних потреб кожного користувача.

Крім цього, було продемонстровано можливості веб-інтерфейсу для ПК у вигляді адміністративної панелі, створеної на React. Цей інструмент доповнює функціональність Telegram-бота, надаючи зручний візуальний інтерфейс для роботи з основними модулями системи. Вебпанель значно полегшує адміністрування, дозволяє краще аналізувати інформацію, ефективніше керувати користувачами та заняттями, а також підвищує загальний комфорт у роботі з платформою dehto.

Таким чином, розділ наочно показав, як усі компоненти системи — від сервера до інтерфейсів Telegram і вебпанелі — утворюють єдиний, взаємопов'язаний механізм, що забезпечує повноцінну, ефективну та зручну освітню взаємодію.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано Telegram-бота для німецької школи dehto, проведено аналіз предметної області та порівняно різних ботів, що працюють на основі месенджера Telegram. На основі цього аналізу були визначені основні вимоги до Telegram-боту. В процесі дослідження була розроблена архітектура, підходи та основні технології реалізації ПО.

Проектування бота включало аналіз варіантів його використання та проектування структури, включаючи моделювання взаємодії між складовими частинами. Також було спроектовано дві бази даних для берігання основної незалежної інформації і для зберігання проміжних даних або даних, які були зібрані в ході використання бота.

Описано алгоритм роботи з ботом та послідовні пункти варіантів його використання. У цьому розділі було розглянуто основний функціонал Telegram-бота: реєстрацію користувачів, розподіл ролей, управління індивідуальними заняттями, розмовними клубами, завданнями та сповіщеннями. Також описано можливості веб-інтерфейсу, який дублює ключові функції бота для зручної роботи через ПК. Система забезпечує гнучку та ефективну взаємодію користувачів з освітнім середовищем.

Архітектура платформи побудована на основі Telegram-бота, який виступає центральним елементом взаємодії між користувачами. Уся логіка реалізована через бекенд на Node.js з розподілом ролей, що забезпечує чітку організацію доступу до функцій. Завдяки гнучкому сценарному управлінню бот автоматизує процеси навчання, адміністрування, сповіщення та збору даних, забезпечуючи зручність, швидкість та контроль у рамках всієї системи.

Бот має зручний інтерфейс, що дозволяє кожному користувачу швидко виконувати свої завдання: адміністрації — керувати системою, викладачам — працювати з заняттями й завданнями, студентам — оплачувати, навчатися та отримувати підтримку. Уся взаємодія відбувається напряду через Telegram.

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Волківський Я.С., Чижмотря О.В. Забезпечення конфіденційності та захисту особистих даних в чат-ботах. Тези XV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», м. Житомир, 28-29 березня 2025 р. – Житомир: Житомирська політехніка, 2025. С. 23-24. – Режим доступу: <https://conf.ztu.edu.ua/wp-content/uploads/2025/04/povnyj-tekst.pdf>
2. Волківський Я.С., Чижмотря О.В. Використання Telegram-ботів для автоматизації освітніх процесів та покращення взаємодії студентів і викладачів. Тези XV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», м. Житомир, 28-29 березня 2025 р. – Житомир: Житомирська політехніка, 2025. С. 15-16. – Режим доступу: <https://conf.ztu.edu.ua/wp-content/uploads/2025/04/povnyj-tekst.pdf>
3. Monobank API Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://api.monobank.ua/>
4. Telegram Bot API [Електронний ресурс] – Режим доступу до ресурсу: <https://core.telegram.org/bots/api>
5. Monobank Open API Specification на GitHub [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/monobankapi/monobank-api-docs>
6. «Launching MVP and Scaling Startups» [Електронний ресурс] – Режим доступу до ресурсу: <https://www.atlassian.com/agile/product-management/mvp>
7. Купер Б. Гнучкий продукт: створення мінімально життєздатного продукту та швидкий збір відгуків клієнтів : навч. посіб. / пер. з англ. – Київ : Наш Формат, 2020. – 280 с.
8. Modrzyk N. Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API. Springer, 2019. 350 с.
9. Node.js Documentation / Node.js Foundation. — 2025. — Режим доступу: <https://nodejs.org/en/docs/>
10. Telegraf Framework Guide / Telegraf. — 2024. — Режим доступу: <https://telegraf.js.org/>

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

11. MongoDB Manual / MongoDB, Inc. — 2025. — Режим доступу:
<https://docs.mongodb.com/manual/>
12. Redis Documentation / Redis Labs. — 2025. — Режим доступу:
<https://redis.io/docs/>
13. Baron J., Tim B., Hisham W. AWS Certified Solutions Architect Official Study Guide: Associate Exam : навч. посіб. — Indianapolis : Wiley, 2018. — 800 с.
14. AWS Well-Architected Framework / Amazon Web Services. — 2025. — Режим доступу: <https://aws.amazon.com/architecture/well-architected/>
15. Heroku Dev Center / Heroku. — 2025. — Режим доступу:
<https://devcenter.heroku.com/>
16. Іванченко М. В., Петренко О. Л. Системне адміністрування : навч. посіб. — Київ : КНЕУ, 2020. — 198 с.
17. PM2 Documentation / Unitech. — 2025. — Режим доступу:
<https://pm2.keymetrics.io/docs/usage/quick-start/>
18. DigitalOcean. Recommended Server Requirements. — 2024. — Режим доступу:
<https://www.digitalocean.com/docs/general-resources/recommended-server-specs/>
19. Ubuntu Server Documentation / Canonical Ltd. — 2025. — Режим доступу:
<https://ubuntu.com/server/docs>
20. Шевченко В. А. Сучасні бази даних: NoSQL-рішення та практичне застосування : навч. посіб. — Київ : Ліра-К, 2021. — 224 с.
21. Introduction to NoSQL / IBM Developer. — 2023. — Режим доступу:
<https://developer.ibm.com/articles/what-is-nosql/>
22. Nielsen Norman Group. User Roles and Permissions — 2024. — Режим доступу:
<https://www.nngroup.com/articles/roles-and-permissions/>
23. Шевчук І. Д. Проєктування розмовних інтерфейсів: створення ботів із продуманим UX : навч. посіб. — Київ : Центр учбової літератури, 2021. — 248 с.
24. Telegram Bot API: Handling Updates and Context / Telegram. — 2024. — Режим доступу: <https://core.telegram.org/bots/api#update>

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижемотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

25. RFC 2818: HTTP Over TLS / IETF. — 2000 (оновлено 2024). — Режим доступу:
<https://datatracker.ietf.org/doc/html/rfc2818>

26. HTTPS / MDN Web Docs. — 2025. — Режим доступу:
<https://developer.mozilla.org/uk/docs/Glossary/HTTPS>

27. Transport Layer Protection Cheat Sheet / OWASP. — 2024. — Режим доступу:
https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

28. TTL (Time To Live) / MDN Web Docs. — 2025. — Режим доступу:
<https://developer.mozilla.org/uk/docs/Glossary/TTL>

29. React – Documentation / Meta Platforms, Inc. — 2025. — Режим доступу:
<https://react.dev/>

30. React Tutorial / W3Schools. — 2024. — Режим доступу:
<https://www.w3schools.com/react/>

ДОДАТКИ

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```
// Initialization File
```

```
import { Telegraf } from "telegraf";
import { createClient } from "redis";
import { MongoClient } from "mongodb";
import { versionBot } from "../../data/general/chats";
import dotenv from 'dotenv';
import { InputFile } from "telegraf/typings/core/types/typegram";

dotenv.config();

async function connectToClubDB() {
  try {
    const client = new MongoClient('mongodb://localhost:27017/?family=4');

    await client.connect();

    console.log("Done");
    return client;
  } catch (error) {
    console.error('\n\nFatal Error to connect to MongoDB:\n', error);
    process.exit(1);
  }
}

export default async function init() {

  console.log(`\n    DehtoBot ${versionBot}\n\n    Developed by Yaroslav Volkivskyi
(TheLaidSon)\n\n\n`);

  console.log("Creating redis client...");
  const redis = createClient();
  redis.on("error", (err) => console.log("Redis Client Error", err));
  console.log("Done");

  console.log("Connecting to redis server...");
  await redis.connect();
  console.log("Done");

  console.log("Connecting to mongodb...")
  const botdb = await connectToClubDB();

  console.log("Creating telegraf bots instanse...");
  const bot = new Telegraf(process.env.MAIN!),
    bot_notification = new Telegraf(process.env.NOTIF!);
  console.log("Done\n");

  bot.use(async (ctx, next) => {
    const originalSendMessage = ctx.telegram.sendMessage,
      originalSendPhoto = ctx.telegram.sendPhoto,
      originalSendDocument = ctx.telegram.sendDocument,
      originalReply = ctx.reply;

    ctx.telegram.sendMessage = async (chatId: string | number, text: string, extra?:
any) => {
      let finalExtra = { ...extra, parse_mode: 'HTML' };

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижемотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

    if (extra && !extra.reply_markup) {
      finalExtra.reply_markup = { remove_keyboard: true };
    }
    return originalSendMessage.call(ctx.telegram, chatId, text, finalExtra);
  };

  ctx.telegram.sendPhoto = async (chatId: string | number, photo: string | InputFile,
  extra?: any) => {
    let finalExtra = { ...extra, parse_mode: 'HTML' };
    if (extra && !extra.reply_markup) {
      finalExtra.reply_markup = { remove_keyboard: true };
    }
    return originalSendPhoto.call(ctx.telegram, chatId, photo, finalExtra);
  };

  ctx.telegram.sendDocument = async (chatId: string | number, document: string |
  InputFile, extra?: any) => {
    let finalExtra = { ...extra, parse_mode: 'HTML' };
    if (extra && !extra.reply_markup) {
      finalExtra.reply_markup = { remove_keyboard: true };
    }
    return originalSendDocument.call(ctx.telegram, chatId, document, finalExtra);
  };

  ctx.reply = async (text: string, extra?: any) => {
    let finalExtra = { ...extra, parse_mode: 'HTML' };
    if (extra && !extra.reply_markup) {
      finalExtra.reply_markup = { remove_keyboard: true };
    }
    return originalReply.call(ctx, text, finalExtra);
  };

  await next();
})

console.warn('\n\n BOT READY TO WORK!\n\n')

// wrap redis with helper functions
const wRedis = ({
  getAll: (id: number) => async () => redis.hGetAll(`${id}`),
  get: (id: number) => async (property: string) => await redis.hGet(`${id}`,
  property),
  set: (id: number) => (property: string) => async (new_value: string) => await
  redis.hSet(`${id}`, property, new_value)
})

return [bot, bot_notification, process.env.MAIN, wRedis, botdb] as const;
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

import init from './init'
import { UserScriptState } from "../../data/general/UserScriptState";
import { Operation } from "./types"
import dbProcess from './dbProcess';
import { ActionType } from './types';
import Requests from './requests';

export default async function arch() {
  const [ bot, bot_notif, token_notif, db, botdb ] = await init(),
    DBProcess = await dbProcess(botdb);
  Requests(DBProcess);

  const onContactMessage = (startState: UserScriptState, action: ActionType<Operation>)
=>
  bot.on('message', async (ctx, next) => {
    const id = ctx.chat.id,
      user = (await db.getAll(id)),
      set = db.set(id),
      message = ctx.message;

    if (user.state === startState) {
      console.log(user['state']);
      if ('contact' in message) {
        action(ctx, user, set, { phone_number: [ message.contact.phone_number,
message.contact.first_name ], text: '', photo: [ '' ], file: [ '' ], stickers: '',
video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`TYPE: ONCONTACTMESSAGE, TEXT&PHOTO = UNDEFINED, NUMBER:
${message.contact.phone_number}, CODE: 0\n`);
      }
      else if ('text' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: message.text, photo: [ ''
], file: [ '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '',
audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&PHOTO = UNDEFINED, TEXT:
${message.text}, CODE: 1\n`);
      }
      else if ('photo' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [
message.photo[0].file_id, message?.caption ?? '' ], file: [ '' ], stickers: '', video: [
'' ], location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, PHOTO GET,
CODE: 2\n`);
      }
      else if ('document' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
message.document.file_id, message?.caption ?? '' ], stickers: '', video: [ '' ],
location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, FILE GET,
CODE: 3\n`);
      }
      else if ('sticker' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: message.sticker.file_id, video: [ '' ], location: [ -1 ], polls: '',
voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, STICKER GET,
CODE: 4\n`);
      }
    }
  })

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    else if ('video' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
        '' ], stickers: '', video: [ message.video.file_id, message?.caption ?? '' ], location:
        [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, VIDEO GET,
CODE: 5\n`);
    }
    else if ('location' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
        '' ], stickers: '', video: [ '' ], location: [ message.location.latitude,
        message.location.longitude ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, LOCATION GET,
CODE: 6\n`);
    }
    else if ('poll' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
        '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: message.poll.question,
        voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, POLL GET,
CODE: 7\n`);
    }
    else if ('voice' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
        '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice:
        message.voice.file_id, audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, VOICE GET,
CODE: 8\n`);
    }
    else if ('audio' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
        '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio:
        message.audio.file_id, video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, AUDIO GET,
CODE: 9\n`);
    }
    else if ('video_note' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
        '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '',
        video_circle: message.video_note.file_id });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, CIRCLE VIDEO
GET, CODE: 10\n`);
    }
    console.log(message);
}
else return next();
});

const onTextMessage = (startState: UserScriptState, action: ActionType<Operation>) =>
bot.on('message', async (ctx, next) => {
    const id = ctx.chat.id,
    user = (await db.getAll(id)),
    set = db.set(id),
    message = ctx.message;

    if (user.state === startState) {
        if ('text' in message) {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

```

        action(ctx, user, set, { phone_number: [ '' ], text: message.text, photo: [ ''
], file: [ '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '',
audio: '', video_circle: '' });
        console.log(`\nTYPE: ONTEXTMESSAGE, OTHERDATA = UNDEFINED, TEXT =
${message.text}, CODE: 1\nstate: ${startState}, message: ${message.text}`);
    }
    else if ('contact' in message){
        action(ctx, user, set, { phone_number: [ message.contact.phone_number,
message.contact.first_name ], text: '', photo: [ '' ], file: [ '' ], stickers: '',
video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`\nTYPE: ONTEXTMESSAGE, NUMBER&TEXT = UNDEFINED, NUMBER GET =
${message.contact.phone_number} by ${message.contact.first_name}, CODE: 0\nstate:
${startState}, message: ${message.contact.phone_number}`);
    }
    else if ('photo' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [
message.photo[0].file_id, message?.caption ?? '' ], file: [ '' ], stickers: '', video: [
'' ], location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`(!)TYPE: ONTEXTMESSAGE, NUMBER&TEXT = UNDEFINED, PHOTO GET, CODE:
2\n`);
    }
    else if ('document' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
message.document.file_id, message?.caption ?? '' ], stickers: '', video: [ '' ],
location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`(!)TYPE: ONTEXTMESSAGE, NUMBER&TEXT = UNDEFINED, FILE GET, CODE:
3\n`);
    }
    else if ('sticker' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: message.sticker.file_id, video: [ '' ], location: [ -1 ], polls: '',
voice: '', audio: '', video_circle: '' });
        console.log(`(!)TYPE: ONTEXTMESSAGE, NUMBER&TEXT = UNDEFINED, STICKER GET, CODE:
4\n`);
    }
    else if ('video' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ message.video.file_id, message?.caption ?? '' ], location:
[ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`(!)TYPE: ONTEXTMESSAGE, NUMBER&TEXT = UNDEFINED, VIDEO GET, CODE:
5\n`);
    }
    else if ('location' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ message.location.latitude,
message.location.longitude ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, LOCATION GET,
CODE: 6\n`);
    }
    else if ('poll' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: message.poll.question,
voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, POLL GET,
CODE: 7\n`);
    }
    else if ('voice' in message) {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice:
message.voice.file_id, audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, VOICE GET,
CODE: 8\n`);
    }
    else if ('audio' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio:
message.audio.file_id, video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, AUDIO GET,
CODE: 9\n`);
    }
    else if ('video_note' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '',
video_circle: message.video_note.file_id });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, CIRCLE VIDEO
GET, CODE: 10\n`);
    }
    }
    else return next();
    console.log(user['name'], '( @', user['username'], ')', '( id:', id, ')', user);
});

const onPhotoMessage = (startState: UserScriptState, action: ActionType<Operation>) =>
bot.on('message', async (ctx, next) => {
    const id = ctx.chat.id,
    user = (await db.getAll(id)),
    message = ctx.message,
    set = db.set(id),
    supportedFormats : string[] = ['.pdf', '.jpeg', '.jpg', '.png', '.heic'];

    if (user.state === startState) {
        console.log(user['state']);
        if ('text' in message) {
            action(ctx, user, set, { phone_number: [ '' ], text: message.text, photo: [ ''
], file: [ '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '',
audio: '', video_circle: '' });
            console.log(`(!)TYPE: ONPHOTOMESSAGE, NUMBER&PHOTO = UNDEFINED, TEXT:
${message.text}, CODE: 1\n`);
        }
        else if ('contact' in message){
            action(ctx, user, set, { phone_number: [ message.contact.phone_number,
message.contact.first_name ], text: '', photo: [ '' ], file: [ '' ], stickers: '',
video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
            console.log(`\nTYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, NUMBER GET =
${message.contact.phone_number} by ${message.contact.first_name}, CODE: 2\nstate:
${startState}, message: ${message.contact.phone_number}`);
        }
        else if ('photo' in message) {
            action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [
message.photo[message.photo.length - 1].file_id, message?.caption ?? '' ], file: [ '' ],
stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '',
video_circle: '' });
            console.log(`TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, PHOTO GET, CODE:
2\n`);
        }
    }
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

    else if ('document' in message) {
        const fileExtension =
message.document.file_name!.substr(message.document.file_name!.lastIndexOf('.')).toLower
Case() || '';

        if (supportedFormats.includes(fileExtension)){
            action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file:
[ message.document.file_id, message?.caption ?? '' ], stickers: '', video: [ '' ],
location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
            console.log(`TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, FILE GET
(${fileExtension}), CODE: 3\n`);
        }
        else{
            action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file:
[ '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '',
video_circle: '' });
            console.log(`(!)TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, UNSUPPORTED
FILE GET (${fileExtension}), CODE: 3\n`);
        }
    }
    else if ('sticker' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: message.sticker.file_id, video: [ '' ], location: [ -1 ], polls: '',
voice: '', audio: '', video_circle: '' });
        console.log(`(!)TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, STICKER GET,
CODE: 4\n`);
    }
    else if ('video' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ message.video.file_id, message?.caption ?? '' ], location:
[ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`(!)TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, VIDEO GET, CODE:
5\n`);
    }
    else if ('location' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ message.location.latitude,
message.location.longitude ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, LOCATION GET,
CODE: 6\n`);
    }
    else if ('poll' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: message.poll.question,
voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, POLL GET,
CODE: 7\n`);
    }
    else if ('voice' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice:
message.voice.file_id, audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, VOICE GET,
CODE: 8\n`);
    }
    else if ('audio' in message) {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		90

```

        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio:
message.audio.file_id, video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, AUDIO GET,
CODE: 9\n`);
    }
    else if ('video_note' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '',
video_circle: message.video_note.file_id });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, CIRCLE VIDEO
GET, CODE: 10\n`);
    }
    }
    else return next();
});

const onDocumentationMessage = (startState: UserScriptState, action:
ActionType<Operation>) =>
bot.on('message', async (ctx, next) => {
    const id = ctx.chat.id,
    user = (await db.getAll(id)),
    message = ctx.message,
    set = db.set(id),
    supportedFormats : string[] = ['.pdf', '.PDF'];

    if (user.state === startState) {
        console.log(user['state']);
        if ('text' in message) {
            action(ctx, user, set, { phone_number: [ '' ], text: message.text, photo: [ ''
], file: [ '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '',
audio: '', video_circle: '' });
            console.log(`(!)TYPE: ONPHOTOMESSAGE, NUMBER&PHOTO = UNDEFINED, TEXT:
${message.text}, CODE: 1\n`);
        }
        else if ('contact' in message){
            action(ctx, user, set, { phone_number: [ message.contact.phone_number,
message.contact.first_name ], text: '', photo: [ '' ], file: [ '' ], stickers: '',
video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
            console.log(`\nTYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, NUMBER GET =
${message.contact.phone_number} by ${message.contact.first_name}, CODE: 0\nstate:
${startState}, message: ${message.contact.phone_number}`);
        }
        else if ('photo' in message) {
            action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [
message.photo[0].file_id, message?.caption ?? '' ], file: [ '' ], stickers: '', video: [
'' ], location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
            console.log(`TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, PHOTO GET, CODE:
2\n`);
        }
        else if ('document' in message) {
            const fileExtension0 =
message.document.file_name!.substr(message.document.file_name!.lastIndexOf('.')).toLower
Case() || '';
            const fileExtension1 =
message.document.file_name!.substr(message.document.file_name!.lastIndexOf('.')).toUpper
Case() || '';
            const fileExtension = fileExtension0 + fileExtension1;

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (supportedFormats.includes(fileExtension0) ||
supportedFormats.includes(fileExtension1)){
            action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file:
[ message.document.file_id, message?.caption ?? '' ], stickers: '', video: [ '' ],
location: [ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
            console.log(`TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, FILE GET
(${fileExtension}), CODE: 3\n`);
        }
        else{
            action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file:
[ '' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '',
video_circle: '' });
            console.log(`(!)TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, UNSUPPORTED
FILE GET (${fileExtension}), CODE: 3_1\n`);
        }
    }
    else if ('sticker' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: message.sticker.file_id, video: [ '' ], location: [ -1 ], polls: '',
voice: '', audio: '', video_circle: '' });
        console.log(`(!)TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, STICKER GET,
CODE: 4\n`);
    }
    else if ('video' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ message.video.file_id, message?.caption ?? '' ], location:
[ -1 ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`(!)TYPE: ONPHOTOMESSAGE, NUMBER&TEXT = UNDEFINED, VIDEO GET, CODE:
5\n`);
    }
    else if ('location' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ message.location.latitude,
message.location.longitude ], polls: '', voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, LOCATION GET,
CODE: 6\n`);
    }
    else if ('poll' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: message.poll.question,
voice: '', audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, POLL GET,
CODE: 7\n`);
    }
    else if ('voice' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice:
message.voice.file_id, audio: '', video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, VOICE GET,
CODE: 8\n`);
    }
    else if ('audio' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio:
message.audio.file_id, video_circle: '' });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, AUDIO GET,
CODE: 9\n`);
    }

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    else if ('video_note' in message) {
        action(ctx, user, set, { phone_number: [ '' ], text: '', photo: [ '' ], file: [
'' ], stickers: '', video: [ '' ], location: [ -1 ], polls: '', voice: '', audio: '',
video_circle: message.video_note.file_id });
        console.log(`\n(!)TYPE: ONCONTACTMESSAGE, NUMBER&TEXT = UNDEFINED, CIRCLE VIDEO
GET, CODE: 10\n`);
    }
    }
    else return next();
});

return [onTextMessage, onContactMessage, onPhotoMessage, onDocumentationMessage, bot,
bot_notif, token_notif, db, DBProcess] as const;
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

import Express from "express";
import jwt from "jsonwebtoken";
import bcrypt from "bcryptjs";
import cors from "cors";
import dotenv from "dotenv";
import { ObjectId } from "mongodb";

dotenv.config();

const app = Express();
app.use(Express.json());
app.use(cors());
app.listen(process.env.API_PORT);

export default function Requests(dbProcess: any) {
  app.post("/web/login", async (req, res) => {
    const { telegramId, pass } = req.body;

    const user = await dbProcess.ShowOneUser(parseInt(telegramId));

    if (user){
      if (await bcrypt.compare(pass, user.web_p)){
        const token = jwt.sign(
          { id: user.id },
          process.env.JWT_SECRET!,
          { expiresIn: "24h" }
        );

        return res.send({
          status: "success",
          role: user.role,
          token,
        });
      }
      else return res.send({ message: "Wrong password" });
    }
    else return res.send({ message: "User not found" });
  });

  app.get("/web/verify", async (req, res) => {
    const token = req.headers.authorization?.split(" ")[1];
    if (token) {
      jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
        if (err) return res.send({ status: "failed", message: "Invalid token"
});
        if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });

        const user = await dbProcess.ShowOneUser(decoded.id);
        return res.send({ status: "success", role: user.role });
      });
    }
  });

  app.get("/web/getme", async (req, res) => {
    const token = req.headers.authorization?.split(" ")[1];
    if (token) {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

```

        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
        });
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            const user = await dbProcess.ShowOneUser(decoded.id);
            return res.send({ status: "success", role: user.role, name: user.name});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

app.get("/web/getlessonsgraph", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1];
    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
        });
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            const lessons = await dbProcess.CountLessons();
            return res.send({ status: "success", trial: lessons.trial, classic:
lessons.classic, clubs: lessons.club});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

app.get("/web/getrolesgraph", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1];
    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
        });
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            const roles = await dbProcess.CountUserRoles();
            return res.send({ status: "success", admin: roles.admin, teacher:
roles.teacher, student: roles.student, guest: roles.guest, supporter: roles.supporter});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

app.get("/web/getdetaskgraph", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1];
    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
        });
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            const count = await dbProcess.CountDeTaskAndSupportChannel();
            return res.send({ status: "success", deTask: count.countDe, support:
count.countSupport});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижморя О.В.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

});

app.get("/web/getallstudents", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1];
    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
});
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            const users = await dbProcess.GetAllStundetsAndGuests();
            return res.send({ status: "success", users: users});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

app.get("/web/getuser", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
        id = req.query.id as string;
    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
});
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            if (!id) return res.send({ status: "failed", message: "Invalid user id"
});
            const user = await dbProcess.ShowOneUser(parseInt(id));
            return res.send({ status: "success", user: user});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

app.get("/web/getallteachers", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1];
    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
});
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            const users = await dbProcess.GetAllTeachers();
            return res.send({ status: "success", user: users});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

app.get("/web/getallusers", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1];
    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
});
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        const users = await dbProcess.ShowAllUsers();
        return res.send({ status: "success", user: users});
    });
}
else return res.send({ status: "failed", message: "Invalid token" });
});

app.get("/web/getuserbyfind", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
        find = req.query.find as string;

    console.log(find);
    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
});
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            if (!find) return res.send({ status: "failed", message: "Invalid user
data to find" });
            const user = await dbProcess.FindUser(find);
            return res.send({ status: "success", user: user});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

app.post("/web/changerole", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
        id = req.body.id,
        role = req.body.role;

    console.log(token, id, role);

    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: err });
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            if (!id) return res.send({ status: "failed", message: "Invalid user id"
});
            if (!role) return res.send({ status: "failed", message: "Invalid user
role" });
            const user = await dbProcess.ChangeUserRole(parseInt(id), role);
            return res.send({ status: "success", user: user});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

app.post("/web/changename", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
        name = req.body.name,
        oid = req.body.oid;

    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: err });

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
        if (!name) return res.send({ status: "failed", message: "Invalid user
name" });
        if (!oid) return res.send({ status: "failed", message: "Invalid user id"
});
        const _user = await dbProcess.ShowOneUser(parseInt(oid));
        const user = await dbProcess.ChangeUserName(new ObjectId(_user._id),
name);
        return res.send({ status: "success", user: user});
    });
}
else return res.send({ status: "failed", message: "Invalid token" });
});

app.post("/web/trialteacher", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
        idTeacher = req.body.idTeacher,
        idStudent = req.body.idStudent;

    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
});
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            if (!idTeacher) return res.send({ status: "failed", message: "Invalid
user id" });
            if (!idStudent) return res.send({ status: "failed", message: "Invalid
user id" });
            const transferToTrial = await
dbProcess.UsersOperationWithGuest(parseInt(idStudent), parseInt(idTeacher), "miro.app",
0, "trial_teacher");
            return res.send({ status: "success", user: transferToTrial});
        });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
});

app.post("/web/transfertoteacher", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
        idTeacher = req.body.idTeacher,
        idStudent = req.body.idStudent;

    if (token) {
        jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
            if (err) return res.send({ status: "failed", message: "Invalid token"
});
            if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
            if (!idTeacher) return res.send({ status: "failed", message: "Invalid
user id" });
            if (!idStudent) return res.send({ status: "failed", message: "Invalid
user id" });
            const transferToTeacher = await
dbProcess.UsersOperationWithGuest(idStudent, idTeacher, "miro.app", 0, "just_teacher");
            return res.send({ status: "success", user: transferToTeacher});
        });
    }
});

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижморя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		98

```

    }
    else return res.send({ status: "failed", message: "Invalid token" });
  });

  app.post("/web/changebalance", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
      id = req.body.id,
      balance = req.body.balance;

    if (token) {
      jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
        if (err) return res.send({ status: "failed", message: "Invalid token"
});
        if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
        if (!id) return res.send({ status: "failed", message: "Invalid user id"
});
        if (!balance) return res.send({ status: "failed", message: "Invalid user
balance" });
        const user = await dbProcess.IndividualChangeUserData(parseInt(id),
'individual_count', parseInt(balance));
        return res.send({ status: "success", user: user});
      });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
  });

  app.post("/web/transfertooanotherteacher", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
      teacherName = req.body.teacherName,
      idStudent = req.body.idStudent;

    if (token) {
      jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
        if (err) return res.send({ status: "failed", message: "Invalid token"
});
        if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
        if (!teacherName) return res.send({ status: "failed", message: "Invalid
user id" });
        if (!idStudent) return res.send({ status: "failed", message: "Invalid
user id" });
        const user = await
dbProcess.IndividualChangeUserData(parseInt(idStudent), 'translate_to_another_teacher',
teacherName);
        return res.send({ status: "success", user: user});
      });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
  });

  app.post("/web/deletestudent", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
      id = req.body.id;

    if (token) {
      jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (err) return res.send({ status: "failed", message: "Invalid token"
    });
    if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
    if (!id) return res.send({ status: "failed", message: "Invalid user id"
    });
    const user = await dbProcess.IndividualChangeUserData(parseInt(id),
'delete_student', '');
    return res.send({ status: "success", user: user});
    });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
    });

    app.post("/web/teachersstudents", async(req, res) => {
        const token = req.headers.authorization?.split(" ")[1],
            id = req.body.id;

        if (token) {
            jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
                if (err) return res.send({ status: "failed", message: "Invalid token"
    });
                if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
                if (!id) return res.send({ status: "failed", message: "Invalid user id"
    });
                const teachers_students = await
dbProcess.GetTeacherStudents(parseInt(id));
                const users = await Promise.all(teachers_students.map(async (item: any)
=> {
                    return await dbProcess.ShowOneUser(item);
                }));
                return res.send({ status: "success", users: users});
            });
        }
        else return res.send({ status: "failed", message: "Invalid token" });
    });

    app.post("/web/teacherstrialstudents", async(req, res) => {
        const token = req.headers.authorization?.split(" ")[1],
            id = req.body.id;

        if (token) {
            jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
                if (err) return res.send({ status: "failed", message: "Invalid token"
    });
                if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
                if (!id) return res.send({ status: "failed", message: "Invalid user id"
    });
                const teachers_students = await
dbProcess.GetTeachersTrialsStudents(parseInt(id));
                const users = await Promise.all(teachers_students.map(async (item: any)
=> {
                    return await dbProcess.ShowOneUser(item);
                }));
                return res.send({ status: "success", users: users});
            });
        }
    });

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		100

```

    }
    else return res.send({ status: "failed", message: "Invalid token" });
  });

  app.post("/web/deleteteacher", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
      id = req.body.id;

    if (token) {
      jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
        if (err) return res.send({ status: "failed", message: "Invalid token"
});
        if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
        if (!id) return res.send({ status: "failed", message: "Invalid user id"
});
        const user = await dbProcess.DeleteTeacherFromPost(parseInt(id));
        return res.send({ status: "success", user: user});
      });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
  });

  app.post("/web/editlink", async(req, res) => {
    const token = req.headers.authorization?.split(" ")[1],
      id = req.body.id,
      link = req.body.link;

    if (token) {
      jwt.verify(token, process.env.JWT_SECRET!, async (err, decoded: any) => {
        if (err) return res.send({ status: "failed", message: "Invalid token"
});
        if (!decoded) return res.send({ status: "failed", message: "Invalid
token" });
        if (!id) return res.send({ status: "failed", message: "Invalid user id"
});
        if (!link) return res.send({ status: "failed", message: "Invalid link"
});
        const user = await dbProcess.IndividualChangeUserData(parseInt(id),
'miro_link', link);
        return res.send({ status: "success", user: user});
      });
    }
    else return res.send({ status: "failed", message: "Invalid token" });
  });
}

```

```

import { ObjectId } from "mongodb";
import { Context, Telegraf } from "telegraf";
import { Update } from "telegraf/typings/core/types/typegram";
import { UserScriptState } from "../../data/general/UserScriptState";

export type ActionType<T> = (ctx: Context<Update>, user: {[x: string]: string}, set:
(key: string) => (value: string) => Promise<number>, additionalData: T) => void;
export type Message = (startState: UserScriptState, action: ActionType<Operation>) =>
Telegraf<Context<Update>>

export type Operation = {
  phone_number: string[];
  text: string,
  photo: string[],
  file: string[],
  stickers: string,
  video: string[],
  location: number[],
  polls: string,
  voice: string,
  audio: string,
  video_circle: string
}

export type MongoDBReturntype = {
  _id: ObjectId;
  title: string;
  teacher: string;
  date: string;
  time: string;
  count: number;
  link: string;
}

export type IndividualArray = {
  [key: string]: any[]
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижемотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		102

```

import axios from "axios";
import dotenv from "dotenv";

dotenv.config();

export default class Monobank{
  private MONOBANK_API_TOKEN = process.env.MONOBANK_API_TOKEN;
  private MONOBANK_API_URL = process.env.MONOBANK_API_URL;

  async createMonobankInvoice(description: string, price: number, reference: {
    typeOfLessons: string, countOfLessons: number, idUser: number }) {
    const invoiceData = {
      amount: price * 100,
      ccy: 980,
      merchantPaymInfo: {
        reference:
`${reference.typeOfLessons}|${reference.countOfLessons}|${reference.idUser}`,
        destination: description,
      },
      validity: 3600
    };

    try {
      const response = await axios.post(`${this.MONOBANK_API_URL}/create`,
invoiceData, {
        headers: {
          'X-Token': this.MONOBANK_API_TOKEN,
          'Content-Type': 'application/json',
        },
      });

      return {
        status: response.data.status,
        invoiceId: response.data.invoiceId,
        pageUrl: response.data.pageUrl,
      };
    } catch (error: any) {
      console.error('Error creating invoice:', error.response?.data ||
error.message);
      throw error;
    }
  }

  async checkInvoiceStatus(invoiceId: string) {
    try {
      const response = await
axios.get(`${this.MONOBANK_API_URL}/status?invoiceId=${invoiceId}`, {
        headers: {
          'X-Token': this.MONOBANK_API_TOKEN,
        },
      });

      return response.data;
    } catch (error: any) {
      if (error.response.status === 404) {
        console.log('Invoice not found');
      }
    }
  }
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		103

```
        return null;
    }

    console.error('Error checking invoice status:', error.response?.data ||
error.message);

    throw error;
}
}
```

```
// Main File
import * as schedule from 'node-schedule';
import arch from './base/main/architecture';
import Start from './data/steps/Start';
import BotCommands from './data/steps/Commands';
import MainMenu from './data/steps/MainMenu';
import IndividualLessons from './data/steps/IndividualLessons';
import EndRootManager from './data/steps/EndRootManager';
import Actions from './data/steps/Actions';
import TeacherOnHour from './data/steps/TeacherOnHour';
import SpeakingClub from './data/steps/SpeakingClub';
import AdminMenu from './data/steps/AdminMenu';
import SummerCamp from './data/steps/SummerCamp';
import UserHandler from './data/steps/UserHandler';
import StudentHandler from './data/steps/StudentHandler';
import TeacherHandler from './data/steps/TeacherHandler';
import DeTask from './data/steps/deTask';
import CareService from './data/steps/CareService';
import AdminNotifications from './data/steps/AdminNotifications';
import Web from './data/steps/Web';

process.env.TZ = 'Europe/Kiev';

async function main() {
  const [ onTextMessage, onContactMessage, onPhotoMessage, onDocumentationMessage, bot,
    notifbot, notiftoken, db, dbProcess ] = await arch();

  await Start(onTextMessage, onContactMessage, bot, notifbot, db, dbProcess);

  await BotCommands(bot, notifbot, db, dbProcess);

  await Actions(bot, dbProcess, db, notifbot);

  await MainMenu(onTextMessage, dbProcess, db);

  await AdminMenu(onTextMessage, dbProcess, db);

  await AdminNotifications(onTextMessage, dbProcess);

  await EndRootManager(onTextMessage, dbProcess);

  await IndividualLessons(onTextMessage, onPhotoMessage, dbProcess, db, notifbot,
    notiftoken!);

  await SpeakingClub(onTextMessage, onPhotoMessage, onDocumentationMessage, dbProcess,
    db, bot, notifbot);

  await DeTask(onTextMessage, onPhotoMessage, dbProcess, db, notifbot, notiftoken!);

  await TeacherOnHour(onTextMessage, dbProcess, notifbot, notiftoken!);

  await SummerCamp(onTextMessage, dbProcess, notifbot);

  await UserHandler(onTextMessage, dbProcess, db);

  await StudentHandler(onTextMessage, dbProcess);
```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижморя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		105

```

await TeacherHandler(onTextMessage, dbProcess);

await CareService(onTextMessage, dbProcess, db);

await Web(onTextMessage, dbProcess, db)

schedule.scheduleJob('* / 5 * * * *', async () => {
    console.log('Notification job');
    await dbProcess.DeleteExpiredClubs();
    await dbProcess.DeleteExpiredIndividualLessons();
    await dbProcess.NotifyUserAboutLesson(bot.telegram);
    await dbProcess.NotifyUserAboutClubLesson(bot.telegram);
    await dbProcess.DeleteTeNotificationEntryData();
    await dbProcess.DeleteExpiredIndividualLessons();
});

await Actions(bot, dbProcess, db, notifbot);

bot.launch();
notifbot.launch();
}

main();

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		106

```

import { JSX, useEffect, useState } from 'react';
import { Navigate } from 'react-router-dom';
import CheckTokenAndGetRole from '../helpers/CheckToken';

export default function ProtectedRoute({ children }: { children: JSX.Element }) {
  const [loading, setLoading] = useState(true);
  const [isAuthenticated, setIsAuthenticated] = useState(false);

  useEffect(() => {
    const checkToken = async () => {
      const token = localStorage.getItem('token');
      if (!token) {
        setLoading(false);
        setIsAuthenticated(false);
        return;
      }

      if (await CheckTokenAndGetRole()) {
        setIsAuthenticated(true);
        setLoading(false);
      }
      else {
        setLoading(false);
        setIsAuthenticated(false);
      }
    };

    checkToken();
  }, []);

  if (loading) return <div>Завантаження...</div>;

  return isAuthenticated ? children : <Navigate to="/login" />;
}

```

```

import { useEffect, useState } from 'react';
import { useNavigate } from 'react-router-dom';
import '../assets/login.css';
import axios from 'axios';
import CheckTokenAndGetRole from '../helpers/CheckToken';
import EventStatus, { callFailedStatus } from '../admin/micro/EventStatus';

export default function Login() {
  const [tId, setTId] = useState('');
  const [title, setTitle] = useState('');
  const [description, setDescription] = useState('');
  const [password, setPassword] = useState('');
  const navigate = useNavigate();

  const errorToLogin = () => {
    setTitle('Помилка!');
    setDescription('Введіть правильні дані');
    callFailedStatus();
  }

  useEffect(() => {
    const check = async () => {
      await CheckTokenAndGetRole() && navigate('/dashboard');
    };

    check();
  }, []);

  const handleLogin = async (e: React.FormEvent) => {
    e.preventDefault();
    try {
      const response = await axios.post('http://127.0.0.1:4000/web/login', {
        telegramId: tId,
        pass: password,
      });

      if (response.data.token) {
        console.log('Login successful');
        localStorage.setItem('token', response.data.token);
        navigate('/dashboard');
      } else errorToLogin();
    } catch (error) {
      console.error('Error logging in: ', error);
    }
  };

  return (
    <div className="login-space">
      <EventStatus title={title} description={description}/>
      <div className="login-container">
        <h2>Вхід в систему</h2>
        <form onSubmit={handleLogin}>
          <input
            id='tId'
            name='tId'
            type="text"

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		108

```

        placeholder="Telegram Id"
        onChange={(e) => setTId(e.target.value)}
        required
    />
    <input
        id='password'
        name='password'
        type="password"
        placeholder="Пароль"
        onChange={(e) => setPassword(e.target.value)}
        required
    />
    <button type="submit">Увійти</button>
</form>
</div>
</div>
);
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		109

```

import { PieChart, Pie, Cell, Tooltip, Legend } from 'recharts';

const COLORS = [
  '#0088FE', // blue
  '#00C49F', // turquoise
  '#FFBB28', // yellow
  '#FF8042', // orange
  '#A28EFF', // purple
  '#FF5E7E', // pink
  '#4CAF50', // green
  '#9E9E9E', // gray
  '#E91E63', // magenta
  '#03A9F4'  // light blue
];

export default function Graph({ data, title, ifEmptyMessage, width, height, fontSize }:
  { data: { name: string, value: number }[], title: string, ifEmptyMessage: string,
  width: number, height: number, fontSize: number }) {
  return (
    <div className="ggraph-wrapper">
      <div style={{ fontSize: `${fontSize}px`, fontWeight: 'bold' }}>{title}</div>
      {data.every(item => item.value === 0) ? (
        <p style={{ textAlign: 'center', marginTop: '100px', color: '#777',
fontSize: `${fontSize / 1.5}px` }}>{ifEmptyMessage}</p>
      ) : (
        <PieChart width={width} height={height}>
          <Pie
            data={data}
            cx="50%"
            cy="50%"
            innerRadius={100}
            outerRadius={120}
            fill="#8884d8"
            paddingAngle={5}
            dataKey="value"
          >
            {data.map((entry, index) => (
              <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
            ))}
          </Pie>
          <Tooltip />
          <Legend />
        </PieChart>
      )}
    </div>
  )
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		110

```

const CloseAllAdd = () => {
  const addMenu = document.querySelectorAll('.additional-menu') as
  NodeListOf<HTMLElement>,
  menuItems = document.querySelectorAll('.menu-item') as NodeListOf<HTMLElement>;

  addMenu.forEach(item => item.classList.remove('active'));
  menuItems.forEach(item => {item.classList.remove('active');
item.classList.add('hoverd')}});
}

const openFindUserIndividual = () => {
  console.log('openFindUserIndividual');
  CloseAllAdd();
  document.querySelector('.find-user-individual')?.classList.add('active');
}

const openAllUsersIndividual = () => {
  console.log('openAllUsersIndividual');
  CloseAllAdd();
  document.querySelector('.all-user-individual')?.classList.add('active');
}

const openFindUserIndividualLessons = () => {
  console.log('openFindUserIndividualLessons');
  CloseAllAdd();
  document.querySelector('.find-user-individual-lessons')?.classList.add('active');
}

const openAllOurTeachersIndividualLessons = () => {
  console.log('openAllOurTeachersLessons');
  CloseAllAdd();
  document.querySelector('.all-our-teachers-individual-
lessons')?.classList.add('active');
}

const openAllOurStudentsIndividualLessons = () => {
  console.log('openAllOurStudentsLessons');
  CloseAllAdd();
  document.querySelector('.all-our-students-individual-
lessons')?.classList.add('active');
}

const openCreateClub = () => {
  console.log("openCreateClub");
  CloseAllAdd();
  document.querySelector('.clubs-create')?.classList.add('active');
}

export default function AddMenu() {
  return (
    <>
    <div className="additional-menu users">
      <div className="add-menu-item">
        <div className="title">
          <div className="title-text green">Знайти користувача за
данними</div>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		111

```

</div>

<div className="content">
  <div className="content-item" onClick={openFindUserIndividual}>
    <div className="group-title">

      <div className="title">Пошук</div>
    </div>
  </div>
</div>

<div className="add-menu-item">
  <div className="title">
    <div className="title-text blue">Всі Користувачі</div>
  </div>

  <div className="content">
    <div className="content-item" onClick={openAllUsersIndividual}>
      <div className="group-title">
        <div className="title">Показати</div>
      </div>
    </div>
  </div>
</div>

</div>

<div className="additional-menu clubs">
  <div className="add-menu-item">
    <div className="title">
      <div className="title-text blue">Клуби</div>
    </div>

    <div className="content">
      <div className="content-item" onClick={openCreateClub}>
        <div className="group-title">
          <div className="title">Додати</div>
        </div>
      </div>

      <div className="content-item">
        <div className="group-title">
          <div className="title">Показати всі</div>
        </div>
      </div>
    </div>
  </div>
</div>

<div className="add-menu-item">
  <div className="title">
    <div className="title-text red">Особові справи</div>
  </div>

  <div className="content">
    <div className="content-item">
      <div className="group-title">

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				112
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<div className="title">Знайти користувача за
данними</div>
</div>
</div>
<div className="content-item">
<div className="group-title">
<div className="title">Показати усіх користувачів</div>
</div>
</div>
</div>
</div>
<div className="additional-menu individual">
<div className="add-menu-item">
<div className="title">
<div className="title-text green">Знайти студента</div>
</div>
<div className="content">
<div className="content-item"
onClick={openFindUserIndividualLessons}>
<div className="group-title">
<div className="title">Пошук</div>
</div>
</div>
</div>
</div>
<div className="add-menu-item">
<div className="title">
<div className="title-text blue">Наші викладачі</div>
</div>
<div className="content">
<div className="content-item"
onClick={openAllOurTeachersIndividualLessons}>
<div className="group-title">
<div className="title">Показати</div>
</div>
</div>
</div>
</div>
<div className="add-menu-item">
<div className="title">
<div className="title-text red">Усі наші студенти</div>
</div>
<div className="content">
<div className="content-item"
onClick={openAllOurStudentsIndividualLessons}>
<div className="group-title">
<div className="title">Показати</div>

```

```

    </div>
  </div>
</div>
</div>
  </div>
</>
)
}
```

```

import { useEffect, useState } from 'react';
import '../assets/adminDashboard.css';
import AddMenu from './micro/AddMenu';
import DashboardHeader from './micro/DashboardHeader';
import { useNavigate } from 'react-router-dom';
import GetNameAndRoleHandler from '../helpers/GetNameAndRole';
import GetLessonsCountHandler from '../helpers/GetLessonsCount';
import Graph from './micro/Graph';
import GetRolesCountHandler from '../helpers/GetRolesCount';
import GetDeTaskAndSupportCountHandler from '../helpers/GetDeTaskAndSupportCount';
import Modal from './micro/Modal';

export default function AdminDashboard() {
  const [name, setName] = useState('User');
  const [role, setRole] = useState('user');
  const [lessonsCount, setLessonsCount] = useState([
    { name: 'Класичні Індивідуальні', value: 0 },
    { name: 'Пробні Індивідуальні', value: 0 },
    { name: 'Клубні', value: 0 }
  ]);
  const [users, setUsers] = useState([
    { name: 'Гості', value: 0 },
    { name: 'Студенти', value: 0 },
    { name: 'Викладачі', value: 0 },
    { name: 'Адміністратори', value: 0 },
    { name: 'Сапортери', value: 0 },
  ]);
  const [deTaskCountAndSupportChannelsCount, setTaskCountAndSupportChannelsCount] =
  useState([
    { name: 'деТаски', value: 0 },
    { name: 'Сапорт канали', value: 0 },
  ])
  const navigate = useNavigate();

  useEffect(() => {
    const getNameAndRole = async () => {
      try{
        const response = await GetNameAndRoleHandler();
        if (response) {
          setName(response.name);
          setRole(response.role);
        } else navigate('/login');
      } catch (error) {
        console.error('Error verifying token', error);
        navigate('/login');
      }
    }

    const getLessonsCount = async () => {
      try{
        const response = await GetLessonsCountHandler();
        if (response) {
          setLessonsCount(response);
        } else navigate('/login');
      } catch (error) {
        console.error('Error verifying token', error);
      }
    }
  });

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				115
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        navigate('/login');
    }
}

const getRolesCount = async () => {
    try{
        const response = await GetRolesCountHandler();
        if (response) {
            setUsers(response);
        } else navigate('/login');
    } catch (error) {
        console.error('Error verifying token', error);
        navigate('/login');
    }
}

const getDeTaskCount = async () => {
    try{
        const response = await GetDeTaskAndSupportCountHandler();
        if (response) {
            setTaskCountAndSupportChannelsCount(response);
        } else navigate('/login');
    } catch (error) {
        console.error('Error verifying token', error);
        navigate('/login');
    }
}

getDeTaskCount();
getRolesCount();
getLessonsCount();
getNameAndRole();
}, []);

const exitHandler = () => {
    localStorage.removeItem('token');
    navigate('/login');
}

return (
    <div className="admin-container">
        <Modal/>
        <DashboardHeader/>

        <div className="admin-content">
            <AddMenu/>

            <div className="user-info">
                <div className="avatar">
                    <div className="avatar-placeholder">{name[0]}</div>
                </div>
                <div className="user-data-and-logout">
                    <div className="user-data">
                        <div className="user-name">{name}</div>
                        <div className="user-role">{role}</div>
                    </div>
                    <div className="logout-button" onClick={exitHandler}>Вийти</div>
                </div>
            </div>
        </div>
    </div>

```

```

</div>

<div className="card">
  <Graph width={350} height={350} fontSize={24} data={lessonsCount}
title="Заплановані заняття" ifEmptyMessage="Немає запланованих занять"/>
  <Graph width={350} height={350} fontSize={24} data={users} title="Користувачі"
ifEmptyMessage="Користувачі відсутні"/>
  <Graph width={350} height={350} fontSize={24}
data={deTaskCountAndSupportChannelsCount} title="Активні деТаски та Сапорт канали"
ifEmptyMessage="Активні завдання відсутні"/>
</div>
</div>
</div>
);
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		117

```

import { useEffect, useState } from "react";

export default function DashboardHeader() {
  const [headerMenu, setHeaderMenu] = useState(false);
  const [clicked, setClicked] = useState(false);
  const [menuClicked, setMenuClicked] = useState('users');

  useEffect(() => {
    const add_menu = document.querySelector(`.additional-menu.${menuClicked}`),
      clickedElement = document.querySelector(`.item-${menuClicked}`),
      add_menu_container = document.querySelector('.additional-menu-container');

    add_menu && add_menu.classList.toggle('active');

    add_menu_container && add_menu_container.classList.toggle('active');

    const otherElements = ['users', 'individual', 'clubs'].filter(item => item !== menuClicked);

    otherElements.forEach(item => {
      const _menu = document.querySelector(`.additional-menu.${item}`),
        _clickedElement = document.querySelector(`.item-${item}`);

      if (_menu) _menu.classList.remove('active');
      if (_clickedElement) {
        _clickedElement.classList.add('hoverd');
        _clickedElement.classList.remove('active');
      }
    });

    if (clickedElement) {
      clickedElement.classList.toggle('active');
      clickedElement.classList.toggle('hoverd');
    }
  }, [menuClicked, clicked]);

  useEffect(() => {
    const headerMenu = document.querySelector('.menu');

    headerMenu && headerMenu.classList.toggle('active');
  }, [headerMenu]);

  const handleClick = (typeofMenu: string) => {
    setClicked(!clicked);
    setMenuClicked(typeofMenu);
  }

  return (
    <div className="admin-header">
      <div className="menu-and-logo">
        <div className="logo">
          dehto
        </div>

        <div className="menu">

```

```

        <div className="menu-item item-users hoverd" onClick={() =>
handleClick('users')}}>
            <div className="title">Користувачі</div>
        </div>
        <div className="menu-item item-individual hoverd" onClick={() =>
handleClick('individual')}}>
            <div className="title">Індивідуальні заняття</div>
        </div>

        <div className="menu-item item-clubs hoverd" onClick={() =>
handleClick('clubs')}}>
            <div className="title">Розмовні клуби</div>
        </div>
    </div>
</div>
</div>
)
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		119

```

import '../../../assets/event.css';
import failedImg from '../../../assets/sources/failed.png';
import successImg from '../../../assets/sources/success.png';

export const callFailedStatus = () => {
  const failedStatus = document.querySelector('.failed-event-status-container');
  if (failedStatus) failedStatus.classList.add('active');

  setTimeout(() => {
    if (failedStatus) failedStatus.classList.remove('active');
  }, 3000);
}

export const callSuccessStatus = () => {
  const successStatus = document.querySelector('.success-event-status-container');
  if (successStatus) successStatus.classList.add('active');

  setTimeout(() => {
    if (successStatus) successStatus.classList.remove('active');
  }, 3000);
}

export default function EventStatus({ title, description }: { title: string,
description: string }) {
  return (
    <>
    <div className="failed-event-status-container">
      <div className="title-with-status">
        <img src={failedImg} alt="" className="status-img" />
        <div className="title">{title}</div>
      </div>
      <div className="description">{description}</div>
    </div>

    <div className="success-event-status-container">
      <div className="title-with-status">
        <img src={successImg} alt="" className="status-img" />
        <div className="title">{title}</div>
      </div>
      <div className="description">{description}</div>
    </div>
    </>
  )
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		120

```

import { useEffect, useState } from 'react';
import '../assets/modal.css';
import close from '../assets/sources/close.png';
import back from '../assets/sources/back.png';
import GetAllStudentsAndGuests from '../helpers/GetAllStudentsAndGuests';
import { useNavigate } from 'react-router-dom';
import GetAllTeachers from '../helpers/GetAllTeachers';
import GetAllUsers from '../helpers/GetAllUsers';
import FindUser from '../helpers/FindUser';
import ChangeUserRole from '../helpers/ChangeUserRole';
import ChangeUserName from '../helpers/ChangeUserName';
import TransferToTrial from '../helpers/TransferToTrial';
import TransferToTeacher from '../helpers/TransferToTeacher';
import GetUser from '../helpers/GetUser';
import DeleteTeacher from '../helpers/DeleteTeacher';
import TeachersStudents from '../helpers/TeachersStudents';
import ChangeBalance from '../helpers/ChangeBalance';
import EditMiroLink from '../helpers/EditLink';
import TransferToAnotherTeacher from '../helpers/TransferToAnotherTeacher';
import DeleteStudent from '../helpers/DeleteStudent';
import TeachersTrialStudents from '../helpers/TeachersTrialStudents';
import EventStatus, { callFailedStatus, callSuccessStatus } from './EventStatus';
import { Students, Teachers, User } from './interfaceExamples';
import { UserInterface } from './interface';

export default function Modal() {
  //
  // USERS //
  //
  const rolesTranscript = {
    'admin': 'Адміністратор',
    'teacher': 'Викладач',
    'student': 'Студент',
    'guest': 'Гість'
  } as { [key: string]: string };

  const [eventStatusTitle, setEventStatusTitle] = useState('');
  const [eventStatusDescription, setEventStatusDescription] = useState('');

  const showFailedStatus = (title: string, description: string) => {
    setEventStatusTitle(title);
    setEventStatusDescription(description);
    callFailedStatus();
  }

  const showSuccessStatus = (title: string, description: string) => {
    setEventStatusTitle(title);
    setEventStatusDescription(description);
    callSuccessStatus();
  }

  const navigate = useNavigate();

  const [allUsers, setAllUsers] = useState<UserInterface[]>([User]);

  const [teachers, setTeachers] = useState<UserInterface[]>(Teachers);

  const [allStudents, setAllStudents] = useState<UserInterface[]>(Students);

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		121

```

const [teacherForTrial, setTeacherForTrial] = useState<string>('');
const [teacherForClassic, setTeacherForClassic] = useState<string>('');

const [userDataToFind, setUserDataToFind] = useState<UserInterface>(User);
const [choosedUser, setChoosedUser] = useState<UserInterface>(User);

const [operationOpened, setOperationOpened] = useState<boolean>(false);
const [operationOpenedAll, setOperationOpenedAll] = useState<boolean>(false);

const [modalFunctionOpened, setModalFunctionOpened] = useState<string>('');
const [modalFunctionOpenedAll, setModalFunctionOpenedAll] = useState<string>('');

const [userRole, setUserRole] = useState<string>('');
const [filterUsers, setFilterUsers] = useState<string>('');

useEffect(() => {
  const getAllStudents = async () => {
    try{
      const response = await GetAllStudentsAndGuests();
      if (response) {
        setAllStudents(response);
      }
      else showFailedStatus('Помилка', 'Не вдалося завантажити студентів');
    } catch (error) {
      console.error('Error verifying token', error);
      navigate('/login');
    }
  }

  const getAllTeachers = async () => {
    try{
      const response = await GetAllTeachers();
      if (response) {
        setTeachers(response);
      }
      else showFailedStatus('Помилка', 'Не вдалося завантажити викладачів');
    } catch (error) {
      console.error('Error verifying token', error);
      navigate('/login');
    }
  }

  const getAllUsers = async () => {
    try{
      const response = await GetAllUsers();
      if (response) {
        setAllUsers(response);
      }
      else showFailedStatus('Помилка', 'Не вдалося завантажити користувачів');
    } catch (error) {
      console.error('Error verifying token', error);
      navigate('/login');
    }
  }

  getAllUsers();
  getAllTeachers();
  getAllStudents();
}, [])

useEffect(() => {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		122

```

    if (filterUsers !== '') {
      const filteredUsers = allUsers.filter((user) => {
        const searchLower = String(filterUsers).toLowerCase();

        return (
          (user.name?.toLowerCase().includes(searchLower) ?? false) ||
          (String(user.id || '').includes(searchLower)) ||
          (user.individual_count?.toString().includes(searchLower) ?? false) ||
          (user.count?.toString().includes(searchLower) ?? false)
        );
      });

      setAllUsers(filteredUsers);
    }
    else {
      const getAllUsers = async () => {
        try{
          const response = await GetAllUsers();
          if (response) {
            setAllUsers(response);
          }
          else showFailedStatus('Помилка', 'Не вдалося завантажити користувачів');
        } catch (error) {
          console.error('Error verifying token', error);
          navigate('/login');
        }
      }

      getAllUsers();
    }

    }, [filterUsers]);

    useEffect(() => {
      const allFunctions = ['change-role-function', 'change-name-function', 'add-to-trial-
function', 'add-to-teacher'];

      if (modalFunctionOpened !== '') {
        console.log(modalFunctionOpened);
        const targetElement = document.querySelector(`.${modalFunctionOpened}`) as
HTMLInputElement;
        const userId = userDataToFind.id;

        allFunctions.filter(item => item !== modalFunctionOpened).forEach(item =>
document.querySelector(`.${item}`)?.classList.remove('active'));

        if (targetElement.classList.contains('active')){
          targetElement.classList.remove('active');
          (document.querySelector(`operation-menu-
individual`))?.querySelector('.functions')?.classList.remove('active');

          switch (modalFunctionOpened) {
            case 'change-role-function':
              document.querySelectorAll('.button-role').forEach(item => {
                item.classList.remove('active');
                item.classList.add('hoverd');
              });

              if (userRole !== '') {
                const changeRole = async () => {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				123
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        try{
            alert(userRole);
            const response = await ChangeUserRole(userId.toString(),
userRole);

            if (response) {
                showSuccessStatus('Успішно', 'Роль успішно
змінена');

                setUserRole('');
            }
            else showFailedStatus('Помилка', 'Не вдалося змінити
роль');
        }
        catch(error){
            showFailedStatus('Помилка', 'Не вдалося змінити роль');
            console.log(error);
        }
    }

    changeRole();
}

break;

case 'change-name-function':
    const input = targetElement.querySelector('.form__field') as
HTMLInputElement;

    const changeName = async () => {
        try{
            const response = await ChangeUserName(userId.toString(),
input.value);

            if (response) {
                showSuccessStatus('Успішно', 'Ім'я успішно змінено');
                input.value = '';
                console.log('name changed');
            }
            else showFailedStatus('Помилка', 'Не вдалося змінити
ім'я');
        }
        catch(error){
            showFailedStatus('Помилка', 'Не вдалося змінити ім'я');
            console.log(error);
        }
    }

    if (input && input.value !== '') {
        changeName();
    }

    break;

case 'add-to-trial-function':
    const addToTrial = async () => {
        try{
            const response = await TransferToTrial(userId.toString(),
teacherForTrial);

            if (response) {
                document.querySelectorAll('.add-to-trial-function >
.button-teacher').forEach(item => {
                    item.classList.remove('active');
                    item.classList.add('hoverd');
                });
            }
        }
    }

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		124

```

        showSuccessStatus('Успішно', 'Користувача успішно додано
до пробних занять викладачу');
        setTeacherForTrial('');
        console.log('added to trial');
    }
    else showFailedStatus('Помилка', 'Не вдалося додати
користувача до пробних занять');
    }
    catch(error){
        showFailedStatus('Помилка', 'Не вдалося додати користувача
до пробних занять');
        console.log(error);
    }
}

if (teacherForTrial !== '') {
    addToTrial();
}

break;

case 'add-to-teacher':
    const addToTeacher = async () => {
        try{
            const response = await TransferToTeacher(userId.toString(),
teacherForClassic);
            if (response) {
                document.querySelectorAll('.add-to-teacher > .button-
teacher').forEach(item => {
                    item.classList.remove('active');
                    item.classList.add('hoverd');
                });
                showSuccessStatus('Успішно', 'Користувача успішно
записано до викладача');
                setTeacherForClassic('');
                console.log('added to teacher');
            }
            else showFailedStatus('Помилка', 'Не вдалося записати
користувача до викладача');
        }
        catch(error){
            showFailedStatus('Помилка', 'Не вдалося записати користувача
до викладача');
            console.log(error);
        }
    }

    if (teacherForClassic !== '') {
        addToTeacher();
    }
    setTeacherForClassic('');
    break;

}
}
else{
    targetElement.classList.add('active');
    document.querySelector('.functions')?.classList.add('active');
}
}
else{

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		125

```

        console.log(modalFunctionOpened);
        allFunctions.forEach(item =>
document.querySelector(`.${item}`)?.classList.remove('active'));
        document.querySelector('.functions)?.classList.remove('active');
    }
    }, [operationOpened]);

    useEffect(() => {
        const allFunctions = ['all-change-role-function', 'all-change-name-function', 'all-
add-to-trial-function', 'all-add-to-teacher'];

        if (modalFunctionOpenedAll !== '') {
            const userId = choosedUser.id;
            console.log(modalFunctionOpenedAll);
            const targetElement = document.querySelector(`.${modalFunctionOpenedAll}`) as
HTMLElement;

            allFunctions.filter(item => item !== modalFunctionOpenedAll).forEach(item =>
document.querySelector(`.${item}`)?.classList.remove('active'));

            if (targetElement.classList.contains('active')){
                targetElement.classList.remove('active');
                (document.querySelector('.operation-menu-individual-after-
all'))?.querySelector('.functions)?.classList.remove('active');

                switch (modalFunctionOpenedAll) {
                    case 'all-change-role-function':
                        document.querySelectorAll('.button-role').forEach(item => {
                            item.classList.remove('active');
                            item.classList.add('hoverd');
                        });

                        if (userRole !== '') {
                            const changeRole = async () => {
                                try{
                                    const response = await ChangeUserRole(userId.toString(),
userRole);
                                    if (response) {
                                        showSuccessStatus('Успішно', 'Роль користувача
зміненна');
                                        console.log('role changed');
                                        setUserRole('');
                                    }
                                    else showFailedStatus('Помилка', 'Не вдалося змінити
роль користувача');
                                }
                                catch(error){
                                    showFailedStatus('Помилка', 'Не вдалося змінити роль
користувача');
                                    console.log(error);
                                }
                            }

                            changeRole();
                        }
                        break;

                    case 'all-change-name-function':
                        const input = targetElement.querySelector('.form__field') as
HTMLInputElement;
                        const changeName = async () => {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		126

```

        try{
            const response = await ChangeUserName(userId.toString(),
input.value);
            if (response) {
                showSuccessStatus('Успішно', 'Ім'я користувача
змінено');
                input.value = '';
                console.log('name changed');
            }
            else showFailedStatus('Помилка', 'Не вдалося змінити ім'я
користувача');
        }
        catch(error){
            showFailedStatus('Помилка', 'Не вдалося змінити ім'я
користувача');
            console.log(error);
        }
    }

    if (input && input.value !== '') {
        changeName();
    }

    break;

case 'all-add-to-trial-function':
    const addToTrial = async () => {
        try{
            const response = await TransferToTrial(userId.toString(),
teacherForTrial);
            if (response) {
                document.querySelectorAll('.add-to-trial-function >
.button-teacher').forEach(item => {
                    item.classList.remove('active');
                    item.classList.add('hoverd');
                });
                showSuccessStatus('Успішно', 'Користувача успішно додано
до пробних занять викладачу');
                setTeacherForTrial('');
                console.log('added to trial');
            }
            else showFailedStatus('Помилка', 'Не вдалося записати
користувача до пробних занять викладача');
        }
        catch(error){
            showFailedStatus('Помилка', 'Не вдалося записати користувача
до пробних занять викладача');
            console.log(error);
        }
    }

    if (teacherForTrial !== '') {
        addToTrial();
    }
    break;

case 'all-add-to-teacher':
    document.querySelectorAll('.add-to-teacher > .button-
teacher').forEach(item => {
        item.classList.remove('active');
        item.classList.add('hoverd');
    });

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				127
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    });
    const addToTeacher = async () => {
      try{
        const response = await TransferToTeacher(userId.toString(),
teacherForClassic);
        if (response) {
          document.querySelectorAll('.add-to-teacher > .button-
teacher').forEach(item => {
            item.classList.remove('active');
            item.classList.add('hoverd');
          });
          showSuccessStatus('Успішно', 'Користувача успішно
записано до викладача');
          setTeacherForClassic('');
          console.log('added to teacher');
        }
        else showFailedStatus('Помилка', 'Не вдалося записати
користувача до викладача');
      }
      catch(error){
        showFailedStatus('Помилка', 'Не вдалося записати користувача
до викладача');
        console.log(error);
      }
    }

    if (teacherForClassic !== '') {
      addToTeacher();
    }
    setTeacherForClassic('');
    break;
  }
}
else{
  targetElement.classList.add('active');
  (document.querySelector('.operation-menu-individual-after-
all'))?.querySelector('.functions')?.classList.add('active');
}
}
else{
  console.log(modalFunctionOpenedAll);
  allFunctions.forEach(item =>
document.querySelector(`.${item}`)?.classList.remove('active'));
  document.querySelector('.functions')?.classList.remove('active');
}
}, [operationOpenedAll]);

//navigation
const afterFindUserGoToOperationMenu = () => {
  const input = document.querySelector('.form__field') as HTMLInputElement;

  if (input.value){
    const getToFindUser = async () => {
      try{
        const response = await FindUser(input.value);

        if (!response) showFailedStatus('Помилка', 'Користувача не знайдено');
        else {
          setUserDataToFind(response);

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		128

```

        document.querySelector('.find-user-individual')?.classList.remove('active');
        document.querySelector('.operation-menu-individual')?.classList.add('active');
    }
}
catch(error){
    showFailedStatus('Помилка', 'Користувача не знайдено');
    console.log(error);
}
}

getToFindUser();
}
else showFailedStatus('Помилка', 'Введіть дані користувача');
}

const backToFindUserAfterOperation = () => {
    document.querySelector('.operation-menu-individual')?.classList.remove('active');
    document.querySelector('.find-user-individual')?.classList.add('active');
}

const backToAllUsersIndividual = () => {
    document.querySelector('.operation-menu-individual-after-all')?.classList.remove('active');
    document.querySelector('.all-user-individual')?.classList.add('active');
}

const closeModal = () => {
    document.querySelectorAll('.modal')?.forEach(item =>
item.classList.remove('active'));
}

const afterChoosingUserGoToOperationMenuIndividual = (e: React.MouseEvent, id: string)
=> {
    const getUserData = async () => {
        try{
            const response = await GetUser(id);
            if (!response) showFailedStatus('Помилка', 'Користувача не знайдено');
            else{
                setChoosedUser(response);
                document.querySelector('.operation-menu-individual-after-all')?.classList.add('active');
                document.querySelector('.all-user-individual')?.classList.remove('active');
            }
        }
        catch(error){
            showFailedStatus('Помилка', 'Користувача не знайдено');
            console.log(error);
        }
    }

    getUserData();
}

// end navigation

const openOrFinishOperation = (e: React.MouseEvent, whatNeedToOpen: string) => {
    e.preventDefault();
    const button = e.target as HTMLElement;

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		129

```

document.querySelectorAll('.functions-button-find-user')
    .forEach(item => item !== button && item.classList.remove('active'));

if (!button.classList.contains('active')) button.classList.add('active');
else button.classList.remove('active');

setOperationOpened(!operationOpened);
setModalFunctionOpened(whatNeedToOpen);
}

const openOrFinishOperationAll = (e: React.MouseEvent, whatNeedToOpen: string) => {
    e.preventDefault();
    const button = e.target as HTMLElement;

    (document.querySelector('.operation-menu-individual-after-
all'))?.querySelectorAll('.functions-button-find-user')
        .forEach(item => item !== button && item.classList.remove('active'));

    if (!button.classList.contains('active')) button.classList.add('active');
    else button.classList.remove('active');

    setOperationOpenedAll(!operationOpenedAll);
    setModalFunctionOpenedAll(whatNeedToOpen);
}

const chooseRole = (e: React.MouseEvent, role: string) => {
    const targetElement = e.target as HTMLElement;

    document.querySelectorAll('.button-role').forEach(item => {
        if (item !== targetElement) {
            item.classList.remove('active');
            item.classList.add('hoverd');
        }
    });

    if (targetElement.classList.contains('active')) {
        targetElement.classList.remove('active');
        targetElement.classList.add('hoverd');
        setUserRole('')
    }
    else{
        setUserRole(role)
        targetElement.classList.add('active');
        targetElement.classList.remove('hoverd');
    }
}

const addTeacherTrial = (e: React.MouseEvent, id: string) => {
    const targetElement = (e.target as HTMLElement).closest('.button-teacher');

    document.querySelectorAll('.add-to-trial-function.active > .button-
teacher').forEach(item => {
        if (item !== targetElement) {
            item.classList.remove('active');
            item.classList.add('hoverd');
        }
    });

    if (targetElement?.classList.contains('active')) {
        targetElement.classList.remove('active');
    }
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		130

```

        targetElement.classList.add('hoverd');
        setTeacherForTrial('')
    }
    else{
        setTeacherForTrial(id)
        targetElement?.classList.add('active');
        targetElement?.classList.remove('hoverd');
    }
}

const addTeacherClassic = (e: React.MouseEvent, id: string) => {
    const targetElement = (e.target as HTMLElement).closest('.button-teacher');

    document.querySelectorAll('.add-to-teacher > .button-teacher').forEach(item => {
        if (item !== targetElement) {
            item.classList.remove('active');
            item.classList.add('hoverd');
        }
    });

    if (targetElement?.classList.contains('active')) {
        targetElement.classList.remove('active');
        targetElement.classList.add('hoverd');
        setTeacherForClassic('')
    }
    else{
        setTeacherForClassic(id)
        targetElement?.classList.add('active');
        targetElement?.classList.remove('hoverd');
    }
}

//
// INDIVIDUALS //
//
const [userDataToFindLesson, setUserDataToFindLesson] = useState<UserInterface>(User);

const [allStudentsIndividual, setAllStudentsIndividual] =
useState<UserInterface[]>([User]);

const [teachersStudentsTrialLessons, setTeachersStudentsTrialLessons] =
useState<UserInterface[]>([User]);

const [operationOpenedLessons, setOperationOpenedLessons] = useState<boolean>(false);
const [operationOpenedAllLessons, setOperationOpenedAllLessons] =
useState<boolean>(false);

const [modalFunctionOpenedLessons, setModalFunctionOpenedLessons] =
useState<string>('');
const [modalFunctionOpenedAllLessons, setModalFunctionOpenedAllLessons] =
useState<string>('');

const [targetTeacherIndividualLessons, setTargetTeacherIndividualLessons] =
useState<UserInterface>(User);
const [operationOpenedTeachers, setOperationOpenedTeachers] = useState<boolean>(false);
const [modalFunctionOpenedTeachers, setModalFunctionOpenedTeachers] =
useState<string>('');

const [transferToAnotherTeacherLessonsId, setTransferToAnotherTeacherLessonsId] =
useState<string>('');

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижморя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		131

```

useEffect(() => {
    const allFunctions = ['change-individual-balance', 'edit-link', 'transfer-to-
another-teacher', 'delete-student'];

    if (modalFunctionOpenedLessons !== '') {
        const userId = userDataToFindLesson.id;
        console.log(modalFunctionOpenedLessons);
        const targetElement = document.querySelector(`.${modalFunctionOpenedLessons}`)
as HTMLElement;

        allFunctions.filter(item => item !== modalFunctionOpenedLessons).forEach(item =>
document.querySelector(`.${item}`)?.classList.remove('active'));

        if (targetElement.classList.contains('active')){
            targetElement.classList.remove('active');
            (document.querySelector('.operation-menu-individual-
lessons'))?.querySelector('.functions')?.classList.remove('active');

            switch (modalFunctionOpenedLessons) {
                case 'change-individual-balance':
                    const inputBalance = targetElement.querySelector('.form__field--
edit-balance-individual') as HTMLInputElement;

                    const changeBalance = async () => {
                        try {
                            const response = await ChangeBalance(userId.toString(),
inputBalance.value);
                            console.log(response);
                            if (response) {
                                showSuccessStatus('Успішно', 'Баланс успішно змінено');
                                inputBalance.value = '';
                            }
                            else showFailedStatus('Помилка', 'Не вдалося змінити
баланс');
                        }
                        catch (error) {
                            showFailedStatus('Помилка', 'Не вдалося змінити баланс');
                            console.log(error);
                        }
                    }

                    if (inputBalance && inputBalance.value !== '') {
                        changeBalance();
                    }

                    break;

                case 'edit-link':
                    const inputLink = targetElement.querySelector('.form__field--edit-
link') as HTMLInputElement;

                    const editLink = async () => {
                        try {
                            const response = await EditMiroLink(userId.toString(),
inputLink.value);

                            if (!response) showFailedStatus('Помилка', 'Не вдалося
змінити посилання');
                            else{
                                showSuccessStatus('Успішно', 'Посилання успішно
змінено');

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		132

```

        inputLink.value = '';
    }
}
catch (error) {
    showFailedStatus('Помилка', 'Не вдалося змінити посилання');
    console.log(error);
}
}

if (inputLink && inputLink.value !== '') {
    editLink();
}

break;

case 'transfer-to-another-teacher':
    const transferToAnotherTeacher = async () => {
        try {
            const response = await
TransferToAnotherTeacher(userId.toString(), transferToAnotherTeacherLessonsId);

            if (response) showSuccessStatus('Успішно', 'Студент
перенесений на іншого викладача');
            else showFailedStatus('Помилка', 'Не вдалося перенести
студента на іншого викладача');
        }
        catch (error) {
            showFailedStatus('Помилка', 'Не вдалося перенести студента
на іншого викладача');
            console.log(error);
        }
    }

    document.querySelectorAll('.transfer-to-another-teacher > .button-
teacher').forEach(item => {
        item.classList.remove('active');
        item.classList.add('hoverd');
    });

    if (transferToAnotherTeacherLessonsId !== '')
transferToAnotherTeacher();

    setTransferToAnotherTeacherLessonsId('');

    break;

case 'delete-student':
    const target = document.querySelector('.agree-with-delete-student-
on-individual-lesson') as HTMLElement;

    const deleteStudent = async () => {
        try {
            const response = await DeleteStudent(userId.toString());

            if (response) showSuccessStatus('Успішно', 'Студент успішно
видалений');
            else showFailedStatus('Помилка', 'Не вдалося видалити
студента');
        }
        catch (error) {
            showFailedStatus('Помилка', 'Не вдалося видалити студента');
        }
    }

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		133

```

        console.log(error);
    }
}

if (target.classList.contains('active')) {
    target.classList.remove('active');
    deleteStudent();
}

break;

}
}
else{
    targetElement.classList.add('active');
    (document.querySelector('.operation-menu-individual-
lessons'))?.querySelector('.functions')?.classList.add('active');
}
}
else{
    console.log(modalFunctionOpened);
    allFunctions.forEach(item =>
document.querySelector(`.${item}`)?.classList.remove('active'));
    (document.querySelector('.operation-menu-individual-
lessons'))?.querySelector('.functions')?.classList.remove('active');
}
}, [operationOpenedLessons]);

useEffect(() => {
    const allFunctions = ['all-change-individual-balance', 'all-edit-link', 'all-
transfer-to-another-teacher', 'all-delete-student'];

    if (modalFunctionOpenedAllLessons !== '') {
        console.log(modalFunctionOpenedLessons);
        const userId = userDataToFindLesson.id;
        const targetElement =
document.querySelector(`.${modalFunctionOpenedAllLessons}`) as HTMLElement;

        allFunctions.filter(item => item !== modalFunctionOpenedAllLessons).forEach(item
=> document.querySelector(`.${item}`)?.classList.remove('active'));

        if (targetElement.classList.contains('active')){
            targetElement.classList.remove('active');
            (document.querySelector('.all-operation-menu-individual-
lessons'))?.querySelector('.functions')?.classList.remove('active');

            switch (modalFunctionOpenedAllLessons) {
                case 'all-change-individual-balance':
                    const inputBalance = targetElement.querySelector('.form__field--
edit-balance-individual') as HTMLInputElement;

                    const changeBalance = async () => {
                        try {
                            const response = await ChangeBalance(userId.toString(),
inputBalance.value);

                            if (response) {
                                showSuccessStatus('Успішно', 'Баланс успішно змінено');
                                inputBalance.value = '';
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        else showFailedStatus('Помилка', 'Не вдалося змінити
баланс');
    }
    catch (error) {
        showFailedStatus('Помилка', 'Не вдалося змінити баланс');
        console.log(error);
    }
}

if (inputBalance && inputBalance.value !== '') {
    changeBalance();
}

break;

case 'all-edit-link':
    const inputLink = targetElement.querySelector('.form__field--edit-
link') as HTMLInputElement;

    const editLink = async () => {
        try {
            const response = await EditMiroLink(userId.toString(),
inputLink.value);

            if (response) {
                showSuccessStatus('Успішно', 'Посилання успішно
змінено');
                inputLink.value = '';
            }
            else showFailedStatus('Помилка', 'Не вдалося змінити
посилання');
        }
        catch (error) {
            showFailedStatus('Помилка', 'Не вдалося змінити посилання');
            console.log(error);
        }
    }

    if (inputLink && inputLink.value !== '') {
        editLink();
    }

    break;

case 'all-transfer-to-another-teacher':
    const transferToAnotherTeacher = async () => {
        try {
            const response = await
TransferToAnotherTeacher(userId.toString(), transferToAnotherTeacherLessonsId);

            if (response) showSuccessStatus('Успішно', 'Студент
перенесений до іншого викладача');
            else showFailedStatus('Помилка', 'Не вдалося перенести
студента до іншого викладача');
        }
        catch (error) {
            showFailedStatus('Помилка', 'Не вдалося перенести студента
до іншого викладача');
            console.log(error);
        }
    }
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		135

```

        document.querySelectorAll('.all-transfer-to-another-teacher >
.button-teacher').forEach(item => {
            item.classList.remove('active');
            item.classList.add('hoverd');
        });

        if (transferToAnotherTeacherLessonsId !== ''){
            transferToAnotherTeacher();
        }

        setTransferToAnotherTeacherLessonsId('');

        break;

        case 'all-delete-student':
            const target = targetElement.querySelector('.agree-with-delete-
student-on-individual-lesson') as HTMLElement;

            const deleteStudent = async () => {
                try {
                    const response = await DeleteStudent(userId.toString());

                    if (response) showSuccessStatus('Успішно', 'Студент успішно
видалений');
                    else showFailedStatus('Помилка', 'Не вдалося видалити
студента');
                }
                catch (error) {
                    showFailedStatus('Помилка', 'Не вдалося видалити студента');
                    console.log(error);
                }
            }

            if (target.classList.contains('active')) {
                target.classList.remove('active');
                deleteStudent();
            }

            break;

        }
    }
    else{
        targetElement.classList.add('active');
        (document.querySelector('.all-operation-menu-individual-
lessons'))?.querySelector('.functions')?.classList.add('active');
    }
}
else{
    console.log(modalFunctionOpened);
    allFunctions.forEach(item =>
document.querySelector(`.${item}`)?.classList.remove('active'));
    (document.querySelector('.all-operation-menu-individual-
lessons'))?.querySelector('.functions')?.classList.remove('active');
}
}, [operationOpenedAllLessons]);

useEffect(() => {
    const allFunctions = ['students', 'delete-teacher', 'trial-students'];

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		136

```

    if (modalFunctionOpenedTeachers !== '') {
        console.log(modalFunctionOpenedTeachers);

        const teacherId = targetTeacherIndividualLessons.id;
        const targetElement = document.querySelector(`.${modalFunctionOpenedTeachers}`)
as HTMLElement;

        allFunctions.filter(item => item !== modalFunctionOpenedTeachers).forEach(item
=> document.querySelector(`.${item}`)?.classList.remove('active'));

        if (targetElement.classList.contains('active')){
            targetElement.classList.remove('active');
            (document.querySelector('.operation-menu-choosed-
teacher'))?.querySelector('.functions')?.classList.remove('active');

            switch (modalFunctionOpenedTeachers) {
                case 'delete-teacher':
                    const deleteTeacher = async () => {
                        try{
                            const response = await DeleteTeacher(teacherId.toString());
                            console.log(response);
                            if (!response) showFailedStatus('Помилка', 'Не вдалося
видалити викладача');
                            else showSuccessStatus('Успішно', 'Викладача успішно
видалено');
                        }
                        catch(error){
                            showFailedStatus('Помилка', 'Не вдалося видалити
викладача');
                            console.log(error);
                        }
                    }

                    const target = document.querySelector('.agree-with-delete-teacher-
on-individual-lesson') as HTMLElement;

                    if (target.classList.contains('active')) {
                        target.classList.remove('active');
                        deleteTeacher();
                    }

                    break;

                }
            }
            else{
                targetElement.classList.add('active');
                (document.querySelector('.operation-menu-choosed-
teacher'))?.querySelector('.functions')?.classList.add('active');
            }
        }
        else{
            console.log(modalFunctionOpenedTeachers);
            allFunctions.forEach(item =>
document.querySelector(`.${item}`)?.classList.remove('active'));
            (document.querySelector('.operation-menu-choosed-
teacher'))?.querySelector('.functions')?.classList.remove('active');
        }
    }, [operationOpenedTeachers]);

    const afterFindUserGoToOperationMenuLessons = () => {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		137

```

const input = document.querySelector('.form__field--name') as HTMLInputElement;

const getToFindUser = async () => {
  try{
    const response = await FindUser(input.value);

    if (!response) showFailedStatus('Помилка', 'Користувача не знайдено');
    else {
      document.querySelector('.find-user-individual-
lessons')?.classList.remove('active');
      document.querySelector('.operation-menu-individual-
lessons')?.classList.add('active');
      setUserDataToFindLesson(response);
    }
  } catch (error) {
    showFailedStatus('Помилка', 'Користувача не знайдено');
    console.error('Error verifying token', error);
  }
}

if (input.value){
  getToFindUser();
}
else showFailedStatus('Помилка', 'Введіть дані користувача');
}

const openOrFinishOperationLessons = (e: React.MouseEvent, whatNeedToOpen: string) => {
  e.preventDefault();
  const button = e.target as HTMLElement;

  document.querySelectorAll('.functions-button-find-user')
    .forEach(item => item !== button && item.classList.remove('active'));

  if (!button.classList.contains('active')) button.classList.add('active');
  else button.classList.remove('active');

  setOperationOpenedLessons(!operationOpenedLessons);
  setModalFunctionOpenedLessons(whatNeedToOpen);
}

const openOrFinishOperationAllLessons = (e: React.MouseEvent, whatNeedToOpen: string) =>
{
  e.preventDefault();
  const button = e.target as HTMLElement;

  document.querySelectorAll('.functions-button-find-user')
    .forEach(item => item !== button && item.classList.remove('active'));

  if (!button.classList.contains('active')) button.classList.add('active');
  else button.classList.remove('active');

  setOperationOpenedAllLessons(!operationOpenedAllLessons);
  setModalFunctionOpenedAllLessons(whatNeedToOpen);
}

const transferToAnotherTeacherLessons = (e: React.MouseEvent, name: string) => {
  const targetElement = (e.target as HTMLElement).closest('.button-teacher');

  document.querySelectorAll('.transfer-to-another-teacher.active > .button-
teacher').forEach(item => {
    if (item !== targetElement) {

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		138

```

        item.classList.remove('active');
        item.classList.add('hoverd');
    }
});

if (targetElement?.classList.contains('active')) {
    targetElement.classList.remove('active');
    targetElement.classList.add('hoverd');
    setTransferToAnotherTeacherLessonsId('');
}
else {
    targetElement?.classList.remove('hoverd');
    targetElement?.classList.add('active');
    setTransferToAnotherTeacherLessonsId(name);
}
}

const agreeWithDeleteStudentLessons = () => {
    document.querySelector('.agree-with-delete-student-on-individual-
    lesson')?.classList.toggle('active');
    document.querySelector('.agree-with-delete-student-on-individual-
    lesson')?.classList.toggle('hoverd');
}

const goToOperationMenuWithTeacherIndividualLessons = (e: React.MouseEvent, id: string)
=> {
    const getUserData = async () => {
        try{
            const response = await GetUser(id);
            if (!response) showFailedStatus('Помилка', 'Користувача не знайдено');
            else{
                setTargetTeacherIndividualLessons(response);
                document.querySelector('.operation-menu-choosed-
                teacher')?.classList.add('active');
                document.querySelector('.all-our-teachers-individual-
                lessons')?.classList.remove('active');
            }
        } catch (error) {
            showFailedStatus('Помилка', 'Користувача не знайдено');
            console.error('Error verifying token', error);
        }
    }

    getUserData();
}

const agreeWithDeleteTeacherLessons = () => {
    document.querySelector('.agree-with-delete-teacher-on-individual-
    lesson')?.classList.toggle('active');
    document.querySelector('.agree-with-delete-teacher-on-individual-
    lesson')?.classList.toggle('hoverd');
}

const openOrFinishOperationTeachers = (e: React.MouseEvent, whatNeedToOpen: string) => {
    e.preventDefault();
    const button = e.target as HTMLElement;

    const getTeachersStudents = async () => {
        try{
            const response = await
TeachersStudents((targetTeacherIndividualLessons.id).toString());

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		139

```

        if (!response) showFailedStatus('Помилка', 'Студентів викладача не
знайдено');
        else setAllStudentsIndividual(response);
    }
    catch(error){
        showFailedStatus('Помилка', 'Студентів викладача не знайдено');
        console.log(error);
    }
}

const getTeachersStudentsTrial = async () => {
    try{
        const response = await
TeachersTrialStudents((targetTeacherIndividualLessons.id).toString());
        if (!response) showFailedStatus('Помилка', 'Студентів викладача в
майбутньому пробних занять не знайдено');
        else setTeachersStudentsTrialLessons(response);
        console.log(response);
    }
    catch(error){
        console.log(error);
    }
}

document.querySelectorAll('.functions-button-find-user')
.forEach(item => item !== button && item.classList.remove('active'));

if (!button.classList.contains('active')){
    button.classList.add('active');

    if (whatNeedToOpen === 'students'){
        getTeachersStudents();
    }
    else if (whatNeedToOpen === 'trial-students'){
        getTeachersStudentsTrial();
    }
}
else button.classList.remove('active');

setOperationOpenedTeachers(!operationOpenedTeachers);
setModalFunctionOpenedTeachers(whatNeedToOpen);
}

const afterAllStudentsIndividualLessonsGotToOperationMenu = (e: React.MouseEvent, id:
string) => {
    const getUserData = async () => {
        try{
            const response = await GetUser(id);
            if (!response) return alert('Користувача не знайдено');
            setDataToFindLesson(response);
        } catch (error) {
            console.error('Error verifying token', error);
        }
    }

    getUserData();
    document.querySelector('.all-our-students-individual-
lessons')?.classList.remove('active');
    document.querySelector('.all-operation-menu-individual-
lessons')?.classList.add('active');
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		140

```

// navigation

const backToFindUserAfterOperationLessons = () => {
    document.querySelector('.operation-menu-individual-
lessons')?.classList.remove('active');
    document.querySelector('.find-user-individual-lessons')?.classList.add('active');
}

const backToAllTeachersLessons = () => {
    document.querySelector('.operation-menu-choosed-
teacher')?.classList.remove('active');
    document.querySelector('.all-our-teachers-individual-
lessons')?.classList.add('active');
}

const backToAllUsersIndividualLessons = () => {
    document.querySelector('.all-operation-menu-individual-
lessons')?.classList.remove('active');
    document.querySelector('.all-our-students-individual-
lessons')?.classList.add('active');
}

return (
    <>
    <EventStatus title={eventStatusTitle} description={eventStatusDescription}/>

    <div className="modal find-user-individual">
        <div className="modal-content">
            <img src={close} alt="" className="exit-button" onClick={closeModal} />
            <div className="modal-header">
                <div className="modal-title">Введіть дані користувача</div>
                <div className="modal-description">[ Telegram Id, повне ім'я, номер
телефону, нік в Telegram ]</div>
            </div>
            <div className="modal-body">
                <div className="form__group field">
                    <input type="input" className="form__field" placeholder="Name"/>
                </div>
            </div>
            <div className="modal-footer">
                <div className="find-user"
onClick={afterFindUserGoToOperationMenu}>Знайти</div>
            </div>
        </div>
    </div>

    <div className="modal operation-menu-individual">
        <div className="modal-content">
            <img src={back} alt="" className="button-back"
onClick={backToFindUserAfterOperation} />
            <img src={close} alt="" className="exit-button" onClick={closeModal} />
            <div className="modal-header">
                <div className="modal-title">{userDataToFind.name}</div>
                <div className="modal-description">[ Виберіть операцію та виконайте
ii ]</div>
            </div>
            <div className="modal-body">
                <div className="info-user">
                    <div className="info-user__item"><b>Telegram Id:</b>
{userDataToFind.id}</div>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		141

```

        <div className="info-user__item"><b>Повне ім'я:</b>
{userDataToFind.name}</div>
        <div className="info-user__item"><b>Роль:</b>
{rolesTranscript[userDataToFind.role || 'guest']}</div>
        <div className="info-user__item"><b>Номер телефону:</b>
{userDataToFind.number}</div>
        <div className="info-user__item"><b>Нік в Telegram:</b>
@{userDataToFind.username}</div>
        { (userDataToFind.role == 'student' || userDataToFind.role ==
'guest') && <div className="info-user__item"><b>Баланс індивідуальних занять:</b>
{userDataToFind.individual_count??0}хв</div> }
        { (userDataToFind.role == 'student' || userDataToFind.role ==
'guest') && <div className="info-user__item"><b>Баланс клубних занять:</b>
{userDataToFind.count ?? 0} шт.</div> }
        { (userDataToFind.role == 'student' || userDataToFind.role ==
'guest') && <div className="info-user__item"><b>Заплановано індивідуальних занять:</b>
{userDataToFind.individual_lessons?.length ?? 0} шт.</div> }
        { (userDataToFind.role == 'student' || userDataToFind.role ==
'guest') && <div className="info-user__item"><b>Записаний(-а) на клубні заняття:</b>
{userDataToFind.recordClubs?.length ?? 0} шт.</div> }
        { userDataToFind.role == 'teacher' && <div className="info-
user__item"><b>Записаний(-а) на клубні заняття:</b> {userDataToFind.recordClubs?.length ??
0} шт.</div> }
        { userDataToFind.role == 'teacher' && <div className="info-
user__item"><b>Кількість студентів:</b> {userDataToFind.registered_students?.length ?? 0}
шт.</div> }
        { userDataToFind.role == 'teacher' && <div className="info-
user__item"><b>Кількість студентів на пробне:</b> {userDataToFind.trial_students?.length ??
0} шт.</div> }
        { userDataToFind.role == 'teacher' && <div className="info-
user__item"><b>Кількість активних деTaskів:</b> {userDataToFind.set_detasks?.length ?? 0}
шт.</div> }
        { userDataToFind.role == 'teacher' && <div className="info-
user__item"><b>Кількість запланованих індивідуальних занять:</b>
{userDataToFind.set_individual_lessons?.length ?? 0} хв</div> }
    </div>
</div>
<div className="modal-footer">
    {userDataToFind.role == 'guest' ?
    <>
        <div className="functions-button-find-user" onClick={{(e) =>
openOrFinishOperation(e, 'change-role-function')}}>Змінити роль</div>
        <div className="functions-button-find-user" onClick={{(e) =>
openOrFinishOperation(e, 'change-name-function')}}>Змінити ім'я</div>
        <div className="functions-button-find-user" onClick={{(e) =>
openOrFinishOperation(e, 'add-to-trial-function')}}>Додати на пробне</div>
        <div className="functions-button-find-user" onClick={{(e) =>
openOrFinishOperation(e, 'add-to-teacher')}}>Додати викладачеві</div>
    </>
    :
        <div className="functions-button-find-user" onClick={{(e) =>
openOrFinishOperation(e, 'change-name-function')}}>Змінити ім'я</div>
    }
</div>
<div className="functions">
    <div className="change-role-function">
        <div className="button-role hoverd" onClick={{(e) =>
chooseRole(e, 'admin')}}>Адміністратор</div>
        <div className="button-role hoverd" onClick={{(e) =>
chooseRole(e, 'student')}}>Студент</div>
    </div>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		142

```

        <div className="button-role hoverd" onClick={ (e) =>
chooseRole(e, 'teacher')}>Викладач</div>
        </div>

        <div className="change-name-function">
        <div className="form__group field">
        <input type="input" className="form__field"
placeholder="Name"/>
        </div>
        </div>

        <div className="add-to-trial-function">
        {teachers.map((item, index) => <div className="button-teacher
hoverd" key={index} onClick={ (e) => addTeacherTrial(e, item.id.toString())}>
        <div className="teacher-main">
        <div className="teacher-name">{item.name}</div>
        <div className="teacher-id">{item.id}</div>
        </div>
        <div className="teacher-stats">
        <div className="teacher-stats__item"><b>Кількість
пробних учнів:</b> {item.trial_students?.length || 0}</div>
        <div className="teacher-stats__item"><b>Кількість
індивідуальних учнів:</b> {item.registered_students?.length || 0}</div>
        <div className="teacher-stats__item"><b>Запланованих
індивідуальних занять:</b> {item.set_individual_lessons?.length || 0}</div>
        <div className="teacher-stats__item"><b>Запланованих
клубних занять:</b> {item.recordClubs?.length || 0}</div>
        </div>
        </div>)}
        </div>

        <div className="add-to-teacher">
        {teachers.map((item, index) => <div className="button-teacher
hoverd" key={index} onClick={ (e) => addTeacherClassic(e, item.id.toString())}>
        <div className="teacher-main">
        <div className="teacher-name">{item.name}</div>
        <div className="teacher-id">{item.id}</div>
        </div>
        <div className="teacher-stats">
        <div className="teacher-stats__item"><b>Кількість
індивідуальних учнів:</b> {item.registered_students?.length || 0}</div>
        </div>
        </div>)}
        </div>
        </div>
        </div>
        </div>

        <div className="modal operation-menu-individual-after-all">
        <div className="modal-content">
        <img src={back} alt="" className="button-back"
onClick={backToAllUsersIndividual} />
        <img src={close} alt="" className="exit-button" onClick={closeModal} />
        <div className="modal-header">
        <div className="modal-title">{choosedUser.name}</div>
        <div className="modal-description">[ Виберіть операцію та виконайте
ii ]</div>
        </div>
        <div className="modal-body">
        <div className="info-user">

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		143

```

        <div className="info-user__item"><b>Telegram Id:</b>
{choosedUser.id}</div>
        <div className="info-user__item"><b>Повне ім'я:</b>
{choosedUser.name}</div>
        <div className="info-user__item"><b>Роль:</b>
{rolesTranscript[choosedUser.role || 'guest']}</div>
        <div className="info-user__item"><b>Номер телефону:</b>
{choosedUser.number}</div>
        <div className="info-user__item"><b>Нік в Telegram:</b>
@{choosedUser.username}</div>
        { (choosedUser.role == 'student' || choosedUser.role == 'guest')
&& <div className="info-user__item"><b>Баланс індивідуальних занять:</b>
{choosedUser.individual_count??0}хв</div> }
        { (choosedUser.role == 'student' || choosedUser.role == 'guest')
&& <div className="info-user__item"><b>Баланс клубних занять:</b> {choosedUser.count ?? 0}
шт.</div> }
        { (choosedUser.role == 'student' || choosedUser.role == 'guest')
&& <div className="info-user__item"><b>Заплановано індивідуальних занять:</b>
{choosedUser.individual_lessons?.length ?? 0} шт.</div> }
        { (choosedUser.role == 'student' || choosedUser.role == 'guest')
&& <div className="info-user__item"><b>Записаний(-а) на клубні заняття:</b>
{choosedUser.recordClubs?.length ?? 0} шт.</div> }
        { choosedUser.role == 'teacher' && <div className="info-
user__item"><b>Записаний(-а) на клубні заняття:</b> {choosedUser.recordClubs?.length ?? 0}
шт.</div> }
        { choosedUser.role == 'teacher' && <div className="info-
user__item"><b>Кількість студентів:</b> {choosedUser.registered_students?.length ?? 0}
шт.</div> }
        { choosedUser.role == 'teacher' && <div className="info-
user__item"><b>Кількість студентів на пробне:</b> {choosedUser.trial_students?.length ?? 0}
шт.</div> }
        { choosedUser.role == 'teacher' && <div className="info-
user__item"><b>Кількість активних деТасків:</b> {choosedUser.set_detasks?.length ?? 0}
шт.</div> }
        { choosedUser.role == 'teacher' && <div className="info-
user__item"><b>Кількість запланованих індивідуальних занять:</b>
{choosedUser.set_individual_lessons?.length ?? 0} хв</div> }
    </div>
</div>
<div className="modal-footer">
    {choosedUser.role == 'guest' ?
    <>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationAll(e, 'all-change-role-function')}>Змінити роль</div>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationAll(e, 'all-change-name-function')}>Змінити ім'я</div>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationAll(e, 'all-add-to-trial-function')}>Додати на пробне</div>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationAll(e, 'all-add-to-teacher')}>Додати викладачеві</div>
    </>
    :
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationAll(e, 'all-change-name-function')}>Змінити ім'я</div>
    }
</div>
<div className="functions">
    <div className="all-change-role-function">
        <div className="button-role hoverd" onClick={(e) =>
chooseRole(e, 'admin')}>Адміністратор</div>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		144

```

        <div className="button-role hoverd" onClick={(e) =>
chooseRole(e, 'student')}>Студент</div>
        <div className="button-role hoverd" onClick={(e) =>
chooseRole(e, 'teacher')}>Викладач</div>
    </div>

    <div className="all-change-name-function">
        <div className="form__group field">
            <input type="input" className="form__field"
placeholder="Name"/>
        </div>
    </div>

    <div className="all-add-to-trial-function">
        {teachers.map((item, index) => <div className="button-teacher
hoverd" key={index} onClick={(e) => addTeacherTrial(e, item.id.toString())}>
            <div className="teacher-main">
                <div className="teacher-name">{item.name}</div>
                <div className="teacher-id">{item.id}</div>
            </div>
            <div className="teacher-stats">
                <div className="teacher-stats__item"><b>Кількість
пробних учнів:</b> {item.trial_students?.length || 0}</div>
                <div className="teacher-stats__item"><b>Кількість
клубних занять:</b> {item.recordClubs?.length || 0}</div>
                <div className="teacher-stats__item"><b>Кількість
індивідуальних учнів:</b> {item.registered_students?.length || 0}</div>
                <div className="teacher-stats__item"><b>Кількість
індивідуальних занять:</b> {item.set_individual_lessons?.length || 0}</div>
            </div>
        </div>)}
    </div>

    <div className="all-add-to-teacher">
        {teachers.map((item, index) => <div className="button-teacher
hoverd" key={index} onClick={(e) => addTeacherClassic(e, item.id.toString())}>
            <div className="teacher-main">
                <div className="teacher-name">{item.name}</div>
                <div className="teacher-id">{item.id}</div>
            </div>
            <div className="teacher-stats">
                <div className="teacher-stats__item"><b>Кількість
індивідуальних учнів:</b> {item.registered_students?.length || 0}</div>
            </div>
        </div>)}
    </div>
</div>
</div>
</div>

<div className="modal all-user-individual">
    <div className="modal-content">
        <img src={close} alt="" className="exit-button" onClick={closeModal} />
        <div className="modal-header">
            <div className="modal-title">Виберіть користувача</div>
            <div className="modal-description">[ Користуйтеся пошуком для
фільтрації ]</div>
        </div>
        <div className="modal-body">
            <div className="all-user-individual-body">
                <div className="form__group field">

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				145
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <input type="input" className="form__field"
placeholder="Name" onChange={(e) => setFilterUsers(e.target.value)}/>
      </div>
      <div className="students-individual">
        {allUsers.map((item, index) =>
          <div className="button-teacher hoverd" key={index}
onClick={(e) => afterChoosingUserGoToOperationMenuIndividual(e, item.id.toString())}>
            <div className="user-main">
              <div className="user-name">{item.name}</div>
              <div className="user-id">{item.id}</div>
            </div>
            <div className="user-stats">
              <div className="user-stats__item"><b>Роль:</b>
{rolesTranscript[item.role]}</div>
              <div className="user-
stats__item"><b>Telegram:</b> @{item.username}</div>
              <div className="user-
stats__item"><b>Телефон:</b> {item.number}</div>
            </div>
          </div>
        )}
      </div>
    </div>
  </div>
</div>

{/* INDIVIDUAL */}

<div className="modal find-user-individual-lessons">
  <div className="modal-content">
    <img src={close} alt="" className="exit-button" onClick={closeModal} />
    <div className="modal-header">
      <div className="modal-title">Введіть дані користувача</div>
      <div className="modal-description">[ Telegram Id, повне ім'я, номер
телефону, нік в Telegram ]</div>
    </div>
    <div className="modal-body">
      <div className="form__group field">
        <input type="input" className="form__field form__field--name"
placeholder="Name"/>
      </div>
    </div>
    <div className="modal-footer">
      <div className="find-user"
onClick={afterFindUserGoToOperationMenuLessons}>Знайти</div>
    </div>
  </div>
</div>

<div className="modal operation-menu-individual-lessons">
  <div className="modal-content">
    <img src={back} alt="" className="button-back"
onClick={backToFindUserAfterOperationLessons} />
    <img src={close} alt="" className="exit-button" onClick={closeModal} />
    <div className="modal-header">
      <div className="modal-title">{userDataToFindLesson.name}</div>
      <div className="modal-description">[ Виберіть операцію та виконайте
її ]</div>
  </div>
</div>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		146

```

    </div>
    <div className="modal-body">
      <div className="info-user">
        <div className="info-user__item"><b>Telegram Id:</b>
{userDataToFindLesson.id}</div>
        <div className="info-user__item"><b>Повне ім'я:</b>
{userDataToFindLesson.name}</div>
        <div className="info-user__item"><b>Роль:</b>
{rolesTranscript[userDataToFindLesson.role || 'guest']}</div>
        <div className="info-user__item"><b>Номер телефону:</b>
{userDataToFindLesson.number}</div>
        <div className="info-user__item"><b>Нік в Telegram:</b>
@{userDataToFindLesson.username}</div>
        { (userDataToFindLesson.role == 'student' ||
userDataToFindLesson.role == 'guest') && <div className="info-user__item"><b>Баланс
індивідуальних занять:</b> {userDataToFindLesson.individual_count??0}хв</div> }
        { (userDataToFindLesson.role == 'student' ||
userDataToFindLesson.role == 'guest') && <div className="info-user__item"><b>Баланс клубних
занять:</b> {userDataToFindLesson.count ?? 0} шт.</div> }
        { (userDataToFindLesson.role == 'student' ||
userDataToFindLesson.role == 'guest') && <div className="info-user__item"><b>Заплановано
індивідуальних занять:</b> {userDataToFindLesson.individual_lessons?.length ?? 0} шт.</div>
}
        { (userDataToFindLesson.role == 'student' ||
userDataToFindLesson.role == 'guest') && <div className="info-user__item"><b>Записаний(-а)
на клубні заняття:</b> {userDataToFindLesson.recordClubs?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Записаний(-а) на клубні заняття:</b>
{userDataToFindLesson.recordClubs?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Кількість студентів:</b>
{userDataToFindLesson.registered_students?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Кількість студентів на пробне:</b>
{userDataToFindLesson.trial_students?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Кількість активних деТасків:</b>
{userDataToFindLesson.set_detasks?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Кількість запланованих індивідуальних занять:</b>
{userDataToFindLesson.set_individual_lessons?.length ?? 0} хв</div> }
      </div>
    </div>
    <div className="modal-footer">
      <>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationLessons(e, 'change-individual-balance')}>Змінити баланс занять</div>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationLessons(e, 'edit-link')}>Змінити лінк</div>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationLessons(e, 'transfer-to-another-teacher')}>Перевести до іншого
викладача</div>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationLessons(e, 'delete-student')}>Видалити студента</div>
      </>
    </div>
    <div className="functions">
      <div className="change-individual-balance">
        <div className="form_group field">
          <input type="input" className="form__field form__field--
edit-balance-individual" placeholder="5"/>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		147

```

        </div>
    </div>

    <div className="edit-link">
        <div className="form__group field">
            <input type="input" className="form__field form__field--
edit-link" placeholder="miro.com"/>
        </div>
    </div>

    <div className="transfer-to-another-teacher">
        {teachers.map((item, index) => <div className="button-teacher
transfer hoverd" key={index} onClick={(e) => transferToAnotherTeacherLessons(e, item.name)}>
            <div className="teacher-main">
                <div className="teacher-name">{item.name}</div>
                <div className="teacher-id">{item.id}</div>
            </div>
            <div className="teacher-stats">
                <div className="teacher-stats__item"><b>Кількість
пробних учнів:</b> {item.trial_students?.length || 0}</div>
                <div className="teacher-stats__item"><b>Індивідуальних
учнів:</b> {item.individual_count || 0}</div>
                <div className="teacher-stats__item"><b>Активних
деТасків:</b> {item.set_detasks?.length || 0}</div>
            </div>
        </div>)}
    </div>

    <div className="delete-student">
        <div className="agree-with-delete-student-on-individual-lesson
hoverd" onClick={agreeWithDeleteStudentLessons}>
            Підтверджую
        </div>
    </div>
</div>
</div>
</div>

<div className="modal all-our-teachers-individual-lessons">
    <div className="modal-content">
        <img src={close} alt="" className="exit-button" onClick={closeModal} />
        <div className="modal-header">
            <div className="modal-title">Викладачі</div>
            <div className="modal-description">[ Виберіть викладача для
подальших дій ]</div>
        </div>
        <div className="modal-body">
            <div className="all-user-individual-body">
                <div className="students-individual">
                    {teachers.map((item, index) => <div className="button-
teacher transfer hoverd" key={index} onClick={(e) =>
goToOperationMenuWithTeacherIndividualLessons(e, item.id.toString())}>
                        <div className="teacher-main">
                            <div className="teacher-name">{item.name}</div>
                            <div className="teacher-id">{item.id}</div>
                        </div>
                        <div className="teacher-stats">
                            <div className="teacher-stats__item"><b>Індивідуальних
учнів:</b> {item.registered_students?.length || 0}</div>
                            <div className="teacher-stats__item"><b>Пробних
учнів:</b> {item.trial_students?.length || 0}</div>
                        </div>
                    </div>)}
                </div>
            </div>
        </div>
    </div>
</div>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		148

```

        </div>
    </div>}}
</div>
</div>
</div>
</div>
</div>

<div className="modal operation-menu-choosed-teacher">
    <div className="modal-content">
        <img src={back} alt="" className="button-back"
onClick={backToAllTeachersLessons} />
        <img src={close} alt="" className="exit-button" onClick={closeModal} />
        <div className="modal-header">
            <div className="modal-
title">{targetTeacherIndividualLessons.name}</div>
            <div className="modal-description">[ Виберіть операцію та виконайте
ii ]</div>
        </div>
        <div className="modal-body">
            <div className="info-user">
                <div className="info-user__item"><b>Telegram Id:</b>
{targetTeacherIndividualLessons.id}</div>
                <div className="info-user__item"><b>Повне ім'я:</b>
{targetTeacherIndividualLessons.name}</div>
                <div className="info-user__item"><b>Роль:</b>
{rolesTranscript[targetTeacherIndividualLessons.role || 'guest']}</div>
                <div className="info-user__item"><b>Номер телефону:</b>
{targetTeacherIndividualLessons.number}</div>
                <div className="info-user__item"><b>Нік в Telegram:</b>
@{targetTeacherIndividualLessons.username}</div>
                { (targetTeacherIndividualLessons.role == 'student' ||
targetTeacherIndividualLessons.role == 'guest') && <div className="info-
user__item"><b>Баланс індивідуальних занять:</b>
{targetTeacherIndividualLessons.individual_count??0}хв</div> }
                { (targetTeacherIndividualLessons.role == 'student' ||
targetTeacherIndividualLessons.role == 'guest') && <div className="info-
user__item"><b>Баланс клубних занять:</b> {targetTeacherIndividualLessons.count ?? 0}
шт.</div> }
                { (targetTeacherIndividualLessons.role == 'student' ||
targetTeacherIndividualLessons.role == 'guest') && <div className="info-
user__item"><b>Заплановано індивідуальних занять:</b>
{targetTeacherIndividualLessons.individual_lessons?.length ?? 0} шт.</div> }
                { (targetTeacherIndividualLessons.role == 'student' ||
targetTeacherIndividualLessons.role == 'guest') && <div className="info-
user__item"><b>Записаний(-а) на клубні заняття:</b>
{targetTeacherIndividualLessons.recordClubs?.length ?? 0} шт.</div> }
                { targetTeacherIndividualLessons.role == 'teacher' && <div
className="info-user__item"><b>Записаний(-а) на клубні заняття:</b>
{targetTeacherIndividualLessons.recordClubs?.length ?? 0} шт.</div> }
                { targetTeacherIndividualLessons.role == 'teacher' && <div
className="info-user__item"><b>Кількість студентів:</b>
{targetTeacherIndividualLessons.registered_students?.length ?? 0} шт.</div> }
                { targetTeacherIndividualLessons.role == 'teacher' && <div
className="info-user__item"><b>Кількість студентів на пробне:</b>
{targetTeacherIndividualLessons.trial_students?.length ?? 0} шт.</div> }
                { targetTeacherIndividualLessons.role == 'teacher' && <div
className="info-user__item"><b>Кількість активних деТасків:</b>
{targetTeacherIndividualLessons.set_detasks?.length ?? 0} шт.</div> }
            </div>
        </div>
    </div>
</div>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		149

```

        { targetTeacherIndividualLessons.role == 'teacher' && <div
className="info-user__item"><b>Кількість запланованих індивідуальних занять:</b>
{targetTeacherIndividualLessons.set_individual_lessons?.length ?? 0} хв</div> }
      </div>
    </div>
    <div className="modal-footer">
      <>
        <div className="functions-button-find-user" onClick={{(e) =>
openOrFinishOperationTeachers(e, 'students')}}>Студенти викладача</div>
        <div className="functions-button-find-user" onClick={{(e) =>
openOrFinishOperationTeachers(e, 'trial-students')}}>Пробні студенти викладача</div>
        <div className="functions-button-find-user" onClick={{(e) =>
openOrFinishOperationTeachers(e, 'delete-teacher')}}>Зняти з полі</div>
      </>
    </div>
    <div className="functions">
      <div className="students">
        {allStudentsIndividual.length > 0 ?
        allStudentsIndividual.map((item, index) => <div
className="button-teacher transfer hoverd" key={index}>
          <div className="teacher-main">
            <div className="teacher-name">{item.name}</div>
            <div className="teacher-id">{item.id}</div>
          </div>
          <div className="teacher-stats">
            <div className="teacher-stats__item"><b>Баланс
індивідуальних занять:</b> {item.individual_count || 0} хв</div>
            <div className="teacher-stats__item"><b>Заплановано
індивідуальних занять:</b> {item.individual_lessons?.length || 0} хв</div>
          </div>
        </div>
        : <div className="no-students" style={{textAlign:
'center'}}>>Немає студентів</div>}
      </div>
      <div className="trial-students">
        {teachersStudentsTrialLessons.length > 0 ?
        teachersStudentsTrialLessons.map((item, index) => <div
className="button-teacher transfer hoverd" key={index}>
          <div className="teacher-main">
            <div className="teacher-name">{item.name}</div>
            <div className="teacher-id">{item.id}</div>
          </div>
          <div className="teacher-stats">
            <div className="teacher-stats__item"><b>Чи заплановане
пробне заняття: </b> {Boolean(item.trial) ? 'Так' : 'Ні'}</div>
          </div>
        </div>
        : <div className="no-students" style={{textAlign:
'center'}}>>Немає студентів</div>}
      </div>
      <div className="delete-teacher">
        <div className="agree-with-delete-teacher-on-individual-lesson
hoverd" onClick={agreeWithDeleteTeacherLessons}>
          Підтверджую
        </div>
      </div>
    </div>
  </div>
</div>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		150

```

<div className="modal all-our-students-individual-lessons">
  <div className="modal-content">
    <img src={close} alt="" className="exit-button" onClick={closeModal} />
    <div className="modal-header">
      <div className="modal-title">Студенти</div>
      <div className="modal-description">[ Виберіть студента для подальших
дій ]</div>
    </div>
    <div className="modal-body">
      <div className="all-user-individual-body">
        <div className="students-individual">
          {allStudents.filter(item => item.individual_count &&
item.individual_count >= 0).map((item, index) => <div className="button-teacher transfer
hoverd" key={index} onClick={(e) => afterAllStudentsIndividualLessonsGotToOperationMenu(e,
item.id.toString())}>
            <div className="teacher-main">
              <div className="teacher-name">{item.name}</div>
              <div className="teacher-id">{item.id}</div>
            </div>
            <div className="teacher-stats">
              <div className="teacher-stats__item"><b>Кількість
індивідуальних занять:</b> {item.individual_count || 0} хв</div>
              <div className="teacher-stats__item"><b>Запланованих
індивідуальних занять:</b> {item.individual_lessons?.length || 0} хв</div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>

  <div className="modal all-operation-menu-individual-lessons">
    <div className="modal-content">
      <img src={back} alt="" className="button-back"
onClick={backToAllUsersIndividualLessons} />
      <img src={close} alt="" className="exit-button" onClick={closeModal} />
      <div className="modal-header">
        <div className="modal-title">{userDataToFindLesson.name}</div>
        <div className="modal-description">[ Виберіть операцію та виконайте
іі ]</div>
      </div>
      <div className="modal-body">
        <div className="info-user">
          <div className="info-user__item"><b>Telegram Id:</b>
{userDataToFindLesson.id}</div>
          <div className="info-user__item"><b>Повне ім'я:</b>
{userDataToFindLesson.name}</div>
          <div className="info-user__item"><b>Роль:</b>
{rolesTranscript[userDataToFindLesson.role || 'guest']}</div>
          <div className="info-user__item"><b>Номер телефону:</b>
{userDataToFindLesson.number}</div>
          <div className="info-user__item"><b>Нік в Telegram:</b>
@{userDataToFindLesson.username}</div>
          { (userDataToFindLesson.role == 'student' ||
userDataToFindLesson.role == 'guest') && <div className="info-user__item"><b>Баланс
індивідуальних занять:</b> {userDataToFindLesson.individual_count??0}хв</div> }
          { (userDataToFindLesson.role == 'student' ||
userDataToFindLesson.role == 'guest') && <div className="info-user__item"><b>Баланс клубних
занять:</b> {userDataToFindLesson.count ?? 0} шт.</div> }
        </div>
      </div>
    </div>
  </div>

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		151

```

        { (userDataToFindLesson.role == 'student' ||
userDataToFindLesson.role == 'guest') && <div className="info-user__item"><b>Заплановано
індивідуальних занять:</b> {userDataToFindLesson.individual_lessons?.length ?? 0} шт.</div>
}
        { (userDataToFindLesson.role == 'student' ||
userDataToFindLesson.role == 'guest') && <div className="info-user__item"><b>Записаний(-а)
на клубні заняття:</b> {userDataToFindLesson.recordClubs?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Записаний(-а) на клубні заняття:</b>
{userDataToFindLesson.recordClubs?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Кількість студентів:</b>
{userDataToFindLesson.registered_students?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Кількість студентів на пробне:</b>
{userDataToFindLesson.trial_students?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Кількість активних деТасків:</b>
{userDataToFindLesson.set_detasks?.length ?? 0} шт.</div> }
        { userDataToFindLesson.role == 'teacher' && <div
className="info-user__item"><b>Кількість запланованих індивідуальних занять:</b>
{userDataToFindLesson.set_individual_lessons?.length ?? 0} хв</div> }
    </div>
</div>
<div className="modal-footer">
    <>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationAllLessons(e, 'all-change-individual-balance')}>Змінити баланс
занять</div>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationAllLessons(e, 'all-edit-link')}>Змінити лінк</div>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationAllLessons(e, 'all-transfer-to-another-teacher')}>Перевести до іншого
викладача</div>
        <div className="functions-button-find-user" onClick={(e) =>
openOrFinishOperationAllLessons(e, 'all-delete-student')}>Видалити студента</div>
    </>
</div>
<div className="functions">
    <div className="all-change-individual-balance">
        <div className="form_group field">
            <input type="input" className="form__field form__field--
edit-balance-individual" placeholder="5"/>
        </div>
    </div>

    <div className="all-edit-link">
        <div className="form_group field">
            <input type="input" className="form__field form__field--
edit-link" placeholder="miro.com"/>
        </div>
    </div>

    <div className="all-transfer-to-another-teacher">
        {teachers.map((item, index) => <div className="button-teacher
transfer hoverd" key={index} onClick={(e) => transferToAnotherTeacherLessons(e, item.name)}>
            <div className="teacher-main">
                <div className="teacher-name">{item.name}</div>
                <div className="teacher-id">{item.id}</div>
            </div>
            <div className="teacher-stats">

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		152


```
@import
url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,100..900;1,100..900
&family=Noto+Sans:ital,wght@0,100..900;1,100..900&display=swap');
```

```
body {
  margin: 0;
  font-family: 'Montserrat';
  background-color: #f5f7fa;
  color: #333;
}
```

```
.admin-container{
  flex: 1;
  display: flex;
  padding: 10px;
  gap: 10px;
  flex-direction: row;
  width: 100%;
  height: 100%;
  box-sizing: border-box;
}
```

```
.admin-content{
  display: flex;
  flex-direction: column;
  position: relative;
  align-items: flex-start;
  flex: 1;
  gap: 10px;
}
```

```
.admin-header {
  background-color: white;
  flex-direction: column;
  color: black !important;
  border-radius: 12px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.05);
  padding: 24px 20px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

```
.avatar {
  width: 48px;
  height: 48px;
  border-radius: 50%;
  background-color: #e1ecf4;
  color: #0077cc;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 24px;
  font-weight: bold;
}
```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		154

```

.logo{
  width: 46px;
  height: 46px;
  border-radius: 50%;
  background-color: #e1ecf4;
  color: #0077cc;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 26px;
  font-weight: bold;
}

.user-data-and-logout{
  display: flex;
  flex-direction: column;
  justify-content: flex-end;
}

.user-info {
  background-color: white;
  padding: 10px;
  border-radius: 12px;
  gap: 10px;
  display: flex;
}

.user-name {
  font-size: 16px;
  font-weight: 600;
}

.user-role {
  font-size: 12px;
  color: #777;
}

.user-data{
  display: flex;
  flex-direction: column;
}

.logout-button {
  margin-top: 2px;
  font-size: 14px;
  font-weight: 600;
  background: white;
  color: rgb(179, 0, 0);
  border: none;
  cursor: pointer;
  transition: all 0.3s ease;
}

.logout-button:hover {
  cursor: pointer;
  color: rgb(255, 0, 0);
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		155

```

.graph-wrapper{
  display: flex;
  flex-direction: column;
  align-items: center;
  padding: 30px;
  background-color: #f5f7fa;
  border-radius: 12px;
  height: fit-content;
  width: fit-content;
  gap: 10px;
}

.card {
  background-color: white;
  border-radius: 12px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.05);
  padding: 24px;
  display: flex;
  flex-wrap: wrap;
  gap: 24px;
  flex-direction: row;
  justify-content: space-around;
  /* flex: 1; */
  width: 100%;
  margin-bottom: 32px;
  box-sizing: border-box;
}

.button-group button {
  margin-right: 10px;
  margin-bottom: 10px;
  padding: 10px 16px;
  border-radius: 6px;
  border: 1px solid #ccc;
  background-color: #fafafa;
  cursor: pointer;
  transition: background-color 0.2s;
}

.button-group button:hover {
  background-color: #eee;
}

.menu-and-logo{
  display: flex;
  align-items: center;
  flex-direction: column;
  gap: 48px;
}

.menu{
  display: flex;
  flex-direction: column;
  overflow: hidden;
  position: relative;
  transition: all .3s ease;
  gap: 8px;
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		156

```

.menu .menu-item{
  display: flex;
  background-color: #f5f7fa;
  align-items: center;
  padding: 10px 22px;
  gap: 8px;
  transition: all .3s ease;
  border-radius: 4px;
  white-space: nowrap;
  cursor: pointer;
}
.menu .menu-item:hover{
  background: rgba(0, 0, 0, 0.10);
  box-shadow: 0px 1px 5px 0px rgba(0, 0, 0, 0.2);
}

.item-users.active, .item-individual.active, .item-clubs.active{
  background: rgba(255, 182, 0, 1);
  box-shadow: 0px 3px 1px -2px rgba(0, 0, 0, 0.2), 0px 2px 2px 0px rgba(0, 0, 0, 0.14),
0px 1px 5px 0px rgba(0, 0, 0, 0.12);
  color: white;
}

.item-users-arrow.active, .item-individual-arrow.active, .item-clubs-arrow.active{
  transform: rotate(180deg);
  filter: invert(1);
}

.menu .menu-item > .title{
  font-style: normal;
  font-weight: 500;
  font-size: 16px;
  line-height: 26px;
  text-transform: none;
}

.phone, .sign-up{
  display: flex;
  gap: 8px;
  border-radius: 4px;
  padding: 8px 11px;
  font-family: 'Montserrat';
  font-style: normal;
  font-weight: 500;
  font-size: 16px;
  line-height: 26px;
}

.phone > img, .sign-up > img{
  width: 24px;
  height: 24px;
}

.social-and-sign{
  display: flex;
  align-items: center;
  gap: 16px;
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		157

```

}

.socials{
  display: flex;
  gap: 20px;
}

.socials > .social{
  width: 28px;
  height: 28px;
}

.sort-container{
  padding: 10px;
  margin-left: 5px;
  border-radius: 8px;
  transition: all .3s ease;
}

.sort-container:hover{
  background: #0000002c;
  cursor: pointer;
}

.sort-container.active{
  background: #D9F1FF;;
}

.sort-container > .sort{
  width: 18.12px;
}

.hello{
  padding: 0px 64.5px;
  display: flex;
  position: relative;
  flex-direction: column;
}

.main-block{
  display: flex;
  justify-content: space-between;
  gap: 16px;
  width: 100%;
}

.main-block > .content{
  display: flex;
  flex-direction: column;
  margin-top: 52px;
  max-width: 759px;
  gap: 20px;
}

.content > .title{
  font-family: 'Noto Sans';
  font-style: normal;
  font-weight: 700;

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		158

```

font-size: 98px;
line-height: 100px;
text-transform: none;
}

.content > .title > span{
  color: #FFB600;
}

.content > .description{
  font-family: 'Montserrat';
  color: #424242;
  font-weight: 400;
  font-size: 24px;
  line-height: 36px;
}

.buttons{
  margin-top: 30px;
  display: flex;
  gap: 30px;
}

.buttons > .button{
  font-family: 'Montserrat';
  font-weight: 500;
  border-radius: 4px;
  padding: 12px 22px;
  color: white;
  text-transform: uppercase;
  box-shadow: 0px 3px 1px -2px #00000033, 0px 2px 2px 0px #00000024, 0px 1px 5px 0px
#0000001F;
}

.button.actual{
  background: #095FCC;
}

.button.all-courses{
  background: #FFB600;
}

.hello-img-right{
  margin-top: -100px;
  width: 1016px;
}

.pages-list{
  display: flex;
  gap: 8px;
  justify-content: center;
  width: 100%;
  height: 16px;
  margin-top: -120px;
}

.pages-list > .page{
  width: 16px;

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		159

```

height: 16px;
border-radius: 50%;
}

.pages-list > .page.active{
background: #FFB600;
}

.pages-list > .page.unactive{
background: #131E29;
}

.img-arrow-down-container{
margin-top: 35px;
margin-bottom: 35px;
width: 100%;
display: flex;
justify-content: center;
}

.arrow-down-continue{
width: 37px
}

.custom-container {
display: flex;
justify-content: center;
align-items: center;
position: relative;
border-right: 2px solid rgba(0, 0, 0, 0.267);
width: 38px;
height: 38px;
cursor: pointer;
user-select: none;
}

.custom-container input {
position: absolute;
opacity: 0;
cursor: pointer;
}

.checkmark {
position: absolute;
inset: 0;
transform: translate(50%, 50%);
height: 15px;
width: 15px;
background-color: transparent;
border: 2px solid rgba(0, 0, 0, 0.6);
border-radius: 4px;
}

.custom-container input:checked + .checkmark::after {
content: '';
position: absolute;
left: 4px;
top: 2px;
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		160

```

width: 5px;
height: 7px;
border: solid #FFB600;
border-width: 0 2px 2px 0;
transform: rotate(45deg);
}

.instrument-panel{
display: flex;
justify-content: space-between;
}

.filter{
display: flex;
gap: 50px;
align-items: center;
}

.instruments{
display: flex;
gap: 50px;
}

.main-instruments{
display: flex;
gap: 10px;
align-items: center;
}

.instrument-container{
border-radius: 4px;
padding: 4px;
background: #ECFCFF;
transition: all .3s ease;
}

.instrument-container > .instrument-icon{
width: 20px;
height: 20px;
}

.instrument-container:hover, .file-intsrument:hover, .file-icon-add:hover{
cursor: pointer;
background: #D9F1FF;
}

.coolinput {
display: flex;
flex-direction: column;
width: fit-content;
position: static;
max-width: 240px;
}

.coolinput label.text {
font-size: 0.75rem;
color: #818CF8;
font-weight: 700;
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		161

```

position: relative;
top: 0.5rem;
margin: 0 0 0 7px;
padding: 0 3px;
background: #e8e8e8;
width: fit-content;
}

.coolinput input[type=text].input {
padding: 11px 10px;
font-size: 0.75rem;
border: 2px #818CF8 solid;
border-radius: 5px;
background: #e8e8e8;
}

.coolinput input[type=text].input:focus {
outline: none;
}

.additional-menu{
position: absolute;
display: flex;
flex-direction: column;
/* flex-wrap: wrap; */
border-radius: 12px;
z-index: 10;
transform: translateY(-40px);
visibility: hidden;
box-shadow: 0px 4px 15px 0px rgba(112, 128, 150, 0.2);
opacity: 0;
transition: all .3s ease;
gap: 30px;
top: 0;
left: 0;
padding: 33px;
background: white;
}

.additional-menu-container{
position: absolute;
width: 100%;
height: 100%;
visibility: hidden;
opacity: 0;
transition: all .3s ease;
}

.additional-menu-container.active{
visibility: visible;
opacity: 1;
}

.additional-menu.active{
visibility: visible;
opacity: 1;
transform: translateY(0px);
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		162

```

.additional-menu > .add-menu-item{
  display: flex;
  flex-direction: column;
  gap: 24px;
  padding: 24px;
  border: 1.5px solid rgba(255, 255, 255, 1);
  border-radius: 8px;
  width: 500px;
  flex: 0 1 auto;
  box-shadow: 0px 4px 15px 0px rgba(112, 128, 150, 0.2);
}

.add-menu-item > .title{
  display: flex;
  font-family: 'Montserrat';
  font-weight: 600;
  line-height: 100%;
  font-size: 16px;
  letter-spacing: 0%;
  align-items: center;
  text-transform: uppercase;
  gap: 16px;
}

.content-item{
  padding: 8px 0px;
  display: flex;
  justify-content: space-between;
  cursor: pointer;
}

.title > .title-img, .group-title > .title-img{
  width: 20px;
  height: 20px;
}

.decline-all-filters{
  width: 28px;
  height: 28px;
}

.group-title{
  display: flex;
  gap: 10px;
  align-items: center;
  font-family: 'Montserrat';
  font-weight: 600;
  line-height: 100%;
  font-size: 16px;
  letter-spacing: 0%;
}

.group-title > .title{
  max-width: 320px;
  transition: all .3s ease;
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		163

```

.content-item > .status{
  padding: 3px 4px;
  border: 1px solid rgba(9, 95, 204, 1);
  color: rgba(9, 95, 204, 1);
  border-radius: 100px;
  display: flex;
  align-items: center;
  justify-content: center;
  color: rgba(9, 95, 204, 1);
  font-family: 'Montserrat';
}

.content-item > .status::before{
  content: 'Йде набір';
}

.green{
  color: rgba(51, 188, 157, 1);
}

.blue{
  color: rgba(9, 95, 204, 1);
}

.purple{
  color: rgba(81, 81, 247, 1)
}

.yellow{
  color: rgba(255, 182, 0, 1)
}

.red{
  color: rgba(248, 106, 133, 1)
}

.lightblue{
  color: rgba(45, 158, 199, 1)
}

.icon-wrapper {
  display: inline-block;
  cursor: pointer;
  transition: transform 0.3s ease;
}

.icon-path {
  transition: fill 0.3s ease;
}

.content-item:hover .icon-path {
  fill: #095FCC;
}

.content-item:hover .icon-wrapper{
  transform: scale(1.5);
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		164

```

.icon-path{transition: all .3s ease;}

.content-item:hover .group-title > .title{
  transform: translateX(10px);
}

.decline-all-filters-container{
  padding: 5px;
  border-radius: 8px;
}

.decline-all-filters-container:hover{
  background: #0000002c;
}

.additional-menu-mobile{
  position: absolute;
  top: 0px;
  right: 0px;
  left: 0px;
  padding: 10px;
  display: flex;
  align-items: flex-end;
  flex-direction: column;
  gap: 10px;
  background: white;
  border-radius: 0px 0px 8px 8px;
  z-index: 999;
  visibility: hidden;
  opacity: 0;
  transform: translateY(-10px);
  transition: all .3s ease
}

.additional-menu-mobile.active{
  visibility: visible;
  opacity: 1;
  transform: translateY(0px);
}

.file-name-and-edit-field{
  width: 100%;
  display: flex;
  position: relative;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
}

.file-name-and-edit-field > .edit-field, .bulk-edit-field {
  text-align: center;
  background: transparent;
  display: none;
  border: none;
  border-bottom: 1px solid #000;
  outline: none;
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмоторя О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		165

```

.bulk-edit-field{
  display: block !important;
  width: 0;
  padding: 0;
  transition: all .3s ease;
}

.bulk-edit-field.active{
  width: 60px;
  padding: 4px;
  margin-right: 5px;
}

.instrument-container.edit{
  display: flex;
}

.file-name-and-edit-field > .file-name{
  display: block;
}

.file-name-and-edit-field > .edit-field.active{
  display: block;
}

.file-name-and-edit-field > .file-name.unactive{
  display: none;
}

```

		Волківський Я.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Чижмотря О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		166