

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:

**«Інформаційна система онлайн-менеджменту
освітніх процесів Digital University UA: модулі
"Обхідні листи" та "Студмістечко"»**

Виконав студент групи ІПЗ-22-1
ЛАГОВСЬКИЙ Адам Віталійович

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЄФРЕМОВ Юрій Миколайович

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

«27» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЛАГОВСЬКИЙ Адам Віталійович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Інформаційна система онлайн-менеджменту освітніх процесів Digital University UA: модулі "Обхідні листи" та "Студмістечко"»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **розробка системи управління освітніми процесами в університеті, що покращить роботу його адміністрації та надасть функціонал для автоматизації та цифровізації рутинних завдань.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т. А.	27.02.2026р.	27.02.2026р.
2	Вакалюк Т. А.	24.04.2026р.	24.04.2026р.
3	Вакалюк Т. А.	02.05.2026р.	02.05.2026р.

РЕФЕРАТ

Кваліфікаційна робота представляє програмний комплекс, розроблений для онлайн-системи управління освітніми процесами «Digital University UA». Практична частина охоплює розробку модулів «Обхідні листи» та «Студмістечко». Теоретична частина викладена у пояснювальній записці обсягом 101 сторінок, яка містить 27 ілюстрацій. Розроблені модулі охоплюють широкий спектр бізнес-логіки університету.

Модуль «Обхідні листи» відповідає за автоматизацію процесу зняття з обліку при відрахуванні чи випуску здобувачів освіти, а також при звільненні працівників, що також включає формування електронних обхідних листів. Реалізовано функціонал для дистанційного погодження документів відповідальними особами, а також оперативний перегляд актуального статусу закриття обхідних листів.

Модуль «Студмістечко» забезпечує комплексне управління житловим фондом університету та процесом поселення, відображаючи статистику доступного житлового фонду, стан кімнат та рух контингенту мешканців гуртожитків. Реалізовано функціонал для управління житловим фондом, вільними місцями та заселенням студентів.

У ході виконання роботи було визначено головні завдання розробки, проведено аналіз існуючих аналогів інформаційних систем та підібрано стек технологій, необхідний для реалізації поставлених вимог до програмного забезпечення.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, ОБХІДНИЙ ЛИСТ, СТУДМІСТЕЧКО, ЖИТЛОВИЙ ФОНД, МОНОЛІТНА АРХІТЕКТУРА, PHP, Yii2, MYSQL, JAVASCRIPT, DOCKER.

					ІПЗ.КР.Б - 121- 26 - ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Інформаційна система онлайн-менеджменту освітніх процесів Digital University UA: модулі "Обхідні листи", та "Студмістечко"	Лім.	Арк.	Аркуші
Розроб.		Лаговський А. В.					2	17
Перевір.		Вакалюк Т. А.						
Керівник								
Н. контр.								
Зав. каф.						Житомирська політехніка, група ІПЗ-22-1		

ABSTRACT

This thesis presents a software package developed for the “Digital University UA” online educational management system. The practical component covers the development of the “Clearance Sheets” and “Student Campus” modules. The theoretical part is presented in a 101-page explanatory note containing 27 illustrations. The developed modules cover a wide range of the university’s business logic.

The “Clearance Sheets” module is responsible for automating the process of deregistration upon expulsion or graduation of students, as well as upon the dismissal of employees, which also includes the generation of electronic circular letters. Functionality has been implemented for remote approval of documents by responsible persons, as well as operational monitoring of the current completion status.

The “Campus” module provides comprehensive management of the university’s housing stock and the check-in process, displaying statistics on available dormitory accommodation, room conditions, and changes in the dormitory resident population. Functionality has been implemented for managing the housing stock, available spaces, and student check-ins.

During the course of the work, the main development tasks were identified, an analysis of existing similar information systems was conducted, and a technology stack was selected that is necessary to meet the software requirements.

KEYWORDS: BACKEND, INFORMATION SYSTEM, STUDENT CAMPUS, HOUSING STOCK, MONOLITHIC ARCHITECTURE, PHP, YII2, JS, LDAP.

					ІПЗ.КР.Б - 121- 26 - ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Інформаційна система онлайн-менеджменту освітніх процесів Digital University UA: модулі "Обхідні листи", та "Студмістечко"	Літ.	Арк.	Аркуші
Розроб.	Лаговський А. В.						2	17
Перевір.	Вакалюк Т. А.					Житомирська політехніка, група ІПЗ-22-1		
Керівник								
Н. контр.								
Зав. каф.								

ЗМІСТ

ВСТУП.....	6
РОЗДІЛ 1. АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ.....	8
1.1. Постановка задачі.....	8
1.2. Аналіз аналогів програмного продукту.....	11
1.3. Вибір архітектури системи.....	16
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	20
Висновки до першого розділу.....	24
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ОНЛАЙН-МЕНЕДЖМЕНТУ ОСВІТНІХ ПРОЦЕСІВ.....	27
2.1. Визначення варіантів використання та структури системи.....	27
2.2. Обґрунтування вибору системи управління базами даних.....	31
2.3. Розробка бази даних системи.....	35
2.4. Реалізація функціоналу модулів «Обхідні листи» та «Студмістечко».....	39
Висновки до другого розділу.....	54
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ОНЛАЙН-МЕНЕДЖМЕНТУ ОСВІТНІХ ПРОЦЕСІВ.....	56
3.1. Архітектура розгортання системи.....	56
3.2. Структура інтерфейсу застосунку.....	59
3.3. Тестування роботи розробленого застосунку.....	68
Висновки до третього розділу.....	73
ВИСНОВКИ.....	75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	77
ДОДАТКИ.....	80
Додаток А.....	81

ВСТУП

В епоху загальної цифровізації українські університети змушені кардинально переосмислювати підходи не лише до організації навчального процесу, а й до управління власною адміністративною та інфраструктурною діяльністю. Глобальний перехід до цифрового суспільства, прискорений пандемією та подальшим поширенням дистанційного й змішаного навчання, оголив системні вади традиційного паперового адміністрування університетів. Застарілі підходи до ведення документообігу довели свою критичну неефективність: вони унеможливають оперативний аналіз даних, багаторазово збільшують ризик виникнення помилок та суттєво ускладнюють доступ до актуальної інформації в режимі реального часу. В умовах, коли студент або викладач може перебувати за сотні кілометрів від університету, необхідність фізичної присутності для підписання паперових документів суттєво ускладнює виконання адміністративних процедур [3, 9].

Актуальність даної роботи зумовлена стрімким зростанням потреби у спеціалізованих цифрових рішеннях для закладів вищої освіти, які б охоплювали не лише навчальну, а й адміністративно-господарську сферу діяльності університету. У багатьох закладах вищої освіти окремі адміністративні процеси досі виконуються із застосуванням паперових документів або розрізнених цифрових засобів для управління такими базовими процесами, як оформлення обхідних листів при відрахуванні або випуску здобувачів освіти та адміністрування студентського житлового фонду. Наслідком цього є хронічне перевантаження адміністративного персоналу рутинними операціями, накопичення помилок через ручне введення даних, неможливість здійснення централізованого моніторингу та відсутність прозорості у розподілі ресурсів закладу. Впровадження спеціалізованих автоматизованих систем управління (АСУ) є сьогодні не технічним нововведенням, а стратегічною необхідністю - це ключовий інструмент для забезпечення прозорості, скорочення операційних витрат та підвищення загальної якості академічних і адміністративних послуг [3, 6, 9].

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

Метою кваліфікаційної роботи є проєктування та реалізація у складі інформаційної системи «Digital University UA» модулів «Обхідні листи» та «Студмістечко», призначених для автоматизації процедури зняття з обліку здобувачів освіти і працівників, а також управління житловим фондом студентських гуртожитків. Розроблені компоненти мають автоматизувати рутинні завдання адміністративного персоналу, налагодити ефективну цифрову взаємодію між студентами, відповідальними особами підрозділів та адміністрацією, а також надати керівництву закладу зручні інструменти для оперативного моніторингу та аналітики.

Для досягнення поставленої мети визначено такі завдання. По-перше, провести аналіз наявних ІТ-рішень для освітнього менеджменту, визначивши їхні переваги та недоліки з точки зору застосування для автоматизації адміністративних процесів вітчизняного ЗВО. По-друге, спроектувати архітектуру розроблюваних модулів та визначити їхню взаємодію в межах єдиної платформи. По-третє, розробити оптимальну логічну та фізичну структуру бази даних, що забезпечить цілісність та узгодженість даних між компонентами системи. По-четверте, реалізувати бізнес-логіку обох модулів на серверній та клієнтській стороні, а також розгорнути вебзастосунок як єдину точку доступу до системи.

Об'єктом дослідження є процеси автоматизації адміністративної діяльності закладу вищої освіти, зокрема оформлення обхідних листів та управління житловим фондом студентських гуртожитків.

Предметом дослідження є алгоритми, програмні фреймворки та інформаційні технології, що застосовуються для створення сучасних цифрових освітніх платформ із урахуванням специфіки нормативних вимог та бізнес-процесів вітчизняних університетів.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

РОЗДІЛ 1. АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ

1.1. Постановка задачі

Сучасні заклади вищої освіти функціонують в умовах стрімкої цифрової трансформації, яка охоплює всі рівні інституційного управління - від навчального процесу до адміністративно-господарської діяльності. Інформаційні системи, що забезпечують ефективне управління інфраструктурою університету, перестали бути опцією та стали стратегічною необхідністю. Постійний рух контингенту здобувачів освіти та працівників, що включає поселення, відрахування, переведення та випуск, вимагає наявності надійних цифрових інструментів, здатних забезпечити комплексний моніторинг ресурсів установи в режимі реального часу. Відсутність таких систем неминуче призводить до зниження ефективності роботи адміністративного апарату та погіршення якості послуг, що надаються здобувачам освіти.

Особливо гострою проблемою залишається цифровізація двох ключових адміністративних процесів. Перший із них — процедура зняття з обліку через оформлення так званого «обхідного листа». Цей документ підтверджує відсутність у особи, яка відраховується або звільняється, будь-яких заборгованостей перед структурними підрозділами університету: бібліотекою, гуртожитком, деканатом, кафедрами та господарськими службами. Другий — управління житловим фондом студентських гуртожитків, що охоплює облік вільних місць, оформлення поселення, переселення та виселення мешканців, а також моніторинг загального стану номерного фонду.

Значна кількість вітчизняних закладів вищої освіти досі застосовує розрізнені або повністю паперові підходи до вирішення цих завдань. Типовий сценарій виглядає так: студент або співробітник, якого відраховують чи звільняють, отримує фізичний аркуш паперу й особисто обходить кожен структурний підрозділ університету для отримання підпису відповідальної

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

особи. Цей процес є надзвичайно трудомістким і залежить від фізичної присутності кожного з учасників, що стає критичним бар'єром в умовах дистанційного навчання або роботи. Він перевантажує адміністративний персонал, призводить до утворення черг, спричиняє тривалу бюрократичну тяганину і, зрештою, суттєво знижує задоволеність академічної спільноти якістю внутрішніх сервісів. Окрім незручностей для кінцевого користувача, ця система унеможлиблює оперативне відстеження заборгованостей у масштабах усієї установи: адміністрація не має централізованого, актуального і цілісного уявлення про стан виконання обхідних листів у будь-який момент часу.

Аналогічна проблема існує у сфері управління гуртожитками. Паперові журнали поселення, розрізнені електронні таблиці та відсутність єдиної бази даних мешканців унеможлиблюють ефективне управління наявним номерним фондом. Комендант не має можливості оперативно отримати актуальні відомості про зайнятість кімнат, корпусів і окремих ліжко-місць. Особливо гострою ця проблема стає в пікові періоди — під час масового заселення студентів на початку навчального року, коли тисячі зарахованих студентів одночасно потребують місця у гуртожитку, або під час масового закриття обхідних листів у сезон випуску. Відсутність інтегрованого рішення призводить до помилок у розподілі місць, конфліктів при переселенні та неможливості швидкого реагування на зміну обставин.

Для подолання зазначених проблем та задоволення сучасних вимог до цифрового університету в рамках даної кваліфікаційної роботи розроблено та впроваджено два спеціалізовані модулі - «Обхідні листи» та «Студмістечко» - у складі існуючої інформаційної системи «Digital University UA». Обидва модулі проєктувалися як органічна частина єдиної платформи, що дозволяє їм обмінюватися даними та забезпечувати наскрізну автоматизацію взаємопов'язаних процесів.

Модуль «Обхідні листи» розроблений, щоб забезпечити повноцінну автоматизацію процедури зняття з обліку. Він охоплює весь цикл — від автоматичної генерації електронного обхідного листа в момент ініціювання

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

відрахування, випуску чи звільнення до фінального підтвердження факту відсутності будь-яких заборгованостей з боку відповідальних підрозділів. Принципово важливою функцією є реалізація дистанційного погодження: замість особистого обходу кабінетів відповідальна особа отримує доступ до власного розділу системи, де може переглянути список студентів або співробітників, що проходять процедуру, та проставити електронну відмітку про відсутність заборгованостей або зафіксувати наявність боргу із вказівкою його причини. Керівництво університету та деканати отримують актуальний зведений звіт після оновлення даних у системі, що дозволяє оперативно контролювати стан закриття обхідних листів за всіма підрозділами і по кожному здобувачу окремо. Система також зберігає повну архівну історію за кожним навчальним роком, що забезпечує можливість відновлення інформації у разі спірних ситуацій.

Не менш важливою складовою розробки є модуль «Студмістечко», призначений для комплексного адміністрування житлового фонду університету. Система організована відповідно до реальної фізичної ієрархії: гуртожиток - корпус - кімната - ліжко-місце. На кожному рівні цієї ієрархії відповідальні особи (коменданти, адміністратори) мають доступ до актуальної інформації про поточний стан фонду. Модуль реалізує повний цикл управління мешканцями: оформлення поселення нового мешканця, переселення між кімнатами в межах одного або різних гуртожитків, обмін кімнатами між двома мешканцями, а також виселення із зберіганням повної хронології записів. Всі операції, що передбачають одночасну зміну декількох записів у базі даних, виконуються атомарно завдяки механізму транзакцій, що гарантує цілісність даних навіть у разі збою. Окремо реалізовано статистичні звіти, що відображають загальний відсоток заповненості кожного гуртожитку, кількість вільних місць у розрізі за типом кімнати (чоловіча/жіноча) та зручний пошук конкретного мешканця за ПІБ з відображенням усіх його поточних та архівних даних щодо проживання.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Впровадження обох модулів у складі єдиної платформи «Digital University UA» вирішує системну проблему інформаційної розрізненості, що характерна для традиційних підходів. Інтеграція між модулями дозволяє, наприклад, автоматично включати до маршруту обхідного листа підрозділ гуртожитку для тих студентів, які є дійсними мешканцями, або не включати його для тих, хто у гуртожитку не проживає. Це виключає ручне налаштування кожного конкретного випадку та мінімізує вплив людського фактора на коректність процедури зняття з обліку.

Таким чином, постановка задачі даної кваліфікаційної роботи полягає у проєктуванні, розробці та інтеграції двох взаємопов'язаних програмних модулів, що автоматизують критично важливі адміністративні процеси сучасного університету — управління обхідними листами та житловим фондом — і тим самим сприяють підвищенню загальної операційної ефективності закладу, прозорості внутрішніх процедур та якості обслуговування здобувачів освіти і персоналу.

1.2. Аналіз аналогів програмного продукту

Moodle - це одна з найвідоміших платформ управління навчанням (LMS) із відкритим вихідним кодом. Її головна мета полягає у забезпеченні взаємодії між викладачами та здобувачами освіти шляхом створення електронних курсів, проведення тестувань та ведення журналів успішності. Завдяки модульній архітектурі система легко розширюється за допомогою плагінів і підтримує міжнародні стандарти e-learning (SCORM, xAPI). Платформа легко інтегрується з корпоративними інструментами зв'язку, такими як Zoom, Microsoft Teams та Google Workspace [16].

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

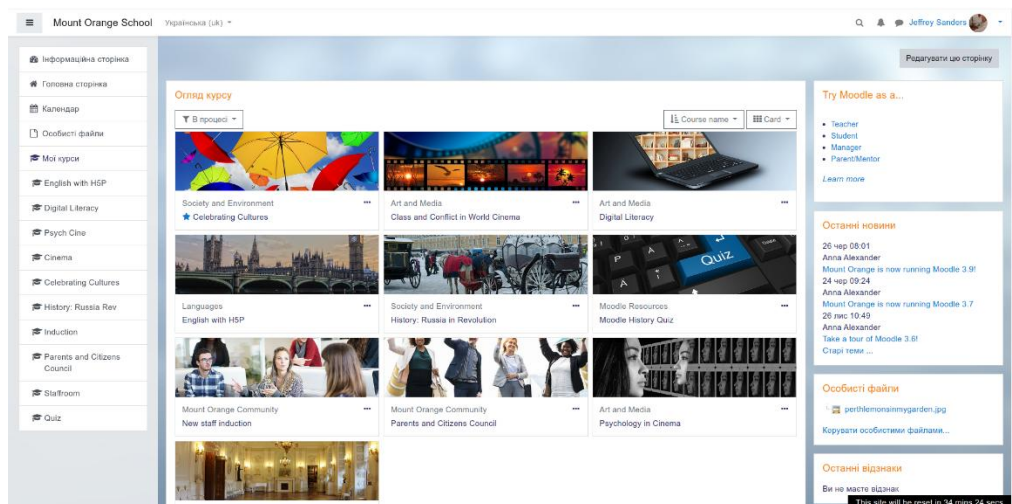


Рисунок 1.1 Платформа Moodle

Перевагами цієї системи є безкоштовна ліцензія, відкритий код, наявність величезної міжнародної спільноти розробників, кросплатформність та висока гнучкість у налаштуванні освітнього процесу.

Головною проблемою Moodle є вузька спеціалізація на процесі навчання. Платформа практично не має вбудованих інструментів для глибокого кадрового діловодства (управління контрактами, табелювання працівників) та формування комплексної адміністративної звітності. Водночас стандартна конфігурація Moodle орієнтована переважно на організацію навчального процесу. Вона не містить спеціалізованого функціоналу для формування та погодження обхідних листів, керування ліжко-місцями в гуртожитках і ведення історії проживання мешканців. Реалізація таких можливостей потребувала б створення додаткових плагінів або інтеграції із зовнішньою інформаційною системою [16].

Canvas LMS, розроблена компанією Instructure, є сучасним стандартом хмарних освітніх рішень, який активно використовується в провідних університетах світу. Система фокусується на високій швидкодії, доступності з мобільних пристроїв та інтуїтивно зрозумілому інтерфейсі (UI/UX). Платформа надає розгорнутий API для підключення зовнішніх сервісів і автоматично синхронізує дані в режимі реального часу [12].

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

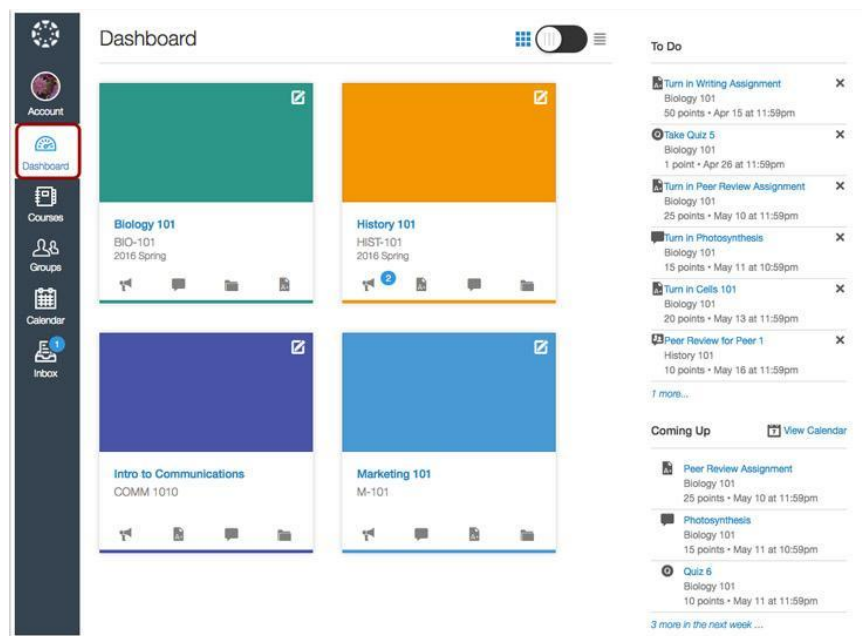


Рисунок 1.2 Система Canvas LMS

Перевагами цієї системи є відсутність потреби у підтримці власних серверів (SaaS-модель), сучасний інтерфейс, висока надійність хмарної інфраструктури, якісна екосистема інтеграцій зі сторонніми сервісами.

Вагомим бар'єром є висока вартість ліцензування. Як і Moodle, Canvas є інструментом для навчання, а не для адміністрування університету: тут відсутні повноцінні модулі для ведення профілів науково-педагогічних працівників чи розширеної статистики контингенту. Canvas LMS має відкритий вихідний код і може використовуватися як у хмарному середовищі, так і за умови самостійного розгортання. Водночас основне призначення системи полягає в організації навчання. Її стандартний функціонал не охоплює керування житловим фондом, поселення й переселення мешканців та погодження електронних обхідних листів [12].

OpenEduCat - це комплексне модульне рішення з відкритим кодом, побудоване на базі потужної ERP-системи Odoo. На відміну від класичних LMS, цей продукт орієнтований саме на управління бізнес-процесами університету. Система охоплює фінансовий облік, управління інфраструктурою, адміністрування студентів та має базовий функціонал для кадрового менеджменту [21].

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

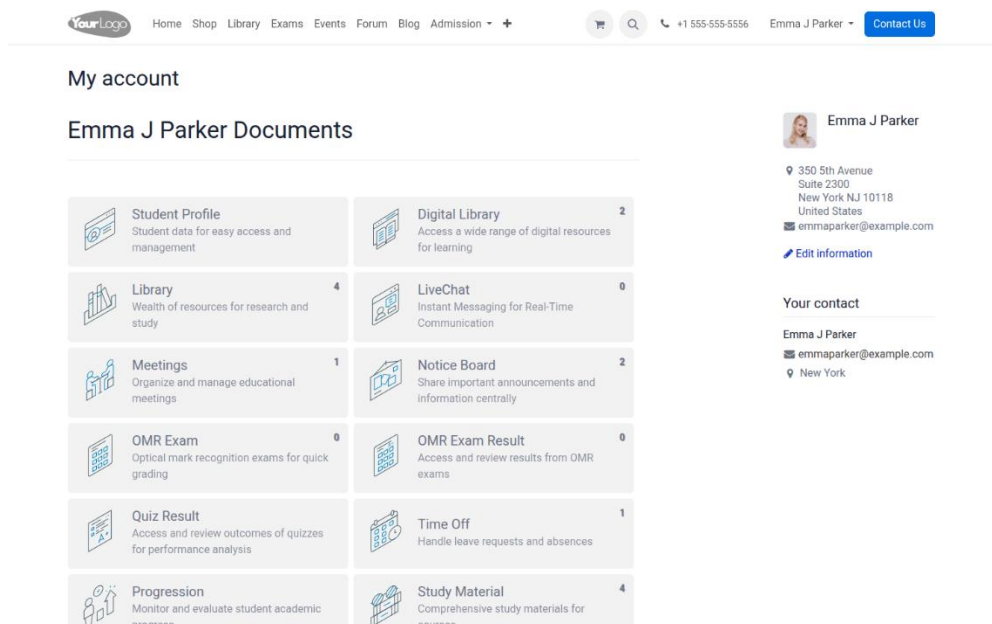


Рисунок 1.3 Система OpenEduCat

Перевагами цієї системи є наявність потужних інструментів аналітики, можливість розгортання на власних серверах, єдина база даних для всіх підрозділів закладу, автоматизація адміністративних завдань.

Адаптація OpenEduCat до специфічних адміністративних процедур конкретного закладу вищої освіти потребує налаштування або розроблення додаткових модулів. Стандартний набір компонентів не забезпечує повної відповідності процесам погодження обхідних листів та деталізованого обліку гуртожитків, кімнат, ліжко-місць і хронології проживання [21].

Unit4 Student Management - це масштабна європейська ERP-система нового покоління, створена спеціально для закладів вищої освіти. Її ключова відмінність полягає у зміщенні фокусу з процесу навчання на адміністрування, управління людським капіталом (HR) та глибоку аналітику (Business Intelligence). Платформа пропонує потужні інструменти для автоматизації роботи відділу кадрів, відстеження статистики вступних кампаній та руху контингенту [29, 38].

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

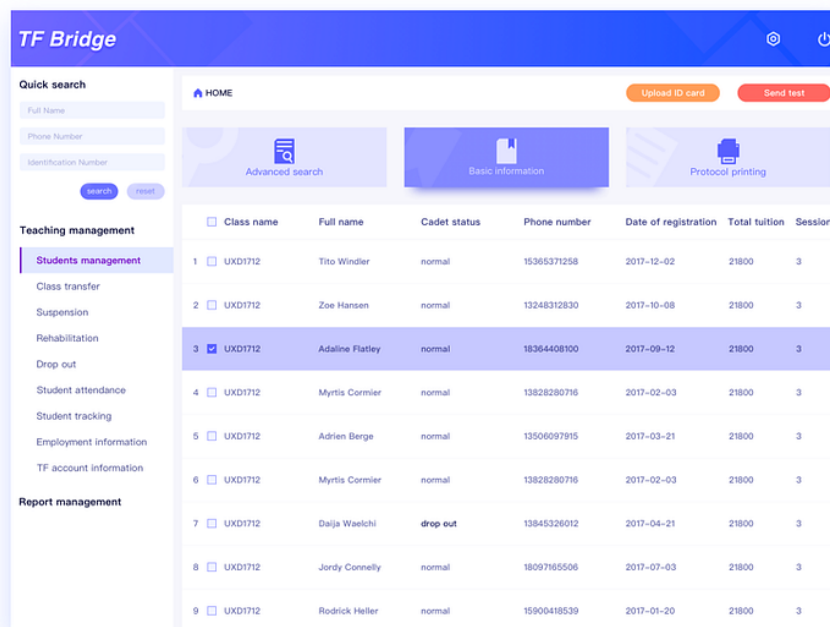


Рисунок 1.4 Система Unit4 Student Management

Перевагами цієї системи є винятковий рівень автоматизації кадрових процесів (від автоматичного генерування контрактів до розрахунку складних схем навантаження НПП). Система містить вбудовані потужні аналітичні дашборди для керівництва, які візуалізують ключові показники ефективності ЗВО у реальному часі. Virізняється високою масштабованістю, стабільністю при екстремальних навантаженнях та відповідає найвищим європейським стандартам корпоративної кібербезпеки.

Головною та майже нездоланною перешкодою є астрономічна вартість ліцензування та технічної підтримки, яка робить систему недоступною для переважної більшості вітчизняних університетів. Платформа має повністю закритий вихідний код, що унеможлиблює самостійне доопрацювання функціоналу силами університетських ІТ-відділів. Процес імплементації такої системи може тривати роками, а її жорстка європейська логіка бізнес-процесів надзвичайно важко та дорого піддається кастомізації під унікальні реалії українського освітнього документообігу та специфічні вимоги вітчизняних нормативно-правових актів [29, 38].

Проведений аналіз демонструє, що існуючі рішення поділяються на дві категорії. Аналіз Moodle, Canvas LMS, OpenEduCat та Unit4 Student

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

Management показав, що наявні програмні продукти охоплюють окремі складові діяльності закладів вищої освіти. Moodle і Canvas LMS орієнтовані переважно на організацію навчального процесу, тоді як OpenEduCat та Unit4 надають ширші засоби для керування адміністративними й організаційними процесами. Водночас жодна з розглянутих систем у стандартній конфігурації не забезпечує одночасного керування електронними обхідними листами та деталізованого обліку гуртожитків, кімнат, ліжко-місць і мешканців. Відповідно, розробка власних спеціалізованих модулів «Обхідні листи» та «Студмістечко» у складі системи «Digital University UA» є науково та практично обґрунтованим завданням.

1.3. Вибір архітектури системи

У сучасних інформаційних системах закладів вищої освіти фундаментальне значення має грамотне проєктування архітектури програмного забезпечення. Це є запорукою стабільності, безпеки та здатності системи до подальшого масштабування. Постійне зростання обсягів даних та кількості активних користувачів — здобувачів освіти, працівників, адміністративного персоналу — вимагає ефективного управління кодовою базою та логікою взаємодії між компонентами. Правильний архітектурний підхід значно спрощує подальший розвиток та технічну підтримку цифрового продукту, а неправильний — може стати причиною системних збоїв, непередбачуваної поведінки та колосальних витрат на рефакторинг у майбутньому.

Вибір архітектурного рішення для будь-якої інформаційної системи є комплексним завданням, що не має єдиної універсальної правильної відповіді. Архітектура визначає не лише спосіб організації коду, а й методи розгортання системи, підходи до забезпечення її безпеки, масштабованості та відмовостійкості. Сучасна індустрія програмного забезпечення пропонує декілька усталених архітектурних підходів, кожен із яких має власну сферу ефективного застосування. Серед найбільш поширених — монолітна архітектура, сервіс-орієнтована архітектура (SOA) та мікросервісна

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

архітектура. Розуміння сильних і слабких сторін кожного підходу є необхідною умовою для обґрунтованого проєктного рішення.

Традиційним та найбільш розповсюдженим підходом є використання монолітної архітектури, за якої всі функціональні компоненти — серверна логіка, взаємодія з базою даних, модулі управління житловим фондом («Студмістечко») та процесами зняття з обліку («Обхідні листи») — об'єднані в єдину цілісну структуру. Така модель забезпечує відносну простоту розробки та високу швидкість початкового розгортання. Усі складові системи функціонують у спільному середовищі, що робить процеси налагодження, тестування та моніторингу максимально прозорими. Крім того, завдяки відсутності мережових затримок між внутрішніми компонентами, моноліт демонструє високу продуктивність під час виконання комплексних запитів, а єдина кодова база суттєво полегшує управління версіями засобами систем контролю версій (VCS). Розробник отримує повну та цілісну картину системи без необхідності одночасно утримувати в голові десятки незалежних сервісів та протоколів їх взаємодії.

Внутрішня організація монолітної системи, як правило, будується на основі архітектурного патерну MVC (Model-View-Controller). Цей патерн розподіляє відповідальність між трьома рівнями: шар моделей інкапсулює бізнес-логіку та правила взаємодії з базою даних, шар контролерів приймає запити від клієнта та керує потоком їх обробки, а шар представлень відповідає за формування кінцевої відповіді. Такий розподіл забезпечує логічну структурованість коду навіть у рамках єдиного застосунку та дозволяє командам розробників паралельно працювати над різними шарами без прямих конфліктів. Фреймворк Yii2, обраний для реалізації даного проєкту, реалізує MVC на глибокому рівні, надаючи вбудовані інструменти для маршрутизації, валідації даних та об'єктно-реляційного відображення через компонент ActiveRecord, що суттєво прискорює розробку [1, 13, 25].

Однак із розширенням функціоналу інформаційної системи монолітна структура може стати надто масивною. Збільшення кількості бізнес-правил

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

підвищує ризик виникнення конфліктів між окремими модулями. Зміни в одній частині коду можуть непередбачувано вплинути на інші компоненти, що ускладнює розробку нового функціоналу та вимагає повного регресійного тестування всієї системи навіть після незначних оновлень. Часом масштабування монолітної системи стає неефективним: якщо лише один модуль зазнає пікового навантаження, доводиться масштабувати весь застосунок цілком, що веде до нераціональних витрат серверних ресурсів.

Проміжним підходом між монолітом та повноцінними мікросервісами є сервіс-орієнтована архітектура (SOA). Вона передбачає виділення окремих функціональних служб, що взаємодіють через стандартизовані протоколи — як правило, SOAP або REST. SOA дозволяє досягти певного рівня незалежності компонентів при збереженні централізованого управління, однак зберігає значну частину складності мікросервісного підходу без усіх його переваг. Для проєктів університетського масштабу SOA нерідко виявляється надмірно ускладненою без відчутного виграшу порівняно з добре структурованим монолітом [1, 5].

Як повноцінна альтернатива моноліту, у сучасній IT-індустрії активно застосовується мікросервісна архітектура. Вона базується на декомпозиції системи на ізольовані невеликі сервіси, кожен з яких виконує власну вузьку бізнес-задачу та розгортається незалежно. Наприклад, у системі «Digital University UA» теоретично можна було б виділити абсолютно незалежні сервіси для обліку кімнат у гуртожитках, обробки заявок на поселення та маршрутизації електронних обхідних листів, що взаємодіяли б між собою через чітко визначені мережеві API. Це дозволяє командам розробників працювати паралельно та незалежно масштабувати лише ті компоненти, які зазнають найбільшого навантаження, — наприклад, виключно модуль поселення в серпні-вересні під час масового заїзду студентів, не торкаючись решти системи.

Попри очевидні переваги, мікросервісний підхід супроводжується значними технічними та інфраструктурними викликами. Він вимагає

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

ретельного планування мережевих з'єднань, впровадження складних систем оркестрації (Kubernetes, Docker Swarm) та розподіленого моніторингу. Розподіленість компонентів неминуче призводить до мережевих затримок, що впливає на загальну швидкодію системи. Особливо гострою проблемою є забезпечення цілісності розподілених транзакцій. Теорема CAP (Consistency, Availability, Partition tolerance) стверджує, що розподілена система не може одночасно гарантувати всі три властивості. Для предметної області, де статус заборгованості студента за гуртожиток має оновлюватися та перевірятися абсолютно синхронно в момент формування обхідного листа, будь-який компроміс у питанні консистентності даних є неприйнятним. Монолітна система з єдиною реляційною базою даних MySQL, на відміну від мікросервісів, повністю задовольняє вимогам ACID і гарантує суворі атомарні транзакції без жодних архітектурних обхідних рішень.

Для реалізації модулів «Обхідні листи» та «Студмістечко» обрано монолітну архітектуру серверного застосунку, побудованого на базі фреймворку Yii2 відповідно до патерну MVC. Обидва модулі використовують спільну кодову базу, єдину систему автентифікації та централізовану реляційну базу даних MySQL [1, 5, 13].

Вибір монолітного підходу зумовлений тісною пов'язаністю даних між модулями. Формування маршруту обхідного листа може залежати від відомостей про проживання здобувача освіти в гуртожитку, а операції поселення, переселення, виселення та виставлення відміток потребують узгодженого оновлення взаємопов'язаних записів. Використання єдиної бази даних дозволяє виконувати такі операції в межах ACID-транзакцій та уникати проблем узгодженості, характерних для розподілених систем.

Клієнт-серверна модель не суперечить монолітній архітектурі. Клієнтська частина виконується у браузері користувача, а серверна бізнес-логіка зосереджена в одному застосунку Yii2, який взаємодіє з базою даних MySQL. Така організація відповідає масштабу розроблюваних модулів, спрощує розгортання, тестування і супровід програмного забезпечення

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Монолітна архітектура в цьому випадку забезпечує централізований контроль над складною бізнес-логікою університету, гарантує узгодженість даних завдяки механізму ACID-транзакцій та робить процес розробки й супроводу максимально ефективним і передбачуваним для команди розробників.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Наступним важливим етапом проєктування стало визначення та обґрунтування технологічного стеку для розробки програмного забезпечення. Вибір конкретних мов програмування, фреймворків та інфраструктурних інструментів є стратегічним рішенням, що безпосередньо впливає на продуктивність команди розробників, довгострокову підтримуваність системи та її здатність відповідати зростаючим вимогам університетського середовища. Кожен інструмент у складі обраного стеку було ретельно проаналізовано з урахуванням специфіки предметної області, наявної інфраструктури та архітектурних рішень, прийнятих на попередньому етапі.

Враховуючи специфіку архітектури вебзастосунків для закладів вищої освіти, базовою серверною мовою програмування було обрано PHP. Це рішення зумовлене рядом вагомих аргументів. По-перше, PHP є однією з найбільш перевірених часом мов для серверної веброзробки: вона живить понад 75% усіх вебсайтів у світі, включаючи такі масштабні платформи, як WordPress та Wikipedia, що підтверджує її здатність витримувати промислові навантаження. По-друге, PHP 8.x, що використовується у проєкті, суттєво відрізняється від ранніх версій завдяки введенню ЛТ-компіляції, типізації аргументів та повернень функцій, іменованих аргументів та атрибутів — що наближає її можливості до строго типізованих мов і значно підвищує надійність написаного коду. По-третє, PHP має розвинену екосистему, вичерпну офіційну документацію та надзвичайно активну міжнародну спільноту, що робить пошук рішень для нестандартних ситуацій швидким і ефективним. Менеджер залежностей Composer дозволяє підключати тисячі

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

перевірених бібліотек, не витрачаючи час на розробку базових утиліт із нуля [4, 19].

Для безпосередньої реалізації бізнес-логіки модулів «Обхідні листи» та «Студмістечко» застосовано високорівневий фреймворк Yii2. Вибір саме цього фреймворку зумовлений кількома ключовими факторами. Yii2 реалізує патерн MVC на рівні архітектурного ядра, що забезпечує чітке розділення відповідальності між компонентами системи і робить кодову базу структурованою та зрозумілою для будь-якого нового розробника, що долучається до команди. Вбудований компонент ActiveRecord надає потужний об'єктно-реляційний інтерфейс для роботи з базою даних: замість написання «сирих» SQL-запитів розробник описує зв'язки між сутностями декларативно, а фреймворк самостійно трансліює ці описи в оптимізовані запити до MySQL. Це принципово важливо для модулів із глибокою ієрархічною структурою даних — зокрема, для опису ланцюжка «гуртожиток — кімната — ліжко-місце — мешканець» у підсистемі «Студмістечко». Вбудований інструмент генерації коду Gii дозволяє автоматично створювати каркаси контролерів, моделей та форм на основі структури бази даних, що значно прискорює розробку рутинних CRUD-операцій. Компонент RBAC (Role-Based Access Control) забезпечує гнучке управління правами доступу різних категорій користувачів — адміністраторів, комендантів, модераторів підрозділів та студентів — без необхідності реалізовувати цю логіку вручну. Нарешті, вбудований механізм валідації моделей перевіряє тип, формат та обов'язковість вхідних значень. Захист від SQL-ін'єкцій забезпечується використанням Query Builder, ActiveRecord і параметризованих запитів, у яких значення передаються окремо від тексту SQL-команди [13, 25, 32].

Як систему управління реляційними базами даних використано MySQL. Це рішення є галузевим стандартом для PHP-застосунків і гарантує стабільність та високу продуктивність під час обробки масивних таблиць із кадровими даними та аналітичними вибірками. MySQL у режимі рушія InnoDB підтримує повноцінні ACID-транзакції та механізм зовнішніх ключів,

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

що є критично важливим для підтримки цілісності ієрархічних залежностей між таблицями обох модулів. Оптимізатор запитів MySQL ефективно обробляє складні конструкції з множинними об'єднаннями таблиць (JOIN), підзапитами та агрегатними функціями, що активно використовуються при формуванні звітів про зайнятість гуртожитків та статус обхідних листів. Для адміністрування бази даних застосовується phpMyAdmin — відкритий вебінтерфейс, що дозволяє виконувати весь спектр операцій зі структурою та даними без необхідності звертатися до командного рядка.

Клієнтська частина системи реалізована з використанням стандартного стеку вебтехнологій: HTML5 для розмітки, CSS3 для стилізації та JavaScript для забезпечення динамічності інтерфейсу. Асинхронна взаємодія з сервером через AJAX-запити дозволяє оновлювати окремі частини сторінки без її повного перезавантаження, що є принципово важливим для операцій, що виконуються часто: виставлення відмітки в обхідному листі, перемикання статусу бронювання ліжко-місця або динамічне завантаження списку вільних кімнат при зміні параметрів фільтра. Для організації повторно використовуваних клієнтських компонентів застосовуються jQuery-плагіни та підключені через Composer AssetBundle-пакети, що дозволяє централізовано керувати версіями клієнтських бібліотек [14].

Для контейнеризації та уніфікації середовища розгортання використано платформу Docker. Конфігурацію компонентів системи визначено у файлі docker-compose.yml. За допомогою Docker Compose запускаються контейнери серверного застосунку Yii2, СУБД MySQL та інші допоміжні сервіси, передбачені фактичною конфігурацією проєкту. Контейнери об'єднано у внутрішню мережу, а для збереження даних MySQL між перезапусками використовується постійний том. Це забезпечує відтворюваність конфігурації та спрощує розгортання системи на різних робочих станціях [2, 20, 35].

Для забезпечення безперебійної та швидкодіючої роботи розроблених програмних модулів було сформовано мінімальні технічні вимоги до апаратного та системного забезпечення сервера. Зазначені вимоги базуються

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

на розрахунку пікового навантаження системи, що виникає в критичні для університету операційні періоди — масове заселення гуртожитків на початку навчального року та масове закриття обхідних листів під час сезону випуску.

Щодо вимог до процесора, сервер повинен мати щонайменше 4 фізичних ядра з базовою тактовою частотою не нижче 3.00 GHz та підтримкою технології багатопотоковості. Це забезпечує ефективну паралельну обробку одночасних HTTP-запитів від великої кількості користувачів: PHP-FPM як менеджер процесів розподіляє вхідні запити між пулом робочих процесів, і наявність достатньої кількості ядер безпосередньо визначає кількість таких процесів, що можуть виконуватися паралельно без деградації продуктивності.

Стосовно оперативної пам'яті, необхідний обсяг становить не менше 32 GB. Це пояснюється тим, що MySQL активно використовує оперативну пам'ять для кешування буферів індексів та часто запитуваних даних (InnoDB Buffer Pool), а PHP-FPM утримує в пам'яті пул активних процесів. Рекомендовано використовувати модулі з підтримкою корекції помилок (ECC), що суттєво підвищує надійність та гарантує цілісність даних під час виконання ресурсомістких операцій, зокрема генерації статистичних звітів та масових вибірок по контингенту мешканців гуртожитків. ECC-пам'ять автоматично виявляє і виправляє одиничні бітові помилки, що є стандартною вимогою для серверного обладнання, де некоректні дані в пам'яті можуть призвести до пошкодження бази даних [7, 15].

Дискова підсистема має бути представлена високошвидкісним твердотільним накопичувачем (SSD) обсягом від 100 GB. Перевага SSD перед традиційними жорсткими дисками (HDD) у контексті серверів баз даних є вирішальною: час випадкового доступу до даних на SSD вимірюється десятками мікросекунд проти мілісекунд на HDD, що безпосередньо впливає на швидкість виконання запитів до MySQL. Крім простору для безпосереднього зберігання бази даних та файлів застосунку, необхідно зарезервувати окремий дисковий ресурс для тимчасового кешування масивних вибірок, автоматичних резервних копій бази даних та системних журналів.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

Щодо операційної системи, рекомендованим середовищем є дистрибутиви сімейства Linux, зокрема Debian-based системи або оптимізований Alpine Linux. Перевага Linux над Windows Server у серверному контексті є очевидною: вища стабільність, нижче споживання ресурсів, відкритий вихідний код та відсутність ліцензійних витрат. Система має підтримувати менеджери управління процесами (systemd або supervisord) для гарантованого автоматичного перезапуску сервісів у разі збою, а також бути сумісною з технологіями контейнеризації Docker. Використання Docker дозволяє упакувати застосунок та всі його залежності в ізольований контейнер, що гарантує ідентичність середовища розробки та виробничого сервера і повністю усуває клас проблем, пов'язаних із розбіжністю версій залежностей.

Мережеве забезпечення сервера повинне гарантувати пропускну здатність каналу на рівні не менше 100 Mbps, а також стабільну внутрішню комунікацію з основним сервером баз даних із мінімальною мережевою затримкою (ping менше 1 ms), що є критично важливим для продуктивності запитів у режимі реального часу. Розміщення вебсервера та сервера баз даних в одній приватній мережі з гарантованою низькою затримкою є стандартною практикою побудови безпечної та продуктивної серверної інфраструктури.

Висновки до першого розділу

У першому розділі було проведено всебічний аналіз предметної області та чітко окреслено мету і ключові завдання кваліфікаційної роботи. Виявлено, що існуючі підходи до управління інфраструктурою закладів вищої освіти часто спираються на розрізнені цифрові рішення або застарілий паперовий документообіг. Формування коректної постановки проблеми стало відправною точкою для проектування модулів «Обхідні листи» та «Студмістечко» інформаційної системи «Digital University UA». Це дозволило встановити пріоритети щодо автоматизації процедур зняття з обліку, дистанційного погодження документів відповідальними підрозділами, а також

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

комплексного управління житловим фондом, розподілу вільних місць та моніторингу процесів поселення.

Дослідження існуючих систем-аналогів (Moodle, Canvas, OpenEduCat, Unit4) підтвердило наукову та практичну доцільність власної розробки. Встановлено, що популярні освітні LMS-системи орієнтовані переважно на навчальний процес і не мають необхідного адміністративного інструментарію для управління гуртожитками чи маршрутизації електронних обхідних листів. Водночас спеціалізовані західні ERP-рішення є економічно недоступними і не враховують специфічних нормативних вимог та бюрократичних процедур українського освітнього простору.

На основі специфіки бізнес-процесів ЗВО було обґрунтовано вибір монолітної архітектури програмного забезпечення. З огляду на надзвичайно тісну зв'язність даних між рухом контингенту, місцем проживання здобувачів, станом їхніх матеріальних чи фінансових заборгованостей та процесом формування обхідних листів, монолітний підхід забезпечує ефективне централізоване управління системою. Це гарантує консистентність реляційної бази даних та дозволяє максимально оптимізувати виконання складних перехресних запитів без мережових затримок, які були б неминучими у мікросервісній архітектурі.

Також здійснено вибір інструментальних засобів: використання мови програмування PHP та високорівневого фреймворку Yii2 забезпечує високу швидкість розробки і безпеку компонентів, а СУБД MySQL гарантує надійність обробки масивних обсягів інформації. Рекомендовані характеристики серверного середовища залежать від кількості одночасних користувачів, обсягу бази даних і складу Docker-контейнерів. Для тестового або пілотного розгортання достатньою є система з багатоядерним процесором, не менше ніж 8 ГБ оперативної пам'яті та SSD-накопичувачем. Остаточні вимоги до середовища експлуатації мають визначатися за результатами навантажувального тестування.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

Таким чином, виконаний системний аналіз і прийняті технічні рішення формують надійну концептуальну основу для практичної реалізації програмного продукту, що забезпечить ефективну цифрову трансформацію управлінських та інфраструктурних процесів університету.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ОНЛАЙН-МЕНЕДЖМЕНТУ ОСВІТНІХ ПРОЦЕСІВ

2.1. Визначення варіантів використання та структури системи

Для забезпечення надійного функціонування модулів «Обхідні листи» та «Студмістечко» в межах інформаційної системи цифрового університету сформульовано низку вимог до розробки. Вони охоплюють як функціональні, так і нефункціональні характеристики, що спрямовані на підтримку високої ефективності, безпеки, масштабованості та сумісності з іншими підсистемами.

Розглянемо основні бізнес-вимоги, які були поставлені для розробленої системи. Обидва модулі створюються з метою автоматизації та цифровізації двох взаємопов'язаних адміністративних процесів університету. Перший — процедура зняття з обліку здобувачів освіти завдяки системі контролю маршруту через різні структурні підрозділи. Другий — комплексне управління житловим фондом студентських гуртожитків. Основною бізнес-ціллю є централізоване управління індивідуальними маршрутами обходу, забезпечення прозорості реєстрації виконання кожного його етапу, а також глибокий моніторинг інфраструктури гуртожитків — від рівня загальної будівлі до конкретного ліжка-місця.

З точки зору архітектурного представлення, система реалізується у вигляді вебзастосунку з монолітною архітектурою. Серверна частина (Backend) побудована на основі PHP-фреймворку Yii2, в якому зосереджена вся бізнес-логіка: управління структурними підрозділами обходу, формування індивідуальних маршрутів студентів, реєстрація відміток проходження та адміністрування користувачів-модераторів. Паралельно на серверній стороні реалізовано багаторівневу ієрархію управління інфраструктурою гуртожитків — від загальних гуртожитків і кімнат до окремих ліжка-місць та даних про їхніх мешканців. Клієнтська частина (Frontend) надає зручні персоналізовані інтерфейси для здобувачів освіти. Уся інформація зберігається в централізованій реляційній базі даних MySQL.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

Адміністрація потребує повноцінної інструментарію для роботи з реєстром структурних підрозділів, які беруть участь в обхідних листах, включаючи операції створення, редагування та видалення. Система повинна дозволяти налаштовувати атрибути кожного підрозділу (статус деканату, статус випускової кафедри, обов'язковість для всіх студентів) та гнучко керувати послідовністю етапів маршруту. Завдяки механізму індивідуальних налаштувань система адаптує маршрут обходу під конкретного здобувача освіти. Відповідальні особи-модератори повинні мати змогу виставляти відмітки про виконання зі збереженням дати, часу та ПІБ модератора в базі даних. Щодо модуля «Студмістечко», адміністрація очікує наявності інструментів для детального контролю інфраструктури: генерація звітів про вільні ліжко-місця з урахуванням статі мешканців кімнати, управління бронюванням ліжок та контроль дат заїзду й виїзду мешканців.

Власники системи очікують гнучких механізмів налаштування для прив'язки відповідальних користувачів до конкретних підрозділів обходу, фіксації всіх дій персоналу в системному журналі, безпечного архівування обхідних листів після їх повного закриття, а також регулярного автоматичного резервного копіювання бази даних.

Об'єктом комп'ютеризації є маршрут зняття з обліку здобувачів освіти через структурні підрозділи університету та багаторівневий процес адміністрування гуртожитків. Основними користувачами системи виступають представники адміністрації, коменданти гуртожитків і відповідальні модератори підрозділів, що взаємодіють із системою через Backend-інтерфейс. Студенти через особистий кабінет на Frontend-частині мають змогу відстежувати статус свого обхідного листа в режимі реального часу.

Система повинна забезпечувати захищену авторизацію користувачів із подальшим призначенням відповідних ролей та прав доступу. В рамках модуля «Обхідні листи» реалізується повний цикл управління структурними підрозділами: їх створення, редагування, включення до маршруту та налаштування послідовності. Передбачено механізм індивідуалізації

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

маршрутів, що дозволяє призначати специфічні структури обходу для окремих здобувачів освіти залежно від їхньої належності до певної кафедри чи факультету. Управління відмітками включає реєстрацію статусу проходження кожного етапу, можливість додавання коментаря, а також автоматичну фіксацію точного часу виконання та ідентифікатора модератора. Система також забезпечує генерацію зведених звітів і безпечне архівування завершених обхідних листів для збереження історичних даних.

Для модуля «Студмістечко» функціональні вимоги охоплюють ведення повного реєстру гуртожитків і кімнат із зазначенням їхнього типу за статтю для розселення, облік вільних і зайнятих ліжко-місць із можливістю їх бронювання, а також реєстрацію мешканців із фіксацією дати заїзду, очікуваної дати виїзду та поточного статусу проживання.

З точки зору сприйняття, система повинна бути достатньо інтуїтивною для повноцінного освоєння працівниками підрозділів та комендантами протягом одного навчального заняття тривалістю до 60 хвилин. Час відгуку інтерфейсу на основні дії, зокрема погодження обхідного листа, не повинен перевищувати 2 секунд, тоді як для складних аналітичних запитів із агрегацією даних по всіх гуртожитках допустима затримка становить до 10 секунд. Усі форми введення даних мають супроводжуватися підказками для запобігання помилкам.

Щодо надійності, система повинна забезпечувати доступність сервісу не менше ніж 99,5% часу протягом календарного місяця, без урахування заздалегідь запланованих технічних робіт, а відновлення роботи після критичного збою має тривати не більше 15 хвилин. Під час масових операцій — генерування великої кількості обхідних листів або хвилі поселень на початку навчального року — система зобов'язана зберігати цілісність даних та обробляти транзакції без втрат інформації завдяки механізму атомарних транзакцій бази даних.

З точки зору продуктивності, система має підтримувати одночасну роботу великої кількості активних користувачів, особливо в пікові

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

навантажувальні періоди — масове поселення у серпні-вересні або закриття обхідних листів у червні під час випуску. Інтерфейс повинен бути адаптивним і коректно відображатися на різних пристроях, включно зі смартфонами, що є критично важливим для студентів при перегляді статусу власного обхідного листа.

Стосовно можливостей розширення та експлуатації, гнучка реляційна структура бази даних дозволяє легко масштабувати функціонал, змінювати маршрути обходу або розширювати житловий фонд університету без необхідності внесення змін до вихідного коду системи.

Після дослідження всіх необхідних характеристик майбутнього програмного забезпечення та виокремлення ключових функцій, що потребують актори системи, було побудовано діаграму використання (рис. 2.1).

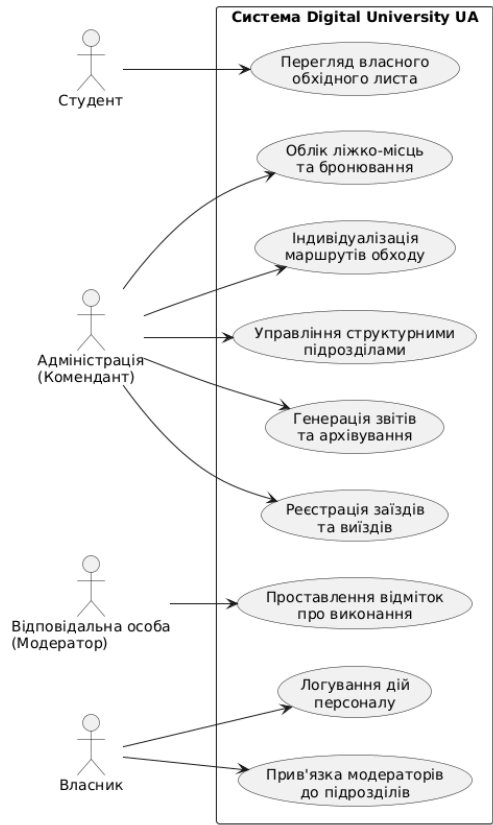


Рисунок 2.1 - Діаграма використання системи

Головними користувачами є адміністратори (коменданти, керівники), які налаштовують інфраструктуру гуртожитків і маршрути обходів, та

модератори підрозділів, які ставлять відмітки. Студенти виступають споживачами інформації через Frontend, де вони контролюють свій маршрут зняття з обліку. Актор «Власник» потребує можливості логувати дії персоналу та керувати глобальними налаштуваннями.

2.2. Обґрунтування вибору системи управління базами даних

У процесі реалізації інформаційної системи було обґрунтовано доцільність використання реляційної системи керування базами даних MySQL, а для її адміністрування — інструмента phpMyAdmin, який забезпечує інтуїтивно зрозуміле середовище для взаємодії з даними через вебінтерфейс. Зазначений вибір зумовлений низкою технологічних та практичних переваг, що вирізняють MySQL серед альтернативних рішень, особливо в контексті розробки складних управлінських систем закладу вищої освіти.

Перш ніж обґрунтовувати вибір конкретної СУБД, необхідно пояснити сам вибір реляційної моделі зберігання даних як такої. Сучасний ринок пропонує широкий спектр альтернативних підходів до організації сховищ: документоорієнтовані бази даних (MongoDB), колонково-орієнтовані (Apache Cassandra), графові (Neo4j) та ключ-значення (Redis). Кожен з цих підходів має власну сферу ефективного застосування. Однак для завдань, що вирішуються в рамках модулів «Студмістечко» та «Обхідні листи», реляційна модель є безальтернативним вибором [30].

Головним аргументом на користь реляційної моделі є надзвичайно високий ступінь взаємозалежності сутностей у предметній області. Інфраструктура модуля «Студмістечко» організована у вигляді жорсткої чотирирівневої ієрархії: гуртожиток — кімната — ліжко-місце — мешканець. Кожен рівень цієї ієрархії нерозривно пов'язаний із попереднім: ліжко-місце не може існувати без кімнати, кімната — без гуртожитку. Запис про мешканця обов'язково посилається на конкретне ліжко-місце і, опосередковано, на фізичну особу в базі даних ЄДЕБО. Порухення будь-якого з цих зв'язків призводить до появи «сирітських» записів і, як наслідок, до некоректної роботи всієї системи. Саме реляційна модель з механізмом зовнішніх ключів

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

(Foreign Keys) на рівні ядра бази даних гарантує, що такі ситуації фізично неможливі: СУБД автоматично відхилить будь-яку спробу видалити батьківський запис, якщо на нього посилаються дочірні записи.

Аналогічна ситуація характерна і для модуля «Обхідні листи». Відмітка студента у базі даних зберігається як запис, що одночасно посилається на ідентифікатор здобувача освіти (з таблиці студентів ЄДЕБО), на конкретний структурний підрозділ університету та на ідентифікатор модератора, який цю відмітку проставив. Ця тристороння залежність ідеально описується реляційною схемою і вкрай складно та ненадійно моделюється в документоорієнтованих сховищах.

Серед реляційних СУБД MySQL конкурує передусім із PostgreSQL, MariaDB та Microsoft SQL Server. Вибір MySQL у межах даного проєкту зумовлений декількома факторами. По-перше, MySQL є де-факто стандартом у стеці LAMP (Linux, Apache, MySQL, PHP), який лежить в основі переважної більшості PHP-застосунків. Фреймворк Yii2, що використовується для реалізації серверної частини, містить нативну, добре оптимізовану підтримку MySQL через компонент ActiveRecord, що значно прискорює розробку та мінімізує кількість ручних SQL-запитів.

По-друге, MySQL відрізняється простотою налаштування та адміністрування порівняно з PostgreSQL, який, попри свою потужність, вимагає більш глибокого розуміння адміністративних процедур. Для університетського середовища, де підтримку системи можуть здійснювати не вузькопрофільні адміністратори СУБД, а розробники широкого профілю, ця перевага є суттєвою.

По-третє, MySQL має чудову документованість, велику міжнародну спільноту та широку екосистему інструментів. Рішення для переважної більшості нестандартних ситуацій, з якими стикається розробник у процесі роботи, вже описані у відкритих джерелах. Це суттєво зменшує час на відлагодження та пошук рішень.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

MySQL характеризується високою продуктивністю, надійністю та здатністю ефективно обробляти великі масиви пов'язаних даних. Для цілей даного проєкту особливе значення мають кілька конкретних технічних можливостей цієї СУБД.

Перша і найважливіша — підтримка транзакцій відповідно до принципів ACID (Atomicity, Consistency, Isolation, Durability). Принцип атомарності гарантує, що будь-яка операція, яка складається з декількох кроків, або буде виконана повністю, або не буде виконана взагалі. Це є критично важливим для модуля «Студмістечко» при реалізації операцій переселення та обміну мешканцями між кімнатами. Операція обміну двох мешканців фактично складається з чотирьох змін у базі даних: закриття двох старих записів про проживання та відкриття двох нових на протилежних ліжках. Якщо після закриття одного або обох попередніх записів про проживання, але до створення обох нових записів відбудеться збій, база даних може залишитися у частково оновленому стані: попередні поселення вже будуть завершені, а нові - створені не повністю. Виконання всіх чотирьох операцій у межах однієї транзакції гарантує, що зміни будуть або збережені разом, або повністю скасовані. MySQL у поєднанні з механізмом `beginTransaction() / commit() / rollBack()` у Yii2 гарантує, що в такій ситуації всі зміни будуть повністю відкочені і база даних повернеться до вихідного коректного стану.

Принцип узгодженості (Consistency) гарантує, що після кожної успішної транзакції база даних переходить із одного коректного стану до іншого. У поєднанні з зовнішніми ключами це означає, що система ніколи не допустить, наприклад, збереження відмітки для неіснуючого студента або поселення мешканця на неіснуюче ліжко-місце.

Другою важливою можливістю є механізм індексування. MySQL дозволяє визначати індекси для конкретних полів таблиць, що значно прискорює виконання запитів до великих масивів даних. У контексті модуля «Обхідні листи» вкрай важливим є індексування поля `student_education_id` у

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

таблиці відміток — це дозволяє системі миттєво знаходити всі відмітки конкретного студента без послідовного перебору мільйонів записів. Аналогічно, в модулі «Студмістечко» індексування поля is_residing у таблиці мешканців дозволяє швидко отримувати список усіх поточних мешканців без залучення архівних даних.

Третьою важливою особливістю є підтримка складних SQL-конструкцій — JOIN, GROUP BY, HAVING, підзапитів та агрегатних функцій. Реалізована в системі логіка підрахунку вільних місць у гуртожитках використовує складні запити з множинними з'єднаннями таблиць (hostel → hostel_room → hostel_room_bed → hostel_room_resident) та фільтрацією через HAVING COUNT(...). Виконання подібних запитів у MySQL відбувається ефективно та передбачувано завдяки вбудованому оптимізатору запитів.

Для адміністрування бази даних MySQL у проєкті використовується phpMyAdmin — відкритий вебінтерфейс для управління MySQL, написаний на PHP. Цей інструмент дозволяє виконувати весь спектр операцій з базою даних без необхідності використання командного рядка: перегляд та редагування структури таблиць, виконання довільних SQL-запитів, імпорт та експорт даних у різних форматах, а також управління правами доступу користувачів бази даних.

У процесі розробки phpMyAdmin відіграв ключову роль на кількох етапах. На початковому етапі він використовувався для швидкого прототипування схеми бази даних: розробник міг візуально створити таблицю, одразу побачити її структуру і скоригувати типи полів або обмеження без необхідності писати повний DDL-скрипт вручну. У процесі налагодження phpMyAdmin дозволяв безпосередньо переглядати стан даних у таблицях після виконання тестових операцій, що значно прискорювало виявлення логічних помилок. Функціонал імпорту та експорту використовувався для розгортання актуальної схеми бази даних на різних серверах (розробницькому та виробничому середовищах).

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

База даних системи організована відповідно до нормальних форм реляційної моделі, що забезпечує відсутність надлишкового дублювання даних та логічну цілісність усієї схеми. Для модуля «Обхідні листи» ключовими таблицями є `bypass_sheet_university_structure` (реєстр структурних підрозділів, що беруть участь в обходах), `bypass_sheet_mark` (відмітки студентів по кожному підрозділу із посиланнями на студента, підрозділ та модератора), `bypass_sheet_user` (прив'язка користувачів-модераторів до підрозділів) та `bypass_sheet_vector` (налаштування порядку відображення підрозділів у маршруті).

Для модуля «Студмістечко» ієрархічна структура відображена у чотирьох взаємопов'язаних таблицях: `hostel` (загальна інформація про гуртожитки), `hostel_room` (кімнати з прив'язкою до гуртожитку та атрибутом статі), `hostel_room_bed` (ліжко-місця з полями бронювання та активності) та `hostel_room_resident` (записи про проживання із полями дат заїзду, виїзду та поточного статусу `is_residing`). Саме завдяки полю `is_residing` система відрізняє поточних мешканців від архівних записів, не видаляючи останніх - що є принципово важливим для збереження повної хронології переміщень кожного студента.

Таким чином, вибір MySQL як СУБД і phpMyAdmin як інструмента адміністрування повністю відповідає технічним вимогам та архітектурній концепції розроблюваної системи. Реляційна модель з підтримкою транзакцій, зовнішніх ключів та складних SQL-запитів гарантує цілісність, продуктивність та надійність зберігання критично важливих адміністративних даних університету.

2.3. Розробка бази даних системи

Для забезпечення надійної та швидкої роботи модулів «Обхідні листи» та «Студмістечко» було розроблено нормалізовану схему реляційної бази даних, яка дозволяє ефективно керувати інформацією без її дублювання та забезпечує фіксацію основних операцій користувачів, пов'язаних із виставленням відміток і зміною даних. Оскільки обидва модулі є частиною

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

єдиної системи «Digital University UA», вони тісно інтегровані з глобальними довідниками структури університету та єдиним реєстром здобувачів освіти.

Для управління житловим фондом у підсистемі «Студмістечко» реалізовано чітку ієрархічну модель, яка дозволяє контролювати інфраструктуру не лише на рівні кімнат, а й опускається до рівня окремого ліжка-місця. Основу становить таблиця гуртожитків, що зберігає загальну інформацію про будівлі - назву, адресу та статус активності. До кожного гуртожитку через зовнішній ключ прив'язані кімнати, де поряд із номером і поверхом зберігається тип статі для поселення, що дозволяє системі автоматично контролювати правила розселення студентів без ручного втручання адміністратора. Кожна кімната своєю чергою деталізується до конкретних ліжок-місць, для яких фіксується інвентарний номер, приналежність до певного факультету чи підрозділу та статус бронювання - це дозволяє університету завчасно розподіляти квоти на місця між структурними підрозділами. Центральною транзакційною таблицею всього модуля є таблиця поселень, яка пов'язує конкретне ліжко-місце з фізичною особою або студентом. Саме тут реєструється весь рух контингенту: дати заїзду, очікувані дати виїзду та актуальний статус проживання. Принципово важливим рішенням є те, що виселення студента не видаляє запис, а лише переводить його у неактивний стан - завдяки цьому система зберігає повну хронологію переміщень кожного мешканця і дозволяє в будь-який момент відновити картину того, хто і де проживав у будь-який конкретний період. Ця багаторівнева структура дозволяє адміністрації точно знати, хто саме і на якому ліжку проживає, та миттєво генерувати аналітичні звіти щодо наявності вільних місць.

Діаграма послідовності для даного модуля зображена на рисунку 2.2.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

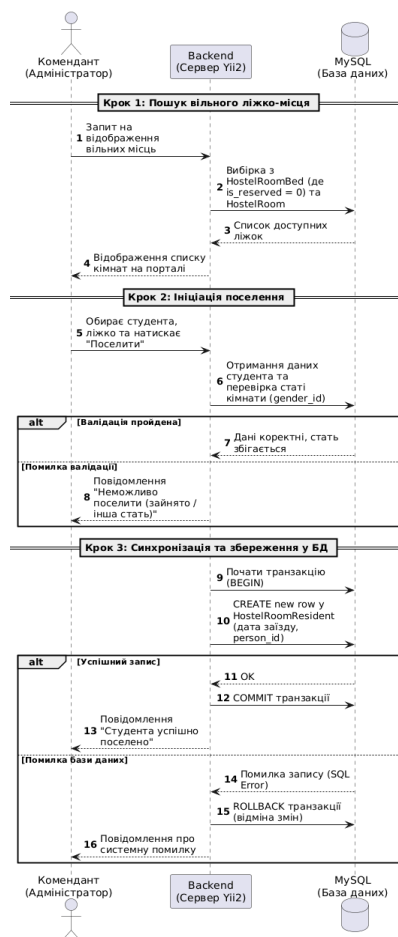


Рисунок 2.2 – Діаграма послідовності для модуля «Студмістечко»

База даних підсистеми «Обхідні листи» базується на системі пов'язаних таблиць, що забезпечують гнучку маршрутизацію здобувача освіти та суворий аудит виконання кожного етапу. Реєстр етапів обходу зберігає налаштування кожного структурного підрозділу, що бере участь у маршруті: посилання на загальний довідник університету, набір бізнес-прапорців, які визначають особливий статус підрозділу - зокрема, чи є він деканатом, випусковою кафедрою або обов'язковим для абсолютно всіх студентів - а також числове поле позиції, що забезпечує правильне сортування послідовності етапів. Окрема таблиця формує зв'язок між структурними підрозділами та користувачами системи: завдяки їй система розуміє, які саме модератори мають авторизацію ставити відмітки від імені конкретного підрозділу - наприклад, уповноважені працівники бухгалтерії або бібліотеки. Для підтримки індивідуалізованих маршрутів передбачено спеціалізовану

таблицю, яка дозволяє формувати специфічні набори підрозділів обходу для конкретних здобувачів освіти, що відрізняються від стандартного маршруту - наприклад, для студентів, які не проживають у гуртожитку. Головною таблицею всього модуля є реєстр відміток, де фіксується сам факт проходження студентом кожного підрозділу: ідентифікатор навчальної картки студента, ідентифікатор підрозділу, статус відмітки та текстовий коментар модератора у разі наявності заборгованості. Критично важливою з точки зору безпеки та прозорості є вимога фіксації ідентифікатора модератора, який вніс зміни, та точного системного часу виконання операції - це формує надійний незмінний журнал аудиту, який унеможливорює несанкціоноване приховування або фальсифікацію відміток.

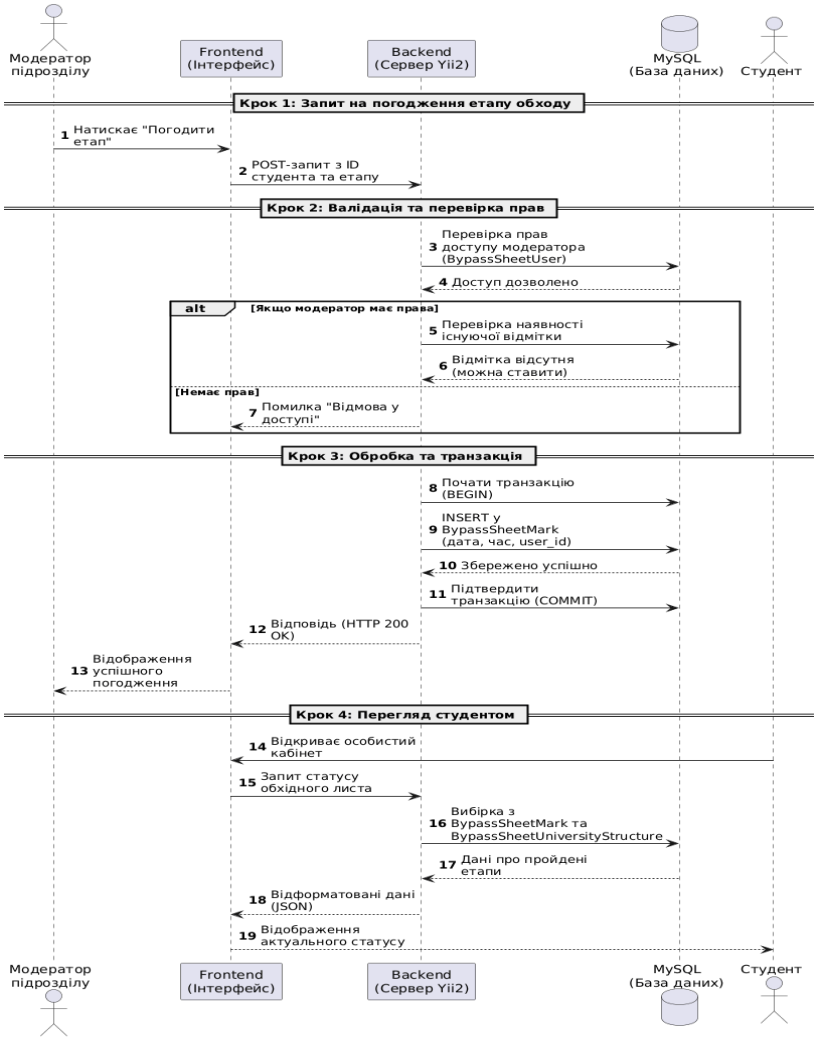


Рисунок 2.3 - Діаграма послідовності для модуля «Обхідні листи»

2.4. Реалізація функціоналу модулів «Обхідні листи» та «Студмістечко»

Розглянемо детальну архітектурну та програмну реалізацію побудованого програмного забезпечення на прикладі модулів «Обхідні листи» та «Студмістечко». Опис охоплює як основні контролери управління, так і допоміжні сервісні класи, що забезпечують виконання складної бізнес-логіки згідно з патерном MVC (Model-View-Controller) та гарантують цілісність даних.

Перейдемо до реалізації функціоналу модуля «Обхідні листи». Процедура збереження відмітки. Основою модуля є процес проставлення відмітки про проходження етапу обхідного листа модератором. За цей функціонал відповідає метод `actionUpdateMark` у контролері бекенду:

```
public function actionUpdateMark(int $education_id, int
$bypass_structure_id): array
{
    Yii::$app->response->format = Response::FORMAT_JSON;
    $data = json_decode(Yii::$app->request->getRawBody(), true);
    $mark = $data['mark'] ?? null;
    $description = $data['description'] ?? null;

    $sheetMark = CircularLettersUniversityStructureService::findBypassSheetMarkOf($education_id, $bypass_structure_id);
    if (is_null($sheetMark)) {
        $sheetMark = new BypassSheetMark();
        $sheetMark->student_education_id = $education_id;
        $sheetMark->bypass_sheet_university_structure_id = $bypass_structure_id;
        if (!is_null($description)) { $sheetMark->mark = false; }
    }
    if (!is_null($mark)) { $sheetMark->mark = $mark; }
    if (!is_null($description)) { $sheetMark->description = $description; }
    $sheetMark->user_id = Yii::$app->user->id;
```

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $sheetMark->datetime      =      Yii::$app->formatter->asDate(new
DateTime('now', new DateTimeZone('Etc/GMT-2')), 'php:Y-m-d H:i:s');
        $sheetMark->save(false);

        return ['success' => true];
    }

```

Цей метод спроектовано для обробки асинхронних (AJAX) запитів, тому він одразу встановлює формат відповіді як JSON і декодує тіло запиту (getRawBody). Спершу система звертається до сервісу CircularLettersUniversityStructureService, щоб перевірити, чи існує вже відмітка для цього студента. Якщо її немає — створюється новий об'єкт BypassSheetMark. Унікальною бізнес-вимогою є те, що якщо модератор залишає коментар (наприклад, "Не здав книжку"), але не ставить статус "Виконано", система автоматично маркує етап як незроблений (\$sheetMark->mark = false). Задля безпеки та аудиту метод завжди жорстко перезаписує user_id ідентифікатором поточного авторизованого співробітника та фіксує точний час транзакції у часовому поясі сервера.

Клієнтський API. Щоб відобразити обхідний лист студенту, розроблено Frontend-контролер BypassSheetController з методом actionGetBypassSheet:

```

public function actionGetBypassSheet(): array
{
    $educationId      =      Yii::$app->request->cookies->
getValue('educationId');
    Yii::$app->response->format = Response::FORMAT_JSON;

    $circularLetterArr = [];
    if
(CircularLettersStudentService::isBypassLetterForStudent($educationId)
) {
        $circularLetterArr      =
CircularLettersStudentService::getLetterForStudent($educationId);
    }
    return ['status' => 'success', 'data' => $circularLetterArr];
}

```

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

Ідентифікатор навчального запису отримується з cookie поточної сесії. За умови ввімкненої в Yii2 перевірки cookie його цілісність контролюється за допомогою криптографічного підпису. Отримане значення додатково має перевірятися на відповідність авторизованому користувачу, оскільки саме розміщення ідентифікатора в cookie не є достатнім механізмом авторизації. Після цього контролер делегує бізнес-логіку сервісу CircularLettersStudentService. Якщо обхідний лист взагалі згенеровано для даного здобувача освіти, сервіс збирає масив структурних підрозділів і відміток, який повертається на клієнтську частину у вигляді структурованого JSON-об'єкта.

Асинхронний запит (JavaScript). Клієнтська логіка виконує AJAX-запит до нашого API через функцію fetchClearanceSheet:

```
const fetchClearanceSheet = async () => {
  const url = `/student/bypass-sheet/get-bypass-sheet`;
  try {
    const response = await getJsonData(url);
    if (response.status === "success" &&
Object.keys(response.data).length > 0) {
      displayClearanceLetter(response.data);
    } else {
      displayNoClearanceMessage();
    }
  } catch (error) {
    displayNoClearanceMessage();
  }
};
```

Функція надсилає запит до раніше описаного API. Використання блоку try...catch гарантує, що у випадку падіння сервера або проблем з мережею користувач не побачить "зламаної" сторінки, а спрацює запасна функція displayNoClearanceMessage, яка ввічливо повідомить, що обхідний лист поки недоступний. Якщо ж дані наявні, вони передаються у функцію рендерингу.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

Маніпуляція DOM-деревом (JavaScript). Дані рендеряться динамічно, створюючи HTML-елементи з відповідними іконками статусу:

```
data.forEach((item) => {
    const value = item.value;
    const iconClass = value === null ? 'fa fa-time' : (value ? 'fa fa-check' : 'fa fa-close');
    const clearanceLetterItem = document.createElement("div");
    clearanceLetterItem.classList.add("clearance-item");
    clearanceLetterItem.innerHTML = `
        <div class="names"><span class="structure-name">${item.name}</span></div>
        <div class="icons">
            ${value === 0 ? `<span class="fa fa-info-circle info-sign" data-tooltip="${item.description}"></span>` : ''}
            <div class="status-icon"><span class="${iconClass}"></span></div>
        </div>`;
    clearanceLetterBlock.appendChild(clearanceLetterItem);
});
```

Цикл проходить по кожному елементу масиву, отриманому з бекенду. Залежно від статусу етапу (виконано, не виконано, ще не оцінено) динамічно підбирається CSS-клас іконки з бібліотеки FontAwesome (fa-check, fa-close або fa-time). Якщо у підрозділу є заборгованість (value === 0), додатково рендериться іконка з інструментом data-tooltip, в який підставляється коментар модератора, що пояснює причину відмови (наприклад, "Не здав студентський квиток"). Це робить інтерфейс максимально інформативним та інтерактивним.

Оптимізація запитів у сервісному шарі. У CircularLettersStudentService інкапсульовано логіку перевірки активності підрозділів:

```
public static function getAllStructureQuery(int $educationId, bool $isActive = true): Query
{
    $bypassSheetUniversityStructure =
    \common\models\circularLetters\BypassSheetUniversityStructure::tableName();
```

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

```

$query = parent::getAllStructureBase($educationId);
if ($isActive) {
    $query->andWhere(["$bypassSheetUniversityStructure.is_active"
=> 1]);
}
return $query;
}

```

Цей метод демонструє патерн Query Builder. Він повертає не готові дані, а об'єкт запиту Query, який пізніше можна розширювати (додавати сортування, пагінацію). Критично важливою є перевірка параметра \$isActive — вона гарантує, що при генерації обхідних листів або звітів закриті чи реорганізовані підрозділи університету (де is_active = 0) не будуть виводитись і не створюватимуть плутанини.

Призначення модератора. Для того, щоб співробітник міг ставити відмітки від імені певного підрозділу, адміністратор має призначити йому права. За це відповідає метод addActionUser:

```

public function addActionUser(): array
{
    Yii::$app->response->format = Response::FORMAT_JSON;
    $data = json_decode(Yii::$app->request->getRawBody(), true);

    if (BypassSheetUser::find()->where(['user_id' => $data['userId'],
'bypass_sheet_university_structure_id' => $data['structureId']])->exists()) {
        return ['success' => false, 'message' =>
ErrorMessageHelperEnum::USER_ALREADY_ADDED->value];
    }

    $bypassSheetUser = new BypassSheetUser();
    $bypassSheetUser->user_id = $data['userId'];
    $bypassSheetUser->bypass_sheet_university_structure_id =
$data['structureId'];
    return ['success' => $bypassSheetUser->save()];
}

```

Логіка обробляє створення зв'язку "Багато до багатьох" між користувачем та структурою. Перед тим як зробити запис у БД, метод виконує

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

перевірку на дублікати (->exists()). Якщо такий користувач вже призначений модератором цього підрозділу, операція блокується і повертається відповідний код помилки, визначений через стандартизований перелік ErrorMessageHelperEnum, що спрощує локалізацію повідомлень.

Видалення прав модератора. Аналогічно реалізовано метод безпечного видалення прав доступу:

```
public function actionDeleteUser(): array
{
    Yii::$app->response->format = Response::FORMAT_JSON;
    $data = json_decode(Yii::$app->request->getRawBody(), true);

    $bypassSheetUser = BypassSheetUser::find()
->where(['bypass_sheet_university_structure_id' => $data['structureId'],
'user_id' => $data['userId']])->one();
    if ($bypassSheetUser && $bypassSheetUser->delete() !== false) {
        return ['success' => true];
    }
    return ['success' => false];
}
```

Замість того, щоб намагатися видалити запис наосліп (що могло б спричинити SQL виняток), контролер спочатку шукає відповідну модель у таблиці BypassSheetUser. Тільки якщо модель знайдено (\$bypassSheetUser не дорівнює null), викликається метод delete().

Пошук користувачів для автодоповнення. Для Frontend-віджету Select2 реалізовано динамічний пошук користувачів з виключенням вже доданих:

```
public function actionGetUserList(int $structureId, string $query = ''):
array
{
    Yii::$app->response->format = Response::FORMAT_JSON;
    $existingUserIds = $this->getBypassUserIds($structureId);

    $queryBuilder = User::find()->select(['id', 'fio', 'username'])
->andFilterWhere(['like', 'fio', $query])
->orWhereFilterWhere(['like', 'username', $query])
```

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

->asArray();

if (!empty($existingUserIds)) {
    $queryBuilder->andWhere(['not in', 'id', $existingUserIds]);
}

return array_map(fn ($user) => ['id' => $user['id'], 'label' =>
$user['fio'] . " (" . $user['username'] . ")"], $queryBuilder->all());
}

```

Цей метод створює потужний SQL-запит за допомогою ActiveQuery. Він шукає збіги введеного користувачем рядка \$query або у ПІБ (fio), або у логіні (username). Найважливіша частина алгоритму — блок andWhere(['not in', ...]). Він відфільтровує масив ID існуючих модераторів, тому адміністратор бачить у випадяючому списку лише тих співробітників, яких ще можна додати до підрозділу. Функція array_map форматує результат спеціально під вимоги віджета Select2 (поля id та label).

Допоміжний метод витягування масиву ID. Для підтримки попереднього методу написано приватний метод оптимізованого витягування стовпця:

```

private function getBypassUserIds(int $structureId): array
{
    return BypassSheetUser::find()
        ->select('user_id')
        ->where(['bypass_sheet_university_structure_id' =>
$structureId])
        ->column();
}

```

Головна перевага цього коду полягає у використанні методу ->column(). Замість того, щоб фреймворк витрачав оперативну пам'ять на створення десятків об'єктів моделі BypassSheetUser з усіма їхніми властивостями, цей метод вказує СУБД повернути простий одновимірний масив (вектор) із значеннями лише одного стовпця user_id. Це значно підвищує швидкодію системи при роботі з великими масивами даних.

Логіка абстрактних сервісів. Перевірка наявності деканату в системі винесена в статичний метод:

```
protected static function isDenearyExist(bool $isActive): bool
```

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

```

{
    return $isActive
        ? BypassSheetUniversityStructure::find()->where(["is_deanery"
=> 1])->andWhere(["is_active" => 1])->exists()
        : BypassSheetUniversityStructure::find()->where(["is_deanery"
=> 1])->exists();
}

```

Метод використовує тернарний оператор для компактного написання коду. Залежно від параметра \$isActive, він перевіряє в БД наявність підрозділу, який позначено системним прапорцем is_deanery = 1. Використання методу exists() замість one() чи count() на рівні SQL трансліюється у запит SELECT EXISTS(...), що є найшвидшим способом дізнатися, чи є хоча б один такий запис у таблиці, не завантажуючи самі дані.

Перейдемо до реалізації функціоналу модуля «Студмістечко». Ініціалізація базового поселення студента. Модуль управління житловим фондом починається з процедури поселення (метод actionCreate):

```

public function actionCreate(): mixed
{
    $params = Yii::$app->request->queryParams;
    $model = new HostelRoomResident();
    $model->hostel_room_bed_id = $params['hostel_room_bed_id'];

    if ($model->loadAll(Yii::$app->request->post()) && $model->saveAll()) {
        return $this->redirect(['view', 'id' => $model->id]);
    }
    return $this->render('create', ['model' => $model]);
}

```

Завдяки використанню методів фреймворку loadAll() та saveAll(), код контролера виглядає дуже мінімалістично. Ці методи інкапсулюють у собі обробку масиву даних \$_POST, автоматичне заповнення атрибутів моделі, виклик валідаторів (чи коректно введено дати, чи існує таке ліжко) та, головне, безпечне збереження запису разом із можливими пов'язаними таблицями в межах однієї транзакції.

Забезпечення цілісності при обміні мешканцями (Транзакції). Найскладніший алгоритм модуля — це обмін ліжко-місцями між двома студентами. Він критично потребує атомарності операцій:

```
public function actionSwapResidents(): Response
{
    $transaction = Yii::$app->db->beginTransaction();
    try {
        if (!$this->completeResidentRecord($resident1, $swapDate)
            || !$this->completeResidentRecord($resident2, $swapDate)) {
            throw new Exception("Failed to complete records");
        }
        $newRes1 = $this->createNewResidentRecord($resident1,
            $resident2->hostel_room_bed_id, $swapDate, $comment);
        $newRes2 = $this->createNewResidentRecord($resident2,
            $resident1->hostel_room_bed_id, $swapDate, $comment);

        if ($newRes1->save(false) && $newRes2->save(false)) {
            $transaction->commit();
            return $this->asJson(['success' => true]);
        }
        throw new Exception("Failed to save records");
    } catch (Throwable $e) {
        $transaction->rollBack();
        return $this->asJson(['success' => false, 'message' => $e-
            >getMessage()]);
    }
}
```

Операція "Обмін" насправді складається з чотирьох послідовних кроків: закриття історії проживання першого мешканця на поточному місці, закриття історії другого, відкриття нової історії для першого на новому місці та відкриття нової історії для другого на старому місці першого. Якщо на будь-якому з цих кроків станеться збій (відключиться світло, пропаде зв'язок з сервером бази даних), система опиниться у "зламаною" стані, де, наприклад, студент виселений, але не заселений нікуди. Саме для цього використовується `Yii::$app->db->beginTransaction()`. Якщо бодай

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

один save() повертає помилку, викидається виняток Exception, який перехоплюється в блоці catch, де виконується \$transaction->rollBack(). Ця команда скасовує всі зміни в базі, гарантуючи абсолютну цілісність інформації.

Аналітика наявності вільних кімнат. Для процесу поселення коменданту необхідно знайти вільну кімнату. Цей запит є складним, оскільки кімната вважається "вільною", якщо кількість активних мешканців менша за кількість фізичних ліжок:

```
$availableRooms = HostelRoom::find()
    ->select(['hr.id as room_id', 'hr.name as room_name'])
    ->from(['h' => 'hostel'])
    ->innerJoin(['hr' => 'hostel_room'], 'h.id = hr.hostel_id')
    ->innerJoin(['hrb' => 'hostel_room_bed'], 'hr.id =
hrb.hostel_room_id')
    ->leftJoin(['hrr' => 'hostel_room_resident'], "hrb.id =
hrr.hostel_room_bed_id")
    ->where(['h.is_active' => 1, 'hr.is_active' => 1,
'hrb.is_reserved' => 0])
    ->groupBy(['hr.id', 'hr.name'])
    ->having('COUNT(DISTINCT hrb.id) > COUNT(DISTINCT CASE WHEN
hrr.is_residing = 1 THEN hrr.hostel_room_bed_id END)')
    ->asArray()->all();
```

Тут застосовано високорівневу оптимізацію SQL-запитів. Замість того, щоб завантажувати всі кімнати в пам'ять сервера і рахувати їх у циклі PHP, запит агрегує дані прямо в MySQL (groupBy). Умова HAVING використовує логічний вираз CASE WHEN всередині агрегатної функції COUNT(). Вона порівнює загальну унікальну кількість ліжок (COUNT(DISTINCT hrb.id)) із кількістю ліжок, де зараз проживають люди (is_residing = 1). Такий підхід переносить агрегування даних на рівень СУБД і зменшує обсяг інформації, що передається до PHP-застосунку. Для кількісної оцінки продуктивності необхідно провести окреме навантажувальне тестування.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Перевірка зайнятості конкретного ліжка. Перед тим, як дозволити адміністратору змінити налаштування ліжка або видалити його, система повинна переконатися, що воно пусте:

```
public
function isEmptyBed(int $bedId): bool
{
    $hasActiveResident = (new Query())->from(['resident' =>
'hostel_room_resident'])->where(
        ['resident.hostel_room_bed_id' => $bedId,
'resident.is_residing' => 1,]
    )->exists();
    return !$hasActiveResident;
}
```

Метод перевіряє наявність активного запису про проживання на заданому ліжку-місці. Метод exists() повертає true, якщо такий мешканець існує. Оскільки метод isEmptyBed() повинен повертати true для вільного ліжка, отриманий результат інвертується оператором !. Така реалізація безпосередньо відображає зміст перевірки й не потребує агрегатної функції COUNT() або SQL-виразу CASE.

Інкапсуляція фільтрів у Search Model. У системі реалізовано табличне відображення історії проживання, де користувач може фільтрувати дані (наприклад, за датою):

```
protected function applyFilters(ActiveQuery $query): void
{
    $query->andWhere(['id' => $this->id,
'hostel_room_bed_id' => $this->hostel_room_bed_id]);
    if (!empty($this->check_in_date)) {
        $query->andWhere(DateHelperService::dateFilter($this->
check_in_date, 'check_in_date'));
    }
}
```

Патерн Search Model дозволяє відділити логіку зберігання даних (Model) від логіки пошуку. Метод andFilterWhere автоматично ігнорує порожні поля, спрощуючи формування умов пошуку. Для роботи з датами використовується

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

виклик зовнішнього сервісу `DateHelperService::dateFilter()`. Це забезпечує коректне перетворення текстової дати з інтерфейсу у формат, який зрозуміє MySQL (наприклад, перетворення діапазону або форматування з `dd.mm.yyyy` у `Y-m-d`).

Зміна порядку відображення гуртожитків. Інтерфейс дозволяє адміністраторам налаштовувати візуальний порядок гуртожитків (кнопки "Вгору" / "Вниз"):

```
public function actionChangePosition($id, $direction): array
{
    $model = $this->findModel($id);
    if ($direction === 'up') {
        $neighbor = Hostel::find()->where(['<', 'position',
$model->position])->orderBy(['position' => SORT_DESC])->one();
    } elseif ($direction === 'down') {
        $neighbor = Hostel::find()->where(['>', 'position',
$model->position])->orderBy(['position' => SORT_ASC])->one();
    }

    if ($neighbor !== null) {
        $currentPosition = $model->position;
        $model->position = $neighbor->position;
        $neighbor->position = $currentPosition;
        if ($model->save(false) && $neighbor->save(false)) return
['success' => true];
    }
    return ['success' => false];
}
```

Алгоритм знаходить "сусідній" запис у базі даних. Якщо напрямок "вгору", ми шукаємо гуртожиток із позицією, меншою за поточну (<), і беремо максимальну з них (SORT_DESC). Далі відбувається класичний обмін змінними (через третю тимчасову змінну `$currentPosition`), і обидві моделі зберігаються.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Валідація даних на рівні моделі. Для захисту бази даних від потрапляння сміттевої або некоректної інформації, кожна модель містить правила перевірки (rules):

```
public function rules(): array
{
    return array_replace_recursive(parent::rules(), [
        [['hostel_room_bed_id', 'check_in_date'], 'required'],
        [['hostel_room_bed_id', 'person_type_id', 'education_id',
        'person_id'], 'integer'],
        [['check_in_date', 'check_out_date'], 'safe'],
        [['comment', 'person_fio'], 'string'],
        [['is_residing'], 'string', 'max' => 1]
    ]);
}
```

Ці правила автоматично застосовуються перед кожним збереженням. required забороняє створення запису без ідентифікатора ліжка та дати заїзду. Правило integer перевіряє, що значення відповідних атрибутів є цілими числами, та запобігає збереженню некоректних даних. Захист від SQL-ін'єкцій забезпечується не валідацією типу, а використанням Query Builder, ActiveRecord і параметризованих запитів, у яких значення передаються окремо від тексту SQL-команди [25, 32].

Запит для повної картки студента. Для побудови комплексної картки з інформацією про те, де саме і на яких умовах проживає здобувач, написано великий SQL запит:

```
$sql = "SELECT      hostel_room_resident.check_in_date,
hostel_room_resident.person_fio,
                hostel_room.name as room_name, hostel_room_bed.number as
room_bed_number,
                hostel.name AS hostel_name, edebo_students.education_id
FROM hostel
INNER JOIN hostel_room ON hostel_room.hostel_id = hostel.id
INNER      JOIN      hostel_room_bed      ON
hostel_room_bed.hostel_room_id = hostel_room.id
LEFT JOIN hostel_room_resident ON hostel_room_bed.id =
hostel_room_resident.hostel_room_bed_id
```

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

```

        LEFT JOIN edebo_students ON edebo_students.education_id =
        hostel_room_resident.education_id
        WHERE hostel_room_resident.id = :id";
        $studentInfo = Yii::$app->db->createCommand($sql)-
        >bindValue(':id', $id)->queryOne();

```

У випадках, коли необхідно зібрати інформацію з понад 5 різних таблиць (Гуртожиток, Кімната, Ліжко, Мешканець, Дані ЄДЕБО), використання зв'язків ActiveRecord (with()) може призвести до проблеми "N+1 запитів" або надмірного навантаження. Розробник обрав шлях написання оптимізованого прямого SQL-запиту через createCommand. Важливо зазначити, що вхідний параметр передається до запиту за допомогою параметризації через bindValue(':id', \$id), завдяки чому значення обробляється окремо від тексту SQL-команди.

Захист від логічних конфліктів при редагуванні фонду. Кімнати в гуртожитку поділяються за статтю. При спробі змінити тип кімнати система виконує захисну перевірку:

```

public function actionChangeGender(): Response
{
    $room = HostelRoom::findOne(Yii::$app->request-
    >post('hostel_room_id'));
    $hasResidents = HostelRoomResident::find()
        ->innerJoin('hostel_room_bed',
        'hostel_room_resident.hostel_room_bed_id = hostel_room_bed.id')
        ->where(['hostel_room_bed.hostel_room_id' => $room->id,
        'hostel_room_resident.is_residing' => true])
        ->exists();

    if ($hasResidents) {
        return $this->asJson(['success' => false, 'message' =>
        "Cannot change room gender with occupants"]);
    }
    // ...
}

```

Зміна статусу кімнати (наприклад, з чоловічої на жіночу) є неприпустимою, якщо в ній наразі проживають особи (прапорець is_residing =

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

true). Алгоритм з'єднує таблицю ліжок та таблицю мешканців і, у разі знаходження хоча б одного студента, блокує операцію з поверненням зрозумілого тексту помилки для користувальницького інтерфейсу. Це класичний приклад дотримання бізнес-правил (Business Constraints) на рівні бекенду.

Алгоритм переселення студента. Для фіксації повної історії рухів студентів, процес "переселення" не є простою зміною атрибута, а являє собою складний процес архівації попереднього проживання:

```
public function actionRelocateInHostel(): Response
{
    $currentResident = HostelRoomResident::findOne($residentId);
    $newBed = HostelRoomBed::findOne($newBedId);

    if (! $this->completeResidentRecord($currentResident,
    $relocationDate)) {
        return $this->asJson(['success' => false]);
    }

    $newResident = $this->createNewResidentRecord($currentResident, $newBedId, $relocationDate,
    $comment);
    if ($newResident->save(false)) {
        return $this->asJson(['success' => true]);
    }
    return $this->asJson(['success' => false]);
}
```

В архітектурі системи закладено принцип збереження історії. Відповідно до нього, метод completeResidentRecord позначає поточний запис як "виселено" і фіксує дату виселення (релевантно даті переселення). Потім метод фабрики createNewResidentRecord створює абсолютно новий запис у таблиці мешканців, копіюючи дані про особу, але встановлюючи нову дату заїзду і прив'язуючи її до нового ідентифікатора ліжка (\$newBedId). Лише після збереження цього нового запису транзакція переселення вважається успішною.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Розглянуті фрагменти вихідного коду демонструють застосування транзакцій, параметризованих запитів, серверної валідації, розмежування прав доступу та асинхронної взаємодії з клієнтською частиною. Використані підходи сприяють підтримуваності програмного коду, цілісності даних і зменшенню ризику виникнення типових помилок.

Висновки до другого розділу

У другому розділі кваліфікаційної роботи було здійснено комплексне проєктування та детальний опис програмної реалізації модулів «Студмістечко» та «Обхідні листи» як складових сучасної цифрової системи управління освітніми процесами університету. За результатами проведеної роботи можна зробити наступні висновки:

Формалізація вимог та проєктування взаємодій. На етапі концептуального проєктування було визначено ключові бізнес-вимоги та функціональні можливості системи. За допомогою діаграм варіантів використання (Use-Case) чітко розмежовано ролі користувачів (студент, модератор підрозділу, адміністратор, комендант гуртожитку) та їхні права доступу. Це дозволило створити безпечне та зрозуміле середовище для кожного учасника процесу.

Оптимізація бази даних. Обґрунтовано вибір реляційної системи управління базами даних, що є критично важливим для збереження складної ієрархічної структури гуртожитків (будівля – кімната – ліжко-місце) та динамічних маршрутів обхідних листів. Спроєктована структура БД (моделі HostelRoomResident, BypassSheetMark тощо) гарантує нормалізацію даних, усунення дублювання та високу швидкість обробки інформації завдяки правильному використанню зовнішніх ключів.

Програмна реалізація за стандартами MVC. Наведено детальний аналіз вихідного коду, який підтверджує суворе дотримання архітектурного патерну Model-View-Controller. Бізнес-логіка відділена від логіки відображення. Розроблені сервісні класи (CircularLettersStudentService) інкапсулюють у собі складні запити, а класи пошуку (Search Models) забезпечують гнучку фільтрацію даних.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Забезпечення цілісності та безпеки даних. Під час реалізації критичних бізнес-процесів (таких як переселення студентів або обмін кімнатами) успішно застосовано транзакції бази даних (beginTransaction(), rollBack()). Для оптимізації аналітичних вибірок замість завантаження важких об'єктів у пам'ять використано прямі SQL-вирази (Expression, HAVING, COUNT(DISTINCT)), що робить систему стійкою до високих навантажень.

Інтерактивність користувацького інтерфейсу. Реалізовано зручний та швидкий клієнтський інтерфейс (Frontend) з використанням асинхронних AJAX-запитів (async/await) та динамічної маніпуляції DOM-деревом. Це дозволяє оновлювати інформацію після виконання асинхронного запиту без повного перезавантаження сторінки.

Отже, прийняті під час проєктування архітектурні та технологічні рішення дозволили створити надійну, масштабовану та продуктивну систему. Вона успішно вирішує завдання автоматизації рутинних процесів поселення студентів та обробки обхідних листів, мінімізує вплив людського фактора і значно підвищує загальну якість надання адміністративних послуг в університеті.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ОНЛАЙН-МЕНЕДЖМЕНТУ ОСВІТНІХ ПРОЦЕСІВ

3.1. Архітектура розгортання системи

Розроблений програмний комплекс має трирівневу клієнт-серверну архітектуру. Перший рівень представлений веббраузером користувача, другий - монолітним серверним застосунком на базі PHP-фреймворку Yii2, третій - реляційною базою даних MySQL. Серверний застосунок побудовано відповідно до патерну MVC, що забезпечує розмежування відповідальності між контролерами, моделями та представленнями

Фізичне розгортання системи виконується у контейнерному середовищі Docker. Склад контейнерів, правила їх запуску, мережеву взаємодію та постійні томи визначено у файлі `docker-compose.yml`. Контейнеризація не змінює монолітного характеру серверного застосунку Yii2, а лише ізолює його середовище виконання та інфраструктурні компоненти [2, 20, 35].

Першим рівнем архітектури є клієнтський вузол - браузер кінцевого користувача, будь то студент, адміністратор чи комендант гуртожитку. Саме тут відображається інтерфейс, побудований із використанням HTML, CSS та JavaScript, і здійснюється комунікація з сервером за допомогою синхронних та асинхронних HTTP-запитів. Застосування технології AJAX є принципово важливим для зручності роботи: операції, що виконуються часто й потребують миттєвого відгуку - зокрема виставлення відмітки в обхідному листі, перемикання статусу бронювання ліжка-місця або динамічне оновлення списку вільних кімнат при зміні параметрів фільтрації - надсилають запити до сервера у фоновому режимі та оновлюють лише відповідну частину сторінки без її повного перезавантаження. Це суттєво скорочує кількість переданих даних і створює відчуття плавної, адаптивної взаємодії, що є особливо важливим під час масових операцій поселення, коли комендант послідовно обробляє десятки записів в рамках одного робочого сеансу. Клієнтський

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

JavaScript також відповідає за первинну валідацію форм та управління станом інтерфейсних компонентів - наприклад, каскадне оновлення випадаючих списків при виборі гуртожитку, кімнати та конкретного ліжка.

Центральним і найбільш складним компонентом усієї системи є основний вебсервер, побудований на базі PHP-фреймворку Yii2. Він виконує роль ядра бізнес-логіки і реалізований відповідно до патерну MVC, де кожен із трьох архітектурних шарів виконує чітко визначену та ізольовану роль. Шар контролерів приймає вхідні HTTP-запити від клієнта та здійснює їх маршрутизацію до відповідного обробника залежно від URL та методу запиту. Контролер відповідає за загальний потік виконання операції: він перевіряє автентифікацію та авторизацію поточного користувача, викликає необхідні методи моделей, обробляє отримані результати та формує відповідь - чи то HTML-сторінку для синхронного запиту, чи то JSON-масив для асинхронного. Саме на рівні контролерів реалізована більшість захисних механізмів: фільтри доступу через компонент AccessControl гарантують, що, наприклад, студент не зможе звернутися до захищених адміністративних дій, навіть знаючи їхні URL-адреси.

Шар моделей є найбільш насиченим з точки зору бізнес-логіки. Моделі в Yii2, що реалізуються через базовий клас ActiveRecord, виконують одночасно кілька функцій: визначають правила валідації вхідних даних, описують зв'язки між сутностями бази даних та надають зручний об'єктно-орієнтований інтерфейс для виконання SQL-запитів. У контексті модуля «Студмістечко» модель мешканця гуртожитку, наприклад, описує правило унікальності - забороняє одночасне активне поселення однієї фізичної особи на два різних ліжко-місця. У модулі «Обхідні листи» модель відмітки містить бізнес-правило, що забороняє збереження відмітки від імені підрозділу, до модераторів якого поточний користувач не належить. Такий підхід централізує перевірку бізнес-правил безпосередньо в моделях, унеможливаючи їх обхід через будь-який інший шлях звернення до даних. Шар представлень відповідає за генерацію фінальної відповіді - HTML-

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

сторінки або JSON-структури - і максимально відокремлений від логіки. Це дозволяє змінювати зовнішній вигляд інтерфейсу без жодного ризику порушити бізнес-логіку.

Третім рівнем архітектури є рівень зберігання даних, представлений СУБД MySQL. Він забезпечує надійне довгострокове зберігання всіх критично важливих даних системи: ієрархічної структури гуртожитків, повної хронологічної історії поселень кожного мешканця, відміток та аудиторського журналу обхідних листів, налаштувань структурних підрозділів та прив'язок модераторів. Доступ до бази даних здійснює серверний застосунок Yii2 через налаштоване з'єднання з MySQL. Для забезпечення цілісності даних використовується рушій InnoDB, який підтримує транзакції та зовнішні ключі [7]. Фізично сервер баз даних рекомендується розміщувати у внутрішній приватній мережі, ізольованій від публічного інтернету, з мінімальною мережевою затримкою до основного вебсервера - це є однією з апаратних вимог, сформульованих у попередньому розділі. Такий підхід унеможливорює прямий зовнішній доступ до бази даних і є стандартною практикою забезпечення безпеки серверної інфраструктури.

Описана трирівнева архітектура забезпечує чітке розмежування відповідальності між клієнтським інтерфейсом, серверною бізнес-логікою та рівнем зберігання даних. Фізичну і логічну взаємодію компонентів наведено на діаграмі розгортання на рисунку 3.1.

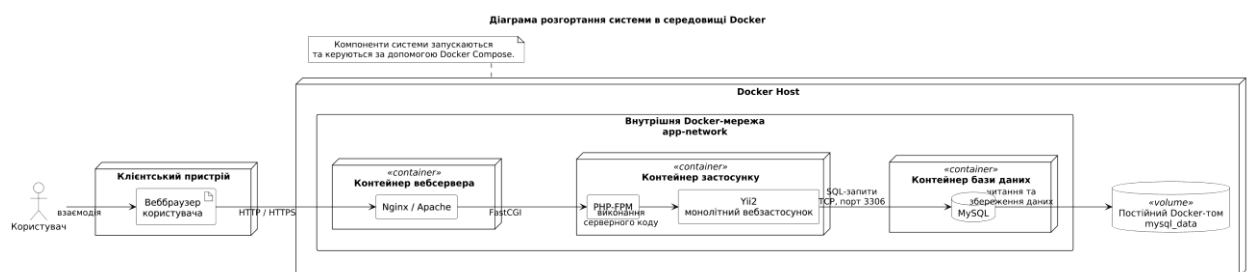


Рисунок 3.1 – Діаграма розгортання розробленої системи управління освітніми процесами

3.2. Структура інтерфейсу застосунку

Розроблена система онлайн-менеджменту освітніх процесів характеризується наявністю інтуїтивно зрозумілого користувацького інтерфейсу та забезпечує повну адаптивність до різноманітних типів пристроїв.

Для персоналізованого доступу до ресурсу передбачено процедуру автентифікації. Зокрема, перед переходом на головну сторінку порталу користувач має пройти авторизацію шляхом введення облікових даних у відповідну форму (рис. 3.2), що гарантує захищений доступ до індивідуального функціоналу відповідно до ролі в системі.

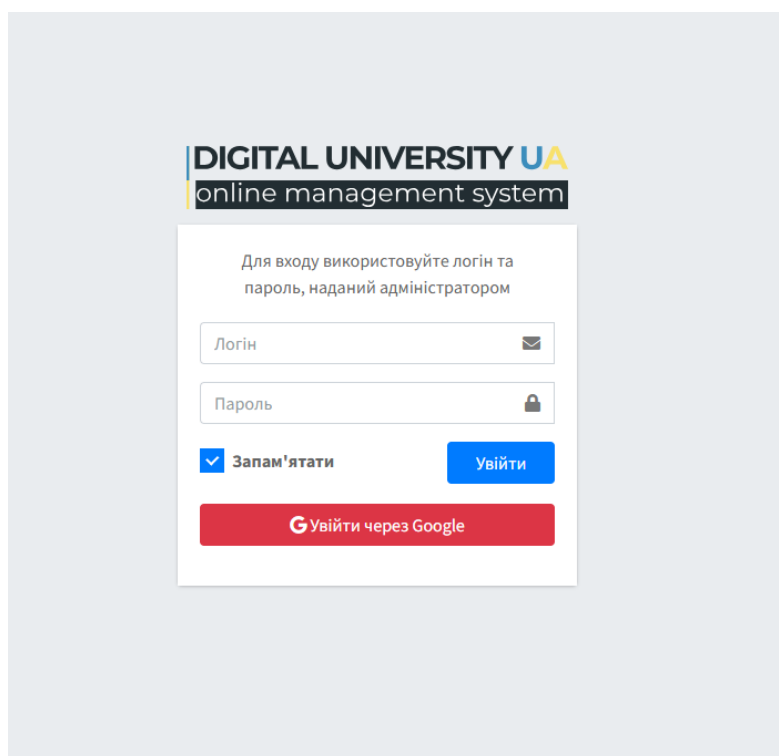


Рисунок 3.2 - Вхід на портал

З міркувань інформаційної безпеки система не передбачає можливості самостійної реєстрації користувачів через веб-інтерфейс. Формування облікових записів покладається на відповідний структурний підрозділ університету, уповноважений здійснювати створення та розповсюдження логінів і паролів для доступу до системи. Централізований контроль за доступом до освітнього середовища мінімізує ризики несанкціонованого проникнення особами, що не мають відношення до університету.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

Після проходження процедури автентифікації користувач автоматично перенаправляється на головну сторінку платформи. Інтерфейс системи організовано таким чином, щоб забезпечити швидкий доступ до основних функціональних компонентів. З цією метою велика кількість модулів і підмодулів була логічно згрупована та представлена у вигляді зручної навігаційної панелі, розміщеної у лівій частині екрана (рис. 3.3).

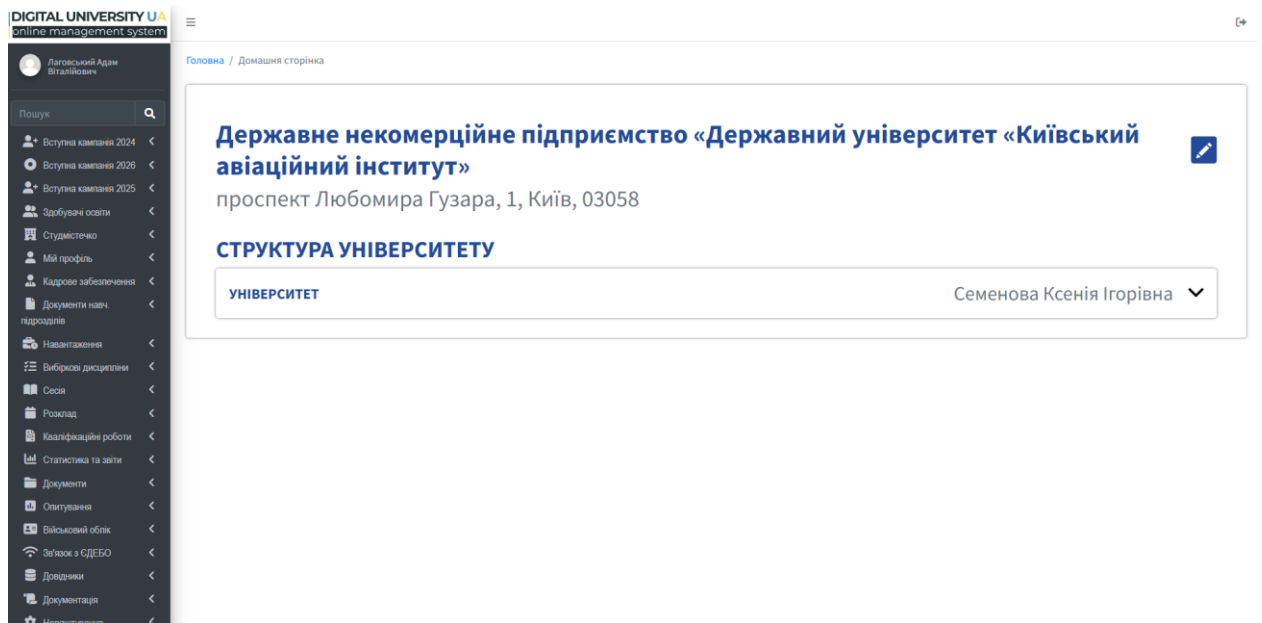


Рисунок 3.3 - Головна сторінка

Серед усіх частин спроектованого застосунку ми розглядатимемо модулі «Студмістечко» та «Обхідні листи», які є одними з ключових в управлінні Університетом. Доступ до першого з них забезпечує вкладка «Студмістечко» (рис. 3.4).

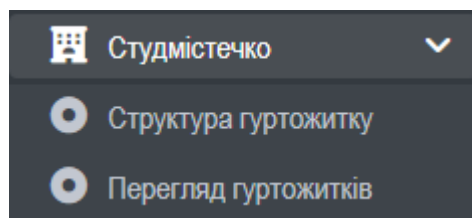


Рисунок 3.4 - Вкладка «Студмістечко»

Доступ до модуля «Обхідні листи» користувач з достатніми правами може отримати через натискання на підменю «Здобувачі освіти» у головному меню системи (рис 3.5).

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

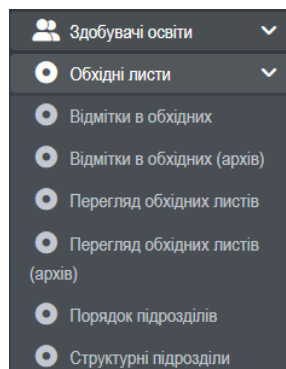


Рисунок 3.5 - Вкладка «Здобувачі освіти»

Огляд системи здійснюється від імені користувача з правами адміністратора. Почнемо огляд системи з управління модулем «Студмістечко», а саме підвкладки «Структура гуртожитку» (рис. 3.6).

Головна / Список гуртожитків

Список гуртожитків +										
Всього 1 запис.										
№ п/п	Порядок	Назва гуртожитку	Дії	Статус	К-ть місць	Заповнено місць	Вільно місць	Резерв	Адреса	Примітка
1		Гуртожиток	  		7	7	0	0	123	Гуртожиток забезпечений необхідними умовами для комфортного проживання студентів. Діє пропускна система та встановлені правила внутрішнього розпорядку.

Рисунок 3.6 – Підвкладка «Структура гуртожитку»

На даній сторінці відображається список гуртожитків університету з детальною інформацією про них. Для управління будівлею було розроблено три активні кнопки, які виконують функції редагування, перегляду кімнат та видалення. При переході на сторінку редагування гуртожитку ми маємо можливість змінити назву, адресу, примітку та чи є будівля активною (рис. 3.7). Кількість місць розраховується автоматично на основі кількості записів про ліжка. Резерв встановлюється у вікні перегляду гуртожитка, через контекстне меню.

Редагування гуртожитку

Назва гуртожитку

Гуртожиток

Адреса

123

Активний

Примітка

Гуртожиток забезпечений необхідними умовами для комфортного проживання студентів. Діє пропускна система та встановлені правила внутрішнього розпорядку.

Зберегти

Відмінити

Рисунок 3.7 – Сторінка редагування гуртожитку

Кнопка перегляду відкриває сторінку зі списком кімнат вибраного гуртожитку (рис. 3.8). На сторінці відображено таблицю з номером кімнати, її типом, поверхом, статусом і приміткою. Для кожного запису доступні дії перегляду, редагування та видалення. Форма редагування кімнати дає змогу змінити її основні характеристики (рис. 3.9)

Список кімнат гуртожитку

Показані 1-6 із 6 записів.

#	Назва кімнати	Дії	Поверх	Тип кімнати	Статус	Примітка
1	1		1	Жіноча		
2	3		1	Чоловіча		
3	5		1	Жіноча		
4	Не задано		1	Чоловіча		
5	2		2	Чоловіча		
6	4		2	Жіноча		

Рисунок 3.8 – Сторінка перегляду гуртожитку

Редагування кімнати

Назва кімнати

1

Поверх

1

Тип кімнати

Жіноча

Активна

Примітка

Зберегти

Відмінити

Рисунок 3.9 – Сторінка редагування кімнати

На сторінці перегляду кімнати користувач може керувати розміщеними в ній ліжко-місцями (рис. 3.10). Під час редагування ліжко-місця можна змінити його номер, статус активності, примітку, ознаку резервування та належність до структурного підрозділу університету (рис. 3.11).

Головна / Список гуртожитків / Список кімнат гуртожитку / Місця у кімнаті

Місця у кімнаті +
Всього 2 записи.

#	Номер місця	Дії	Підрозділ	Статус	Резерв	Примітка
1	1	 	Аерокосмічний факультет			123
2	2	 	Кафедра електроніки, робототехніки і технологій моніторингу та Інтернету речей			

Рисунок 3.10 – Сторінка перегляду кімнати

Головна / Список гуртожитків / Список кімнат гуртожитку / Місця у кімнаті / Редагування місця

Редагування місця

Номер місця

1

Підрозділ

Аерокосмічний факультет

☒ Активне ☐ Резерв

Примітка

123

Зберегти Відмінити

Рисунок 3.11 – Сторінка редагування місця в кімнаті

Головною сторінкою модуля є підменю «Перегляд гуртожитків» (рис. 3.12), на якому користувач може переглянути повний список гуртожитків. У таблиці наведено номер, назву, статус, загальну кількість ліжко-місць, кількість зайнятих, вільних і зарезервованих місць, адресу та примітку.

Рисунок 3.12 – Сторінка перегляду гуртожитків

Після натискання кнопки перегляду відкривається сторінка огляду гуртожитку, яка має два режими відображення: візуальний із розташуванням кімнат (рис. 3.13) і табличний (рис. 3.14). В обох режимах передбачено перехід до картки здобувача освіти після натискання на його ПБ. У табличному режимі також доступна фільтрація записів за даними мешканця.

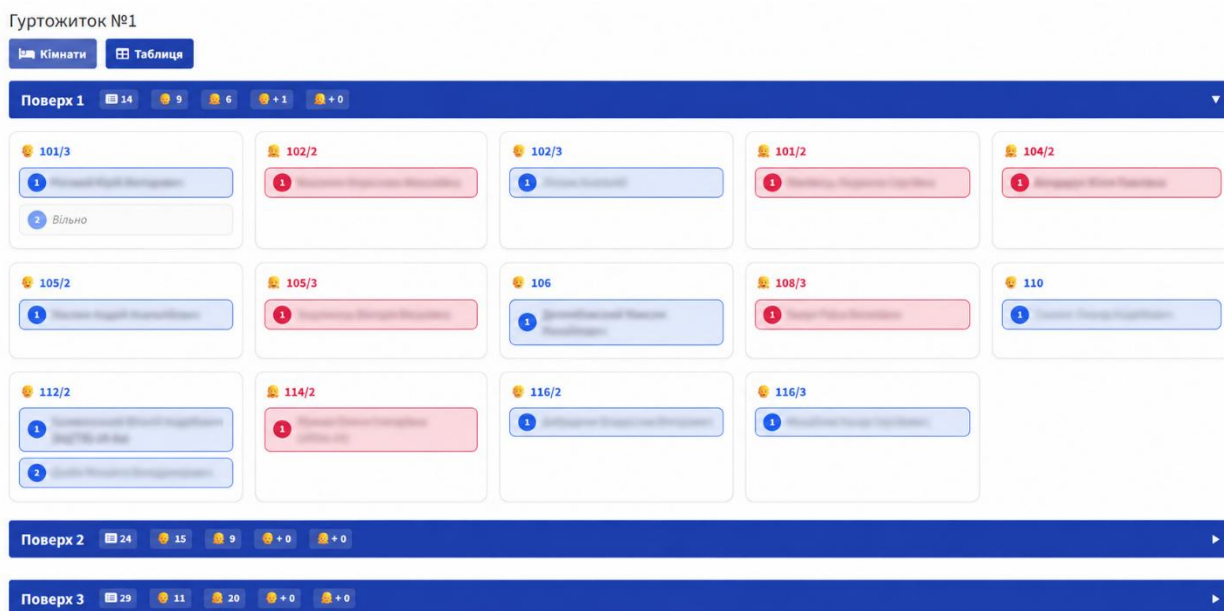


Рисунок 3.13 – Візуальне відображення кімнат

Кімната	Місце	Стать	Факультет	ПІБ	Тип особи	Дата народження / Група	Примітка медична	Примітка (пільга)	Примітка (здоров'я)	Примітка (інше)
Bci	Bci	Bci	Bci	Введіть ПІБ	Bci	Введіть групу	Примітка	Примітка	Примітка	Примітка
Поверх 1										
1	1	ч			Здобувач вищої освіти					
2	1	ч			Здобувач вищої освіти					
3	1	ж			Здобувач вищої освіти					
4	1	ч			Здобувач вищої освіти					
595	1	ж			Здобувач вищої освіти					
596	1	ж			Здобувач вищої освіти					
597	1	ч			Здобувач вищої освіти					
598	1	ж			Здобувач вищої освіти					
599	1	ч			Здобувач вищої освіти					
600	1	ж			Здобувач вищої освіти					
601	1	ч			Здобувач вищої освіти					
602	1	ч			Здобувач вищої освіти	БЦ(ТВ)-25-8а				
602	2	ч			Здобувач вищої освіти	ЗТВн-25				
603	1	ж			Здобувач вищої освіти	зМОН-24				
604	1	ч			Здобувач вищої освіти					
605	1	ч			Здобувач вищої освіти					
Поверх 2										
606	1	ч			Здобувач вищої освіти					
607	1	ч			Здобувач вищої освіти					
608	1	ч			Здобувач вищої освіти					
609	1	ч			Здобувач вищої освіти					
610	1	ч			Здобувач вищої освіти					
611	1	ж			Здобувач вищої освіти					
612	1	ж			Здобувач вищої освіти					
613	1	ж			Здобувач вищої освіти					
615	1	ч			Здобувач вищої освіти					
615	1	ч			Здобувач вищої освіти					
616	1	ж			Здобувач вищої освіти					

Рисунок 3.14 – Табличне відображення кімнат

Наступним ключовим модулем є «Обхідні листи». В ньому міститься шість підпунктів, включаючи архіви. Почнемо з сторінки «Відмітки в обхідних». Це головне робоче місце модератора структурного підрозділу. Сторінка виводить список активних студентів (переважно випускників), яким згенеровано обхідний лист. Список зручно відсортований та підтримує фільтрацію (рис. 3.15).

Відмітки в обхідних листах Показані 1-15 із 157 записів.							
#	Група	Факультет	Форма навчання	Статус фізкультурника	Адресований факультет	Управління навчально-методичної роботи	
1	Б-134-22-1-1/Б	АНО	Денно	Контракт	Позначено на картці		
2	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
3	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
4	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
5	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
6	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
7	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
8	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
9	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
10	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
11	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
12	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
13	Б-134-22-1-1/Б	АНО	Денно	Екстерн			
14	Б-134-22-1-1/Б	АНО	Денно	Екстерн			

Рисунок 3.15 – Сторінка «Відмітки в обхідних»

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

Наступною сторінкою є «Відмітки в обхідних листах (архів)». Єдиною відмінністю між цими сторінками є те, що студенти, відображені в архіві, вже або завершили навчання, або були відраховані (рис. 3.16).

Відмітки в обхідних листах (архів)									
Показані 226-240 із 357 записів.									
Освітня програма		Рівень випуску/відрядження		Форма навчання		Основа фінансування		Дата завершення навчання/відрядження	
#	П.І.Б.	Група	Факультет	Форма навчання	Основа фінансування	Дата завершення навчання/відрядження		Аерокосмічний факультет	Управління навчально-методичної роботи
226	Кравчук Олександр Сергійович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	16.06.2025 (Відраховано із ЗО)			
227	Карчевський Андрій Олександрович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	30.06.2025 (Завершено навчання)			
228	Горбенко Ярослав Вадимович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	30.06.2025 (Завершено навчання)			
229	Герца Данило Михайлович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	30.06.2025 (Завершено навчання)			
230	Добрянський Данило Вікторович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	30.06.2025 (Завершено навчання)			
231	Філоненко Олександр Євгенович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	30.06.2025 (Завершено навчання)			
232	Самко Андрій Володимирович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	30.06.2025 (Завершено навчання)			
233	Ханн Михайло Юрійович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	30.06.2025 (Завершено навчання)			
234	Гайдаш Олексій Леонідович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	30.06.2025 (Завершено навчання)			
235	Кобзар Назарій Олександрович	Ба-272-21-1-ТО	АКФ	Денна	Контракт	30.06.2025 (Завершено навчання)			
236	Діоміда Дмитро Олександрович	Ба-272-21-1-ТО	АКФ	Денна	Бюджет	30.06.2025 (Завершено навчання)			

Рисунок 3.16 – Сторінка «Відмітки в обхідних листах (архів)»

Сторінка «Перегляд обхідних листів» (рис. 3.17) надає адміністрації університету (наприклад, працівникам деканату) зведений звіт. На відміну від сторінки "Відмітки", де працівник бачить лише свій підрозділ, ця сторінка генерує повну матрицю. Для кожного студента циклом підвантажуються повний масив його відміток по всіх підрозділах.

Перегляд обхідних листів											
Показані 1-15 із 2 268 записів.											
Освітня програма		Рівень випуску/відрядження		Форма навчання		Основа фінансування		Дата завершення навчання/відрядження		Дата завершення навчання/відрядження	
#	П.І.Б.	Група	Факультет	Форма навчання	Основа фінансування	Деканат	Випускова кафедра	Аерокосмічний факультет	Сектор планування соціальних витрат	Управління навчально-методичної роботи	Управління навчально-методичної роботи
1	Старіков Андрій Валерійович	Б-134-22-1-ЛВ	АКФ	Денна	Контракт						
2	Мозок Денис Вячеславович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
3	Полосмак Максим Олександрович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
4	Маніков Матвій Олександрович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
5	Паталах Гліб Данилович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
6	Савінов Антон Олексійович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
7	Кутюков Михайло Сергійович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
8	Виноградський Ярослав Олексійович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
9	Троїма Олександр Вікторович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
10	Кузьменко Євгеній Павлович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
11	Модак Роман Миколайович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
12	Буринський Данило Олексійович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
13	Шульченко Богдан Геннадійович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
14	Петруня Максим Петрович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						
15	Плісюк Владислав Володимирович	Б-134-22-1-ЛВ	АКФ	Денна	Бюджет						

Рисунок 3.17 – Сторінка «Перегляд обхідних листів»

Сторінка «Перегляд обхідних листів (архів)» (рис. 3.18) аналогічно до попередньої сторінки, але звертається до контролера архіву. За допомогою

спадного списку можна обрати навчальний рік та освітню програму і побачити повністю заповнені та закриті обхідні листи історичних випусків. Це необхідно для вирішення спірних питань, якщо випускник через роки звертається за дублікатом документів.

☒ Перегляд обхідних листів (архів)
Показані 1-15 із 4 495 записів.

Освітня програма	Рік випуску/вдодування	Група	Факультет	Форма навчання	Основа фінансування	Дата завершення навчання/вдодування	Деканат	Випускова кафедра	Аерокосмічний факультет	Сектор планування соціальних витрат	Управління навчально-методичної роботи
П.І.Б.											
1	Ткаченко Денис Сергійович	Б-131-24-1-МК	АКФ	Денна	Бюджет	27.01.2025 (Відраховано із ЗО)			Відраховано		
2	Чабарай Андрій Андрійович	Б-131-24-1-МК	АКФ	Денна	Бюджет	23.06.2025 (Відраховано із ЗО)					
3	Костяний Артем Дмитрович	Б-134-24-1-ЛВ	АКФ	Денна	Бюджет	14.09.2025 (Відраховано із ЗО)					
4	Забайрачній Олексій Олександрович	Б-134-24-1-ЛВ	АКФ	Денна	Бюджет	27.01.2025 (Відраховано із ЗО)					
5	Мурашов Владислав Сергійович	Б-134-24-1-ЛВ	АКФ	Денна	Бюджет	23.06.2025 (Відраховано із ЗО)					
6	Скрябіньський Данило Олександрович	Б-134-24-1-ЛВ	АКФ	Денна	Бюджет	23.06.2025 (Відраховано із ЗО)					
7	Шаловал Сергій Владиславович	Б-134-24-1-ЛВ	АКФ	Денна	Бюджет	31.08.2025 (Відраховано із ЗО)					
8	Хорошо Сергій Володимирович	Б-134-24-1-ЛВ	АКФ	Денна	Бюджет	27.01.2025 (Відраховано із ЗО)					
9	Петьмак Іван Сергійович	Б-134-24-1-ЛВ	АКФ	Денна	Контракт	23.06.2025 (Відраховано із ЗО)					
10	Бондаревський Єгор Васильович	Б-134-24-1-ОС	АКФ	Денна	Бюджет	23.06.2025 (Відраховано із ЗО)					
11	Людський Антон Віталійович	Б-141-24-1-МН	АКФ	Денна	Бюджет	23.06.2025 (Відраховано із ЗО)					
12	Мазуревич Данило Анатолійович	Б-141-24-1-СЕ	АКФ	Денна	Бюджет	23.06.2025 (Відраховано із ЗО)					
13	Лі Олександр Олександрович	Б-174-24-1-АТ	АКФ	Денна	Бюджет	23.06.2025 (Відраховано із ЗО)					

Рисунок 3.18 – Сторінка «Перегляд обхідних листів (архів)»

Сторінка «Порядок підрозділів» (рис. 3.19) дозволяє адміністратору системи налаштовувати візуальний вигляд обхідного листа (вектор маршруту). Адміністратор за допомогою кнопок "Вгору" / "Вниз" змінює позицію підрозділу.

☒ Порядок структурних підрозділів
Показані 1-5 із 5 записів.

#	Порядок	Структурний підрозділ
1		Деканат
2		Випускова кафедра
3	Вниз	Аерокосмічний факультет
4	Вгору Вниз	Сектор планування соціальних витрат
5	Вгору	Управління навчально-методичної роботи

Рисунок 3.19 – Сторінка «Порядок підрозділів»

Сторінка «Структурні підрозділи» (рис. 3.20) є панеллю управління конфігурацією самого процесу обхідних листів. Тут визначається, які саме існуючі підрозділи університету (кафедри, відділи, гуртожитки) братимуть участь у маршруті обхідного листа. Адміністратор може додавати нові вузли маршруту або видаляти неактуальні. Найважливіша функція цієї сторінки - призначення модераторів (сторінка редагування (рис. 3.21)). Адміністратор використовує динамічний пошук, щоб знайти співробітника університету (виключаючи тих, хто вже призначений, завдяки SQL-фільтру NOT IN) і

надати йому право ставити відмітки від імені конкретного підрозділу. Тільки призначені тут працівники отримають доступ до сторінки "Відмітки" для цього підрозділу.

Головна / Обхідні листи / Структурні підрозділи для відміток в обхідних листах

Структурні підрозділи +						
Показані 1-20 із 84 записів.						
#	Структурний підрозділ	Чи активний	Чи є деканом	Чи є випусковою кафедрою	Відображення в обхідному листі	Контактна інформація
1	Аерокосмічний факультет	Так	Ні	Ні		
2	Факультет авіації, електроніки та телекомунікацій	Так	Так	Ні		
3	Факультет екологічної безпеки, інженерії та технологій	Так	Так	Ні		
4	Факультет економіки та бізнес-адміністрування	Так	Так	Ні		
5	Факультет комп'ютерних наук та технологій	Так	Так	Ні		
6	Факультет лінгвістики та соціальних комунікацій	Так	Так	Ні		
7	Факультет міжнародних відносин	Так	Так	Ні		
8	Факультет наземних споруд і аеродромів	Так	Так	Ні		
9	Факультет транспорту, менеджменту і логістики	Так	Так	Ні		
10	Юридичний факультет	Так	Так	Ні		
11	Кафедра прикладної механіки та інженерії матеріалів	Так	Ні	Так		
12	Кафедра завадних двигунів	Так	Ні	Так		
13	Кафедра конструкції літальних апаратів	Так	Ні	Так		
14	Кафедра підтримання льотної придатності повітряних суден	Так	Ні	Так		
15	Кафедра гідрогазових систем	Так	Ні	Так		
16	Кафедра аеродинаміки та безпеки польотів літальних апаратів	Так	Ні	Так		
17	Кафедра загальної фізики	Так	Ні	Так		
18	Кафедра комп'ютеризованих спектроскопічних систем та технологій	Так	Ні	Так		
19	Кафедра автоматизації та енергоменеджменту	Так	Ні	Так		
20	Кафедра технологій аеропортів	Так	Ні	Так		

Рисунок 3.20 – Сторінка «Структурні підрозділи»

Головна / Обхідні листи / Структурні підрозділи для відміток в обхідних листах / Аерокосмічний факультет / Редагування

Редагування структурного підрозділу

Структурний підрозділ

Аерокосмічний факультет

Чи активний

Так

Чи для всіх студентів

Ні

Чи є випусковою кафедрою

Ні

Чи є деканом

Ні

Відображення в обхідному листі

Контактна інформація

Відповідальні особи

Оберіть особу...

testuser (testuser)

Маслаков Дмитро Дмитрович (diguni_mdd)

Структурні підрозділи

Оберіть підрозділ...

Структурні підрозділи відсутні

Зберегти
Відмінити

Рисунок 3.21 – Сторінка редагування

3.3. Тестування роботи розробленого застосунку

Тестування є обов'язковим етапом розробки програмного забезпечення, що дозволяє виявити дефекти та переконатися у відповідності системи заявленим функціональним вимогам. У межах даної кваліфікаційної роботи проводилось функціональне тестування розроблених модулів «Обхідні листи» та «Студмістечко» методом «чорної скриньки» (Black Box Testing). Суть цього методу полягає в тому, що тестувальник взаємодіє із системою виключно

через її зовнішній інтерфейс, не маючи доступу до внутрішньої структури коду, та перевіряє відповідність реакцій системи очікуваним результатам для заданих вхідних умов.

Таблиця 3.1 – Результати функціонального тестування

ID	Модуль	Сценарій	Очікуваний результат	Фактичний результат	Статус
ТС-01	Обхідні листи	Відкриття списку структурних підрозділів авторизованим адміністратором	Відображено перелік активних і неактивних підрозділів	Перелік завантажено без помилок	Пройдено
ТС-02	Обхідні листи	Створення підрозділу з коректними даними	Новий запис збережено і відображено у списку	Запис створено, виконано перехід на сторінку редагування	Пройдено
ТС-03	Обхідні листи	Видалення підрозділу, з яким пов'язані відмітки	Видалення заблоковано, показано повідомлення	Операцію заблоковано	Пройдено
ТС-04	Обхідні листи	Повторне призначення того самого модератора	Дублювання заблоковано	Система відобразила повідомлення про наявний запис	Пройдено
ТС-05	Обхідні листи	Виставлення позитивної відмітки	Статус, час і користувача збережено	Дані збережено та відображено	Пройдено
ТС-06	Обхідні листи	Негативна відмітка без коментаря	Збереження заблоковано	Відображено помилку валідації	Пройдено

ТС-07	Студмістечко	Поселення на вільне ліжко	Створено активний запис про проживання	Запис створено, ліжко позначено як зайняте	Пройдено
ТС-08	Студмістечко	Поселення без обов'язкових даних	Збереження заблоковано	Відображено повідомлення біля обов'язкових полів	Пройдено
ТС-09	Студмістечко	Переселення на зайняте ліжко	Операцію заблоковано	Система не створила новий запис	Пройдено
ТС-10	Студмістечко	Обмін місцями між двома мешканцями	Старі записи закрито, два нові створено в одній транзакції	Обмін виконано без часткового оновлення даних	Пройдено
ТС-11	Студмістечко	Зміна типу кімнати за наявності мешканців	Операцію заблоковано	Відображено повідомлення про необхідність виселення	Пройдено
ТС-12	Студмістечко	Редагування архівного запису	Редагування недоступне	Архівний запис відкрито лише для перегляду	Пройдено

Для кожного функціонального блоку було складено тест-кейси, які охоплюють як позитивні сценарії (коректна поведінка системи при правильних вхідних даних), так і негативні (поведінка системи при некоректних або граничних вхідних даних).

Тестування функціоналу управління обхідними листами охопило перевірку адміністративних можливостей, роботу механізму виставлення відміток, а також коректність відображення даних для студента.

Першим кроком перевірялося відображення списку структурних підрозділів. Авторизований адміністратор переходить на відповідну сторінку модуля — система коректно завантажує таблицю з переліком усіх підрозділів (активних та неактивних), що підтвердило правильність роботи контролера та пошукової моделі.

Наступним тестом перевірялося створення нового структурного підрозділу. При заповненні форми коректними даними та збереженні новий підрозділ одразу з'явився у загальному переліку, а система виконала перенаправлення на сторінку редагування — поведінка відповідає очікуваній.

Для перевірки захисту від некоректного видалення була здійснена спроба видалити підрозділ, для якого вже існують відмітки студентів. Система заблокувала операцію та відобразила відповідне повідомлення про помилку, що підтверджує наявність захисного механізму на рівні бізнес-логіки.

Перевірка механізму призначення модераторів показала, що при введенні імені співробітника у поле пошуку система динамічно виводить відфільтрований список, а після вибору — успішно прив'язує користувача до підрозділу. При спробі призначити вже доданого модератора повторно система повертає відповідне повідомлення про дублювання і блокує запис, що є коректною поведінкою.

Центральний функціонал модуля — виставлення відміток — тестувався у двох сценаріях. У позитивному сценарії модератор натискає на статус студента, підтверджує проходження, і система миттєво оновлює іконку на зелену галочку, фіксуючи час та ідентифікатор виконавця. У негативному сценарії модератор вказує, що студент не пройшов підрозділ, і додає коментар із причиною — статус змінюється на відповідний, а коментар відображається при наведенні курсора на іконку. Спроба залишити відмітку «не виконано» з

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

порожнім полем коментаря була заблокована системою з виведенням повідомлення про обов'язковість заповнення.

Перевірка студентського кабінету підтвердила, що авторизований студент коректно бачить повний перелік підрозділів маршруту свого обхідного листа з актуальними статусами (виконано / не виконано / ще не підписано). Перегляд архіву відміток за минулий рік показав, що система правильно фільтрує дані за вибраним навчальним роком і не дозволяє редагувати архівні записи. Нарешті, тестування сторінки «Порядок підрозділів» підтвердило, що переміщення записів вгору та вниз відбувається без перезавантаження сторінки, а спроба перемістити перший елемент ще вище коректно блокується.

Тестування модуля управління гуртожитками охопило перевірку ієрархічної структури даних, процесу поселення та виселення мешканців, а також складних операцій переселення та обміну між мешканцями.

При відкритті головної сторінки модуля авторизованим адміністратором система коректно відобразила список гуртожитків із загальною інформацією — кількість кімнат, наявних місць та поточна заповненість. Перегляд сторінки конкретної кімнати показав, що система правильно відображає перелік ліжок із зазначенням статусу кожного (зайнято, вільно або заброньовано) та ПІБ мешканця.

Тестування процесу поселення включало два сценарії. У позитивному — комендант обирає вільне ліжко, заповнює форму з коректними даними і зберігає запис: система успішно реєструє поселення і оновлює статус ліжка. У негативному — спроба зберегти форму без вибору ліжка та дати заїзду призвела до виведення повідомлень про помилки валідації для кожного обов'язкового поля, що підтверджує коректну роботу правил моделі HostelRoomResident.

Найбільш відповідальною операцією, яка потребувала ретельної перевірки, є переселення студента. При виборі мешканця та вказуванні нового вільного ліжка система архівує попередній запис про проживання (фіксує дату

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

виселення) і одночасно створює новий запис із датою переселення — таким чином повна історія руху студента по гуртожитках зберігається. Спроба переселити студента на вже зайняте ліжко коректно заблокована з виведенням відповідного повідомлення.

Особливу увагу при тестуванні приділено операції обміну мешканцями між кімнатами, яка передбачає одночасну модифікацію чотирьох записів у базі даних. Було підтверджено, що після успішного обміну обидва старих записи архівуються, а нові записи на протилежних ліжках створюються коректно, зберігаючи при цьому оригінальні дати заїзду та всі пов'язані дані. Атомарність операції гарантується механізмом транзакцій, тому часткового збою даних зафіксовано не було.

Тестування операції виселення підтвердило, що після встановлення дати виселення запис набуває статусу «не проживає», ліжко відображається як вільне, а вся інформація зберігається в архіві для можливого перегляду в майбутньому.

Для перевірки захисту від логічних конфліктів було здійснено спробу змінити стать кімнати (з чоловічої на жіночу), у якій наразі проживають мешканці. Система заблокувала зміну та повідомила, що операція можлива лише після виселення всіх мешканців. Аналогічна зміна для порожньої кімнати пройшла успішно. Крім того, пошук конкретного студента за ПІБ повернув повну картку з інформацією про гуртожиток, кімнату та ліжко, що підтвердило коректну роботу SQL-запиту з множинними з'єднаннями таблиць.

Висновки до третього розділу

Проведене тестування охопило всі ключові функціональні блоки розроблених модулів «Обхідні листи» та «Студмістечко». У межах виконаного набору функціональних тестів фактичні результати відповідали очікуванім. Дефектів, які блокують виконання перевірених сценаріїв, не виявлено. Водночас отримані результати стосуються лише наведених функціональних сценаріїв і не замінюють навантажувальне тестування, аудит безпеки та тривалу дослідну експлуатацію.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Граничні та некоректні вхідні дані (порожні обов'язкові поля, спроби дублювання, операції над зайнятими ресурсами) коректно опрацьовуються системою з виведенням інформативних повідомлень для користувача. Складні атомарні операції (обмін мешканцями, переселення) зберігають цілісність даних завдяки механізму транзакцій бази даних. Архівні записи захищені від редагування, що гарантує достовірність та незмінність історичних даних.

Отримані результати тестування підтверджують, що розроблене програмне забезпечення коректно виконує перевірені функціональні сценарії та може бути використане як основа для пілотного впровадження після проведення додаткового навантажувального тестування, аудиту безпеки та перевірки в середовищі експлуатації.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У третьому розділі кваліфікаційної роботи було вирішено комплекс завдань, пов'язаних із розгортанням, документуванням та верифікацією розроблених програмних модулів «Обхідні листи» та «Студмістечко» у складі інформаційної системи управління освітніми процесами університету.

Архітектурне рішення системи базується на клієнт-серверній моделі з монолітним серверним застосунком на базі Yii2. У серверному застосунку зосереджено бізнес-логіку модулів «Обхідні листи» та «Студмістечко», а реляційна СУБД MySQL використовується для централізованого зберігання взаємопов'язаних даних. Застосування патерну MVC забезпечує структурованість коду та розмежування відповідальності між компонентами системи.

У ході опису інтерфейсу модуля «Обхідні листи» детально розглянуто шість ключових сторінок, їхні функції та права доступу для різних ролей. Встановлено, що система реалізує чітке розмежування доступу: адміністратори керують конфігурацією підрозділів та призначенням модераторів, рядові працівники структурних підрозділів мають доступ виключно до механізму виставлення відміток у рамках своєї зони відповідальності, а студенти переглядають статус власного обхідного листа у режимі реального часу. Архівні розділи системи є захищеними від редагування, що гарантує незмінність та достовірність накопичених історичних даних попередніх випусків.

Аналогічно розглянуто інтерфейс модуля «Студмістечко», що охоплює повний цикл управління житловим фондом університету. Описано принципи роботи сторінок перегляду ієрархії гуртожитків (будівля – кімната – ліжко-місце), форм поселення та виселення мешканців, а також складних операцій переселення і обміну. Встановлено, що комендант отримує повний інструментарій для ведення актуального реєстру мешканців: система зберігає повну хронологію переміщень кожного студента, не дозволяє виконувати логічно суперечливі дії (наприклад, поселення на зайняте ліжко або зміну типу

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

кімнати за наявності мешканців), а пошук за ПІБ дозволяє миттєво отримати зведену інформацію про місце проживання конкретного здобувача освіти.

Проведене функціональне тестування методом «чорної скриньки» охопило всі ключові сценарії роботи з обома модулями. Для модуля «Обхідні листи» підтверджено коректну роботу механізмів управління структурними підрозділами, блокування дублювання модераторів, виставлення відміток з фіксацією аудиту та динамічного керування порядком відображення підрозділів. Для модуля «Студмістечко» верифіковано процеси поселення, переселення та виселення мешканців, захист від логічних конфліктів (заборона зміни типу кімнати при наявності мешканців, заборона поселення на зайняте ліжко), а також атомарність складних операцій обміну на рівні транзакцій бази даних. За результатами всіх проведених тестів критичних дефектів виявлено не було, а фактична поведінка системи у всіх перевірених сценаріях відповідала очікуваній [11, 36].

Таким чином, результати третього розділу підтверджують, що розроблені модулі є повністю функціональними, відмовостійкими і коректно інтегрованими в загальну архітектуру інформаційної системи університету, що підтверджує можливість переходу до етапу пілотної експлуатації та подальшої перевірки системи в умовах реального навантаження.

		Лаговський А. В.			ПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Patterns of Enterprise Application Architecture / Martin Fowler. – Addison-Wesley Professional, 2002. – 560 с.
2. Docker Compose Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.docker.com/compose/>
3. Цифровізація у вищій освіті: інституційні підходи до викладання та навчання / Ю. Б. Семеняко, І. В. Брюховецька, О. Є. Бохонько. – Запоріжжя : Класичний приватний університет, 2023. – 186 с.
4. PHP Manual [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/manual/en/>
5. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems / Martin Kleppmann. – O'Reilly Media, 2017. – 616 с.
6. Про вищу освіту : Закон України від 01.07.2014 № 1556-VII [Електронний ресурс] – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/laws/show/1556-18>
7. MySQL 8.4 Reference Manual [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.4/en/>
8. Design Patterns: Elements of Reusable Object-Oriented Software / Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. – Addison-Wesley Professional, 1994. – 416 с.
9. Цифрова трансформація освіти і науки / Міністерство освіти і науки України [Електронний ресурс] – Режим доступу до ресурсу: <https://mon.gov.ua/tag/tsifrova-transformatsiya-osviti-i-nauki>
10. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics / Jennifer Niederst Robbins. – 5th ed. – O'Reilly Media, 2018. – 808 с.
11. The Art of Software Testing / Glenford J. Myers, Corey Sandler, Tom Badgett. – 3rd ed. – Wiley, 2011. – 256 с.

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ІЗ	Арк.
		Вакалюк Т. А.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

- 12.Canvas LMS : official source code repository [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/instructure/canvas-lms>
- 13.The Definitive Guide to Yii 2.0 [Електронний ресурс] – Режим доступу до ресурсу: <https://www.yiiframework.com/doc/guide/2.0/en>
- 14.JavaScript Guide – MDN Web Docs [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
- 15.MySQL Crash Course / Ben Forta. – 2nd ed. – Addison-Wesley Professional, 2024. – 304 с.
- 16.Moodle Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.moodle.org/>
- 17.Software Requirements / Karl Wieggers, Joy Beatty. – 3rd ed. – Microsoft Press, 2013. – 672 с.
- 18.Bray T. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259 [Електронний ресурс] – Режим доступу до ресурсу: <https://www.rfc-editor.org/rfc/rfc8259>
- 19.PHP Cookbook: Modern Code Solutions for Professional Developers / Eric A. Mann. – O'Reilly Media, 2023. – 431 с.
- 20.Docker Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.docker.com/>
- 21.OpenEduCat Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://doc.openeducat.org/>
- 22.SQL Antipatterns: Avoiding the Pitfalls of Database Programming / Bill Karwin. – Pragmatic Bookshelf, 2010. – 328 с.
- 23.phpMyAdmin Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.phpmyadmin.net/>
- 24.Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 / Robin Nixon. – 6th ed. – O'Reilly Media, 2021. – 826 с.
- 25.Yii 2.0 API Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.yiiframework.com/doc/api/2.0>

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ІЗ	Арк.
		Вакалюк Т. А.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

- 26.Clean Code: A Handbook of Agile Software Craftsmanship / Robert C. Martin. – Prentice Hall, 2008. – 464 с.
- 27.Security: Authorization. The Definitive Guide to Yii 2.0 [Электронный ресурс] – Режим доступа до ресурсу: <https://www.yiiframework.com/doc/guide/2.0/en/security-authorization>
- 28.UML Distilled: A Brief Guide to the Standard Object Modeling Language / Martin Fowler. – 3rd ed. – Addison-Wesley Professional, 2003. – 208 с.
- 29.Unit4 Higher Education ERP Software [Электронный ресурс] – Режим доступа до ресурсу: <https://www.unit4.com/industries/higher-education-erp-software>
- 30.Redis in Action / Josiah L. Carlson. – Manning Publications, 2013. – 320 с.
- 31.PHP & MySQL: Server-side Web Development / Jon Duckett. – Wiley, 2022. – 672 с.
- 32.SQL Injection Prevention Cheat Sheet / OWASP Foundation [Электронный ресурс] – Режим доступа до ресурсу: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
- 33.Creating Software with Modern Diagramming Techniques: Build Better Software with Mermaid / Ashley Peacock. – Pragmatic Bookshelf, 2023. – 156 с.
- 34.Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design / Michael J. Hernandez. – 4th ed. – Addison-Wesley Professional, 2021. – 640 с.
- 35.Compose File Reference / Docker Documentation [Электронный ресурс] – Режим доступа до ресурсу: <https://docs.docker.com/reference/compose-file/>
- 36.Certified Tester Foundation Level Syllabus. Version 4.0.1 / International Software Testing Qualifications Board [Электронный ресурс] – Режим доступа до ресурсу: https://istqb.org/wp-content/uploads/2024/11/ISTQB_CTFL_Syllabus_v4.0.1.pdf

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ІЗ	Арк.
		Вакалюк Т. А.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

- 37.Про Єдину державну електронну базу з питань освіти : Постанова Кабінету Міністрів України від 15.07.2025 № 872 [Електронний ресурс] – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/laws/show/872-2025-п>
- 38.Unit4 Student Management – Cloud Service Description. Version 1.4, May 2020 [Електронний ресурс] – Режим доступу до ресурсу: <https://info.unit4.com/rs/400-HYB-295/images/Unit4-Student-Management-SaaS-Service-Description-v.1.4-May-2020-%28EN%29.pdf>

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

<?php

namespace backend\controllers\circularLetters;

use backend\models\circularLetters\StudentSearch;
use common\models\circularLetters\BypassSheetMark;
use
common\services\circularLetters\CircularLettersStudentService;
use
common\services\circularLetters\CircularLettersUniversityStructureService;

use DateTime;
use DateTimeZone;
use Yii;
use yii\data\ArrayDataProvider;
use yii\filters\AccessControl;
use yii\filters\VerbFilter;
use yii\web\Controller;
use yii\web\Response;

class BypassSheetMarkController extends Controller
{
    public function behaviors(): array
    {
        return [
            'verbs' => [
                'class' => VerbFilter::class,
                'actions' => [
                    'delete' => ['post'],
                ],
            ],
            'access' => [
                'class' => AccessControl::class,
                'rules' => [
                    [
                        'allow' => true,

```

		Лаговський А. В.			ІПЗ.КР.Б - 121- 26 - ІЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

        'actions' => ['index', 'view', 'create',
'update',
                                'delete',      'update-mark',
'report'],

        'roles' => ['@']
    ],
    ['allow' => false]
]
]
];
}

public function actionIndex(): mixed
{
    $students =                                     =
CircularLettersStudentService::getFormattedStudents();
    $structures = CircularLettersUniversityStructureService
::getBelondingUniversityStructures();

    $searchModel = new StudentSearch();
    $searchModel->load(\Yii::$app->request->queryParams);
    $filteredStudents = $searchModel->filter($students);
    setlocale(LC_ALL, 'uk_UA.UTF-8');

    $dataProvider = new ArrayDataProvider([
        'allModels' => $filteredStudents,
        'sort' => [
            'attributes' => [
                'person_fio' => [
                    'asc' => ['person_fio' => SORT_ASC],
                    'desc' => ['person_fio' => SORT_DESC],
                ],
            ],
            'sortFlags' => SORT_LOCALE_STRING,
        ],
        'pagination' => ['pageSize' => 15],
    ]);

    return $this->render('index', [

```

		Лаговський А. В.			ІІЗ.КР.Б - 121- 26 - ІІЗ	Арк.
		Вакалюк Т. А.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
        'structures'   => $structures,
    ));
}

    public function actionUpdateMark(int $education_id, int
    $bypass_structure_id): array
    {
        Yii::$app->response->format = Response::FORMAT_JSON;

        $data = json_decode(Yii::$app->request-
        >getRawBody(), true);

        $mark = $data['mark'] ?? null;
        $description = $data['description'] ?? null;

        $sheetMark =
        CircularLettersUniversityStructureService::findBypassSheetMarkOf(
            $education_id,
            $bypass_structure_id
        );

        if (is_null($sheetMark)) {
            $sheetMark = new BypassSheetMark();
            $sheetMark->student_education_id = $education_id;
            $sheetMark->bypass_sheet_university_structure_id =
            $bypass_structure_id;
            if (!is_null($description)) {
                $sheetMark->mark = false;
            }
        }

        if (!is_null($mark)) { $sheetMark->mark =
        $mark; }

        if (!is_null($description)) { $sheetMark->description =
        $description; }

        $sheetMark->user_id = Yii::$app->user->id;
    }
}

```

		Лаговський А. В.			ІІЗ.КР.Б - 121- 26 - ІІЗ	Арк.
		Вакалюк Т. А.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $date = new DateTime('now', new DateTimeZone('Etc/GMT-
2'));

        $sheetMark->datetime = Yii::$app->formatter->asDate($date,
'php:Y-m-d H:i:s');

        $sheetMark->save(false);

        $sheetDateMark
=
CircularLettersUniversityStructureService::getDataOfUserMark(
        $education_id,
        $bypass_structure_id
    );

    return [
        'success' => true,
        'fio'      => $sheetDateMark['fio'],
        'datetime' => Yii::$app->formatter->asDatetime($date,
'php:d.m.Y H:i'),
    ];
}

    public function actionReport(): string
    {
        $students
=
CircularLettersStudentService::getFormattedStudents(false);

        $searchModel = new StudentSearch();
        $searchModel->load(\Yii::$app->request->queryParams);
        $filteredStudents = $searchModel->filter($students);
        setlocale(LC_ALL, 'uk-UA.UTF-8');

        $dataProvider = new ArrayDataProvider([
            'allModels' => $filteredStudents,
            'sort'      => ['attributes' => ['person_fio' =>
['asc' => ['person_fio' => SORT_ASC],
'desc' => ['person_fio' => SORT_DESC]]],
            'sortFlags' => SORT_LOCALE_STRING],
            'pagination' => ['pageSize' => 15],
        ]);
    }

```

		Лаговський А. В.			ІІЗ.КР.Б - 121- 26 - ІІЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

        $allRecords    = $dataProvider->getModels();
        $studentLetters = [];
        foreach ($allRecords as $record) {
            $educationId = $record['education_id'];
            $studentLetters[$educationId] =
CircularLettersStudentService

::getLetterForStudent($educationId);
        }

        return $this->render('report', [
            'searchModel'    => $searchModel,
            'dataProvider'    => $dataProvider,
            'studentLetters' => $studentLetters,
        ]);
    }
}

<?php

namespace backend\controllers\circularLetters;

use common\models\circularLetters\BypassSheetMark;
use common\models\circularLetters\BypassSheetStudent;
use common\models\circularLetters\BypassSheetUniversityStructure;
use common\models\circularLetters\BypassSheetUser;
use common\models\User;
use common\services\helper\ErrorMessageHelperEnum;
use common\services\helper\MessageHelperEnum;
use Yii;
use yii\web\Controller;
use yii\web\Response;

class BypassSheetController extends Controller
{
    public function actionUpdate($id): mixed
    {
        $model = $this->findModel($id);

```

		Лаговський А. В.			ІІЗ.КР.Б - 121- 26 - ІІЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

        $title = $model->display_name ?: $model->
>universitySettingsUniversityStructure->title;

        if ($model->load(Yii::$app->request->post()) && $model->
>save(false)) {}

        $bypassUserIds = self::getBypassUserIds($id);
        $users = User::find()
            ->select(['id', 'fio', 'username'])
            ->where(['id' => $bypassUserIds])
            ->asArray()->all();

        return $this->render('update', [
            'title' => $title,
            'model' => $model,
            'users' => $users,
        ]);
    }

    public function actionAddUser(): array
    {
        Yii::$app->response->format = Response::FORMAT_JSON;
        $data = json_decode(Yii::$app->request->getRawBody(),
true);

        if (BypassSheetUser::find()->where([
            'user_id' => $data['userId'],
            'bypass_sheet_university_structure_id' =>
$data['structureId']
        ])->exists()) {
            return [
                'success' => false,
                'message' =>
ErrorMessageHelperEnum::USER_ALREADY_ADDED->value,
            ];
        }

        $bypassSheetUser = new BypassSheetUser();
        $bypassSheetUser->user_id = $data['userId'];

```

		Лаговський А. В.			ІІЗ.КР.Б - 121- 26 - ІІЗ	Арк.
		Вакалюк Т. А.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $bypassSheetUser->bypass_sheet_university_structure_id =
$data['structureId'];
        return ['success' => $bypassSheetUser->save()];
    }

    public function actionDeleteUser(): array
    {
        Yii::$app->response->format = Response::FORMAT_JSON;
        $data = json_decode(Yii::$app->request->getRawBody(),
true);

        $bypassSheetUser = BypassSheetUser::find()->where([
            'bypass_sheet_university_structure_id' =>
$data['structureId'],
            'user_id' => $data['userId']
        ])->one();

        if ($bypassSheetUser && $bypassSheetUser->delete() !==
false) {
            return ['success' => true];
        }
        return ['success' => false];
    }

    public function actionGetUserList(int $structureId, string
$query = ''): array
    {
        Yii::$app->response->format = Response::FORMAT_JSON;
        $existingUserIds = $this->getBypassUserIds($structureId);

        $queryBuilder = User::find()
            ->select(['id', 'fio', 'username'])
            ->andFilterWhere(['like', 'fio', $query])
            ->orWhereWhere(['like', 'username', $query])
            ->asArray();

        if (!empty($existingUserIds)) {
            $queryBuilder->andWhere(['not in', 'id',
$existingUserIds]);

```

		Лаговський А. В.			ІІЗ.КР.Б - 121- 26 - ІІЗ	Арк.
		Вакалюк Т. А.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    return array_map(
        fn ($user) => ['id' => $user['id'], 'label' =>
$user['fio']." (".$user['username'].")"],
        $queryBuilder->all()
    );
}

public function actionDelete($id): Response
{
    $transaction = Yii::$app->db->beginTransaction();
    try {
        $bypassSheetMark = BypassSheetMark::findOne([
            'bypass_sheet_university_structure_id' => $id
        ]);
        if ($bypassSheetMark) {
            Yii::$app->session->setFlash(
                'bypass-sheet-error',

                ErrorMessageHelperEnum::BYPASS_SHEET_STRUCTURE_HAS_MARKS->value
            );
        } else {

            BypassSheetStudent::deleteAll(['bypass_sheet_university_structure_id'
=> $id]);

            $this->findModel($id)->delete();
            $transaction->commit();
        }
    } catch (\Exception $e) {
        $transaction->rollBack();
        Yii::$app->session->setFlash('bypass-sheet-error',
            'Помилка під час видалення підрозділу: ' . $e-
>getMessage());
    }
    return $this->redirect(['index']);
}

```

```

        private static function getBypassUserIds(int $structureId):
array
    {
        return BypassSheetUser::find()
            ->select('user_id')
            ->where(['bypass_sheet_university_structure_id' =>
$structureId])
            ->column();
    }
}

```

<?php

```

namespace common\services\circularLetters;

use
common\models\circularLetters\base\BypassSheetUniversityStructure;
use common\models\circularLetters\BypassSheetUser;
use yii\db\ActiveQuery;
use yii\db\Query;

class CircularLettersStudentService extends
AbstractCircularLettersStudentService
{
    protected static function isDenearyExist(bool $isActive): bool
    {
        return $isActive
            ? BypassSheetUniversityStructure::find()
                ->where(["is_deanery" => 1])
                ->andWhere(["is_active" => 1])
                ->exists()
            : BypassSheetUniversityStructure::find()
                ->where(["is_deanery" => 1])
                ->exists();
    }

    protected static function isGraduationDepartmentExist(bool
$isActive): bool
    {

```

| | | | | | | |
|------|------|------------------|--------|------|-------------------------|------|
| | | Лаговський А. В. | | | ІПЗ.КР.Б - 121- 26 - ІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 90 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        return $isActive
        ? BypassSheetUniversityStructure::find()
            ->where(["is_graduation_department" => 1])
            ->andWhere(["is_active" => 1])
            ->exists()
        : BypassSheetUniversityStructure::find()
            ->where(["is_graduation_department" => 1])
            ->exists();
    }

    public static function getAllStructureQuery(int $educationId,
bool $isActive = true): Query
    {
        $bypassSheetUniversityStructure =

\common\models\circularLetters\BypassSheetUniversityStructure::tableNa
me();

        $query = parent::getAllStructureBase($educationId);

        if ($isActive) {
            $query->
>andWhere(["$bypassSheetUniversityStructure.is_active" => 1]);
        }

        return $query;
    }

    protected static function getBypassUserStructures(int $userId,
bool $isReport): array
    {
        $bypassStructureTable =
BypassSheetUniversityStructure::tableName();
        $bypassUserTable = BypassSheetUser::tableName();
        $query = BypassSheetUniversityStructure::find();

        if ($isReport) {
            $query->innerJoin(
                $bypassUserTable,

```

| | | | | | | |
|------|------|------------------|--------|------|-------------------------|------|
| | | Лаговський А. В. | | | ІПЗ.КР.Б - 121- 26 - ІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 91 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

"$bypassUserTable.bypass_sheet_university_structure_id"           =
$bypassStructureTable.id"
        )->andWhere(["$bypassUserTable.user_id" => $userId]);
    }

    $query->andWhere(["$bypassStructureTable.is_active" =>
1]);

    return $query->all();
}
}

```

<?php

```

namespace backend\controllers\hostel;

use common\models\hostel\HostelRoom;
use common\models\hostel\HostelRoomBed;
use common\models\hostel\HostelRoomResident;
use common\services\helper\ErrorMessageHelperEnum;
use Throwable;
use Yii;
use yii\web\Response;

class HostelViewController extends SecureController
{
    // --- Отримання кімнат з вільними місцями ---
    public function actionGetRooms(string $hostelId, ?string
$genderId = null): array
    {
        Yii::$app->response->format = Response::FORMAT_JSON;

        $query = HostelRoom::find()
            ->select(['hr.id as room_id', 'hr.name as room_name'])
            ->from(['h' => 'hostel'])
            ->innerJoin(['hr' => 'hostel_room'], 'h.id =
hr.hostel_id')
            ->innerJoin(['hrb' => 'hostel_room_bed'], 'hr.id =
hrb.hostel room id')

```

| | | | | | | |
|------|------|------------------|--------|------|-------------------------|------|
| | | Лаговський А. В. | | | ІПЗ.КР.Б - 121- 26 - ІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 92 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        ->leftJoin(['hrr' => 'hostel_room_resident'],
            "hrb.id = hrr.hostel_room_bed_id AND
            (hrr.is_residing = 1 OR NOT EXISTS (
                SELECT 1 FROM hostel_room_resident hrr2
                WHERE hrr2.person_id = hrr.person_id AND
hrr2.is_residing = 1
            )))
        ->where(['h.id' => $hostelId, 'h.is_active' => 1,
            'hr.is_active' => 1, 'hrb.is_reserved' => 0,
'hrr.is_active' => 1]);

        if ($genderId !== null) {
            $query->andWhere(['hr.gender_id' => $genderId]);
        }

        $availableRooms = $query
            ->groupBy(['hr.id', 'hr.name'])
            ->having('COUNT(DISTINCT hrb.id) > COUNT(DISTINCT CASE
WHEN hrr.is_residing = 1 THEN hrr.hostel_room_bed_id END)')
            ->orderBy(['hr.name' => SORT_ASC])
            ->asArray()->all();

        return ['success' => true, 'data' => array_map(fn($i) =>
['id' => $i['room_id'], 'name' => $i['room_name']], $availableRooms)];
    }

    // --- Поселення студента ---
    public function actionCheckInHostel(): Response
    {
        $request          = Yii::$app->request;
        $hostelRoomBed     = HostelRoomBed::findOne($request-
>post('hostel_room_bed_id'));

        $hostelRoomResident = new HostelRoomResident();
        $hostelRoomResident->hostel_room_bed_id = $hostelRoomBed-
>id;

        $hostelRoomResident->person_type_id      = $request-
>post('person_type_id');

```

| | | | | | | |
|------|------|------------------|--------|------|--------------------------|------|
| | | Лаговський А. В. | | | ІІЗ.КР.Б - 121- 26 - ІІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 93 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        $hostelRoomResident->education_id            = $request->
>post('education_id');

        $hostelRoomResident->person_fio              = $request->
>post('person_fio');

        $hostelRoomResident->check_in_date            = $request->
>post('check_in_date');

        $hostelRoomResident->comment                  = $request->
>post('comment');

        $hostelRoomBed->is_reserved = 0;

        if ($hostelRoomResident->save(false) && $hostelRoomBed->
>save(false)) {
            return $this->asJson(['success' => true, 'data' =>
$hostelRoomResident]);
        }
        return $this->asJson(['success' => false, 'errors' =>
$hostelRoomResident->getErrors()]);
    }

    // --- Виселення студента ---
    public function actionCheckOutHostel(): Response
    {
        $request = Yii::$app->request;

        $hostelRoomResident =
HostelRoomResident::findOne($request->
>post('hostel_room_resident_id'));

        $hostelRoomResident->check_out_date            = $request->
>post('check_out_date');

        $hostelRoomResident->is_residing               = false;

        if ($hostelRoomResident->save(false)) {
            return $this->asJson(['success' => true, 'data' =>
$hostelRoomResident]);
        }
        return $this->asJson(['success' => false, 'errors' =>
$hostelRoomResident->getErrors()]);
    }
}

```

| | | | | | | |
|------|------|------------------|--------|------|-------------------------|------|
| | | Лаговський А. В. | | | ІПЗ.КР.Б - 121- 26 - ІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 94 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

// --- Переселення студента ---
public function actionRelocateInHostel(): Response
{
    $request          = Yii::$app->request;
    $residentId       = $request->post('resident_id');
    $newBedId         = $request->post('new_hostel_room_bed_id');
    $comment          = $request->post('comment');
    $relocationDate   = $request->post('relocation_date');

    $currentResident  =
HostelRoomResident::findOne($residentId);

    if (! $this->completeResidentRecord($currentResident,
    $relocationDate)) {
        return $this->asJson(['success' => false, 'errors' =>
    $currentResident->getErrors()]);
    }

    $newResident      =
    $this->createNewResidentRecord($currentResident, $newBedId, $relocationDate,
    $comment);

    if ($newResident->save(false)) {
        return $this->asJson(['success' => true, 'data' =>
    $newResident]);
    }
    return $this->asJson(['success' => false, 'errors' =>
    $newResident->getErrors()]);
}

// --- Обмін мешканцями між кімнатами ---
public function actionSwapResidents(): Response
{
    $request          = Yii::$app->request;
    $resident1        = HostelRoomResident::findOne($request->post('resident_id_1'));
    $resident2        = HostelRoomResident::find()->where([
        'hostel room bed id' => $request->post('room bed id'),

```

| | | | | | | |
|------|------|------------------|--------|------|-------------------------|------|
| | | Лаговський А. В. | | | ІПЗ.КР.Б - 121- 26 - ІЗ | Арк. |
| | | Вакалюк Т. А. | | | | |
| Змн. | Арк. | № докум. | Підпис | Дата | | 95 |

```

        'is_residing'          => 1
    ]->one();
    $swapDate = $request->post('swap_date');
    $comment  = $request->post('comment');

    $transaction = Yii::$app->db->beginTransaction();

    try {
        if          (!$this->completeResidentRecord($resident1,
$swapDate) ||
                !$this->completeResidentRecord($resident2,
$swapDate)) {
            throw new
\ErrorException(ErrorMessageHelperEnum::FAILED_TO_COMPLETE_RESIDENT_RECORDS
->value);
        }

        $newResident1          =          $this-
>createNewResidentRecord($resident1,    $resident2->hostel_room_bed_id,
$swapDate, $comment);
        $newResident2          =          $this-
>createNewResidentRecord($resident2,    $resident1->hostel_room_bed_id,
$swapDate, $comment);

        if    ($newResident1->save(false)    &&    $newResident2-
>save(false)) {
            $transaction->commit();
            return $this->asJson(['success' => true]);
        }
        throw new
\ErrorException(ErrorMessageHelperEnum::FAILED_TO_SAVE_RESIDENT_RECORDS-
>value);

    } catch (Throwable $e) {
        $transaction->rollBack();
        return $this->asJson(['success' => false, 'message' =>
$e->getMessage()]);
    }
}

```

| | | | | | | |
|------|------|------------------|--------|------|--------------------------|------|
| | | Лаговський А. В. | | | ІІЗ.КР.Б - 121- 26 - ІІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 96 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

// --- Зміна типу кімнати ---
public function actionChangeGender(): Response
{
    $room = HostelRoom::findOne(Yii::$app->request->post('hostel_room_id'));

    $hasResidents = HostelRoomResident::find()
        ->innerJoin('hostel_room_bed',
            'hostel_room_resident.hostel_room_bed_id = hostel_room_bed.id')
        ->where(['hostel_room_bed.hostel_room_id' => $room->id, 'hostel_room_resident.is_residing' => true])
        ->exists();

    if ($hasResidents) {
        return $this->asJson([
            'success' => false,
            'message' =>
                ErrorMessageHelperEnum::CANNOT_CHANGE_ROOM_TYPE_WITH_OCCUPANTS->value,
        ]);
    }

    $genders = Genders::find()->select('id')->asArray()->column();

    $room->gender_id = ($room->gender_id == $genders[0]) ?
        $genders[1] : $genders[0];

    if ($room->save(false)) {
        return $this->asJson(['success' => true,
            'new_gender_id' => $room->gender_id]);
    }

    return $this->asJson(['success' => false, 'errors' =>
        $room->getErrors()]);
}

// --- Допоміжні методи ---
protected function completeResidentRecord(HostelRoomResident
    $resident, string $checkoutDate): bool
{

```

| | | | | | | |
|------|------|------------------|--------|------|-------------------------|------|
| | | Лаговський А. В. | | | ІПЗ.КР.Б - 121- 26 - ІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 97 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        $resident->is_residing    = 0;
        $resident->check_out_date = $checkoutDate;
        return $resident->save(false);
    }

    protected function createNewResidentRecord(
        HostelRoomResident $oldResident,
        int $newBedId,
        string $checkinDate,
        string $comment
    ): HostelRoomResident {
        $newResident          = new HostelRoomResident();
        $newResident->attributes = $oldResident->attributes;
        $newResident->id        = null;
        $newResident->isNewRecord = true;
        $newResident->hostel_room_bed_id = $newBedId;
        $newResident->check_in_date      = $checkinDate;
        $newResident->check_out_date     = null;
        $newResident->comment            = $comment;
        $newResident->is_residing        = 1;
        return $newResident;
    }
}

const fetchClearanceSheet = async () => {
    const url = `/student/bypass-sheet/get-bypass-sheet`;
    try {
        const response = await getJsonData(url);
        if (response.status === "success" && response.data &&
            Object.keys(response.data).length > 0) {
            displayClearanceLetter(response.data);
        } else {
            displayNoClearanceMessage();
        }
    } catch (error) {
        console.error("Error fetching data:", error);
        displayNoClearanceMessage();
    }
};

```

| | | | | | | |
|------|------|------------------|--------|------|--------------------------|------|
| | | Лаговський А. В. | | | ІІЗ.КР.Б - 121- 26 - ІІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 98 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

function displayClearanceLetter(data) {
    if (!data) return;

    const clearanceLetterBlock =
document.getElementById("clearanceSheetContainer");
    clearanceLetterBlock.innerHTML = "";

    data.forEach((item) => {
        const value = item.value;
        const contacts = item.contact_details ?? '';
        const iconClass = value === null ? 'fa fa-time' : (value ? 'fa
fa-check' : 'fa fa-close');

        const clearanceLetterItem = document.createElement("div");
        clearanceLetterItem.classList.add("clearance-item");
        clearanceLetterItem.dataset.status = `${value}`;

        if (item.value === 0) clearanceLetterItem.title =
"Заборгованість";
        if (item.value === null) clearanceLetterItem.title = "Відмітки
ще немає";
        if (item.value) clearanceLetterItem.title = "Усе
добре";

        clearanceLetterItem.innerHTML = `
            <div class="names">
                <span class="structure-name">${item.name}</span>
                <span class="contacts">${contacts}</span>
            </div>
            <div class="icons">
                ${value === 0 ? `<span class="fa fa-info-circle info-sign"
data-tooltip="${item.description}
?? 'Немає примітки'`></span>` : ''}
                <div class="status-icon" data-status-icon="${value}">
                    <span class="${iconClass}"></span>
                </div>
            </div>`;

        clearanceLetterBlock.appendChild(clearanceLetterItem);
    });
}

```

| | | | | | | |
|------|------|------------------|--------|------|-------------------------|------|
| | | Лаговський А. В. | | | ІПЗ.КР.Б - 121- 26 - ІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 99 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    });
}

document.addEventListener("DOMContentLoaded", () => {
    fetchClearanceSheet();

    const container =
document.getElementById('clearanceSheetContainer');
    container.addEventListener('click', (event) => {
        const target = event.target;
        if (target.classList.contains('info-sign')) {
            const description = target.getAttribute('data-tooltip') ||
'Немає примітки';
            copyToClipboard(description);
            showToast();
        }
    });

function copyToClipboard(text) {
    if (navigator.clipboard) {
        navigator.clipboard.writeText(text).catch(console.error);
    } else {
        const textarea = document.createElement('textarea');
        textarea.value = text;
        document.body.appendChild(textarea);
        textarea.select();
        try { document.execCommand('copy'); } catch (e) {}
        document.body.removeChild(textarea);
    }
}

function showToast() {
    const toast = document.getElementById('toast');
    toast.classList.add('show');
    setTimeout(() => toast.classList.remove('show'), 2000);
}

const displayNoClearanceMessage = () => {

```

| | | | | | | |
|------|------|------------------|--------|------|-------------------------|------|
| | | Лаговський А. В. | | | ІПЗ.КР.Б - 121- 26 - ІЗ | Арк. |
| | | Вакалюк Т. А. | | | | 100 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

const container =
document.getElementById("clearanceSheetContainer");
container.innerHTML = `
    <div class="no-semesters-message">
        
        <p>Обхідний лист ще недоступний...</p>
    </div>`;
};

```

| | | | | | | |
|------|------|------------------|--------|------|-------------------------|------|
| | | Лаговський А. В. | | | ІПЗ.КР.Б - 121- 26 - ПЗ | Арк. |
| | | Вакалюк Т. А. | | | | 101 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |