

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:

«Вебплатформа для підтримки соціальних зв'язків»

Виконав студент групи ІПЗп-22-1
ДУБРОВ Михайло Юрійович

Керівник роботи:
ЧИЖМОТРИЯ Олена Геннадіївна

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2026

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
«16» липня 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: ДУБРОВ Михайло Юрійович

Керівник роботи: Чижмотря Олена Геннадіївна

Тема роботи: «Вебплатформа для підтримки соціальних зв'язків»,
затверджена Наказом закладу вищої освіти від **«16» липня 2026 р., №396.с**

Вихідні дані для роботи: об'єктом дослідження є процес проектування та програмної реалізації вебплатформи для підтримки соціальних зв'язків, що базується на архітектурі динамічної обробки та представлення користувацького контенту; предметом дослідження є методи побудови реляційної структури бази даних, архітектурні рішення та програмні засоби PHP, MySQL, WAMP64 для забезпечення цілісності даних, швидкодії та ефективної комунікації між користувачами.

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	16.07.2026	16.07.2026
2	Чижмотря О.Г.	16.07.2026	16.07.2026
3	Чижмотря О.Г.	16.07.2026	16.07.2026

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра на тему “Вебплатформа для підтримки соціальних зв’язків” містить 84 сторінки друкованого тексту. Робота ілюстрована 30 рисунками та містить 17 таблиць. Перелік використаних джерел налічує 30 найменувань і займає 3 сторінки. Пояснювальна записка також містить 10 сторінок додатків. Загальний обсяг роботи становить 90 сторінок.

Мета роботи полягає у проєктуванні вебплатформи для підтримки соціальних зв’язків, яка забезпечує активну комунікацію між користувачами.

У першому розділі проведено аналіз предметної області та існуючих рішень у сегменті соціальних мереж, визначено цільову аудиторію, сформовано функціональні та нефункціональні вимоги, а також обґрунтовано вибір технологічного стеку для побудови серверної архітектури.

У другому розділі виконано об’єктно-орієнтоване проєктування: розроблено UML-моделі архітектури, спроектовано реляційну базу даних, та алгоритм взаємодії користувачів.

У третьому розділі здійснено реалізацію серверної та клієнтської частин, впроваджено механізми аутентифікації та обробки контенту.

КЛЮЧОВІ СЛОВА: WEB PLATFORM, SOCIAL NETWORK, PHP, MYSQL, WAMP64, DATABASE, WEB BASED SYSTEM, USER CONTENT.

					ІПЗ.КР.Б. – 121 – 26 – ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дубров М.Ю.			Вебплатформа для підтримки соціальних зв’язків	Літ.	Арк.
Керівник		Чижмотря О.Г.					3
							90
Рецензент		Морозов А.В.				ФІКТ Гр. ІПЗп-22-1	
Зав. каф.		Вакалюк Т.А.					

ABSTRACT

The explanatory note to the bachelor's thesis titled "A Web Platform for Supporting Social Connections" consists of 84 pages of printed text. The thesis is illustrated with 30 figures and contains 17 tables. The list of references includes 30 items and takes up 3 pages. The explanatory note also contains 10 pages of appendices. The total volume of the work is 90 pages.

The purpose of this project is to design a web platform for fostering social connections that facilitates active communication among users.

In the first chapter analyzes the subject area and existing solutions in the social networking segment, identifies the target audience, defines functional and non-functional requirements, and justifies the choice of the technology stack for building the server architecture.

In the second chapter, object-oriented design was carried out: UML architecture models were developed, a relational database was designed, and a user interaction algorithm was modeled.

In the third chapter, the server-side and client-side components were implemented. and mechanisms for authentication and content processing were put in place.

KEYWORDS: WEB PLATFORM, SOCIAL NETWORK, PHP, MYSQL, WAMP64, DATABASE, WEB BASED SYSTEM, USER CONTENT

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижморя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. ПРЕДМЕТНА ОБЛАСТЬ ТА ПОСТАНОВКА ЗАДАЧІ	10
1.1. Постановка задачі та огляд існуючих соціальних платформ	10
1.2. Функціональні та нефункціональні вимоги до системи.....	19
1.3. Вибір і обґрунтування архітектурного рішення	23
1.4. Вибір інструментальних засобів для реалізації	27
Висновки до першого розділу	31
РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ.....	32
2.1. Моделювання варіантів використання та поведінки системи	32
2.2. Проєктування логічної структури бази даних	38
2.3. Проєктування компонентної структури та взаємодії модулів	46
2.4. Розробка основних алгоритмів роботи платформи	53
Висновки до другого розділу.....	57
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЄКТУ	58
3.1. Програмна реалізація серверної частини та підсистеми безпеки ...	58
3.2. Проєктування та реалізація клієнтського інтерфейсу	61
3.3. Налаштування середовища, розгортання та приклади інтерфейсу	64
3.4. Тестування програмної системи	72
Висновки до третього розділу	76
ВИСНОВКИ	77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	79
ДОДАТКИ	82

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних

ПЗ - програмне забезпечення

СУБД - система управління базами даних

HTML - HyperText Markup Language - мова гіпертекстової розмітки

CSS - Cascading Style Sheets – каскадні таблиці стилів

PHP - Hypertext Preprocessor - скриптова мова програмування для створення вебзастосунків

MySQL - система управління реляційними базами даних

WAMP - Windows, Apache, MySQL, PHP - набір програмного забезпечення для розробки вебсайтів у середовищі Windows

SQL - Structured Query Language - мова стуктурованих запитів

HTTP/HTTPS - HyperText Transfer Protocol (Secure) - протоколи передачі даних

UI/UX - User Interface / User Experience - інтерфейс та досвід користувача

UML - Unified Modeling Language - уніфікована мова моделювання

JSON - JavaScript Object Notation - текстовий формат обміну даними

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. У сучасному цифровому світі соціальні мережі перетворилися з інструментів звичайного спілкування на складні екосистеми, що потребують інтенсивної взаємодії користувачів та обробки великих обсягів контенту. Реалізація проєкту вимагає створення механізмів авторизації, динамічного формування стрічки контенту та забезпечення комунікації в реальному часі.

Актуальність проєкту зумовлена необхідністю створення ефективної архітектури для соціальної мережі, яка б забезпечувала баланс між швидкістю роботи та цілісністю даних. Використання технологічного стеку PHP та MySQL у середовищі WAMP64 дозволяє реалізувати надійний серверний фундамент, проте ключовим викликом залишається проєктування структури бази даних, здатної масштабувати зв'язки між користувачами та оперативно обробляти контент.

Крім того, розробка повноцінної вебплатформи на PHP вимагає системного підходу до архітектури: від моделювання бізнес - логіки за допомогою UML до забезпечення захисту персональних даних та оптимізації запитів до БД.

З огляду на зростаючі вимоги користувачів створення власної архітектури соціальної мережі, яка здатна ефективно керувати користувацьким контентом та забезпечувати високу швидкість відгуку системи, є вкрай актуальним науково-практичним завданням.

Об'єкт дослідження - процес проєктування та програмної реалізації вебплатформи для підтримки соціальних зв'язків, що базується на архітектурі динамічної обробки та представлення користувацького контенту.

Предмет дослідження - методи побудови реляційної структури бази даних, архітектурні рішення та програмні засоби PHP, MySQL, WAMP64 для забезпечення цілісності, швидкодії та ефективної комунікації між користувачами.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Мета роботи - полягає у проєктуванні та розробці програмного комплексу, який забезпечує взаємодію користувачів, ефективне управління медіаконтентом та динамічну обробку соціальних зв'язків.

Для досягнення цієї мети було визначено такі завдання:

- проаналізувати предметну область, дослідити принципи побудови сучасних соціальних мереж та провести огляд існуючих аналогів;
- обґрунтувати вибір технологічного стеку PHP, MySQL, WAMP64 та визначити функціональні й нефункціональні вимоги до архітектури системи;
- спроектувати об'єктно-орієнтовану структуру соціальної мережі, включаючи UML-моделі архітектури та логіку взаємодії користувачів;
- розробити схему реляційної бази даних, яка забезпечує цілісність даних при високій інтенсивності запитів;
- реалізувати основні модулі системи: автентифікацію, профіль користувача, систему публікацій, стрічку контенту та механізми соціальної взаємодії: лайки, коментарі, підписки, а також розробити адміністративну панель та модуль модерації, впровадити систему управління користувачами, обробки скарг на контент, блокування, видалення і дотримання правил спільноти користувачами;
- розробити функціонал для обміну повідомленнями та реалізувати пошукову систему для швидкого знаходження контенту та користувачів;
- впровадити алгоритм рекомендацій контенту на основі підписок на друзів для персоналізації стрічки;
- провести багаторівневе тестування системи, забезпечити механізми захисту даних та виконати розгортання соціальної мережі у середовищі WAMP64.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

Методи дослідження. Під час роботи застосовувалися методи системного та порівняльного аналізу для огляду існуючих рішень в розділі соціальних мереж та формування вимог до функціональності системи. Використано методи об'єктно-орієнтованого проєктування для побудови архітектури серверної клієнтської частини, а також методи моделювання реляційних баз даних для забезпечення ефективного зберігання та обробки користувацького контенту.

Практичне значення одержаних результатів. Розроблена соціальна мережа Layttle є програмним продуктом, що забезпечує користувачам платформу для активної комунікації, обміну медіаконтентом та управління власними соціальними зв'язками. Практичне значення роботи полягає у створенні архітектурної основи системи, яка забезпечує цілісність даних, швидкодію при високій інтенсивності запитів та можливість подальшого масштабування функціоналу. Розроблені програмні рішення можуть бути використані як фундамент для побудови спеціалізованих соціальних мереж або інтерактивних комунікаційних систем. Зокрема, створена структура реляційної бази даних та оптимізовані SQL-запити можуть бути використані як готовий шаблон або навчальний посібник для розробки аналогічних багатокористувацьких систем із високим рівнем динамічної взаємодії. Практичну цінність також має вдале поєднання класичної серверної архітектури на базі шаблону MVC та сучасних асинхронних технологій обміну даними AJAX і Fetch API. Такий підхід доводить свою ефективність у реальних умовах, забезпечуючи мінімальний час відгуку інтерфейсу та зниження навантаження на серверне середовище. Розроблені скрипти для обробки, перевірки та безпечного завантаження мультимедійних файлів можуть бути легко адаптовані для використання в інших вебзастосунках, що потребують роботи з великими обсягами графічного чи відеоконтенту.

Готовий програмний продукт разом із детальною документацією створює міцну базу для подальших наукових досліджень та експериментів у галузі оптимізації соціальних мереж.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. ПРЕДМЕТНА ОБЛАСТЬ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Постановка задачі та огляд існуючих соціальних платформ

Предметною областю даного дипломного проєкту є сфера розробки веборієнтованих соціальних мереж, що забезпечують комунікацію користувачів та обмін контентом. Аналіз предметної області показує, що соціальні мережі стали фундаментальним інструментом взаємодії в сучасному світі, де швидкість обміну інформацією та надійність зберігання даних є критичними показниками.

Процес створення повноцінної соціальної мережі є комплексною інженерною задачею, що вимагає глибокого розуміння архітектурних патернів[1]. Класична розробка таких систем передбачає побудову клієнт-серверної архітектури, де Frontend-частина забезпечує інтерактивний інтерфейс для користувача[2], а Backend-частина реалізує бізнес-логіку: керування сесіями, обробку запитів до бази даних, алгоритми рекомендацій підписок на друзів та захист від несанкціонованого доступу[3].

Використання готових CMS або шаблонних рішень для створення соціальної мережі часто обмежує функціональні можливості та масштабованість продукту. Власник ресурсу стає залежним від сторонніх платформ, які мають обмежену гнучкість у налаштуванні алгоритмів взаємодії між користувачами чи змінні структури даних. Натомість розробка власної архітектури системи дозволяє: оптимізувати структуру бази даних під специфічні потреби мережі: дружба, підписка, коментування. Забезпечити повний контроль над безпекою даних та процесом аутентифікації користувачів, реалізувати унікальний набір функцій, що неможливо в умовах використання стандартних конструкторів.

Таким чином, розробка власного програмного продукту з використанням стеків PHP та MySQL у середовищі WAMP64 [4] є актуальним завданням, оскільки дозволяє створити високоефективну та адаптивну систему соціальних зв'язків, оптимізовану під конкретні вимоги проєкту, що чітко ілюструє порівняння шаблонного підходу та інженерної реалізації (рис.1.1.).

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

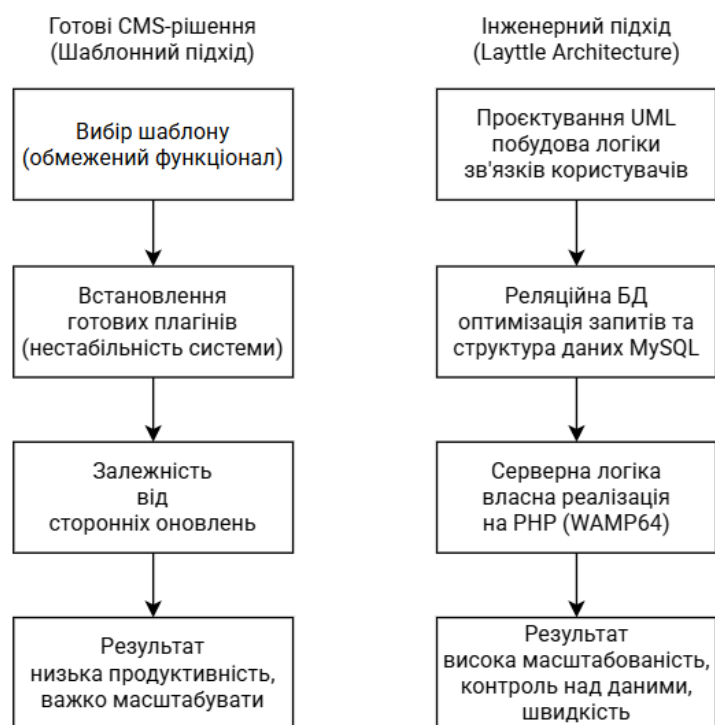


Рис.1.1. Порівняння шаблонного підходу до розробки соціальних мереж та інженерної реалізації системи Layttle

Усвідомлення обмежень готових шаблонних рішень призвело до необхідності еволюції підходів до побудови соціальних мереж. Основною ідеєю проєкту Layttle є відмова від статичних CMS на користь гнучкої архітектури, де складні технічні процеси автоматизуються на рівні серверної логіки. Замість використання обмежених візуальних конструкторів, користувач отримує функціонал для динамічної взаємодії з контентом, що забезпечується високою продуктивністю середовища PHP/MySQL. Загальний принцип функціонування системи Layttle та архітектурної взаємодії користувача з платформою представлено у вигляді концептуальної схеми (рис.1.2.).

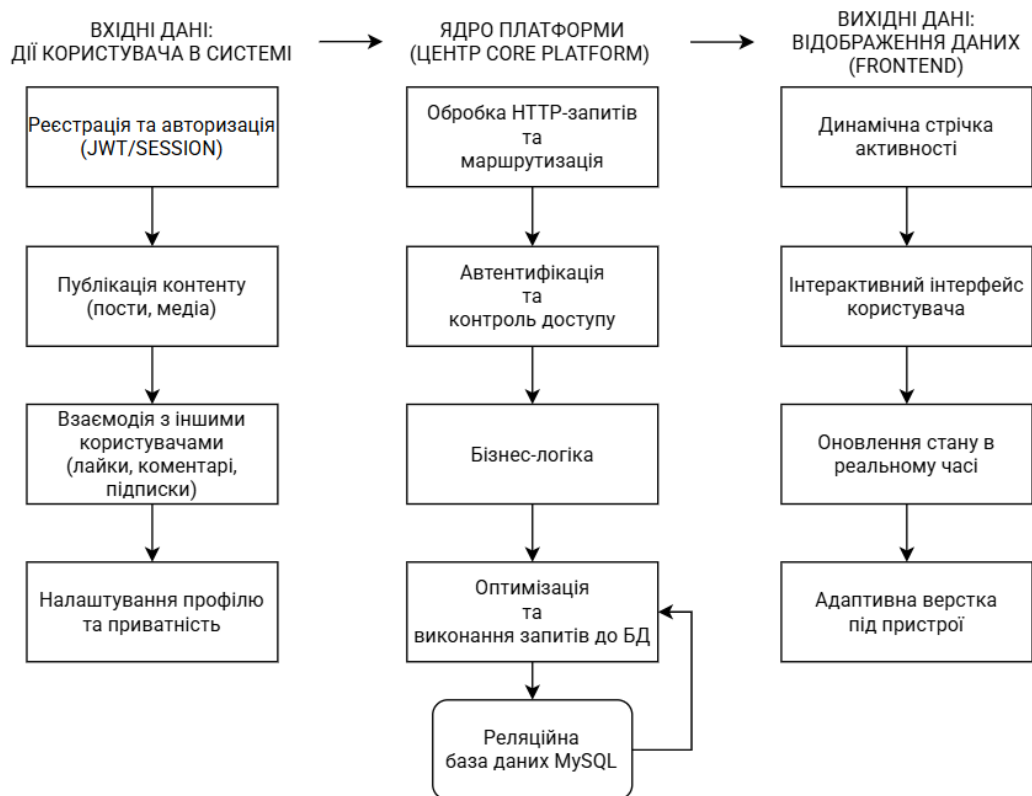


Рис.1.2. Концептуальна схема функціонування соціальної мережі Layttle

Під час проєктування соціальної мережі Layttle ключовим завданням стало створення максимально зручного та оптимізованого середовища для взаємодії користувачів із платформою. Дослідження досвіду популярних глобальних соцмереж зокрема Facebook, Instagram та X показує, що їхні інтерфейси часто перевантажені зайвими елементами, рекламними інтеграціями та величезною кількістю налаштувань, що лише відволікає аудиторію та ускладнює процес простого і швидкого обміну контентом.

Головною проблемою багатьох традиційних вебплатформ [5] полягає в тому, що користувачеві доводиться пристосовуватися до перевантаженої архітектури: навіть банальна публікація допису чи пошук потрібних даних вимагають забагато зайвих рухів. На противагу цьому, концепція Layttle зосереджена на максимальному спрощенні та автоматизації стандартних дій за допомогою зручної блокової структури. Людина взаємодіє виключно з базовими, звичними елементами на кшталт текстів, медіафайлів та обговорень, тоді як усю складну технічну магію розмітки та оптимізації платформа бере на себе.

Такий підхід дає змогу повністю зануритися у спілкування та створення контенту, не витрачаючи ресурсів на вивчення тяжкої навігації чи подолання бар'єрів, притаманних величезним сучасним медіаресурсам.

У сучасних умовах соціальні мережі перетворилися на складні екосистеми, де продуктивність платформи залежить від швидкості обробки користувацького контенту. Найбільш трудомістким завданням при розробці такої системи є не просто верстка інтерфейсу, а забезпечення взаємодії між користувачами та алгоритмічної підтримки стрічки контенту. Глобальні лідери ринку, такі як Facebook, Instagram та X, базуються на величезних архітектурних потужностях, що дозволяє їм підтримувати мільйони активних користувачів.

Основною метою є створення полегшеної архітектури, яка автоматизує управління контентом. На відміну від існуючих аналогів, де користувач стикається з перевантаженими інтерфейсами, Layttle впроваджує принцип “інтуїтивної структури”. Користувач взаємодіє з функціоналом публікацій, коментарів, підписок, тоді як система в той час у фоновому режимі автоматично забезпечує валідність структури даних, швидкодію та адаптивність відображення контенту на будь-яких пристроях.

Фундаментальним рішенням є вибір архітектури на основі компонентного підходу, що дозволяє динамічно формувати сторінки без перезавантаження сервера. Використання підходу Schema-Driven UI дозволяє відокремити конфігурацію сторінки від її програмного коду. Уся структура соціальної взаємодії зберігається у форматі JSON [6], що дозволяє системі обробляти запити користувачів та миттєво генерувати інтерфейс, виключаючи необхідність ручного оновлення кожного окремого шаблону.

Для забезпечення візуальної цілісності та узгодженості інтерфейсу використовується метод Atomic Design. Це дозволяє розробити унікальний набір компонентів - від базових елементів стрічки контенту до складних медіагалерей профілю. Така архітектура гарантує стабільність верстки при високих навантаженнях, що є критично важливим для соціальних мереж, що візуально демонструє принцип роботи методу (рис.1.3.).

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Аналіз ринкових аналогів показує, чому Layttle є доцільною альтернативою:

Facebook - перевантажений застарілими функціями та непрозорими алгоритмами, що призводить до сповільнення відгуку системи. Layttle акцентує фокус на прозорих механізмах зв'язків.

Instagram - має жорстку прив'язку до медіа-форматів, що обмежує користувачів у виборі способів комунікації. Layttle пропонує гнучкість у роботі з контентом за рахунок динамічних блоків.

X - орієнтований на короткі повідомлення, проте не забезпечує глибини соціальної взаємодії. Layttle об'єднує швидкість обміну даними з розвиненою структурою соціальних зв'язків.

Отже, головним завданням даної роботи є проєктування Layttle як комплексної вебплатформи, що приховує від користувача технічну складність роботи з базами даних та серверами. Досягнення цієї мети вимагає дослідження архітектурних патернів побудови соціальних систем, методів ефективної серіалізації даних та організації клієнт-серверної взаємодії за допомогою PHP та MySQL. Такий підхід дозволяє створити продукт, що поєднує високу швидкість розробки та стабільність роботи, які недоступні при використанні масових шаблонних рішень.

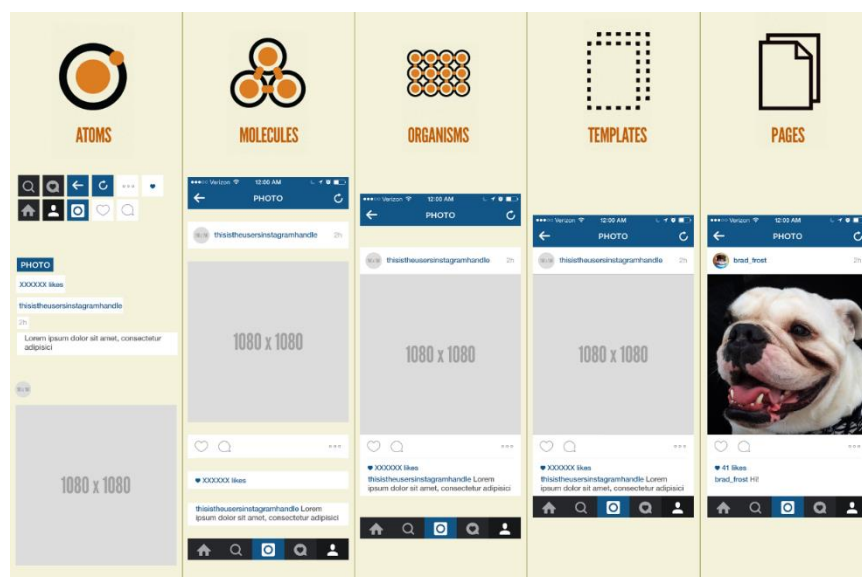


Рис.1.3. Зображення принципу роботи методу Atomic Design

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Facebook - функціонує як масштабна глобальна екосистема, побудована на базі монолітної архітектурної концепції, яка об'єднує безліч інструментів для комунікації та роботи з інформацією. Головна стратегія платформи полягає в тому, щоб утримати аудиторію всередині єдиного простору завдяки постійній генерації та доставці індивідуально підібраного контенту.

Логіка функціонування цієї платформи спирається на складні алгоритми сортування інформації та розгалужену мережу взаємних зв'язків. Головними плюсами Facebook вважаються потужна технічна база, колосальний потенціал для просування рекламних компаній і здатність обробляти мільярди активних дій одночасно, що й гарантує стабільно високу залученість аудиторії.

Головні мінуси цієї системи криються в надмірній складності її архітектури. Користувач виявляється повністю підпорядкованим закритим внутрішнім алгоритмам, які самостійно вирішують, які матеріали виводити на екран, що суттєво знижує особистий контроль над власним соціальним колом. Будь-які спроби адаптувати стрічку чи змінити логіку видачі під особисті запити наштовхуються на закритість налаштувань, які розробники навмисно ховають від очей аудиторії. Увесь механізм заточений на те, щоб максимально подовжити час присутності в системі, а не на зручність і комфорт конкретного користувача, на відміну від принципово іншого підходу, закладеного в проєкті Layttle. Ознайомитися із загальним оформленням стрічки новин цього сервісу можна на рис.1.4.

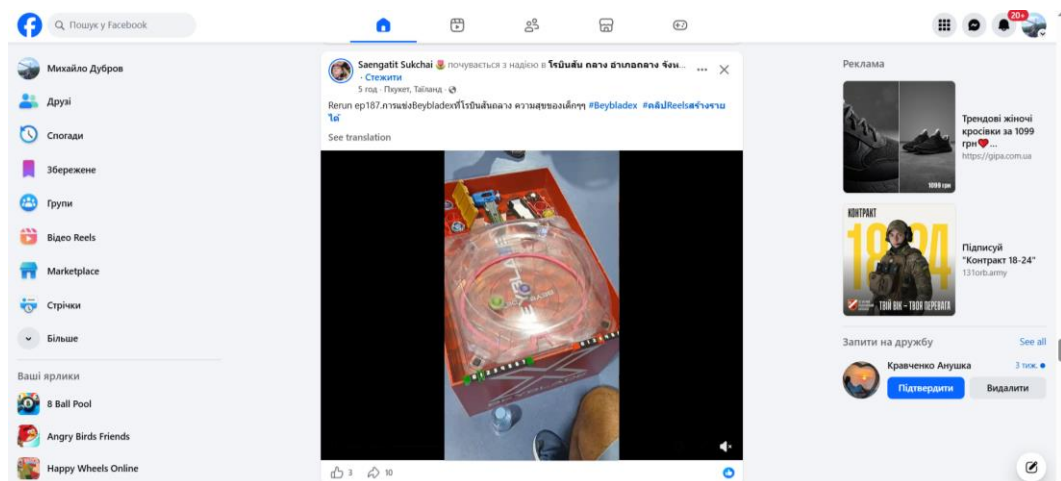


Рис.1.4. Інтерфейс стрічки новин Facebook

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Instagram - робить акцент на специфічній візуально-орієнтованій моделі споживання медіаконтенту. Філософія цього сервісу базується на перетворенні звичайного спілкування на безперервний потік зображень і динамічних відеороликів, де головним каталізатором інтересу виступають візуальна естетика та миттєвий відгук аудиторії.

Головний акцент у роботі цього сервісу зроблено на активній роботі з важкими медіафайлами: платформа самостійно займається налаштуванням швидкості показу зображень чи роликів, щоб стрічка прокручувалася максимально плавно. До ключових переваг Instagram належать налагоджена система доставки візуальних матеріалів та широкий арсенал вбудованих функцій для оперативного редагування картинок або відео. Завдяки цьому люди можуть створювати якісний контент прямо в застосунку, не вдаючись до допомоги сторонніх графічних редакторів.

Водночас такі сильні сторони водночас породжують і суттєві бар'єри, що заважають повноцінному спілкуванню. Сервіс встановлює жорсткі рамки, а його внутрішня будова сильно обмежує роботу зі звичайними текстами чи посиланнями, змушуючи авторів підлаштовуватися під суворі стандарти медіаформатів. Інтерфейс буквально забитий рекламними блоками та підказками алгоритмів, які раз у раз відволікають людину від подій у житті її ближнього кола. До того ж закриті й постійно мінливі правила видачі стрічки перетворюють цей майданчик на досить непередбачуване середовище для побудови довготривалих спільнот. Натомість концепція проєкту Layttle зосереджена на протилежному принципі - показувати виключно публікації тих користувачів, на яких людина підписана самостійно. Поглянути на типовий інтерфейс цього сервісу можна на рис.1.5.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

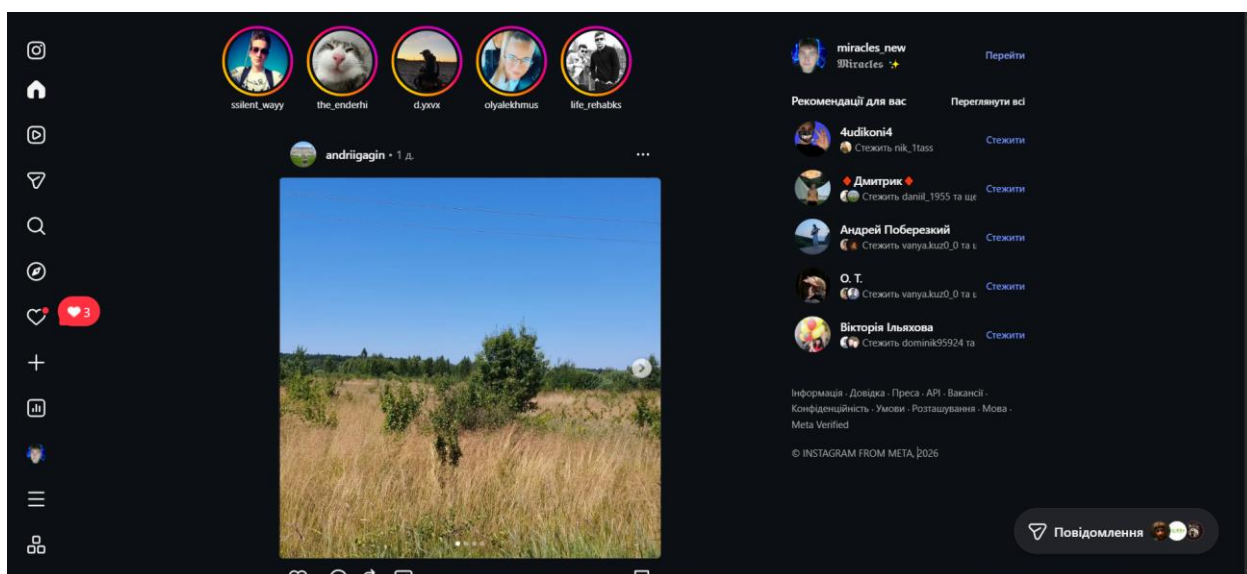


Рис.1.5. Інтерфейс стрічки новин Instagram

X (Twitter) - це відома соціальна платформа, яка побудована навколо концепції миттєвої трансляції коротких текстових нотаток та медіафайлів для швидкого реагування на свіжі події в режимі реального часу. Головна мета цього сервісу полягає в створенні динамічного інформаційного потоку, де на першому місці стоять швидкість появи нового контенту в реальному часі.

Архітектура та робочі алгоритмічні процеси цього сервісу повністю оптимізовані під обробку величезного потоку частих запитів і практично миттєву доставку інформації кінцевому споживачеві. До сильних сторін платформи належить її унікальна роль як потужного інструменту швидкого інформування та масштабної дискусійної площадки, яка дозволяє людям безперешкодно та без зайвих витрат часу на оформлення ділитися новинами з величезною кількістю підписників.

Водночас такі особливості створюють чимало мінусів, особливо якщо йдеться про побудову міцних і тривалих соціальних контактів. Внутрішня будова цього сервісу створює чималий інформаційний шум, у якому дуже важко зосередитися на спілкуванні з близькими людьми. Екран переповнений блоками актуальних трендів, рекламою та тригерами, які часто провокують гострі дискусії чи конфлікти, відволікаючи від справжніх приятелів. До того ж закриті алгоритми видачі постів і постійні зміни правил доступу до API [7] роблять

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

спільноти вкрай вразливими до зовнішніх факторів. На відміну від цього середовища, платформа Layttle націлена на забезпечення спокійної та затишної атмосфери, де на перший план виходить якісний контент від людей, яким дійсно довіряють, без агресивного нав'язування чужих трендів чи суперечливих тем. Побачити, як виглядає інтерфейс сервісу X, можна на рис.1.6.

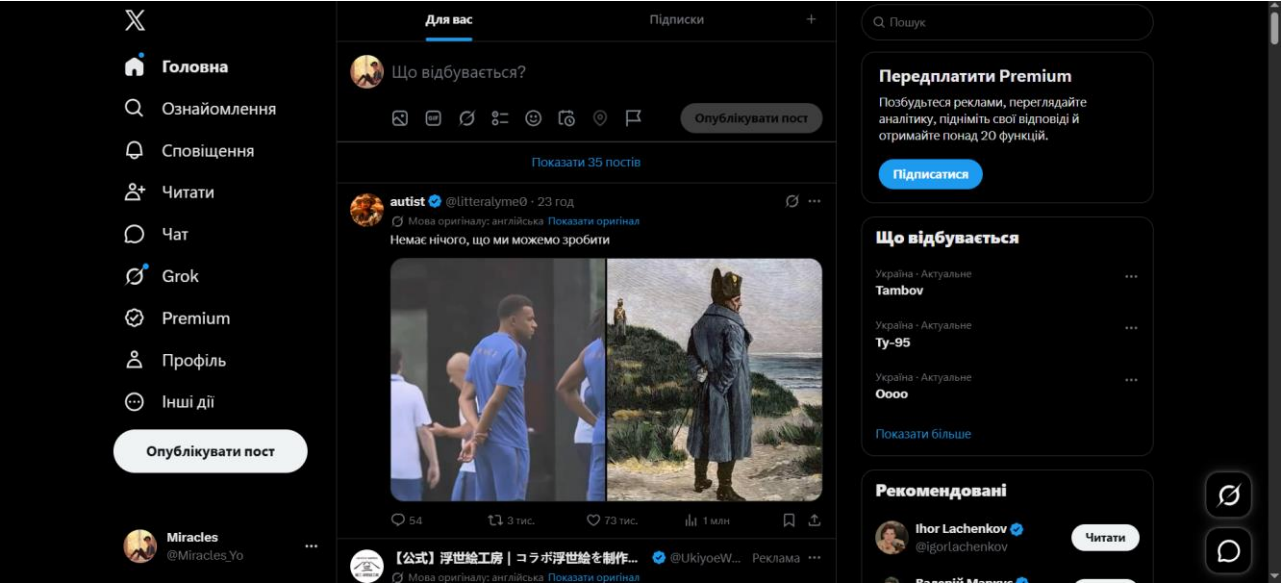


Рис.1.6. Інтерфейс стрічки новин X

Для систематизації результатів проведеного аналізу та обґрунтування функціональних вимог до розроблюваної системи Layttle, було проведено порівняння ключових параметрів існуючих соціальних мереж. У таблицю (табл.1.1.) наведено порівняльну характеристику платформ Facebook, Instagram та X за технічними та функціональними критеріями, що дає змогу виявити архітектурні переваги власної реалізації системи.

Таблиця 1.1.

Порівняння систем конструювання вебресурсів

Функціональний критерій	Facebook	Instagram	X (Twitter)	Layttle
Архітектура даних	Монолітна	Монолітна	Монолітна	Модульна
Алгоритми упорядкування	Нав'язливі	Візуальні	Агресивно-трендові	Користувацькі
Рекламна модель	Агресивна	Нав'язлива	Надмірно персоналізована	Відсутня
Налаштування стрічки	Обмежене алгоритмічне	Обмежене візуальне	Обмежене тренди	Лише підписки користувача
Контроль приватності	Складний	Базовий	Базовий	Високий

Формат контенту	Універсальний	Візуальний	Візуальний	Гнучкий
Масштабованість	Залежить від корпорації	Залежить від корпорації	Залежить від корпорації	Власна інженерна
Використання ресурсів	Надлишкове	Високе	Високе	Оптимізоване

Виконаний аналіз соц.мереж показав помітну різницю: сьогодні практично не існує проєктів, які б пропонували гармонійний баланс в управлінні соціальними зв'язками. Попри колосальні масштаби та мільйонні аудиторії таких гігантів, як Facebook, Instagram чи X, кожен із них страждає від низки вроджених проблем. Серед них виділяються закриті монолітні структури, тотальна залежність від закритих корпоративних алгоритмів, агресивна реклама, неможливість вільно налаштувати власну стрічку контенту без стороннього втручання, а також дефіцит повноцінного контролю над власними персональними даними.

З огляду на це, створення власної соціальної платформи Layttle є абсолютно виправданим, логічним та технічно обґрунтованим кроком. Розроблюваний проєкт вдало поєднує простоту сприйняття й прозорість взаємодії із гнучкою сучасною архітектурою. Відмова від зайвого функціонального навантаження дає змогу суттєво оптимізувати ресурси системи, повернути користувачам повний контроль над власним соціальним колом завдяки показу публікацій виключно від друзів та підписок, а також забезпечити високу інженерну масштабованість. У результаті виходить надійне, швидкісне й безпечне середовище для спілкування, вільне від агресивного тиску алгоритмів чи нав'язливої комерційної реклами.

1.2. Функціональні та нефункціональні вимоги до системи

Правильне визначення потенційної аудиторії та детальний розбір сценаріїв взаємодії із системою становлять фундаментальну основу для формування якісних технічних вимог до майбутнього додатку. Чітке усвідомлення запитів майбутніх користувачів дає змогу вибудувати інтуїтивний інтерфейс, зробити

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

взаємодію максимально комфортною та спрямувати ресурси на реалізацію найважливіших функцій. Ґрунтуючись на попередньому дослідженні ринку, для соціальної мережі Layttle можна виокремити три ключові категорії користувачів, кожна з яких ставить перед платформою свої унікальні цілі та вимоги.

Першою категорією є активні користувачі та представники онлайн-спільнот, які прагнуть спілкуватися в середовищі без нав'язливої реклами. Це користувачі, яким важлива швидкість завантаження контенту та відсутність зайвого інформаційного шуму. Основна проблематика існуючих гігантів Facebook, Instagram, X полягає в тому, що стрічки контенту перенасичені алгоритмічною рекламою та нерелевантними рекомендаціями. Платформа Layttle вирішує цю проблему через надання чистої стрічки контенту, яка формується виключно на основі підписок користувача, забезпечуючи комфортне та безпечне споживання інформації.

Другою категорією аудиторії є блогери, лідери думок та контент-мейкери. Це професіонали, яким необхідно регулярно публікувати різноплановий контент: тексти, медіафайли, галереї та взаємодіяти з підписниками. Їхньою головною проблемою у великих соцмережах є залежність від закритих, непередбачуваних алгоритмів сортування, які можуть знижувати охоплення публікацій. Рішенням від вебплатформи є прозорий хронологічний порядок відображення контенту та гнучкі інструменти управління власним профілем і публікаціями без штучних обмежень чи візуальної диктатури.

Третьою категорією складають адміністратори локальних спільнот та невеликих тематичних груп. Вони потребують інструментів для організації взаємодії між учасниками без надлишкової складності монолітних систем. Проблематика полягає в тому, що існуючі платформи пропонують перевантажені інструменти модерації та складну ієрархію прав. На відміну від громіздких аналогів, розроблювана вебплатформа Layttle вирішує це завдання шляхом впровадження модульної архітектури, яка дозволяє легко керувати внутрішніми зв'язками спільноти, забезпечуючи високу швидкість відгуку системи та повний контроль над структурою даних.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Зважаючи на детально опрацьовані вимоги користувачів, ми виділили головні сценарії взаємодії, які наочно демонструють здатність платформи задовольняти найрізноманітніші запити. У межах стандартного робочого процесу взаємодія із сервісом передбачає проходження процедури авторизації, комфортний перегляд стрічки контенту, яка складається виключно з публікацій людей на яких підписані користувачі, а також легке створення власних постів за допомогою зручних та зрозумілих блоків.

Зовсім інакше виглядає процес керування контентом та адміністрування платформи. Завдяки продуманій модульній будові та використанню технології Schema-Driven UI, будь-які зміни у профілі чи нові публікації оновлюються миттєво у фоновому режимі, позбавляючи потреби постійно перезавантажувати сторінку. Усі інформаційні потоки фіксуються в чітко структурованому форматі, що забезпечує миттєвий рендеринг елементів інтерфейсу та стабільну роботу комунікаційних механізмів. Задіяні потреби чудово демонструють здатність системи гнучко підлаштовуватися під особисті потреби будь-якої категорії користувачів.

Можливості майбутнього програмного продукту бере акцент на чітко окреслений набір функцій, які успішно будуть закривати комунікаційні та налаштовуванні потреби. В основу платформи покладено модель розмежування прав доступу на основі ролей (Role-Based Access Control). Оскільки ключова ідея соціальної мережі Layttle вимагає обов'язкового входження в систему заради безпеки й захисту соціальних контактів, тут навмисно виключено категорію випадкових гостей: будь-яка активність стає доступною виключно після успішної реєстрації та авторизації. Уся логіка побудована навколо трьох головних рівнів - авторизованого користувача, модератора та адміністратора. Детальний опис функціонала для кожної ролі наведено в таблиці (табл.1.2.).

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Таблиця 1.2.

Функціональні вимоги до системи за ролями користувачів

Роль користувача	Функціональні можливості у системі
Авторизований користувач	Управління власним профілем та персональними налаштуваннями безпеки. Створення, редагування та видалення власних публікацій, дописів і медіафайлів. Перегляд стрічки публікацій, сформованої на основі підписок на інших користувачів. Взаємодія з контентом, коментування, ставлення уподобайок, чатування. Управління списком друзів та підписок. Завантаження та управління власними медіаресурсами.
Модератор	Перегляд списку користувачів та їхньої активності на платформі. Тимчасове або постійне блокування деструктивного контенту, приховування публікацій із порушенням правил платформи, а також винесення офіційних попереджень користувачам щодо порушення контентної політики. Обробка скарг від спільноти, перегляд і надання відповідей у служби підтримки.
Адміністратор	Повний контроль над програмним комплексом, що включає всі права модератора з розширеними повноваженнями. Можливість видалення облікових записів користувачів та пов'язаного з ними контенту. Управління обліковими записами та правами модераторів. Доступ до повної адміністративної панелі керування платформою, системних налаштувань та моніторингу цілісності бази даних.

На протигагу функціональним можливостям, нефункціональні критерії відповідають за загальні параметри надійності та продуктивності продукту, визначаючи швидкість і ефективність його роботи.

Головним аспектом є вимоги до продуктивності та швидкодії, які визначають, наскільки миттєво система реагує на дії користувача та, як раціонально витрачає системні ресурси. Стандартний час відповіді серверного API під час виконання типових дій як-от завантаження стрічки публікацій, публікація нового допису чи перехід у профіль - тримається в суворих межах 300-400 мілісекунд. З огляду на те, що інтерфейс клієнтської частини робиться інтерактивним та базується на новітніх компонентах, оновлення контенту й промальовування елементів у процесі прокручування відбуваються бездоганно плавно, позбавляючи будь-яких ривків чи затримок. Аби гарантувати швидке завантаження фотографій і відеороликів, передбачено їх попереднє стиснення ще на серверній стороні до моменту збереження, а також надійне кешування статичних даних.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

Особливу увагу приділено надійності та стабільності роботи, які безпосередньо спираються на ретельно підібрану технологічну інфраструктуру. Цільовий показник безперебійної експлуатації платформи встановлено на рівні 99,9%, що гарантує її доступність цілодобово. Для абсолютного захисту від втрати цінної інформації в реляційній базі даних MySQL [8] налаштовано щоденне автоматичне створення резервних копій. Крім того, на випадок будь-яких непередбачуваних серверних збоїв передбачено вбудовані алгоритми автоматичного перезапуску служб, що зводить можливий час простою до мінімуму.

Окрему увагу приділено питанням безпеки, адже сервіс працює з особистою інформацією учасників та їхніми приватними листуваннями. Увесь трафік між додатком і сервером передається виключно через захищений протокол HTTPS [9] за допомогою надійних SSL/TLS-сертифікатів. Користувацькі паролі ніколи не потрапляють до бази в чистому вигляді на серверній стороні їх обов'язково переводять у криптографічний хеш. Щоб убезпечити систему від найпоширеніших загроз на кшталт SQL-ін'єкцій чи XSS-атак, вхідні дані проходять автоматичне очищення, а запити до бази формуються лише через параметризовані команди. Крім того, за авторизацію та контроль активних сесій відповідають безпечні токени, які надійно захищають акаунти від будь-яких спроб стороннього втручання.

1.3. Вибір і обґрунтування архітектурного рішення

Успіх і продуктивність комунікаційної платформи Layttle безпосередньо спираються на ретельний розбір підходів до роботи з динамічними інформаційними масивами та правильний підбір системи керування базами даних. Головна технічна складність тут полягає в тому, щоб поєднати дві протилежні вимоги: забезпечити абсолютну надійність і цілісність для облікових записів, списків друзів і приватних повідомлень, а водночас дати максимальну свободу й гнучкість для налаштування стрічок, самих публікацій та вкладених медіафайлів. Оскільки формат дописів та характер взаємодій постійно

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

змінюються самі користувачі створюють різноманітні за структурою та глибиною об'єкти, класичні схеми реляційних баз даних вимагають серйозної адаптації. У процесі проєктування ретельно було вивчено три ключові патерни зберігання подібної інформації.

Перший шлях базується на класичному реляційному підході з глибокою нормалізацією кожної окремої сутності. У сучасному світі соціальна мережа передбачає акцент уваги під лайки, коментарі чи медіафайли окремих таблиць. Попри всю свою структурну строгість і надійність, такий метод суттєво гальмує роботу з мінливим контентом. Найбільшим мінусом тут стає заплутаність запитів та відчутне просідання продуктивності під час збирання стрічки контенту. Наприклад, щоб вивести на екран хоча б кілька десятків постів із їхніми авторами, відмітками подобається та коментарями, серверній частині на PHP доводиться вибудовувати складні багатотабличні об'єднання типу JOIN, що створює надлишкове навантаження на середовище WAMP64 і помітно затримує реакцію інтерфейсу.

Альтернативним варіантом є перехід на документні бази даних NoSQL, яскравим прикладом яких є MongoDB. У цьому випадку кожен пост разом із прикріпленими коментарями та медіафайлами пакується в єдиний цілісний JSON-документ, що дозволяє блискавично зчитувати дані для формування стрічки. Проте ставка виключно на NoSQL-модель створює чимало труднощів із безпекою та керуванням зв'язками між людьми. Оскільки платформа вимагає бездоганної цілісності профільних даних, ролей та взаємних підписок, брак звичних ACID-транзакцій значно ускладнює контроль надійності та захищеності інформації на рівні бекенду.

Найзбалансованішим і найбільш вдалим рішенням для платформи Layttle виявився гібридний шлях: залучення реляційної СУБД MySQL, розгорнутої в середовищі WAMP64, але з використанням спеціальних колонок типу JSON. Такий вибір дозволив успішно вирішити класичну архітектурну потребу. З одного боку, стандартні таблиці MySQL із чіткими зв'язками гарантують абсолютну безпеку й цілісність найважливішої інформації профілів, ролей та

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

параметрів безпеки. З іншого боку, завдяки вбудованій підтримці JSON, різноманітні блоки контенту, внутрішня структура публікацій та медіа зберігаються гнучко в межах тих самих реляційних таблиць. Це дає змогу PHP-бекенду легко й ефективно опрацьовувати будь-які запити, гармонійно поєднуючи надійність класичних баз даних із гнучкістю документоорієнтованих систем. Побачити схему реалізації цього архітектурного підходу можна на (рис.1.7.)



Рис.1.7. Архітектурна діаграма гібридної моделі зберігання даних

Починаючи з релізу 5.7.8., СУБД MySQL стала вбудованою підтримкою роботи з JSON-даними. Завдяки цьому з'явилася можливість пакувати складні структури дописів, медіафайли та змінюванні блоки контенту в спеціальну колонку content формату JSON, яка знаходиться всередині звичайної реляційної таблиці posts.

Вибраний архітектурний підхід помітно прискорює роботу платформи Layttle. Оскільки всі головні деталі та параметри публікації зібрані в єдиному полі, сервер на базі PHP миттєво отримує повністю готову структуровану інформацію для швидкого формування стрічки за допомогою одного лаконічного SQL-запиту: `SELECT content_json FROM posts WHERE id = ?`. Окрім цього, таке рішення надзвичайно спрощує подальше вдосконалення системи. Поява додаткових форматів медіа чи зміна логіки взаємодії більше не потребують складних міграцій та перебудови структури баз даних у середовищі WAMP64 базова схема залишається незмінною та стабільною, адже тепер достатньо просто оновити внутрішню начинку JSON-документа.

Крім того, сучасні можливості MySQL дозволяють індексувати віртуальні колонки, згенеровані безпосередньо з JSON-даних. Це розв'язує проблему пошуку за вмістом дописів, даючи змогу системі швидко знаходити специфічну інформацію всередині збережених конфігурацій.

Підсумовуючи, поєднання традиційної реляційної моделі MySQL разом із вбудованими можливостями JSON дає проєкту найкраще: омріяну гнучкість NoSQL-рішень та беззаперечну надійність повноцінної ACID-архітектури. Саме такий продуманий гібридний симбіоз перетворюється на ідеальний інструмент для створення сучасних вебплатформ із динамічним контентом.

Для розгортання та хостингу нашої соціальної мережі Layttle, ми обрали надійну інфраструктурну модель. Оскільки бекенд-логіка та робота з реляційною базою даних вимагають стабільного середовища виконання, серверна частина та взаємодія з СУБД адаптовані під можливості вебсервера. Водночас для швидкого й безперебійного доступу користувачів до інтерфейсу застосовується сучасний хостинг із підтримкою автоматичного розгортання. Завдяки вбудованій системі CI/CD, усі зміни в репозиторії миттєво підхоплюються системою, яка самостійно збирає та оновлює вебсторінки. Це гарантує стабільну роботу комунікаційної платформи та швидку доставку контенту з будь-якого куточка світу.

Для реляційної бази даних передбачено, що вона функціонує локально, у майбутньому для промислової експлуатації планується залучення хмарних сервісів за моделлю керованих баз даних (Database-as-a-Service). Перенесення бази даних на керовану хмарну платформу дозволить автоматизувати створення бекапів, забезпечити надійне шифрування інформації та гарантувати високу доступність сервісу для всіх користувачів платформи.

Для ефективного управління медіафайлами та зображеннями в архітектурі в майбутньому буде закладено інтеграцію зі спеціалізованим хмарним сервісом Cloudinary. Оскільки користувачі активно завантажують власний контент, фото та медіа для публікацій, зберігати їх виключно локально в середовищі WAMP64 або перевантажувати ними реляційну базу даних є неефективним рішенням при масштабуванні. У перспективі повномасштабного розгортання платформи такий сервіс, як Cloudinary, братиме на себе всю рутинну роботу: автоматичну оптимізацію та стиснення зображень, зміну їх розмірів, конвертацію у сучасні вебформати та швидку доставку через мережу CDN. Це дозволить зберегти

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

високу швидкодію системи та забезпечити миттєве завантаження медіаконтенту для користувачів з будь-яких пристроїв Windows, Linux, Android, iOS.

Для успішного запуску створеного повноцінного проєкту обов'язковою умовою є наявність власного унікального доменного імені. Це важливо не лише для впізнаваності бренду, а й для технічної безпеки правильного налаштування політик CORS обміну ресурсами між різними джерелами та підключення захищених SSL-сертифікатів для шифрування трафіку між користувачами та сервером.

Для надійного захисту процедури входу, реєстрації та публічних форм від різноманітних ботів і скриптів передбачено використання сучасного інструменту безпеки Cloudflare Turnstile. Будь-який відкритий вебпроєкт неминуче стикається з ризиками [10] масового спаму та спроб підбору паролів методом brute-force. Впровадження цієї системи дозволяє перевіряти користувачів ще на етапі взаємодії з інтерфейсом, відсіюючи автоматизовані запити. Такий підхід значно зменшує зайве навантаження на локальний сервер та базу даних, оскільки спам-атака блокується ще до того, як бекенд почне виконувати основну бізнес-логіку.

1.4. Вибір інструментальних засобів для реалізації

Вибір інструментальних засобів є фундаментальним і критично важливим етапом проєктування будь-якого програмного забезпечення[11]. Від обґрунтованості цього вибору напряду залежать швидкість розробки, стабільність функціонування системи, рівень її продуктивності, безпека обробки конфіденційних даних [12], а також зручність подальшого супроводу, масштабування та модифікації кодової бази. У цьому розділі послідовно схарактеризовано, детально проаналізовано та обґрунтовано технологічний стек, задіяний для створення клієнтської і серверної частин вебплатформи, а також підібрано необхідне локальне середовище для розробки та тестування.

На відміну від сучасних тенденцій масового переходу на складні мікросервісні архітектури чи важкі асинхронні платформи, в основі нашої

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

вебплатформи лежить перевірена роками, стабільна та високоефективна класична архітектура на базі мови PHP у поєднанні з реляційною системою управління базами даних MySQL. Такий вибір зумовлений кількома вагомими чинниками. PHP спочатку створювався спеціально для вебпрограм, що забезпечує мінімальний поріг входження та наявність величезної кількості вбудованих інструментів для роботи з HTTP-запитами, сесіями, куки та заголовками. Використання нативного об'єктно-орієнтованого PHP-бекенду гарантує передбачуване та стабільне виконання бізнес-логіки, безпечну фільтрацію й обробку користувацьких даних захист від SQL-ін'єкцій, а також безперебійну взаємодію з реляційною базою даних через стандартні механізми такі як PDO. На відміну від бездержавних stateless архітектур деяких сучасних фреймворків, класичний підхід дозволяє швидко і зручно керувати сесансами користувачів авторизація, ролі, права доступу, що критично важливо для соціальних платформ із динамічним контентом.

Для забезпечення повноцінного циклу розробки, тестування та налагодження програмного забезпечення [13] безпосередньо на робочій станції розробника було обрано інтегроване локальне середовище WAMP64 (Windows, Apache, MySQL, PHP). WAMP64 розгортає повноцінний вебсервер Apache [14], який за своєю конфігурацією та принципами роботи максимально наближений до реальних хостингових серверів.

Наявність вбудованого інтерфейсу phpMyAdmin дозволяє наочно проєктувати структуру бази даних, створювати зв'язки між таблицями Foreign Keys, виконувати експорт, імпорт та проводити оптимізацію індексів у режимі реального часу.

Можливість швидкого редагування файлу php.ini дає змогу змінити ліміт пам'яті для завантаження медіафайлів, часу виконання скриптів max_execution_time, увімкнення чи вимкнення необхідних розширень.

Фронтенд-частина вебплатформи розроблена з використанням класичних, надійних і перевірених часом вебтехнологій, що в сукупності забезпечують

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

миттєвий відгук інтерфейсу, кросбраузерність та комфортну взаємодію для користувачів із будь-яких пристроїв.

Оскільки бекенд побудований на PHP, що передбачає генерацію HTML-сторінок на сервері, клієнтська частина формується за допомогою адаптивної HTML5-верстки [15]. Використання семантичних тегів забезпечує правильну структуру документа, покращує доступність Accessibility та позитивно впливає на пошукову оптимізацію SEO.

Клієнтський JavaScript [16] відповідає за життєвий цикл стрічки контенту динамічне підвантаження даних, анімацію профільних елементів, роботу віджетів швидкого реагування, лайки, коментарі та інше без перезавантаження сторінки через AJAX-запити.

Бекенд-частина розроблена з акцентом на високу швидкість обробки транзакцій та структурованість інформації. Такий підхід є оптимальним, оскільки він забезпечує надійне опрацювання вхідних API-запитів, керування сесіями користувачів, логування подій та стабільне збереження інформації[17].

Контентна вебплатформа передбачає жорсткі зв'язки типу “один до багатьох”, користувач - його дописи, допис - його коментарі та “багато до багатьох”, користувачі у взаємних підписах або лайках.

Завдяки правильному індексуванню полів primary keys, indexes на часто запитовані колонки на кшталт user_id чи created_at, база даних здатна опрацьовувати декілька запитів за секунду без затримок. Це повністю задовольняє поточні та перспективні потреби проєкту в швидкодії та масштабованості. Без необхідності використання складної NoSQL системи.

Додатково варто зупинитися на середовищі розробки та інструментах контролю версій, які безпосередньо впливають на якість і безпеку управління проєктом. Для написання, редагування, налагодження та професійного рефакторингу програмного коду використовувалося спеціалізоване інтегроване середовище розробки PhpStorm[18] від компанії JetBrains. Будучи галузевим стандартом для PHP-розробки, PhpStorm забезпечує глибокий статичний аналіз коду, миттєве виявлення синтаксичних та логічних помилок, зручну навігацію

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

по архітектурі проєкту, а також потужні вбудовані інструменти для роботи з базами даних через пряме підключення до локального сервера MySQL та інтеграцію з терміналом. Додатково, тісна інтеграція середовища з системою контролю версій Git [19] дозволяє ефективно керувати гілками розробки, фіксувати зміни, проводити тестування та забезпечувати надійне резервне копіювання всіх компонентів вебплатформи на кожному етапі її життєвого циклу. Такий підхід до підбору програмного забезпечення створює надійний фундамент для стабільного функціонування всієї системи в цілому.

Окрім вибору базових серверних та клієнтських технологій, важливим аспектом реалізації проєкту є організація структури вихідного коду та механізмів моніторингу його виконання. У середовищі PhpStorm реалізовано модульну структуру проєктної документації та скриптів, де кожен компонент від контролерів обробки запитів до модулів відображення інтерфейсу ізольований у відповідних директоріях за принципом єдиної відповідальності. Це дозволяє уникнути монолітної заплутаності коду, полегшує підключення сторонніх бібліотек при потребі та прискорює пошук можливих помилок під час виконання інтеграційних тестів. Організація логування подій на рівні вебсервера Apache та конфігураційних файлів PHP надає змогу оперативно відстежувати системні попередження, аварійні завершення скриптів чи нестандартні HTTP-запити, що гарантує високий рівень контрольованості та прозорості всієї розробленої вебплатформи на етапі її розгортання та експлуатації.

Такий комплексний підхід до вибору програмного забезпечення, систем керування базами даних, середовищ розробки та протоколів безпеки забезпечує не лише високу ефективність на етапі створення платформи, але й закладає міцний фундамент для її подальшої підтримки, модернізації та масштабування. Усі обрані інструменти тісно інтегровані між собою, утворюючи єдину злагоджену екосистему, яка гарантує стабільність функціонування соціальної вебплатформи Layttle навіть за умов зростаючого навантаження та збільшення обсягів користувацького контенту.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

Висновки до першого розділу

У першому розділі було здійснено детальне дослідження предметної області та доведено актуальність створення соціальної мережі Layttle. Огляд існуючих платформ дав змогу виділити їхні сильні та слабкі сторони, а також знайти вільну нішу для нашого продукту. Крім того, визначено основну аудиторію та описано базові сценарії взаємодії користувачів із платформою: від реєстрації й створення особистого профілю до публікації дописів, коментарів та спілкування у стрічці публікацій. На базі проведеного аналізу сформовано чіткі вимоги до майбутньої системи та обґрунтовано вибір технологічного стеку. Було доведено, що для надійного зберігання даних і гнучкої роботи з публікаціями ідеально підходить поєднання нативного PHP, середовища WAMP64 та реляційної СУБД MySQL із підтримкою форматів JSON. Також у розділі визначено ролеву модель доступу, яка розмежовує можливості звичайних користувачів, модераторів та адміністраторів платформи, а також виділено ключові нефункціональні показники продуктивності й безпеки. Усі зібрані дані сформували міцний фундамент для подальшого проєктування та реалізації архітектури соціальної мережі.

Окрім загального аналізу ринкових аналогів та формування базових вимог, у першому розділі було детально досліджено специфіку організації сучасних соціальних мереж та тенденції розвитку вебтехнологій. Особливу увагу приділено вивченню архітектурних патернів побудови клієнт-серверних додатків, що дозволило обрати класичну модель MVC (Model-View-Controller) як найбільш оптимальну для проєктування вебплатформи Layttle. Такий підхід забезпечує чітке логічне відокремлення інтерфейсу користувача від бізнес-логіки та шару управління даними, що критично важливо для подальшого масштабування та супроводу програмного продукту.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1. Моделювання варіантів використання та поведінки системи

Моделювання варіантів використання є обов'язковим етапом об'єктно-орієнтованого проєктування. Цей процес дає змогу чітко окреслити межі програмного комплексу, виділити головних дійових осіб та детально формалізувати сценарії їхньої щоденної взаємодії з платформою. Демонстраційна розширена діаграма варіантів використання зображена на (рис.2.1.).

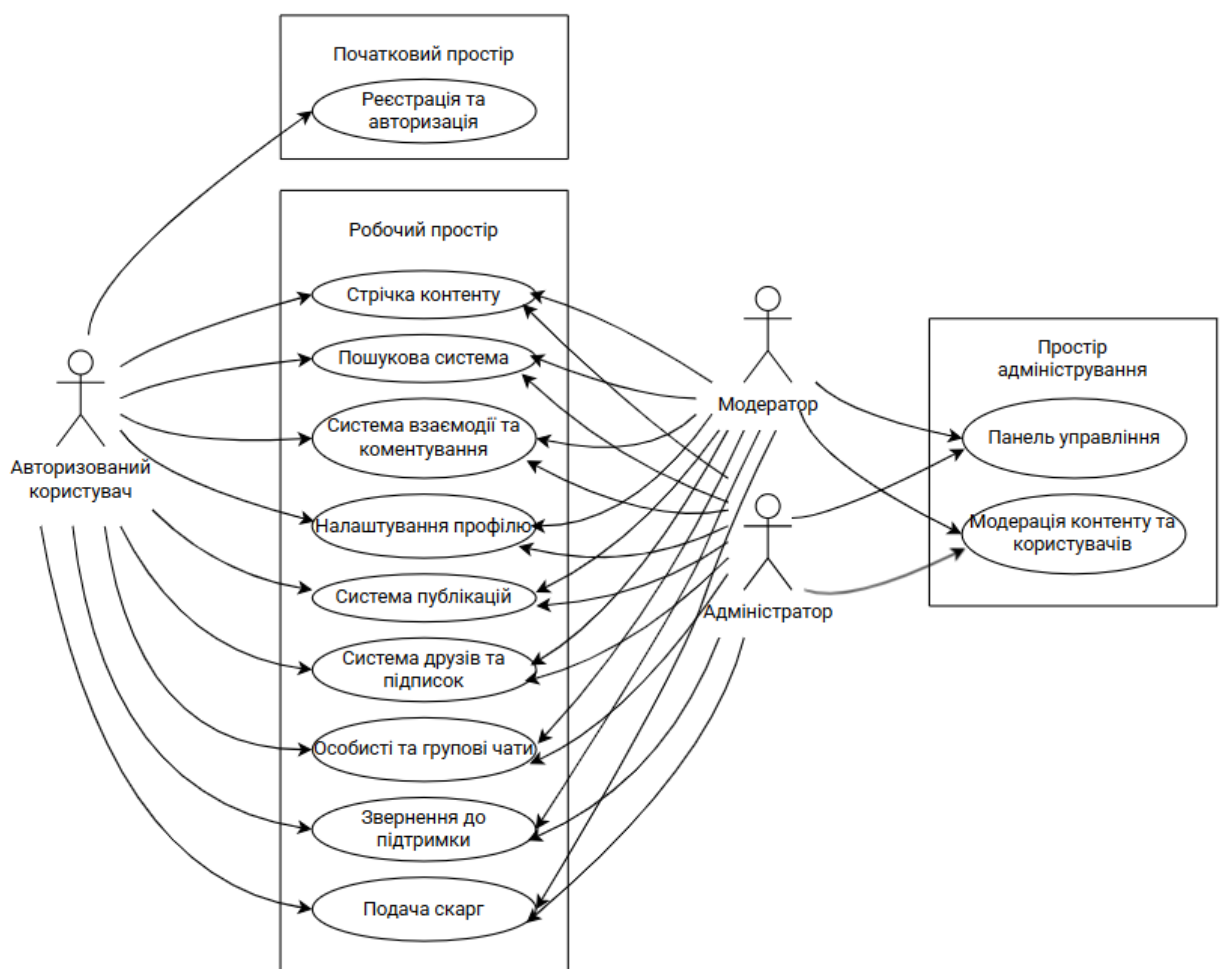


Рис.2.1. Діаграма варіантів використання системи

На основі представленої моделі, яка використовує управління доступом на основі ролей, в архітектурі платформи виділено чотирьох незалежних акторів. Їхня детальна ідентифікація зони відповідальності наведені у таблиці (табл.2.1.).

Таблиця 2.1.

Ідентифікація акторів системи

Актор	Опис ролі та зона відповідальності
Авторизований користувач	Авторизований користувач. Повноправний учасник соціальної мережі. Має доступ до робочого простору, де може створювати власні публікації, спілкуватися в особистих та групових чатах, керувати стрічкою публікацій, додавати друзів, налаштовувати свій профіль, а також звертатися до служби підтримки чи подавати скарги.
Модератор	Це співробітник або уповноважений учасник команди платформи, який працює в просторі адміністрування. Його головне завдання полягає у перевірці контенту, опрацюванні скарг від користувачів та контролю за дотриманням правил внутрішньої спільноти.
Адміністратор	Це системний керівник проєкту, який володіє найвищим рівнем доступу до всіх підсистем Layttle. Окрім стандартних можливостей, він керує глобальною панеллю управління, контролює роботу платформи та відповідає за її технічне забезпечення.

З огляду на специфіку та масштаби розроблюваної соціальної мережі Layttle, весь функціонал і сценарії взаємодії користувачів із системою спроектовані та розбиті на три логічні зони: початковий простір, робочий простір та простір адміністрування.

На рівні початкового простору реалізовано ідентифікацію в платформі. Зокрема, кожен новачок системи має змогу пройти процедуру створення облікового запису та безпечної авторизації для отримання повного доступу до функціоналу соцмережі.

Основне навантаження зосереджено у робочому просторі, де розгортається вся активність користувача. Тут задіяний повноцінний набір інструментів для комунікації та самовираження: перегляд і персоналізація стрічки контенту, швидкий пошук людей за допомогою пошукової системи, а також публікація власних дописів з медіафайлами. Крім того, робочий простір охоплює управління профілем, взаємодію через систему лайків та коментарів, розвиток мережі друзів чи підписок, активне спілкування в особистих і групових чатах у реальному часі, а також можливість звернутися до техпідтримки чи надіслати скаргу.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

За безперестанну роботу системи, порядок та розв'язання конфліктів відповідає простір адміністрування. У цій зоні актори з розширеними повноваженнями модератори та адміністратори здійснюють модерацію контенту, контролюють дотримання правил спільноти та розв'язують спірні ситуації. Своєю чергою, адміністратор володіє виключним доступом до спеціальної панелі управління для глобального моніторингу платформи.

Завдяки такій архітектурній структурі є чітке розуміння обов'язків: звичайні користувачі зосереджені на соціальній взаємодії в робочій зоні, тоді як персонал управління забезпечує безпеку та підтримку порядку.

Для глибшого розуміння внутрішньої логіки та підготовки до програмної реалізації, для ключових процесів платформи було сформовано детальні специфікації. У них покроково розписані сценарії поведінки користувача, системні передумови та реакції на можливі помилки. Зокрема, у таблиці (табл.2.2.). наведено специфікацію базового прецеденту “Реєстрація та авторизація”, який гарантує безпечний доступ до особистого кабінету та керування сесіями.

Таблиця 2.2.

Специфікація прецеденту "Реєстрація та Авторизація"

Атрибут	Опис
Назва прецеденту	Реєстрація та Авторизація
Актор	Неавторизований користувач
Мета	Отримати доступ до закритого функціоналу соціальної мережі Layttle стрічки контенту, чатів, профілю шляхом створення акаунта або входу в систему.
Передумови	Неавторизований користувач намагається перейти на будь-яку сторінку платформи, але система блокує перегляд і автоматично перенаправляє його на форму входу чи реєстрації.
Основний сценарій (Успішний)	1. Система перехоплює спробу неавторизованого доступу та редиректить неавторизованого користувача на сторінку авторизації. 2. Неавторизований користувач обирає необхідну вкладку: “Увійти”, якщо акаунт уже є або “Зареєструватися”. 3. Неавторизований користувач заповнює поля форми: ім'я, email, пароль та підтверджує дію. 4. Клієнтська частина проводить первинну перевірку заповненості полів та надсилає запит на сервер. 5. Сервер обробляє дані, створює нового користувача в базі MySQL або перевіряє існуючі дані та ініціює сесію.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

	6. Сервер повертає успішну відповідь, після чого система перенаправляє користувача до його персонального робочого простору стрічки контенту.
Альтернативний сценарій (Помилки)	4а. Помилка валідації на клієнті: введено невірний формат пошти або занадто короткий пароль — форма підсвічує помилку, блокуючи відправку. 5а. Конфлікт при реєстрації: користувач із такою поштою вже зареєстрований — сервер повертає помилку, інтерфейс виводить сповіщення про необхідність увійти. 5б. Помилка авторизації: введено невірний пароль або неіснуючий email — сервер відхиляє запит, виводиться загальне повідомлення про помилку входу без розкриття деталей безпеки.
Постумови	Неавторизований користувач набуває статусу авторизованого користувача. У системі формується активна сесія, що відкриває повний доступ до всього функціоналу платформи.

Головною особливістю платформи є зручна взаємодія користувачів із динамічним контентом та медіафайлами в режимі реального часу. У таблиці (табл.2.3.) наведено інженерну специфікацію прецеденту «Створення та публікація дописів», який виступає центральним і найбільш інтерактивним елементом клієнтської частини.

Таблиця 2.3.

Специфікація прецеденту "Створення та публікація дописів"

Атрибут	Опис
Назва прецеденту	Створення та публікація дописів
Актор	Авторизований користувач
Мета	Створити новий публічний чи приватний допис із текстовим описом та медіафайлами і розмістити його в своєму профілі.
Передумови	Користувач успішно авторизований у системі, перебуває у своєму робочому просторі та має активне інтернет-з'єднання.
Основний сценарій (Успішний)	1. Користувач відкриває форму створення публікації у своєму профілі. 2. Система відображає інтерактивне поле введення тексту та кнопку завантаження зображень/медіа. 3. Користувач вводить текстовий вміст допису та обирає файли для прикріплення. 4. Клієнтський інтерфейс здійснює первинну перевірку розміру та формату завантажених файлів. 5. Користувач натискає кнопку "Опублікувати". 6. Система формує запит FormData та передає дані на сервер через POST API. 7. Сервер приймає запит, валідує текстовий контент, безпечно зберігає медіафайли на сервері та записує інформацію про новий пост у базу даних MySQL. 8. Сервер повертає успішну відповідь, після чого стрічка контенту автоматично оновлюється, відображаючи новий допис.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Альтернативний сценарій (Помилки)	4а. Некоректний формат або перевищення ліміту файлу: система блокує відправку запиту, виводить сповіщення про помилку та пропонує обрати інший файл. ба. Втрата з'єднання з мережею: система перехоплює мережеву помилку під час відправки, сповіщає користувача та тимчасово зберігає чернетку локально. 7а. Помилка обробки на сервері: сервер відхиляє запит через збій бази даних або внутрішню помилку, клієнтська частина виводить повідомлення про неможливість публікації та пропонує спробувати пізніше.
Постумови	Новий допис успішно збережений у базі даних і стає доступним у стрічці контенту користувача та його підписників.

Окрім повсякденної взаємодії користувачів, безперебійність роботи та безпека забезпечуються адміністраторами та модераторами. У таблиці (табл.2.4.) наведено детальну специфікацію прецеденту модерації контенту, яка відображає захисні механізми платформи та процес зміни статусів публікацій чи акаунтів адміністративним складом.

Таблиця 2.4.

Специфікація прецеденту "Модерація контенту та користувачів"

Атрибут	Опис
Назва прецеденту	Модерація контенту та користувачів
Актор	Модератор, Адміністратор
Мета	Обмежити публічний доступ до дописів, коментарів або заблокувати обліковий запис користувача, що порушує внутрішню політику та правила вебплатформи.
Передумови	Модератор або адміністратор успішно авторизовані в системі, перебувають в панелі управління та ідентифікували проблемний допис чи акаунт-порушника.
Основний сценарій (Успішний)	- Модератор знаходить необхідний контент чи профіль за допомогою системи скарг або внутрішньої фільтрації. - Модератор обирає проблемну публікацію або профіль та натискає дію "Заблокувати" або "Видалити контент". - Система вимагає обов'язкового зазначення причини блокування з випадаючого списку чи текстового поля. - Модератор вводить обґрунтування та підтверджує виконання операції. - Сервер оновлює запис у базі даних MySQL, змінюючи статус допису/акаунта з активного на заблокований. - Система автоматично генерує сповіщення про порушення та надсилає його власнику контенту із зазначенням причини. - Інтерфейс оновлюється, а заблокований елемент зникає із загальної стрічки публікацій вебплатформи.

Альтернативний сценарій (Помилки)	5а. Конкурентний доступ (контент вже видалено власником): сервер не знаходить активного запису в базі даних, система виводить модератору сповіщення про неактуальність запиту та оновлює таблицю скарг. 5б. Обмеження прав: спроба модератора застосувати дії щодо захищеного акаунта іншого модератора чи адміністратора система перевіряє рівень доступу та відхиляє операцію, виводячи помилку доступу.
Постумови	Контент який порушує правила спільноти стає недоступним для перегляду звичайними користувачами, а статус облікового запису порушника змінюється відповідно до накладених адміністративних обмежень.

Окрім публікації дописів та перегляду контенту, важливим елементом платформи є пряма комунікація між користувачами. У таблиці (табл.2.5.) наведено детальну специфікацію прецеденту обміну повідомленнями в чаті, яка відображає комунікаційні механізми платформи та процес передачі даних у режимі реального часу.

Таблиця 2.5.

Специфікація прецеденту "Обмін повідомленнями в чаті"

Атрибут	Опис
Назва прецеденту	Обмін повідомленнями в чаті.
Актор	Авторизований користувач.
Мета	Надіслати та отримати текстове повідомлення в особистому або груповому чаті в режимі реального часу.
Передумови	Користувач авторизований, відкрив розділ повідомлень і обрав активний діалог із користувачем чи групою.
Основний сценарій (Успішний)	1. Користувач вводить текстове чи медіа повідомлення з можливим вибором смайликів у поле введення. 2. Натискає кнопку "Надіслати" або клавішу Enter. 3. Клієнтський скрипт формує пакет даних і передає його на сервер. 4. Сервер отримує повідомлення, зберігає його в реляційній базі даних MySQL. 5. Система миттєво доставляє повідомлення в активний чат іншому користувачу або учасникам групи. 6. Інтерфейс автоматично прокручується до нового повідомлення.
Альтернативний сценарій (Помилки)	1а. Порожнє повідомлення: якщо поле пусте або містить самі пробіли, кнопка відправки неактивна. 3а. Втрата зв'язку: система фіксує розрив мережі, виводить сповіщення про помилку надсилання та пропонує повторити спробу.
Постумови	Повідомлення надійно збережене в базі даних та відображається у вікні діалогу обох співрозмовників.

Сформовані специфікації прецедентів чітко визначають взаємодію клієнтської частини з сервером. Вони детально ілюструють, що кожна дія користувача чи адміністратора нерозривно пов'язана з обробкою на бекенді: перевіркою прав доступу, валідацією даних, управлінням сесіями та запитам до бази даних. Зібрана деталізація виступає практичною основою для подальшого проєктування архітектури класів, розробки API-маршрутів та реалізації програмної логіки платформи.

2.2. Проєктування логічної структури бази даних

Вибір системи управління базами даних є ключовим етапом розробки, оскільки архітектура вебплатформи поєднує в собі високі вимоги до надійності зв'язків: друзі, чати, підписки та гнучкості при зберіганні контенту пости, мультимедіа. Серед доступних рішень розглядалися реляційні та документоорієнтовані СУБД. Попри переваги NoSQL-систем неструктурованих даних, для проєкту було обрано реляційну систему MySQL версії 9.1.0. Вона забезпечує сувору транзакційну цілісність ACID, що критично важливо для збереження особистих повідомлень, взаємин між користувачами та системних логів. Крім того, дев'ята версія має потужну оптимізацію для роботи з типами даних JSON, що дозволяє гнучко адаптувати параметри профілів та конфігурації медіафайлів без широкої структури таблиць.

Завдяки гібридному підходу, реляційні зв'язки гарантують безпеку та унікальність облікових записів, тоді як вбудовані JSON-поля дають можливість ефективно зберігати динамічні метадані дописів та налаштувань користувача. Це спрощує архітектуру сховища, знижує кількість зайвих зв'язків між таблицями та прискорює загальну швидкість обробки запитів на бекенді.

Логічна структура бази даних вебплатформи охоплює взаємопов'язані таблиці, які комплексно забезпечують управління обліковими записами, стрічкою контенту, системою лайків і коментарів, механізмом дружби, особистими та груповими чатами, а також модулем адміністрування. Для уникнення дублювання інформації та забезпечення цілісності даних на рівні

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

сховища, більшість реляційних таблиць нормалізовано. Загальну архітектуру основних сутностей, їхні атрибути, первинні й зовнішні ключі, а також логічні зв'язки між ними відображено у вигляді діаграми “сутність-зв'язок” на (рис.2.2.), яка формує цілісний інформаційний простір.

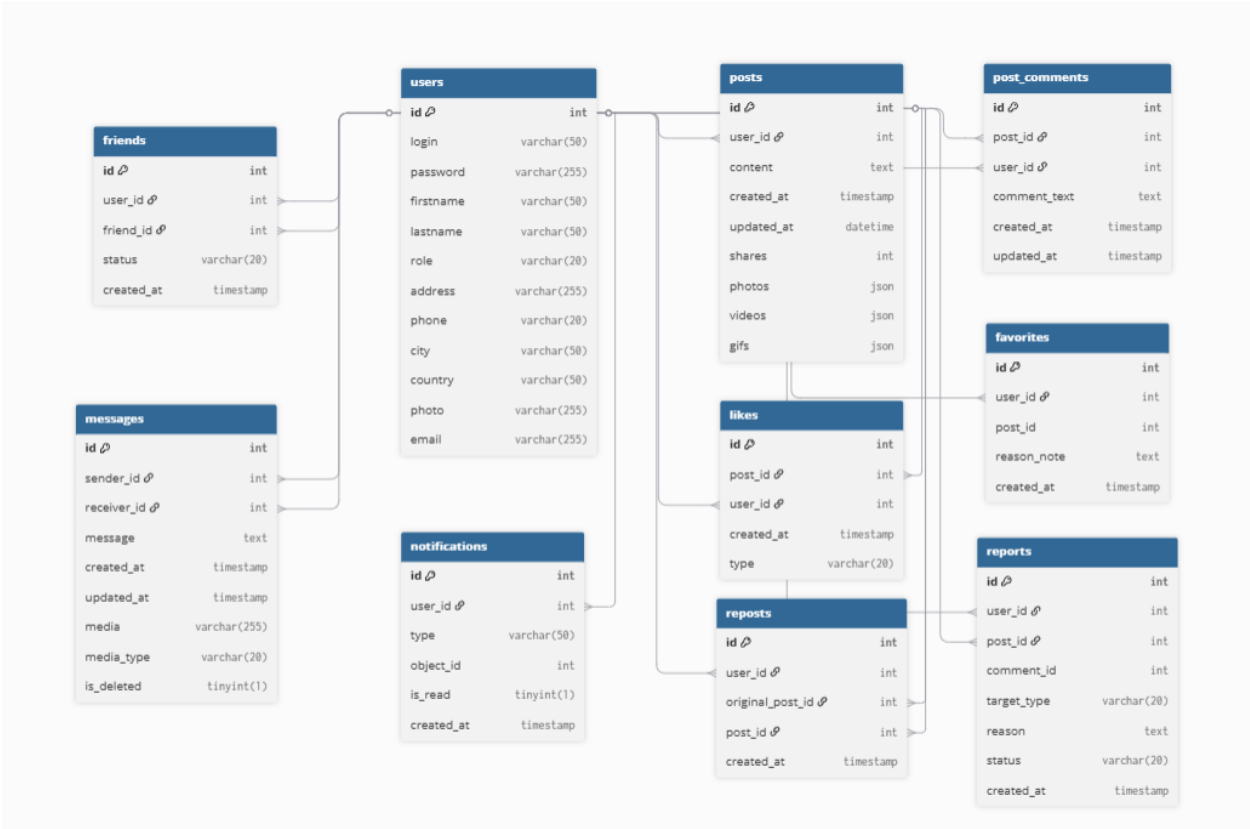


Рис. 2.2. Схема реляційної бази даних Layttle

Фундаментом системи є таблиця користувачів **users**, яка зберігає профільні дані користувачів та виступає головною батьківською сутністю для більшості функціональних модулів платформи. На основі первинних та зовнішніх ключів реалізовано такі ключові зв'язки: взаємодія між користувачами та публікаціями **posts** побудована за принципом “один-до-багатьох”. Один користувач має можливість створювати необмежену кількість дописів, проте кожен окремий пост прив'язаний до свого автора через поле **user_id**. Аналогічним чином організовані залежності для коментарів постів **post_comments** та вподобань **likes**, що гарантує точну ідентифікацію походження кожного елемента активності.

Для забезпечення зв'язків між користувачами реалізовано таблицю дружби **friends**, яка дає змогу переглядати контент друзів та легше знаходити своїх серед

підписок для бажаного чатування. Модуль приватних повідомлень messages використовує подвійну прив'язку до сутності користувачів sender_id та receiver_id, що дозволяє створювати захищені канали персонального листування.

Архітектура також включає спеціалізовані таблиці для адміністративного контролю та сповіщень. Зокрема, модуль скарг reports забезпечує гнучке відстеження порушень як стосовно публікацій, так і відносно коментарів завдяки використанню полів target_type та прив'язці до відповідних ідентифікаторів.

Таке архітектурне рішення забезпечує логічну ізоляцію даних, мінімізує надлишковість інформації та створює надійну основу для стабільного функціонування всіх сервісів соціальної мережі в умовах високого навантаження.

Проектування апаратно-інформаційного ядра реалізовано на основі реляційної моделі, яка забезпечує високу цілісність, нормалізацію та швидку обробку запитів. Логічна структура бази даних охоплює сукупність взаємопов'язаних таблиць, специфікації яких детально наведено у таблицях (табл.2.5. - табл.2.14.) пояснювальної записки.

Таблица 2.5.

Структура таблиці users

Назва поля	Тип даних	Обмеження	Опис
id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор користувача.
login	VARCHAR (255)	NOT NULL	Логін користувача в системі.
password	VARCHAR (255)	NOT NULL	Хешований пароль облікового запису.
firstname	VARCHAR (50)	NOT NULL	Ім'я користувача.
lastname	VARCHAR (50)	NOT NULL	Прізвище користувача.
role	VARCHAR (20)	NOT NULL, DEFAULT 'user'	Роль користувача в системі.
address	VARCHAR (255)	NOT NULL	Поштова адреса користувача.
phone	VARCHAR (20)	NOT NULL	Контактний номер телефону.
city	VARCHAR (50)	NOT NULL	Місто проживання.

продовж.табл.2.5.

country	VARCHAR (50)	NOT NULL	Країна проживання.
photo	VARCHAR (255)	NULL, DEFAULT NULL	Фото.
email	VARCHAR (255)	NOT NULL	Електронна пошта користувача.

Таблиця (табл.2.6.) posts містить текстовий контент дописів, медіа вкладення у форматі JSON фото, відео, GIF, а також лічильник поширень та часові мітки створення й редагування.

Таблиця 2.6.

Структура таблиці posts

Назва поля	Тип даних	Обмеження	Опис
id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор публікації.
user_id	INT	FK, NOT NULL	Посилання на автора допису.
content	TEXT	NOT NULL	Текстовий вміст публікації.
created_at	TIMESTAMP	NULL, DEFAULT CURRENT_TIMESTAMP	Час створення допису.
updated_at	DATETIME	NULL, DEFAULT NULL	Час останнього редагування публікації.
shares	INT	NULL, DEFAULT NULL	Лічильник поширень допису.
photos	JSON	NULL, DEFAULT NULL	Масив прикріплених фотографій у форматі JSON.
videos	JSON	NULL, DEFAULT NULL	Масив прикріплених відеофайлів у форматі JSON.
gifs	JSON	NULL, DEFAULT NULL	Масив анімацій GIF у форматі JSON.

Для організації інтерактивної взаємодії користувачів та обговорення контенту на платформі використовується модуль коментарів.

Таблиця post_comments забезпечує збереження коментарів до публікацій, фіксує зв'язок із автором і дописом, а також містить часові мітки створення та оновлення. (табл.2.7.).

Таблиця 2.7.

Структура таблиці post_comments

id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор коментаря.
post_id	INT	FK, NOT NULL	Посилання на публікацію, до якої залишено коментар.
user_id	INT	FK, NOT NULL	Посилання на автора коментаря.
comment_text	TEXT	NOT NULL	Текстовий зміст коментаря.
created_at	TIMESTAMP	NULL, DEFAULT CURRENT_TIMESTAMP	Дата та час створення коментаря.
updated_at	TIMESTAMP	NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	Дата та час останнього редагування коментаря.

Для реалізації можливості оцінювання та висловлення емоційної реакції на контент у соціальній мережі використовується модуль вподобань. Таблиця likes (табл.2.8.) забезпечує фіксацію зв'язку між користувачами та публікаціями, які вони вподобали, а також зберігає тип реакції та час її створення.

Таблиця 2.8.

Структура таблиці likes

Назва поля	Тип даних	Обмеження	Опис
id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор вподобання.
post_id	INT	FK, NOT NULL	Посилання на публікацію, якій поставлено реакцію.
user_id	INT	FK, NOT NULL	Посилання на користувача, який залишив реакцію.
created_at	TIMESTAMP	NULL, DEFAULT CURRENT_TIMESTAMP	Дата та час створення вподобання.
type	VARCHAR (50)	NULL, DEFAULT '♥'	Візуальне оформлення реакції емодзі.

Для реалізації можливості збереження улюблених публікацій та додавання їх до списку обраного в соціальній мережі використовується спеціальний модуль. Таблиця favorites (табл.2.9.) забезпечує фіксацію зв'язку між користувачами та обраними дописами, а також дозволяє залишати нотатки або причини додавання і фіксує час створення запису.

Таблиця 2.9.

Структура таблиці favorites

Назва поля	Тип даних	Обмеження	Опис
id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор вподобання.
user_id	INT	FK, NOT NULL	Посилання на публікацію, якій поставлено реакцію.
reason_note	TEXT	NULL, DEFAULT NULL	Текстова нотатка або причина збереження в обране.
created_at	TIMESTAMP	NULL, DEFAULT CURRENT_TIMESTAMP	Дата та час додавання публікації до обраного.
post_id	INT	FK, NOT NULL	Посилання на збережену публікацію.

Для реалізації можливості поширення публікацій та створення репостів у соціальній мережі використовується спеціальний модуль. Таблиця reposts (табл.2.10.) забезпечує репост посту інших користувачів на своїй власній сторінці з вказуванням автора оригінального посту, а також фіксує час створення запису.

Таблиця 2.10.

Структура таблиці reposts

Назва поля	Тип даних	Обмеження	Опис
id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор репосту.
user_id	INT	FK, NOT NULL	Посилання на користувача, який здійснив репост.
original_post_id	INT	FK, NOT NULL	Посилання на оригінальну публікацію, яку поширюють.
created_at	TIMESTAMP	NULL, DEFAULT CURRENT_TIMESTAMP	Дата та час створення репосту.
post_id	INT	FK, NOT NULL	Посилання на згенерований запис репосту в системі.

Для забезпечення модерації контенту, безпеки та обробки скарг від користувачів у соціальній мережі використовується спеціальний модуль. Таблиця reports (табл.2.11.) забезпечує фіксацію звернень щодо порушень, зберігає інформацію про скарги на публікації або коментарі, причину скарги, поточний статус розгляду та час створення запису.

Таблиця 2.11.

Структура таблиці reports

Назва поля	Тип даних	Обмеження	Опис
id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор скарги.
user_id	INT	FK, NOT NULL	Посилання на користувача, який надіслав скаргу.
post_id	INT	FK, NULL	Посилання на публікацію, на яку скаржаться.
comment_id	INT	FK, NULL	Посилання на коментар, на який скаржаться.
target_type	ENUM ('post', 'comment')	NOT NULL	Тип об'єкта скарги (публікація чи коментар).
reason	TEXT	NOT NULL	Причина або опис суті скарги.
status	ENUM ('pending', 'reviewed', 'resolved')	NULL, DEFAULT 'pending'	Поточний стан розгляду скарги адміністрацією.
created_at	TIMESTAMP	NULL, DEFAULT CURRENT_TIMESTAMP	Дата та час створення скарги.

Для організації соціальних зв'язків, управління запитами в друзі та формування кола спілкування користувачів у соціальній мережі використовується спеціальний модуль. Таблиця friends (табл.2.12.) забезпечує фіксацію взаємин між користувачами, зберігає ідентифікатори ініціатора та друга, поточний статус зв'язку очікування, прийнято чи відхилено та час створення запису.

Таблиця 2.12.

Структура таблиці friends

Назва поля	Тип даних	Обмеження	Опис
id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор запису дружби.
user_id	INT	FK, NOT NULL	Посилання на користувача, який надіслав запит у друзі.
friend_id	INT	FK, NOT NULL	Посилання на користувача, якого додають у друзі.
status	ENUM	NULL, DEFAULT 'pending'	Поточний статус зв'язку очікування, прийнято, відхилено.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

продовж.табл.2.12.

	('pending', 'accepted', 'rejected')		
created_at	TIMESTAMP	NULL, DEFAULT CURRENT_TIMESTAP	Дата та час створення зв'язку.

Для забезпечення приватного спілкування та обміну медіафайлами між користувачами соціальної мережі реалізовано модуль системи повідомлень. Таблиця messages (табл.2.13.) забезпечує збереження історії листування, фіксує ідентифікатори відправника та отримувача, текстовий вміст, прикріплені медіафайли, а також статус логічного видалення повідомлення.

Таблиця 2.13.

Структура таблиці messages

Назва поля	Тип даних	Обмеження	Опис
id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор запису дружби.
sender_id	INT	FK, NOT NULL	Посилання на користувача, який відправив повідомлення.
receiver_id	INT	FK, NOT NULL	Посилання на користувача, якому адресовано повідомлення.
message	TEXT	NOT NULL	Текстований вміст приватного повідомлення.
created_at	TIMESTAMP	NULL, DEFAULT CURRENT_TIMESTAMP	Дата та час відправлення повідомлення.
updated_at	TIMESTAMP	NULL, DEFAULT NULL	Дата та час останнього редагування повідомлення.
media	VARCHAR (255)	NULL, DEFAULT NULL	Посилання або шлях до прикріпленого медіафайлу.
media_type	ENUM ('image', 'video')	NULL, DEFAULT NULL	Тип прикріпленого медіафайлу зображення або відео.
Is_deleted	TINYINT(1)	NULL, DEFAULT 0	Прапорець логічного видалення (0 – активне, 1 – видалене).

Для забезпечення своєчасного інформування користувачів про активність у системі та події в соціальній мережі реалізовано модуль сповіщень. Таблиця (табл.2.14.) notifications забезпечує збереження сповіщень, фіксує ідентифікатор користувача, тип події, посилання на зв'язаний об'єкт, статус прочитання та час створення запису.

Таблиця 2.14.

Структура таблиці messages

Назва поля	Тип даних	Обмеження	Опис
id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор сповіщення.
user_id	INT	FK, NOT NULL	Посилання на користувача, якому призначено сповіщення.
type	VARCHAR(50)	NOT NULL	Тип або категорія сповіщення.
object_id	INT	NOT NULL	Ідентифікатор об'єкта, до якого стосується сповіщення.
is_read	TINYINT(1)	NULL, DEFAULT 0	Статус прочитання (0 - не прочитано, 1 - прочитано) .
created_at	TIMESTAMP	NULL, DEFAULT NULL	Дата та час створення сповіщення.

Ключовим архітектурним та інноваційним рішенням вебплатформи Layttle є оптимізований підхід до збереження мультимедійного контенту всередині користувацьких публікацій. Замість створення складних розгалужених реляційних зв'язків та окремих дочірніх таблиць для кожного типу файлів, у таблиці posts реалізовано спеціалізовані поля photos, videos та gifs, які зберігають структуровані набори даних у форматі JSON.

Фізично ці дані представляють собою впорядкований масив об'єктів або списків шляхів файлів, що дозволяє одному допису динамічно утримувати до кількох медіафайлів одночасно.

2.3. Проєктування компонентної структури та взаємодії модулів

Для забезпечення процесу розробки, локального тестування та повноцінного функціонування розробленого програмного комплексу Layttle

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

було обрано локально-інтегровану серверну архітектуру на базі програмного пакета Wamp64. Такий підхід дозволяє розгорнути повноцінний вебсервер безпосередньо на робочій станції, об'єднавши всі необхідні службові компоненти в єдиному середовищі виконання.

Роль кінцевої точки взаємодії виконує будь-який сучасний веббраузер на боці користувача. Через нього надсилаються HTTP/HTTPS-запити до локального сервера, здійснюється рендеринг інтерфейсу, обробка динамічних подій за допомогою JavaScript управління модальними вікнами, асинхронні запити fetch, робота з галереями та медіаконтентом.

Локальний вебсервер Apache приймає вхідні запити від клієнта, здійснює маршрутизацію через системний файл router.php та запускає інтерпретатор PHP для виконання бізнес-логіки застосунку. На цьому ж рівні опрацьовуються завантажені користувачем медіафайли зображення, відео, які зберігаються у локальній файловій структурі проєкту.

Управління реляційною базою даних здійснюється за допомогою СУБД MySQL у складі середовища Wamp64. На цьому вузлі розгорнуто всі структуровані таблиці системи users, posts, post_comments, likes, favorites, reposts, reports, friends, messages, notifications, де забезпечується виконання SQL-запитів, підтримка зовнішніх ключів та збереження мультимедійного контенту у форматі JSON.

Подібна локальна конфігурація забезпечує ізольованість середовища розробки, високу швидкість виконання запитів під час тестування та повну сумісність між усіма елементами програмного забезпечення платформи.

На (рис.2.3.) наведено розроблену діаграму розгортання програмного комплексу Layttle, яка відображає локальну інфраструктуру та взаємодію компонентів системи в середовищі розробки та тестування.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

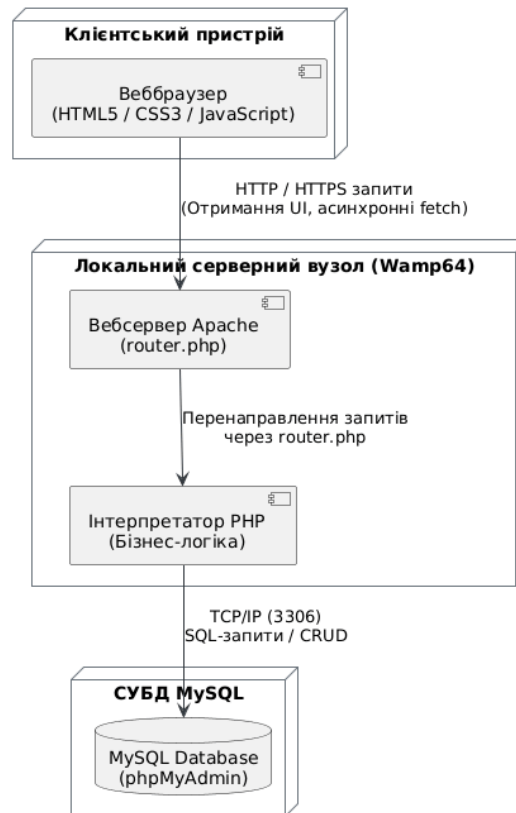


Рис.2.3. Діаграма розгортання системи

Програмна реалізація базується на класичному архітектурному патерні MVC (Model-View-Controller) із власним модульним ядром. Така структура забезпечує чітке розділення відповідальності між обробкою вхідних запитів, бізнес-логікою та візуальним представленням даних.

Життєвий цикл опрацювання запитів у системі побудований за такою логікою:

Увесь зовнішній трафік надходить через єдину точку входу в кореневій директорії `index.php`, після чого системний маршрутизатор `core/Router.php` аналізує вхідний URL-шлях та спрямовує запит до відповідного контролера наприклад, `ProfileController`, `ChatController`, `SiteController` чи `UsersController`.

Базові компоненти платформи реалізовані у каталозі `core/Core.php`, `Controller.php`, `Model.php`, `DB.php`, де налаштовано з'єднання з базою даних MySQL, управління сесіями `Session.php` [20] та базові механізми запитів.

Контролери взаємодіють із моделями `Users.php` та обробниками `api/post_handler.php`, виконуючи CRUD-операції, валідацію вхідних даних та

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмоторя О.Г.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

керування завантаженням медіаконтентом, який фізично зберігається у локальній директорії uploads/posts/, а посилання на нього фіксуються в базі даних.

Формування інтерфейсу користувача відбувається через шаблонізатор core/Template.php та структуру подань views/, розділену за функціональними модулями профіль, чат, авторизація, стрічка контенту та загальні макети layouts/.

Проектування предметної області соціальної мережі базується на принципах об'єктно-орієнтованого програмування та відображає ключові соціальні взаємодії між користувачами платформи.

Основою доменної моделі є клас User, який акумулює облікові дані учасника та виступає головним ядром системи. Від нього беруть початок ключові функціональні зв'язки:

Сутність користувача тісно пов'язана з модулем публікацій Post, де кожен допис містить текстовий вміст та багатокомпонентні медіавкладення, організовані у вигляді нативних JSON-масивів.

Соціальна та інтерактивна активність охоплює зв'язки з коментарями, емодзі-реакціями, збереженими матеріалами та системою взаємних сповіщень.

Модуль комунікації забезпечує управління приватними повідомленнями та дружніми зв'язками між відправниками й отримувачами всередині вебплатформи, що детально відображає структуру класів системи (рис.2.4.).

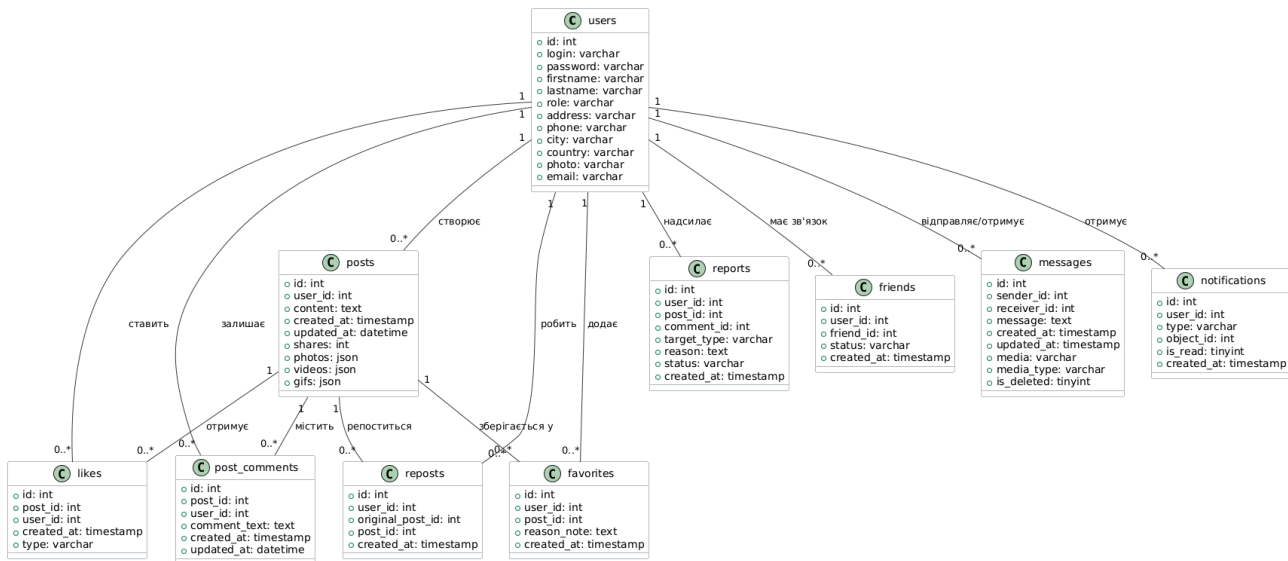


Рис.2.4. Діаграма класів предметної області

Усі компоненти утворюють цілісну систему без ізольованого виділення фронтенд-частини в окремий застосунок. Графічне відображення модульної ієрархії та структури файлів зафіксовано на діаграмі компонентів нижче (рис.2.5.).

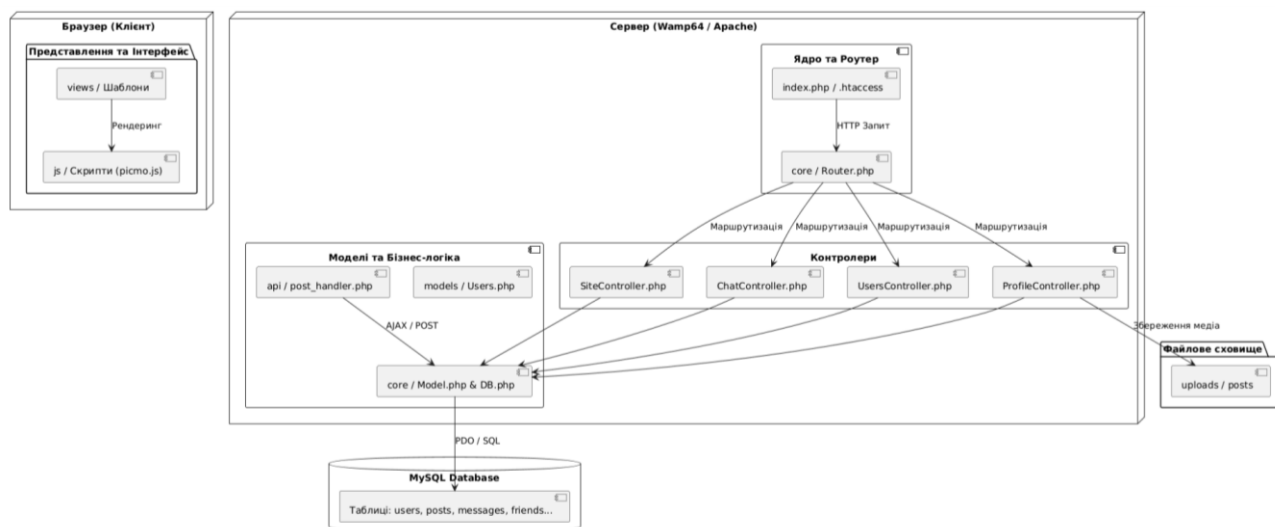


Рис.2.5. Схема компонентів та зв'язків між ними

Організація обміну інформацією між клієнтською частиною та серверним середовищем проєкту реалізована за допомогою стандартного протоколу [21] HTTP, поєднуючи класичний серверний рендеринг та асинхронні запити AJAX без перезавантаження сторінок. Авторизація та підтримка активної сесії користувача контролюються на серверній стороні за допомогою вбудованих механізмів PHP-сесій, що гарантує безпеку та ізольованість облікових записів.

Для наочної демонстрації хронологічного порядку виклику методів, обміну повідомленнями та взаємодії між програмними об'єктами системи використовуються діаграми послідовності. На відміну від статичних схем, вони детально відображають динаміку виконання конкретного сценарію. На (рис.2.6.) наведено діаграму послідовності для процесу створення та публікації нового допису користувачем у соціальній мережі, що демонструє повний шлях передачі даних від інтерфейсу форми до збереження запису в базі даних.

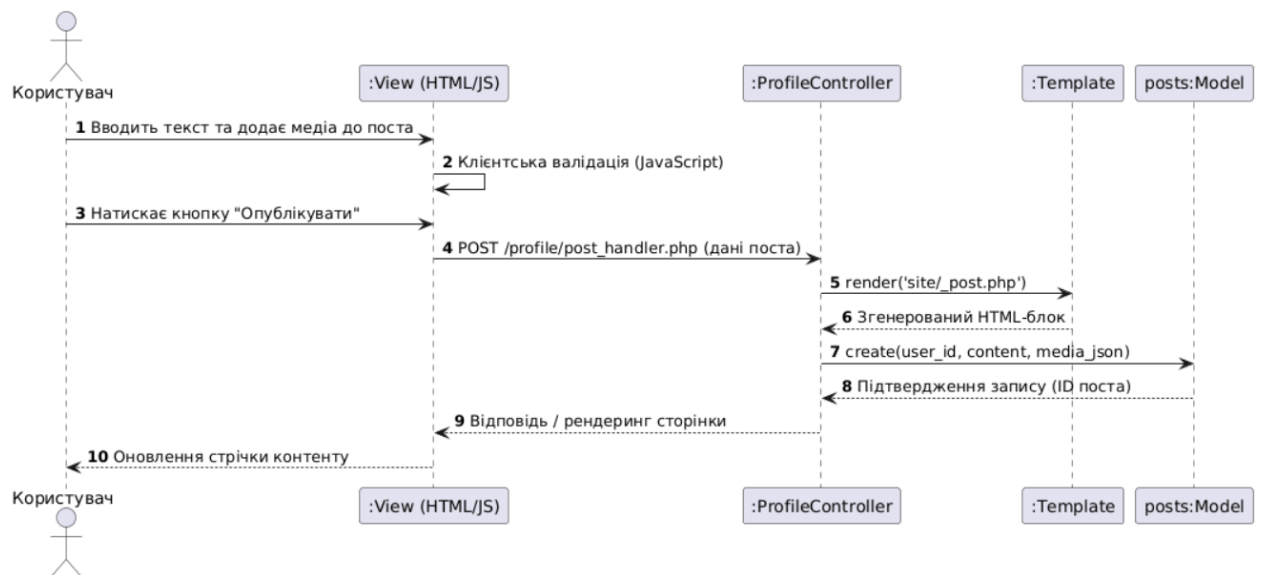


Рис.2.6. Діаграма послідовності процесу публікації нового допису

Логіка комунікації між користувачами всередині вебплатформи починається з того, що учасник відкриває інтерфейс чату ChatView та заповнює форму приватного повідомлення, додаючи за потреби текстовий вміст або медіафайли. Клієнтський скрипт здійснює первинну перевірку введених даних, після чого ініціюється відправка запиту. За допомогою асинхронного AJAX-виклику на серверний обробник ChatController передається JSON-пакет із даними відправника, отримувача та текстом повідомлення.

Маршрутизатор системи перенаправляє вхідний запит до ChatController, який виконує перевірку активної сесії користувача та валідує права доступу. Після цього контролер викликає метод збереження моделі повідомлень MessageModel, передаючи туди ідентифікатори користувачів і текст самого повідомлення. Успішне виконання запиту на додавання нового рядка до таблиці messages супроводжується формуванням відповіді та оновленням сторінки або поверненням користувача до інтерфейсу діалогу з відображенням актуальної історії листування, як це показано на (рис.2.7.).

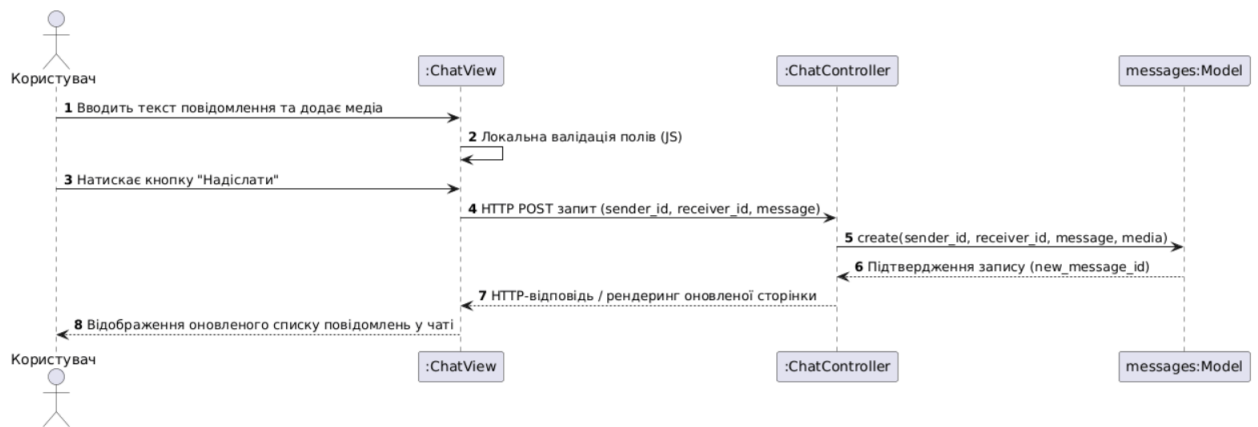


Рис.2.7. Діаграма послідовності процесу надсилання приватного повідомлення

Окрім створення публікацій та приватного листування, важливим функціональним елементом вебплатформи Layttle є взаємодія користувачів зі стрічкою контенту. Під час завантаження головної сторінки SiteController серверне середовище виконує запит до бази даних MySQL, збираючи масив актуальних дописів разом із прикріпленими медіафайлами у форматі JSON, інформацією про авторів та кількістю емодзі-реакцій і коментарів.

Користувач має можливість переглядати стрічку, залишати текстові коментарі під постами через форму зворотного зв'язку або виставляти швидкі емодзі-вподобайки за допомогою інтерактивної піксель-панелі. Клієнтський інтерфейс надсилає асинхронний запит на серверний обробник, після чого база даних оновлює відповідні записи у таблицях post_comments або likes, а інтерфейс миттєво відображає зміни в реальному часі без перезавантаження сторінки, що зображено на (рис.2.8).

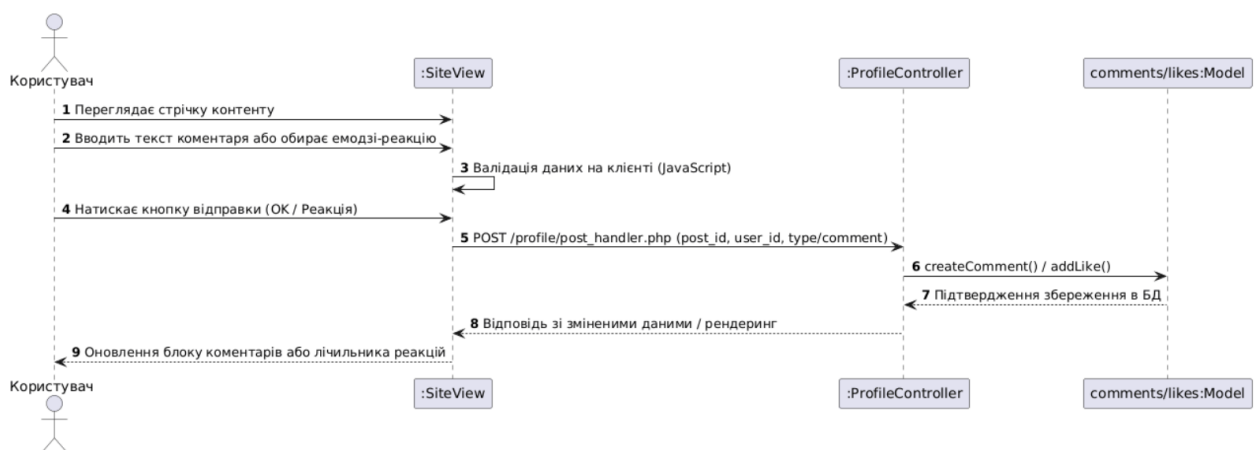


Рис. 2.8. Діаграма послідовності процесу коментування та реагування на публікацію в стрічці.

2.4. Розробка основних алгоритмів роботи платформи

Було спроектовано детальні алгоритми основних взаємодій. Нижче наведено блок-схеми та опис послідовності дій для ключових модулів системи: публікації контенту, взаємодії зі стрічкою контенту, а також обміну приватними повідомленнями в реальному часі.

Створення та оприлюднення нової публікації є базовим сценарієм для будь-якої соціальної мережі. Логіка цього процесу охоплює перевірку даних на клієнті, безпечну передачу на бекенд та фіксацію в базі даних MySQL, що чітко демонструє покрокову логіку виконання цієї операції (рис.2.9.). Застосування багаторівневої валідації дозволяє мінімізувати навантаження на серверні ресурси, усуваючи невалідні або порожні пакети даних ще на етапі браузерної обробки. Крім того, архітектурне рішення передбачає механізми захисту від повторного надсилання форм, що підвищує загальну стійкість системи до непередбачуваних дій користувача.

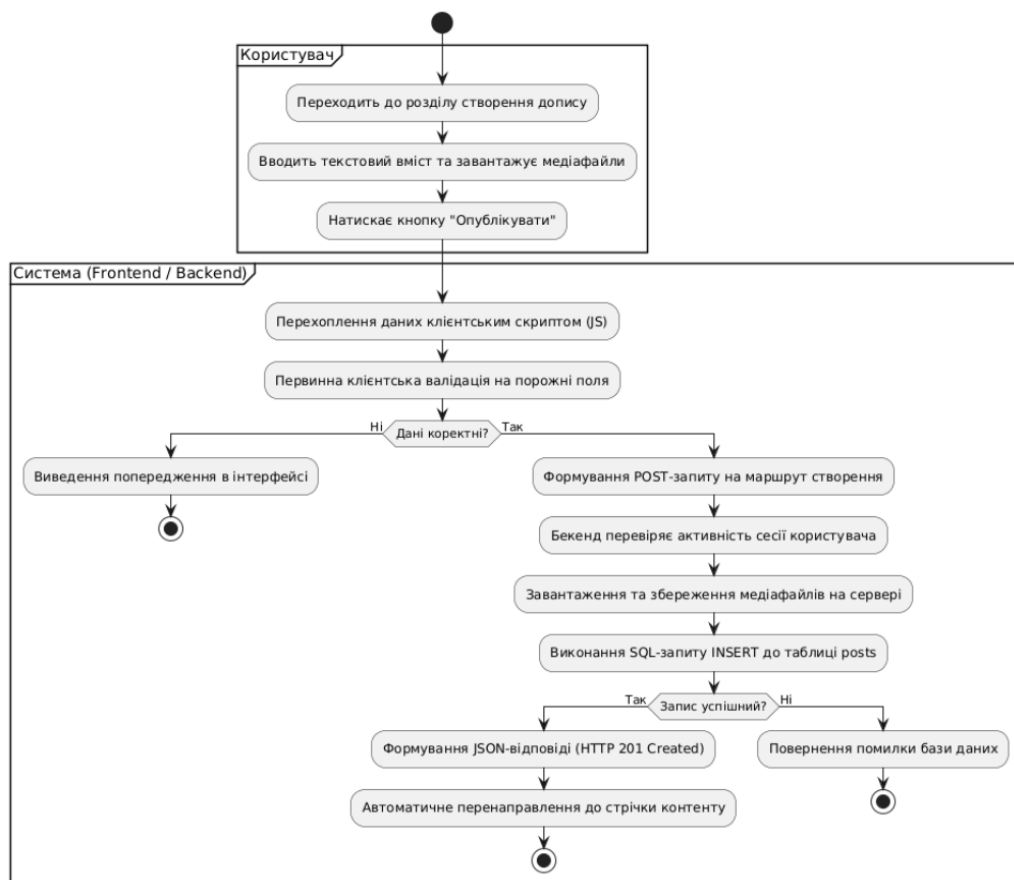


Рис.2.9. Алгоритм процесу публікації нового допису користувачем

Життєвий цикл створення допису розпочинається з ініціації форми на клієнті. Після введення контенту та прикріплення медіафайлів спрацьовує локальна перевірка JavaScript, що дозволяє уникнути зайвих запитів до сервера у разі порожніх полів. Успішно проаналізований пакет надсилається на бекенд, де контролер перевіряє права сесії, обробляє медіа та формує SQL-запит на додавання запису до бази даних. У разі успіху користувач миттєво перенаправляється на оновлену стрічку контенту.

Процес обробки завантажених мультимедійних файлів включає перевірку їхнього розширення, обмеження максимального розміру та генерацію унікальних ідентифікаторів імен, аби уникнути конфліктів у файловій системі сервера.

Додатково на рівні серверного середовища виконується кешування тимчасових параметрів запиту та перевірка цілісності потоку даних, що мінімізує ризики виникнення збоїв під час паралельної обробки кількох мультимедійних потоків від різних активних користувачів платформи.

Згідно з цією схемою, взаємодія розпочинається з того, що користувач переглядає стрічку контенту та обирає конкретну публікацію для подальшої реакції. Залежно від типу обраної дії, алгоритм розгалужується на два можливі сценарії: написання коментаря або встановлення емодзі-вподобайки. У першому випадку користувач вводить текст у спеціальну форму коментаря та натискає кнопку відправки “OK”, тоді як у другому відкриває панель емодзі та обирає потрібну реакцію. Після цього сформований запит передається на серверну частину за допомогою POST-запиту, де бекенд насамперед визначає ідентифікатор поточного користувача та унікальний ID самого допису.

На наступному етапі система знову розгалужує виконання залежно від типу операції. Якщо це коментар, виконується валідація довжиною введеного тексту, після чого запис успішно зберігається в таблицю `post_comments` бази даних., що чітко відтворює порядок виконання цієї дії так само з емодзі-вподобайками. (рис.2.10.).

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

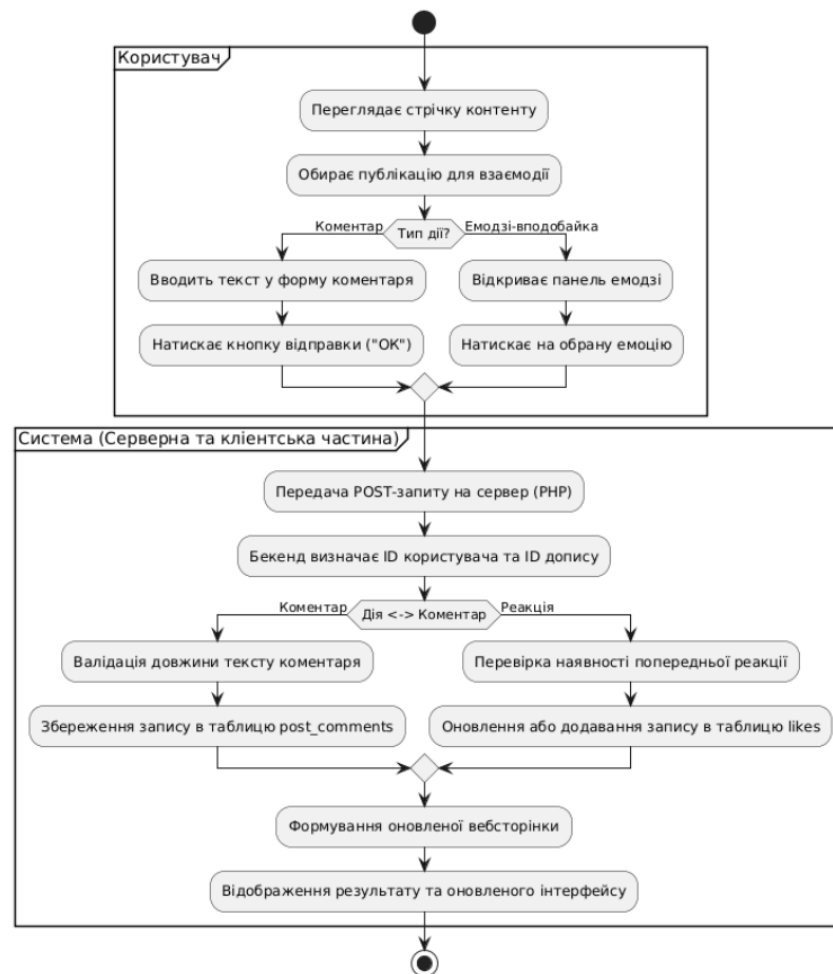


Рис.2.10. Алгоритм процесу коментування та реагування на публікацію

Логіка платформи передбачає альтернативні сценарії взаємодії залежно від дій користувача - чи то публікація повноцінного текстового коментаря, чи то швидке залишення емодзі-вподобайки під дописом. Оброблена серверними скриптами на PHP інформація передається через стандартні запити форми або посилання, забезпечуючи оновлення даних та їх збереження у базі даних MySQL. Своєю чергою, модуль приватних повідомлень відповідає за організацію прямого та безпечного листування між користувачами.

Згідно з цією схемою, процес надсилання приватного повідомлення розпочинається з того, що користувач переходить до розділу чатів, обирає співрозмовника, вводить текст чи додає медіаконтент та натискає кнопку “Надіслати”. На стороні системи формується асинхронний AJAX-запит, після чого контролер перевіряє статус авторизації та параметри. У разі активності та існування одержувача повідомлення записується до таблиці messages бази даних,

формується успішна відповідь і новий елемент миттєво відображається в активному діалозі, що наочно відображає алгоритм виконання цієї процедури (рис.2.11.).

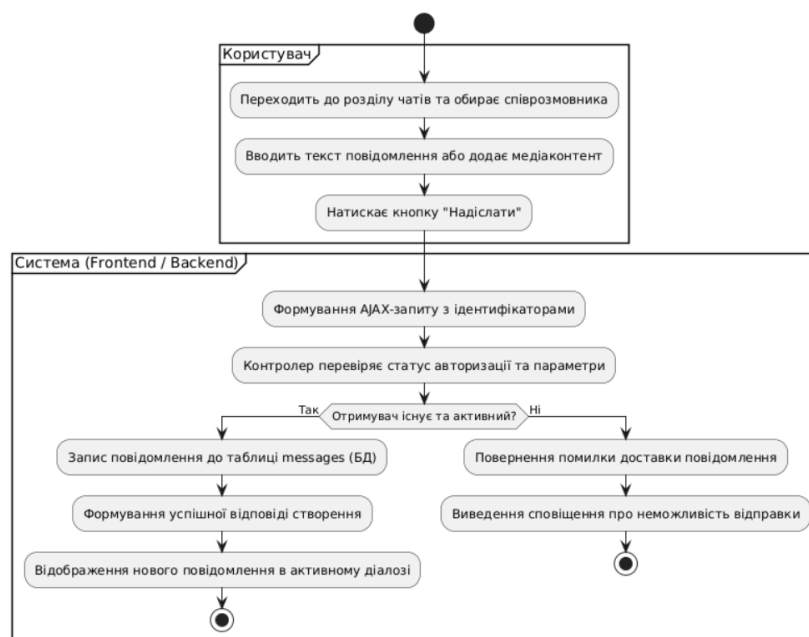


Рис.2.11. Алгоритм процесу надсилення приватного повідомлення

Процес побудований на перевірці надійності зв'язку та валідації ідентифікаторів користувачів відправника та отримувача. Після проходження перевірок на бекенді, дані фіксуються в таблиці повідомлень бази даних MySQL, після чого клієнтська частина автоматично розширює вікно діалогу новим елементом без потреби оновлювати сторінку браузера.

Контролер на бекенді здійснює перевірку статусу авторизації та вхідних параметрів, після чого аналізує чи існує отримувач і чи є він активним. Якщо перевірка дає негативний результат “Ні”, система формує помилку доставки повідомлення та виводить відповідне сповіщення про неможливість відправки.

У разі успішного проходження всіх перевірок “Так”, запис із повідомленням надійно фіксується в таблиці messages бази даних MySQL. Далі сервер формує успішну відповідь про створення нового елемента листування, а клієнтська частина автоматично розширює вікно активного діалогу без потреби повністю оновлювати сторінку браузера.

Висновки до другого розділу

У другому розділі здійснено комплексне проектування архітектури та поведінкової логіки вебплатформи Layttle, що охоплює весь спектр робіт від загальної концепції до деталізації алгоритмів внутрішніх компонентів.

Сформовано структуру взаємодії компонентів вебзастосунку на базі класичного шаблону MVC (Model-View-Controller) із розмежуванням відповідальності між клієнтським інтерфейсом та серверним середовищем виконання у середовищі Wamp64.

Визначено ключових суб'єктів взаємодії в межах соціальної мережі авторизований користувач, модератор, адміністратор та формалізовано їхні базові права доступу до функціональних модулів платформи.

Розроблено модель даних та організацію сховища на базі реляційної СУБД MySQL, де оптимізовано таблиці для зберігання користувацьких профілів, стрічки контенту, коментарів, емодзі-вподобайок та приватних повідомлень.

За допомогою діаграм послідовності та активності деталізовано хронологію і топологію передачі повідомлень для ключових процесів: публікації нових дописів, взаємодії зі стрічкою контенту, а також обміну особистими повідомленнями у чаті.

Спроектвані логічні, структурні та алгоритмічні моделі повністю відображають специфіку розробленої соціальної мережі, задовільняють початкові вимоги технічного завдання та створюють документаційну базу для переходу до етапу безпосередньої програмної реалізації проєкту.

Зокрема, детально описано алгоритми клієнтської та серверної валідації даних, процеси захищеної обробки завантажених медіафайлів із генерацією унікальних ідентифікаторів імен, а також механізми асинхронної взаємодії через AJAX-запити для забезпечення миттєвого оновлення інтерфейсу чатів і стрічки дописів. Застосування сучасних підходів до проектування архітектури дозволило мінімізувати навантаження на серверне середовище, оптимізувати виконання SQL-запитів до реляційної бази даних та попередити можливі збої.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЄКТУ

3.1. Програмна реалізація серверної частини та підсистеми безпеки

Проєкт реалізовано мовою програмування PHP із залученням локального серверного середовища Wamp64. В основу архітектури покладено класичний патерн Model-View-Controller (MVC), що дозволяє чітко розмежувати обробку бізнес-логіки, роботу з даними та представлення інтерфейсу. Програмний код структуровано у вигляді взаємопов'язаних модулів окремих контролерів для обробки запитів, моделей для взаємодії з базою даних та маршрутизаторів, що забезпечує гнучкість, надійність і зручність подальшого супроводу платформи.

Для комунікації із реляційною базою даних MySQL у проєкті залучено стандартні механізми підключення та виконання запитів на основі розширень PHP, зокрема PDO. Це дає можливість стабільно опрацьовувати паралельні запити від користувачів без надмірного навантаження на локальний сервер. Параметри підключення до бази визначаються через конфігураційні файли середовища, що дозволяє гнучко керувати доступом до даних під час локального розгортання платформи.

Задля збереження взаємної цілісності інформації залучено зовнішні ключі Foreign Keys з налаштованими правилами каскадного видалення. Зокрема, у разі видалення профілю учасника мережі архітектура передбачає автоматичне очищення пов'язаних із ним матеріалів публікацій, коментарів, емодзі-вподобайок та мультимедійних файлів. З метою прискорення пошуку та обробки великих масивів інформації задіяно індексацію таблиць, а найважливіші операції проводяться у режимі транзакцій MySQL, що унеможливорює виникнення конфліктів чи неповного запису даних при випадкових збоях.

Функціональні можливості вебплатформи присвячено питанням автентифікації, захисту даних [22] PHP та контролю доступу. Зокрема, процедура створення нового облікового запису супроводжується обов'язковою перевіркою введених параметрів, перевіркою складності пароля через регулярні вирази та його хешуванням з використанням алгоритмів на кшталт bcrypt.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Для захисту приватних розділів та управління правами користувачів на платформі задіяно механізми перевірки сесій і токенів доступу. Перед тим як надати доступ до захищених маршрутів, сервер здійснює послідовну перевірку запиту. Спочатку виконується дешифрування та валідація підпису ідентифікатора, після чого система звертається до бази даних для перевірки поточного стану облікового запису. За рахунок фіксації версії активного сеансу в базі даних, у разі зміни користувацьких даних чи виходу з акаунту відбувається оновлення цього ключа, що призводить до миттєвого відкликання попередніх дозволів.

Оскільки функціонал соціальної мережі орієнтований на публікацію авторського контенту, взаємодію стрічки та персоналізацію профілів, особливу увагу приділено безпечній роботі з медіафайлами. Будь-які завантажені користувачем зображення проходять попередню перевірку на сервері за допомогою вбудованих інструментів PHP [23]: перевіряється формат файлу, його допустимі розміри та розширення для запобігання завантаженню виконуваних скриптів. Успішно опрацьовані файли розміщуються у відповідних каталогах сховища, тоді як у базі даних фіксуються виключно безпечні посилання на них.

Надійність виконання серверних операцій підтримується завдяки впровадженню єдиного підходу до перехоплення виняткових ситуацій. Усі потенційно критичні ділянки коду в контролерах обгорнуті в блоки виявлення помилок try-catch, що виключає ймовірність аварійного падіння бекенду при неправильних запитах або збоях зв'язку з базою даних. У разі виникнення помилки формується акуратна відповідь у форматі JSON із відповідним HTTP-статусом для клієнтської частини, що дозволяє інтерфейсу коректно відобразити попередження користувачеві без розкриття службових деталей серверного середовища.

Застосування вищезазначених архітектурних і безпекових рішень дозволило створити стабільну, захищену та продуктивну серверну основу, яка повністю задовольняє бізнес-логіку проєкту та забезпечує цілісність користувацьких даних.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

Для забезпечення централізованої обробки всіх вхідних запитів на серверній стороні реалізовано паттерн єдиної точки входу `index.php`. Завдяки цьому будь-який HTTP-запит від клієнтської частини перенаправляється через правила модуля `mod_rewrite` вебсервера Apache в єдиний диспетчер маршрутів.

Диспетчер здійснює розбір URI-рядка, визначає цільовий контролер та метод, а також передає у масиві параметрів дані з глобальних змінних `$_GET`, `$_POST`, `$_FILES`. Такий підхід дозволяє реалізувати чисті та зрозумілі для людського ока URL-адреси, щоб приховати реальну структуру файлової системи сервера від потенційних злоумисників та спростити логування подій на рівні додатку.

Оскільки вебплатформа опрацьовує користувацький контент та взаємодіє з автентифікованими сесіями, серверний код містить низку вбудованих захисних фільтрів. Завдяки повній відмові від динамічного конструювання SQL-запитів шляхом простої взаємодії рядків і переходу на використання підготовлених виразів Prepared Statements в середовищі PDO [24], будь-які введені користувачем дані автоматично відображаються на рівні драйвера бази даних до моменту виконання запиту. Усі текстові поля, що вводяться користувачами такі як, коментарі, дописи, оновлення профілів тощо, проходять фільтрацію за допомогою спеціалізованих функцій та відображення спеціальних символів `htmlspecialchars` перед виведенням назад у браузер. Для критичних POST-запитів, як от зміна пароля, редагування профілю, видалення контенту на бекенді генеруються унікальні токени сесії, валідація яких унеможливорює несанкціоноване виконання команд зі сторонніх шкідливих ресурсів.

Застосування вищезазначених архітектурних і безпекових рішень дозволило створити стабільну, захищену та продуктивну серверну основу, яка повністю задовольняє бізнес-логіку проєкту та забезпечує цілісність користувацьких даних.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

3.2. Проектування та реалізація клієнтського інтерфейсу

Клієнтську частину вебплатформи Layttle реалізовано на основі класичного стека вебтехнологій - HTML5, CSS3 [25] та JavaScript, які виконуються безпосередньо у браузері користувача. Інтерфейс формується за допомогою серверних скриптів PHP, що забезпечує динамічну генерацію сторінок та гнучку взаємодію з реляційною базою даних.

Архітектура фронтенд-складника організована у вигляді структурованих шаблонів і логічних модулів, що охоплюють систему навігації, розділ авторизації та реєстрації, інтерфейс персонального профілю, стрічку контенту, а також спеціалізовані форми для публікації дописів і надсилання приватних повідомлень. Для керування станом користувацької сесії та передачі параметрів між сторінками задіяно вбудовані механізми сесій PHP, що дозволяє уникати зайвого навантаження на клієнтську частину та гарантує швидкий відгук інтерфейсу під час навігації по платформі.

Візуальне оформлення ресурсів вебплатформи реалізовано за допомогою сучасних стандартів каскадних таблиць стилів із підтримкою адаптивної верстки. Гармонійне поєднання базових класів komponування разом із динамічними змінними дозволило сформувати єдиний стилістичний простір для соціальної мережі. Окрім цього, серверна шаблонізація гарантує надійне відокремлення загальних елементів інтерфейсу від персонального контенту, який генерують учасники всередині власних акаунтів.

З метою підвищення надійності під час заповнення форм, створення дописів чи надсилання повідомлень задіяно клієнтські сценарії на JavaScript, що відповідають за попередню обробку та збереження введеної інформації. Інтерфейс акумулює користувацькі зміни у полях введення та організовано передає їх на бекенд через захищені POST-запити, мінімізуючи ризики втрати даних та запобігаючи зайвому навантаженню під час активного листування чи завантаження важких медіафайлів.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

Упроваджені архітектурні підходи дозволили створити швидкий, стабільний та зручний у користуванні інтерфейс соціальної мережі, який тісно інтегрований із серверним середовищем та повністю задовільняє потреби учасників у комунікації без використання зайвих сторонніх фреймворків.

Для забезпечення чистоти коду, простоти підтримки та уникнення дублювання HTML-розмітки у проєкті реалізовано концепцію модульних шаблонів. Замість монолітних файлів сторінок, інтерфейс платформи розбито на логічні компоненти, які динамічно підключаються серверною логікою.

Хедер, бічне меню швидкого доступу та футер винесені в окремі шаблонні файли, що дозволяє змінювати загальну структуру навігації в єдиному місці без необхідності редагування кожної сторінки окремо. Відображення дописів побудовано на основі універсального шаблону картки, який приймає масив даних із бази та заповнює відповідні теги. Це гарантує єдиний візуальний стиль для текстових публікацій, медіафайлів та блоків взаємодії з контентом.

Елементи взаємодії з підвищеною динамікою форми редагування профілю, підтвердження видалення дописів, надсилання повідомлень реалізовані у вигляді прихованих контейнерів, які активуються за допомогою клієнтських скриптів при виникненні відповідних подій.

Хоча бекенд платформи спирається на класичну серверно-орієнтовану генерацію сторінок, для ключових елементів інтерактивності задіяно сучасні можливості JavaScript та роботу з асинхронними запитами Fetch API та AJAX.

Користувачі можуть ставити вподобайки, залишати коментарі або підписуватися на інших користувачів без повного перезавантаження сторінки. Клієнтський скрипт перехоплює подію кліку, формує асинхронний запит до відповідного PHP-контролера, отримує JSON-відповідь і завантажує потрібний інтерфейс.

Перед відправкою даних на сервер, JavaScript виконує перевірку введеної інформації: наявність обов'язкових полів, відповідність довжини тексту бази даних, перевірку розширень завантажуваних файлів.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

Окрему увагу в процесі розробки клієнтської частини було приділено візуальній реалізації системи реакцій та інтерактивних панелей. Зокрема, під кожним постом розроблено випадаючу панель швидкого вибору емодзі, яка з'являється за допомогою легких CSS-переходів [26] та обробки подій миші на JavaScript. Це дозволяє користувачеві миттєво обирати бажану емоційну реакцію, від радощів та захоплень, до суму та обурення. Після чого лічильник вподобань оновлюється в реальному часі без перезавантаження всієї сторінки.

Перегляд фотографій у постах підтримує індикатор каруселі, що реалізовано за допомогою масиву зображень та локального стану клієнтського скрипта, який перемикає активні індекси картинок без звернення до сервера.

Система пошуку користувачів та друзів працює на основі інтерактивних списків із кнопками швидкого керування зв'язками, додавання або видалення за допомогою мініатюрних іконок із підтвердженням дії. Усі форми від створення нового допису у блоці “Що у вас нового, друже?” до редагування персональних контактних даних у профілі проходять попередню валідацію на стороні клієнта. Це виключає надсилання порожніх або не правильно оформлених запитів, знижуючи навантаження на PHP-сервер і підвищуючи загальну плавність роботи вебресурсу.

Завдяки поєднанню класичної серверної логіки PHP та застосуванню асинхронних скриптів JavaScript, вебплатформа Layttle демонструє високу швидкість завантаження, мінімальне споживання системних ресурсів та забезпечує приємний, чуйний користувацький досвід у стилімізованому космічному-інтерфейсі.

Застосування локального кешування для статичних елементів дизайну та мінімізація важких скриптів дозволили досягти високих показників швидкодії та стабільності роботи вебзастосунку в умовах інтенсивного користувацького трафіку.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

3.3. Налаштування середовища, розгортання та приклади інтерфейсу

Функціонал взаємодії з користувачем розпочинається з модуля автентифікації та реєстрації, який реалізовано у вигляді зручних форм для введення облікових даних. Інтерфейс підтримує класичний вхід за допомогою електронної пошти та пароля, а також забезпечує перевірку активної сесії користувача. Під час створення нового акаунту реалізовано можливість попереднього перегляду та завантаження фотографії профілю безпосередньо через елементи форми.

Безпека та коректність обробки даних на клієнтській і серверній сторонах охоплюють низку захисних механізмів. Під час реєстрації чи зміни пароля впроваджено логіку валідації складності введеної комбінації символів, що виключає використання надто простих чи вразливих паролів. Процедури підтвердження дій супроводжуються генерацією системних сповіщень, які інформують учасника про успішне виконання операцій або вказують на конкретні помилки при заповненні полів. Наочний вигляд головного екрана входу та авторизації на платформі представлено на (рис.3.1.)



Рис.3.1. Інтерфейс форми авторизації користувачів

Для забезпечення зручного переходу між сеансами роботи та надання можливості новим користувачам приєднатися до вебплатформи, інтерфейс містить інтерактивне посилання на створення нового облікового запису.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Клієнтська частина динамічно перенаправляє користувача до розширеної форми реєстрації, яка містить поля для введення логіна, імені, прізвища, а також перевірку введеного пароля. Зовнішній вигляд цієї форми представлено на (рис.3.2.).

Рис.3.2. Інтерфейс форми реєстрації користувачів

Після успішної автентифікації користувач потрапляє на головну сторінку вебплатформи, яка є центром для комунікації. Верхня частина інтерфейсу містить фіксовану панель навігації: фірмовий логотип платформи, рядок пошуку для швидкого знаходження користувачів чи друзів, а також блок швидкого доступу до персонального профілю, чатів або друзів.

Основну робочу область сторінки займає стрічка друзів, де в реальному часі відображаються публікації виключно тих учасників, на яких підписаний користувач. Для зручної навігації передбачено можливість вибирати контент за певний період часу. Вона дозволяє миттєво сортувати дописи за періодом створення: обирати “Усі пости”, а також фільтрувати публікації за категоріями “Сьогодні”, “Місяць”, “3 місяці”, “Рік” або “Більше року”. Наочний вигляд головного екрана стрічки контенту та фільтрів вибору категорій за часом на вебплатформі представлено на (рис.3.3.).

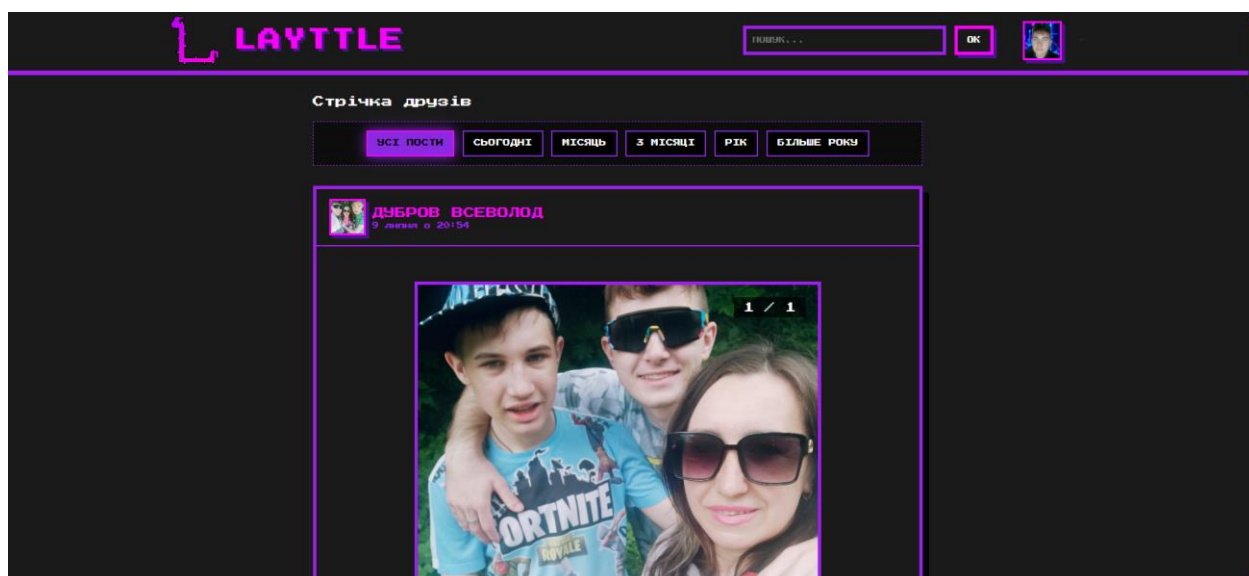


Рис.3.3. Головний екран стрічки публікацій

Переглядаючи контент у профілях друзів, користувачі можуть взаємодіяти з публікаціями за допомогою розширеної системи емодзі-вподобайок, залишати коментарі, а також редагувати чи видаляти власні дописи та повідомлення. Детальний інтерфейс взаємодії з публікацією друга, панель емодзі-реакцій та блок коментарів наведено на (рис.3.4.).

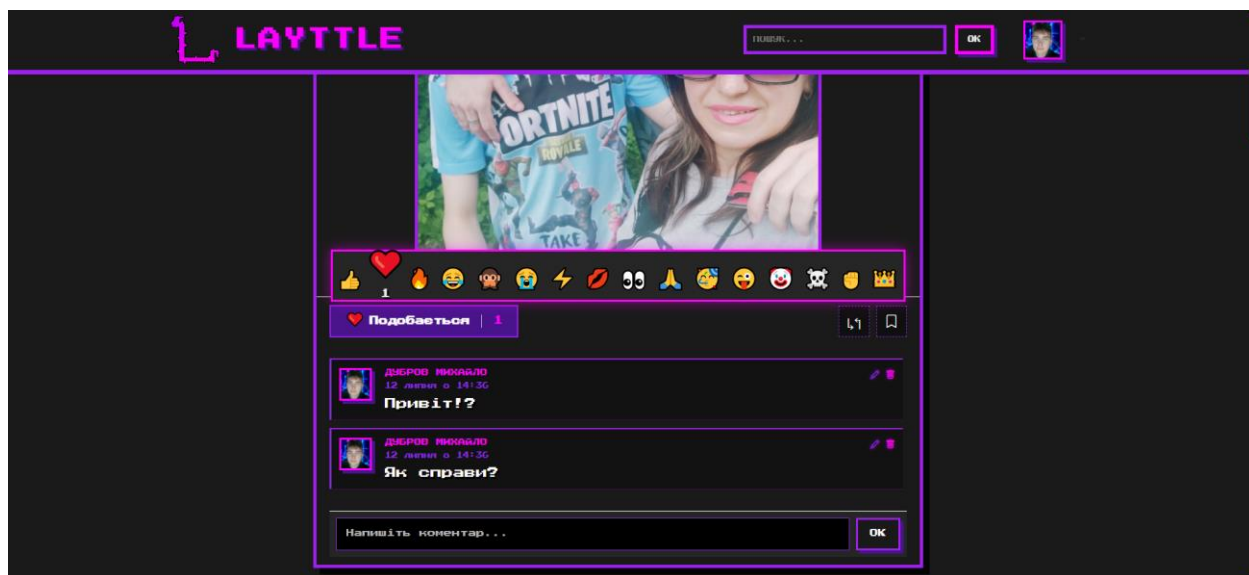


Рис.3.4. Взаємодія з публікацією друга

Архітектура пошукового механізму вебплатформи базується на комбінованому принципі фільтрації даних, що охоплює як глобальний пошук користувачів системи, так і список друзів конкретного профілю. Пошук бажаних користувачів відбувається по прізвищу, імені або логіну користувача

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

забезпечуючи високу швидкість обробки інформації на рівні реляційної бази даних через захищені PDO-вирази.

Глобальний пошук розгорнутий прямо в хедері сайту для максимальної зручності користувачів. При введенні пошукового запиту при натисканні кнопки “OK”, система формує запит через роутер до контролера, який одразу перевіряє базу за трьома полями пошуку: по прізвищу, імені та логіну і виводить користувачів, що збігається з пошуковим запитом. Наочний вигляд реалізації глобального пошуку користувачів на вебплатформі представлено на (рис.3.5.)

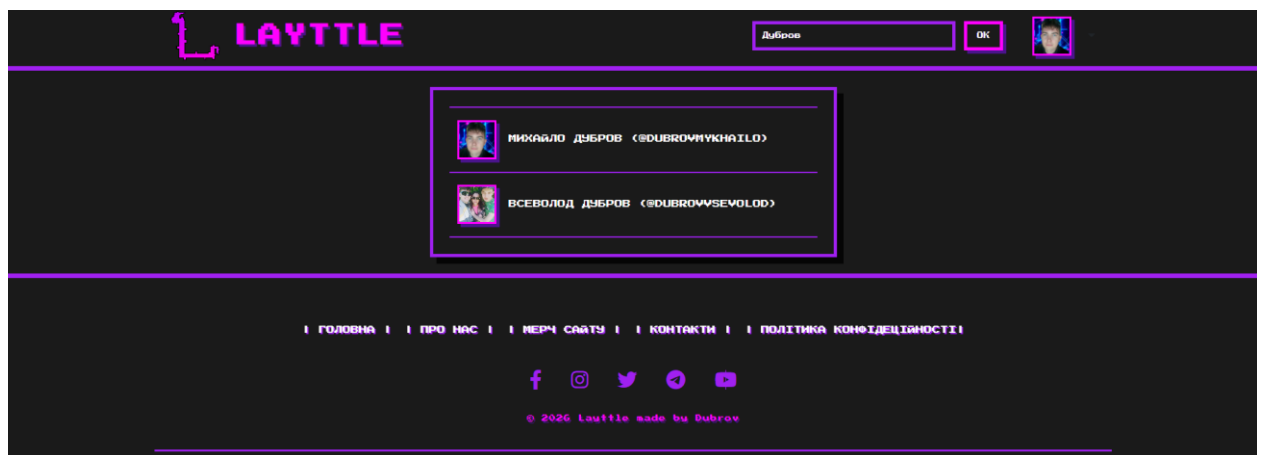


Рис.3.5. Інтерфейс глобального пошуку користувачів

Водночас для списку друзів зроблена окрема логіка. Цей пошук працює всередині конкретного профілю і прив'язаний до таблиці зв'язків. Тобто система спершу дивиться на статус дружби, щоб це були реально підтверджені друзі, а вже потім фільтрує їх за введеними даними. При цьому весь інтерфейс створений через єдиний гнучкий шаблон, в якому є: аватарка користувача [27], ім'я, прізвище, логін, а також кнопка для миттєвого видалення з друзів.

Такий підхід забезпечує високу точність вибору та не допускає появу не друзів у списку. Завдяки оптимізованим SQL-запитам до бази даних та використанню асинхронних технологій, обробка запитів на видалення чи пошук відбувається миттєво. Це дозволяє користувачеві комфортно керувати власним соціальним колом без затримок у роботі інтерфейсу та зайвого навантаження на систему. Зовнішній вигляд інтерфейсу пошуку та керування списком друзів наведено на (рис.3.6.).

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

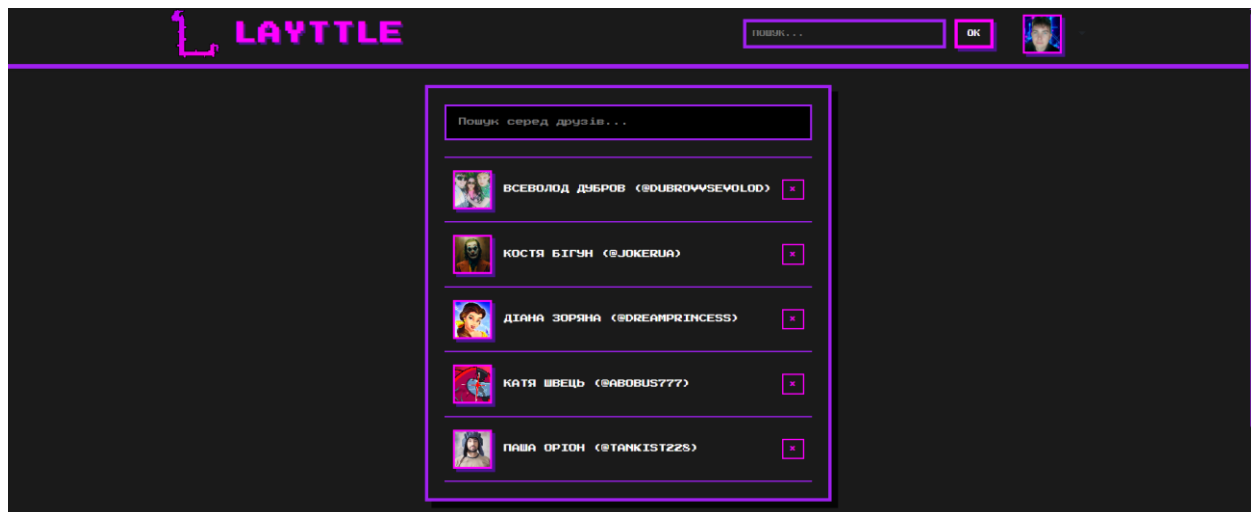


Рис.3.6. Пошук та керування списком друзів

Інтерфейс персонального профілю та стрічки користувача побудований за принципом блокової структури. Робоча область сторінки профільних даних поділена на функціональні блоки, які забезпечують зручний доступ до особистої інформації, медіафайлів та налаштувань. Загальний вигляд верхньої частини сторінки профілю користувача представлено на (рис.3.7.)

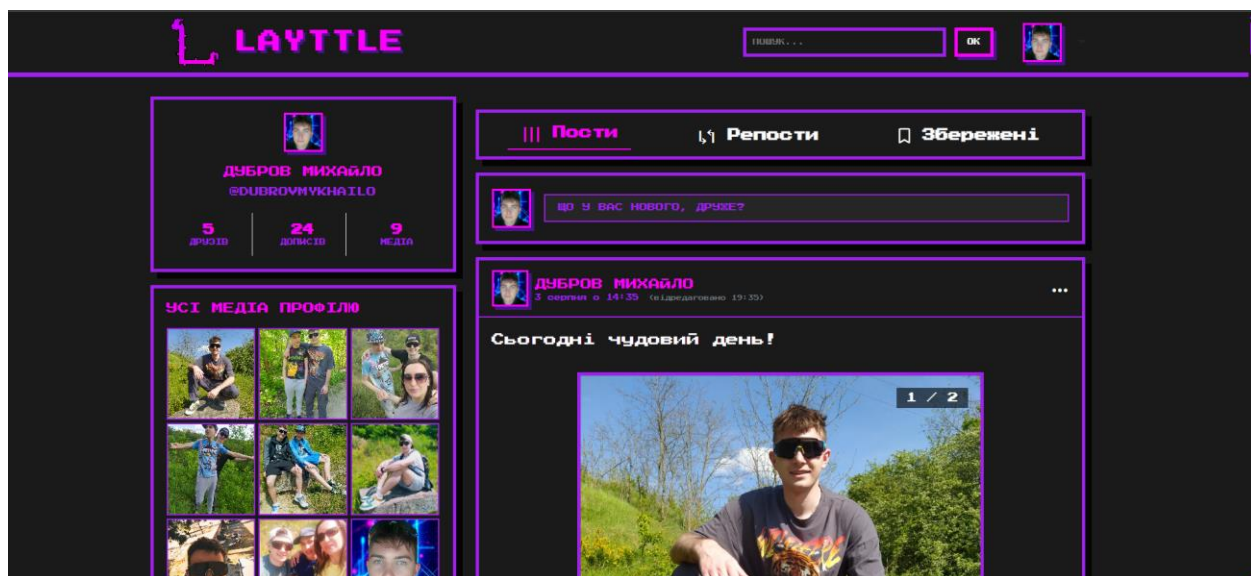


Рис.3.7. Особистий профіль користувача

Головна сторінка профілю містить інформаційний блок акаунту із кількістю друзів, опублікованих дописів і завантажених медіафайлів. Окремою секцією винесено галерею “Усі медіа профілю”, яка забезпечує швидкий перегляд фотографій та мультимедійного контенту, завантаженого

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

користувачем. Продовження структури особистого профілю з відображенням медіагалереї та контактних даних наведено на (рис.3.8.).

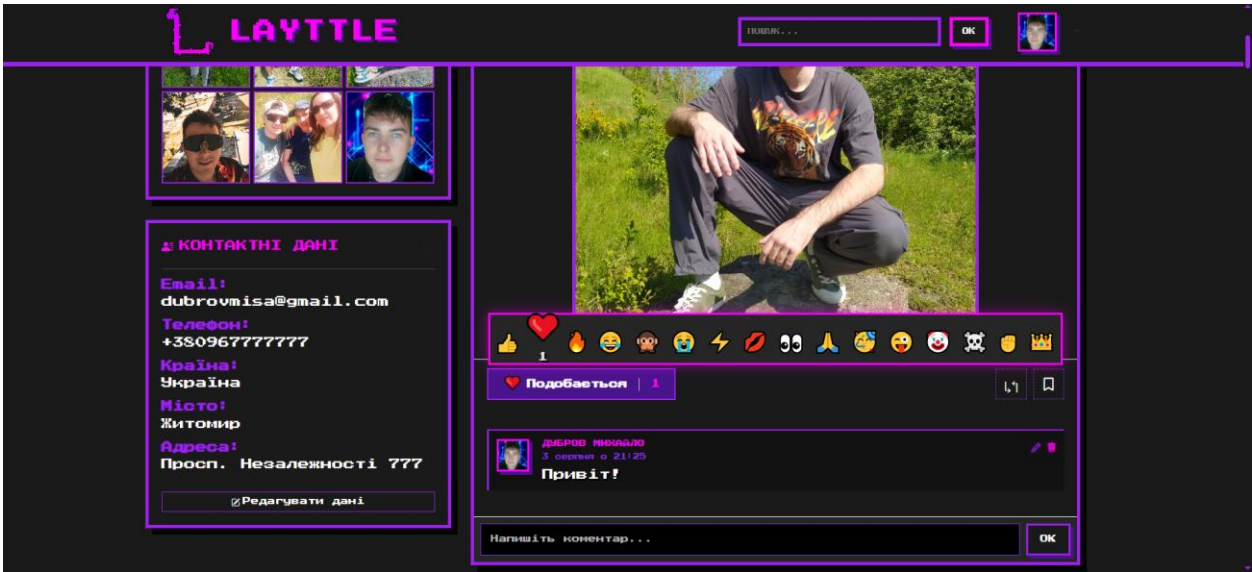


Рис.3.8. Особистий профіль користувача продовження

У центральній частині сторінки розміщено інтерактивну стрічку з вкладками “Пости”, “Репости” та “Збережені”. Блок створення публікації “Що у вас нового, друже?” дозволяє учасникам миттєво ділитися текстовим контентом, фотографіями чи відеоматеріалами. Кожен пост супроводжується системою реакцій емої-вподобайок, блоком коментарів із можливістю редагування чи видалення дописів, а також додатковими інструментами взаємодії. Деталізацію розділів з репостами та збереженими дописами відображено на (рис.3.9.).

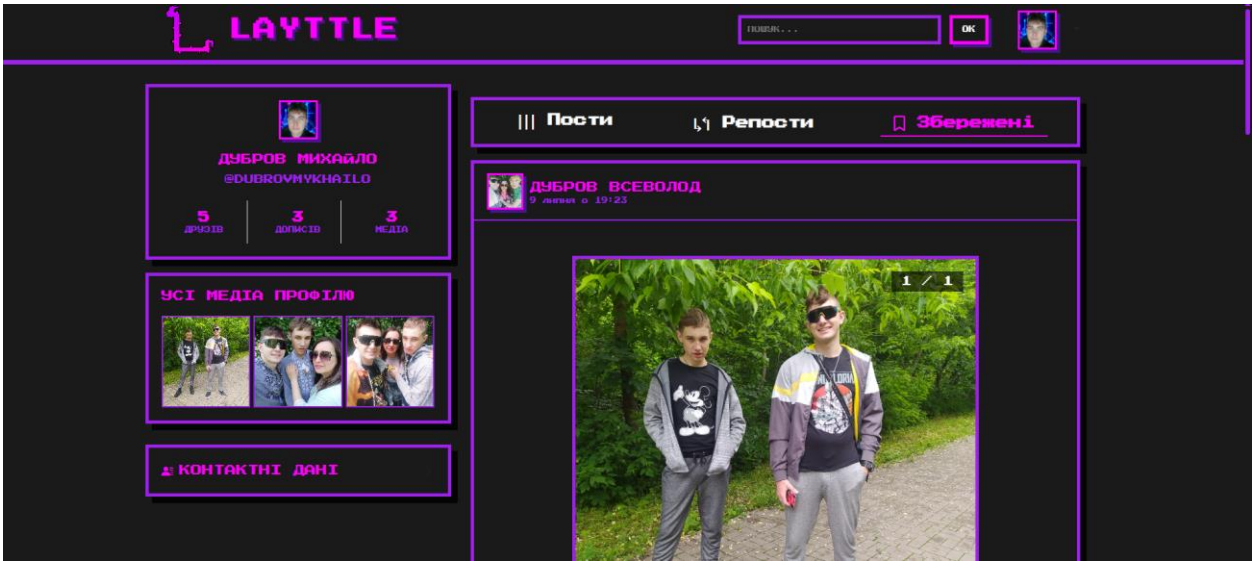


Рис.3.9. Відображення репостів та збережених постів в особистому профілі

Для власних профілів передбачено зручний модуль редагування контактних даних таких як: Email, телефон, країна, місто, адреса, а при перегляді сторінок інших користувачів інтерфейс динамічно адаптується, пропонуючи кнопки “Додати в друзі” та “Написати повідомлення”. Адаптивний вигляд інтерфейсу при перегляді сторінки іншого користувача представлено на (рис.3.10.).

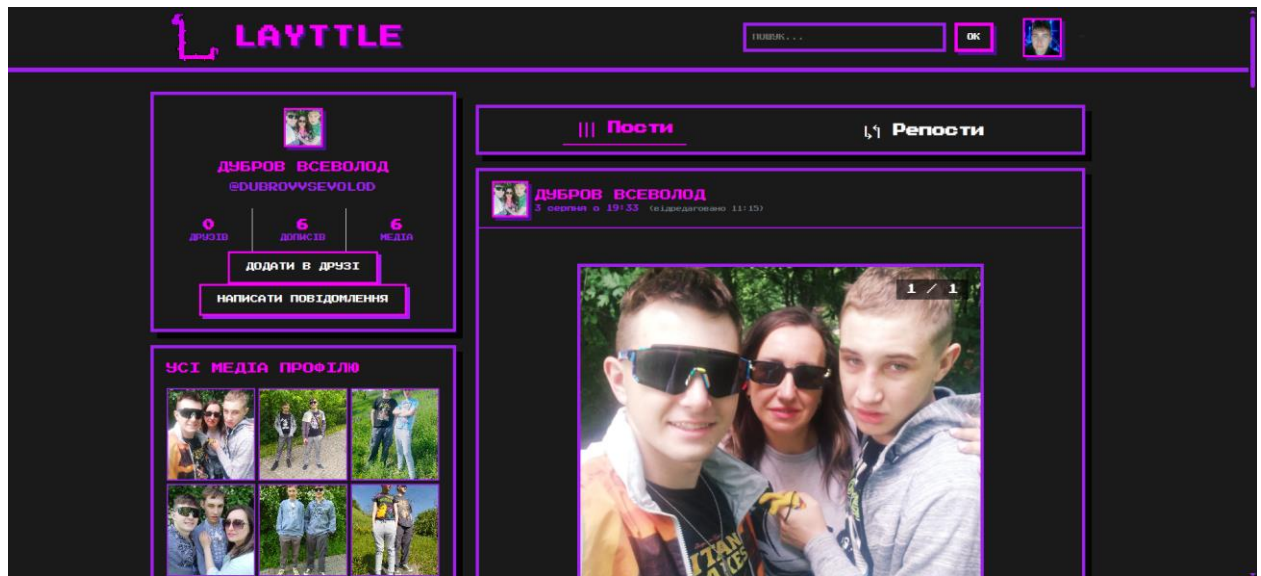


Рис.3.10. Інтерфейс перегляду іншого профілю з можливістю комунікації

Інтерфейс обміну повідомленнями реалізовано як багатокомпонентну динамічну систему, що забезпечує двосторонню комунікацію в реальному часі. Архітектура модуля поєднує асинхронні AJAX-запити для передачі даних без перезавантаження сторінки, групування повідомлень за часом. Система підтримує введення текстових блоків із можливістю автоматичного форматування та вставлення емодзі, а також завантаження мультимедійних файлів зображень та відео. Медіафайли проходять валідацію на стороні сервера, після чого автоматично адаптуються під сітку повідомлень із підтримкою перегляду у повному екрані. Загальний вигляд інтерфейсу чату, календарне групування повідомлень та фірмовий стиль оформлення блоків представлено на (рис.3.11.).



Рис.3.11. Загальний вигляд чату

Інтеграція бібліотеки Емої-март налаштована через кастомний механізм Shadow DOM [28], що дозволяє повністю стилізувати палітру піктограм під загальну концепцію платформи. Реалізовано логіку редагування тексту та заміни медіафайлів із фіксацією часу зміни. Система динамічно виводить мітки часу створення повідомлення та статус «змінено» у разі внесення коригувань. Наочну демонстрацію інтегрованої палітри емодзі безпосередньо в інтерфейсі обміну повідомленнями наведено на (рис.3.12.).

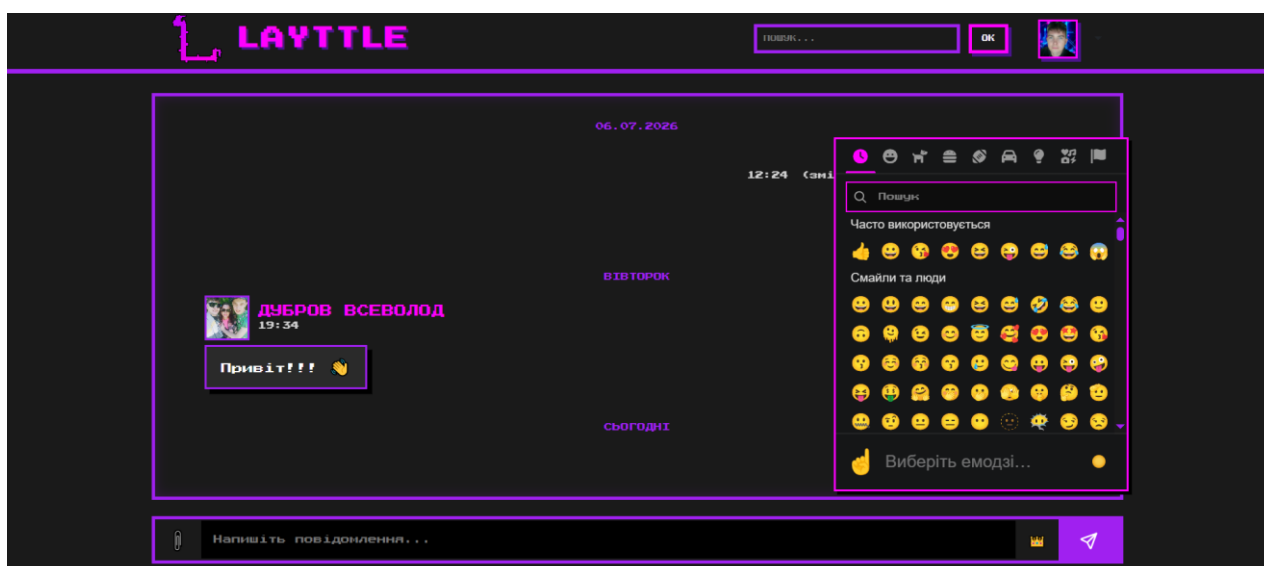


Рис.3.12. Демонстрація можливості використання емодзі в чаті

Повідомлення групуються за календарним принципом “СЬОГОДНІ”, “ВІВТОРОК” або конкретна дата. Кожне повідомлення містить аватар

користувача, повне ім'я та прізвище, виведені у фірмовому стилі вебплатформи. Приклад адаптивного виведення завантажених користувачем зображень та медіаконтенту всередині повідомлень чату зображено на (рис.3.13.).

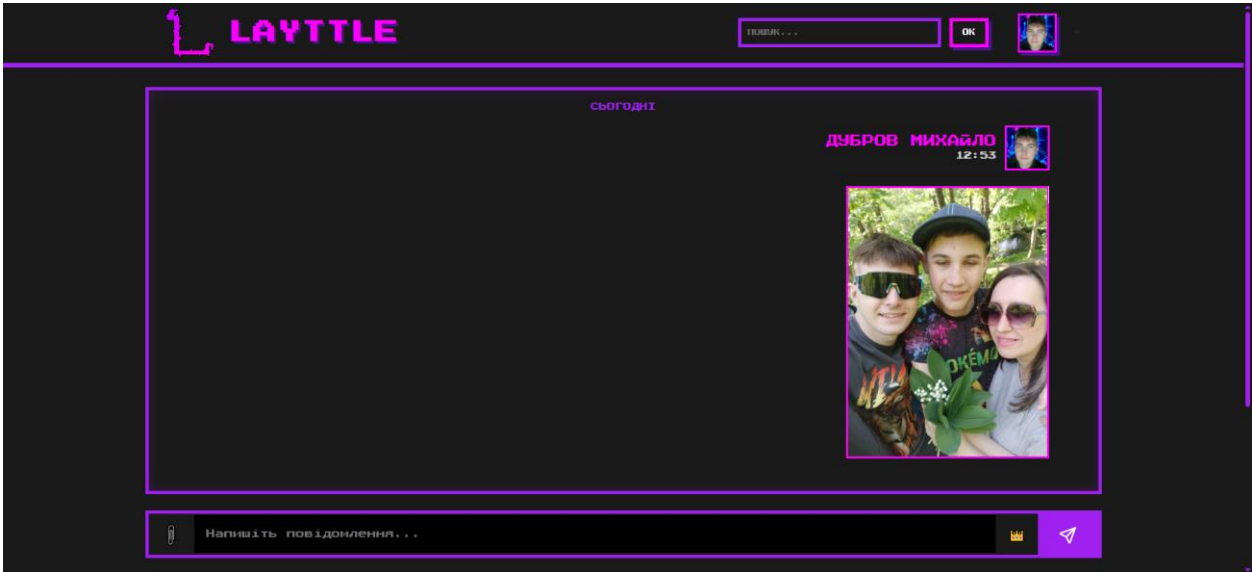


Рис.3.13. Відображення мультимедійного контенту

Узагальнюючи описані технічні рішення, розгорнута архітектура та клієнтські інтерфейси формують повноцінну програмну екосистему для комунікації та роботи з контентом. Завдяки використанню захищених PDO-запитів до реляційної бази даних та асинхронних механізмів забезпечено надійність зберігання інформації та високу швидкість пошуку й обміну повідомленнями. Програмне забезпечення [29] спроектоване з розрахунком на майбутнє масштабування: планується інтеграція адміністративного блоку, який включатиме панель управління, систему скарг та інструменти фільтрації недопустимого контенту. Модульна структура вебплатформи дає змогу безперешкодно впроваджувати нові функціональні модулі без порушення цілісності вже наявної архітектури.

3.4. Тестування програмної системи

Для перевірки працездатності розробленої вебплатформи, коректності взаємодії клієнтської частини з сервером та забезпечення базового рівня безпеки було проведено тестування основних функціональних модулів.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Оскільки ключовим елементом безпеки є захист облікових записів користувачів, особливу увагу було приділено тестуванню системи реєстрації та авторизації. Перевірено коректність збереження паролів у базі даних MySQL. Вхідні паролі користувачів не зберігаються у відкритому вигляді, а проходять криптографічне перетворення за допомогою вбудованих функцій PHP[30] `password_hash` та `password_verify`, що виключає можливість взлому при потенційному витoku даних із БД. Протестовано роботу глобальних масивів сесій (`$_SESSION`) рhрдля утримання активного стану користувача після успішного входу та коректність їх знищення під час виходу з системи.

Тестування інтерфейсу та серверної логіки виконувалося локально в середовищі WAMP64. Було протестовано роботу форм реєстрації та логіну на наявність порожніх полів або некоректних даних із виведенням відповідних сповіщень для користувача. Перевірено виконання SQL-запитів через захищені PDO-з'єднання. Протестовано додавання нових записів: постів, коментарів, додавання в друзі та коректність їх відображення у стрічці й профілі. Перевірено працездатність пошуку користувачів за прізвищем, ім'ям або логіном, а також сортування публікацій за часовими періодами. Протестовано надсилання та оновлення повідомлень у чаті в реальному часі, а також коректність завантаження медіафайлів.

Усі проведені перевірки підтвердили стабільну роботу вебплатформи, коректність обробки запитів на сервері та надійність захисту користувацьких даних за допомогою хешування паролей.

Для досягнення максимальної об'єктивності результатів тестування розробленого вебдодатка було застосовано комплексний підхід, що охоплює модульне, інтеграційне та системне тестування. Як основне робоче середовище використовувався локальний сервер WAMP64, що містить інтерпретатор мови PHP, реляційну систему керування базами даних MySQL для збереження структурованих даних користувачів, постів, зв'язків та повідомлень, вебсервер Apache із налаштованим модулем `mod_rewrite` для забезпечення чистого роутингу запитів MVC-архітектури

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Окрім базової перевірки хешування паролів, під час експериментальних досліджень було здійснено тестування захищеності вебплатформи від поширених вебвразливостей. Усі запити до бази даних реалізовано через захищені PDO-з'єднання з використанням підготовлених виразів Prepared Statements. Тестування шляхом введення спеціальних SQL-конструкцій у поля форм авторизації, пошуку та чату підтвердило повну неможливість маніпуляції базою даних з боку користувача. Проведено тестове введення HTML-тегів та JavaScript-сценаріїв у поля створення постів, коментарів та повідомлень чату.

Перевірено поведінку системи при спробі прямого переходу за захищеними URL-адресами сторінки профілю, налаштувань, чатів без попередньої авторизації. Система демонструє коректне перенаправлення на форму входу, що гарантує конфіденційність даних.

У рамках тестування клієнтської частини та взаємодії у реальному часі було оцінено: відсутність зайвого перезавантаження сторінок при надсиланні повідомлень і лайків, час відгуку інтерфейсу склав в межах 200-300 мс у локальному середовищі. Протестовано завантаження зображень різних форматів JPEG, PNG, WEBP та інші відео-формати.

Серверна валідація успішно відсіювала файли, що перевищували ліміти або не відповідали дозволеним типам, а алгоритми масштабування забезпечили коректне відображення прев'ю в галереях та стрічках без порушення верстки.

Окремим етапом експериментальних досліджень стала перевірка кросбраузерності та адаптивності користувацького інтерфейсу. Тестування відображення вебплатформи проводилося у найпопулярніших сучасних вебпереглядачах, зокрема Google Chrome, Mozilla Firefox, Safari та Microsoft Edge. Усі стилістичні оформлення, анімації та інтерактивні елементи, включаючи кастомні модальні вікна та палітру емодзі, відображалися коректно незалежно від обраного браузера.

Детальному тестуванню було піддано складну бізнес-логіку взаємодії між користувачами. Зокрема, проведено перевірку повного життєвого циклу публікації: від її створення з прикріпленими медіафайлами до редагування та

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

остаточного видалення з бази даних і файлового сервера. Успішно перевірено механізм репостів та збереження публікацій у вкладку “Збережені”, де система коректно генерує зв'язки в таблицях без дублювання самого контенту, що значно економить дисковий простір сервера. У модулі управління друзями протестовано випадки: додавання в друзі, а також миттєве видалення користувача зі списку. Всі вказані операції виконуються без збоїв із паралельним оновленням лічильників в інформаційному блоці профілю.

Загалом, результати комплексного тестування підтверджують, що розроблена соціальна вебплатформа повністю відповідає висунутим технічним вимогам. Архітектура бази даних витримує тестові навантаження, клієнтська частина забезпечує високий рівень інтерактивності завдяки асинхронній передачі даних, а реалізовані механізми безпеки надійно захищають систему від несанкціонованого доступу.

Окремим етапом тестування стала перевірка продуктивності та стабільності роботи вебплатформи в умовах паралельного виконання запитів кількома віртуальними користувачами. За допомогою вбудованих інструментів розробника PhpStorm та моніторингу навантаження вебсервера Apache у середовищі WAMP64 було перевірено поведінку системи при одночасному надсиланні запитів до баз даних із різних сесій. Використання оптимізованих індексів у таблицях зв'язків дозволило уникнути блокування таблиць table locking та забезпечити мінімальний час генерації сторінок.

Також було проведено стрес-тестування механізму завантаження медіаконтенту шляхом масового надсилання графічних файлів максимального допустимого розміру. Система продемонструвала високу стійкість: скрипти обробки зображень коректно виділяли необхідний обсяг оперативної пам'яті у файлі конфігурації php.ini, виконували генерацію мініатюр thumbnails без зависання потоку та своєчасно очищували тимчасові буфери.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

Висновки до третього розділу

У третьому розділі було успішно реалізовано практичну частину проєкту створено повноцінну та функціональну соціальну вебплатформу “Layttle”, а також виконано її тестування в локальному середовищі.

На базі мови PHP та локального середовища Wamp64 побудовано стійку архітектуру за патерном MVC. Особливу увагу зосереджено на захисті даних: впроваджено валідацію вхідних параметрів, надійне криптографічне хешування паролів за допомогою алгоритмів password_hash, а також контроль активних сесій (\$_SESSION) для запобігання несанкціонованому доступу. Завантажені користувачами медіафайли проходять обов'язкову серверну перевірку формату й розміру.

Інтерфейс охоплює всі необхідні модулі для повноцінної комунікації: форми авторизації, стрічки публікацій із можливістю сортування за часовими періодами, глобальний пошук користувачів та розширену систему приватних чатів із підтримкою емодзі та мультимедіа. Застосування асинхронних запитів дозволило забезпечити миттєвий відгук інтерфейсу під час взаємодії з дописами та повідомленнями.

Для перевірки працездатності вебплатформи було проведено комплексне функціональне тестування. Вони охопили перевірку роботи форм, коректність виконання SQL-запитів, стабільність роботи пошукових алгоритмів, а також безпеку обробки сесій і захисту паролів. Усі проведені перевірки підтвердили стабільність, відмовостійкість та повну готовність розробленої платформи до практичної експлуатації.

Окрему увагу під час тестування було приділено перевірці коректності взаємодії клієнтської та серверної частин, а також оцінці швидкодії системи під час паралельної обробки запитів від кількох віртуальних користувачів. Проведені експерименти продемонстрували відсутність критичних помилок, високу швидкість завантаження сторінок та коректне відображення елементів інтерфейсу.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було успішно спроектовано, розроблено та локально розгорнуто повноцінну соціальну вебплатформу “Layttle”. Практична реалізація охопила повний цикл створення сучасного вебзасобу для комунікації, управління персональними профілями, публікаціями та обміну повідомленнями у реальному часі.

На базі мови PHP та середовища Wamp64 реалізовано надійну серверність за патерном MVC. Завдяки взаємодії з реляційною базою даних MySQL через захищені PDO-з’єднання, досягнуто високої цілісності й безпеки збережених даних. Здійснено глибоку інтеграцію механізмів автентифікації, перевірки складних паролів за допомогою криптографічних алгоритмів password_hash, контролю активних сесій (\$_SESSION) та обов'язкової серверної валідації завантажених користувачами медіафайлів.

Розроблено повноцінний особистий профіль користувача, який містить контактні дані, інтерактивну галерею з фото- та відеоконтентом. Профіль динамічно відображає публікації поточного користувача, його репости та збережені пости.

Реалізовано стрічку контенту із можливістю гнучкого сортування за часовими періодами (сьогодні, місяць, 3 місяці, рік тощо). Створено повноцінне вікно створення та редагування постів із підтримкою мультимедіа. Користувачі можуть залишати коментарі з функціями їх редагування та видалення, фіксовано точну дату й час виставлення або редагування публікацій і коментарів, а також є можливість ставити емодзі-вподобайки.

Впроваджено функціонал підписок на інші профілі з можливістю додавання в друзі. Реалізовано також дві незалежні пошукові системи: глобальний пошук користувачів за прізвищем, ім'ям та логіном, а також локальний пошук безпосередньо серед власних друзів.

Створено розширений модуль обміну приватними повідомленнями, який забезпечує завантаження медіаконтенту, вибір емодзі через інтерактивну панель,

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

а також деталізоване відображення дати й години відправки, статусу редагування та імені й прізвища співрозмовника з аватаром.

Проведено комплексне функціональне тестування основних модулів вебплатформи, перевірено коректність виконання складних запитів до бази даних, стабільність роботи форм, асинхронних запитів та захисту сесій.

Окрему увагу під час проєктування приділено оптимізації клієнтської частини з використанням сучасних стандартів HTML5, CSS3 та асинхронних технологій JavaScript, Fetch API та AJAX. Це дозволило забезпечити миттєвий відгук інтерфейсу при виконанні базових операцій таких, як додавання вподобайок, надсилання повідомлень чи перемикання елементів галереї без повного перезавантаження сторінок. Застосування модульних шаблонів для структуризації сторінок значно спростило подальшу підтримку коду та масштабування вебресурсу.

Окрім вирішення прикладних задач із розробки інтерфейсу та бекенд-логіки, у рамках кваліфікаційної роботи було детально опрацьовано питання архітектурної масштабованості та оптимізації продуктивності вебплатформи. Зокрема, використання реляційної СУБД MySQL дозволило побудувати чіткі зв'язки між таблицями користувачів, постів, коментарів, лайків та повідомлень за допомогою зовнішніх ключів Foreign Keys та каскадного видалення даних, що підтримує її цілісність на найвищому рівні.

У процесі проєктування інтерфейсу особливу увагу приділено принципам сучасного UI/UX-дизайну. Розроблена адаптивна верстка на основі каскадних таблиць стилів CSS3 забезпечує коректне та унікальне відображення на фоні вже існуючих соціальних мереж. Застосування різноманітних анімацій, спливаючих панелей емодзі та динамічних індикаторів каруселі медіафайлів суттєво підвищує загальну зручність взаємодії користувача із системою.

Усі поставлені перед розробкою завдання виконано в повному обсязі, створений програмний продукт повністю відповідає початковим вимогам технічного завдання, демонструє високу відмовостійкість, безпеку та готовність до практичного розгортання в умовах реального мережевого середовища.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мартін Р. С. Чиста архітектура. Мистецтво розробки програмного забезпечення. Київ : Фабула, 2019. 368 с.
2. Фаулер М. Архітектура корпоративних програмних застосунків. Київ : Вільямс, 2018. 544 с.
3. Фленаган Д. JavaScript. Повне керівництво. Київ : Жива мова, 2019. 1080 с.
4. WampServer Documentation. WampServer Official. URL: <https://wampserver.aviatechno.net/>
5. ДСТУ ISO/IEC 12207:2016. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення (ISO/IEC 12207:2008, IDT). Київ : ДП «УкрНДНЦ», 2018. 120 с.
6. JSON Data Interchange Format (ECMA-404). JSON.org. URL: <https://www.json.org/>
7. MDN Web Docs: Using Fetch API. Mozilla Developer Network. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
8. MySQL 9.1.0 Reference Manual. MySQL Developer Zone. URL: <https://dev.mysql.com/doc/refman/9.1/en/>
9. MDN Web Docs: Using files from web applications (FormData / FileReader). Mozilla Developer Network. URL: https://developer.mozilla.org/en-US/docs/Web/API/File/Using_files_from_web_applications
10. Про захист інформації в інформаційно-телекомунікаційних системах: Закон України від 05.07.1994, № 80/94-ВР. Відомості Верховної Ради України. 286 с.
11. ДСТУ ISO/IEC 25010:2016. Інженерія систем і програмного забезпечення. Вимоги до якості програмного продукту та оцінювання (SQuaRE). Моделі якості системи та програмного продукту. Київ : ДП «УкрНДНЦ», 2018. 34 с.
12. OWASP Top Ten: Web Application Security Risks. OWASP Foundation. URL: <https://owasp.org/www-project-top-ten/>

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижморя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

- 13.Nielsen J. Usability Engineering. Academic Press, 1993. 362 p.
- 14.RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. IETF. URL: <https://datatracker.ietf.org/doc/html/rfc7230>
- 15.W3C HTML5 Specification. World Wide Web Consortium (W3C). URL: <https://www.w3.org/TR/html5/>
- 16.Фрімен Е., Робсон Е. Вивчаємо програмування на JavaScript. Київ : Фабула, 2020. 640 с.
- 17.Apache HTTP Server Documentation. The Apache Software Foundation. URL: <https://httpd.apache.org/docs/>.
- 18.PhpStorm Documentation. JetBrains. URL: <https://www.jetbrains.com/phpstorm/documentation/>
- 19.Git: Distributed Version Control System. Git SCM. URL: <https://git-scm.com/doc>
- 20.MDN Web Docs: Client-side web storage (LocalStorage / sessionStorage). Mozilla Developer Network. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_api/Client-side_storage
- 21.OWASP Password Storage Cheat Sheet. OWASP Foundation.
URL:https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
- 22.RFC 6455: The WebSocket Protocol. IETF. URL: <https://datatracker.ietf.org/doc/html/rfc6455>
- 23.PHP Documentation: Introduction. PHP.net. URL: <https://www.php.net/manual/en/introduction.php>
- 24.PHP: PDO (PHP Data Objects) - Manual. PHP.net. URL: <https://www.php.net/manual/en/book.pdo.php>
- 25.MDN Web Docs: Cascading Style Sheets (CSS). Mozilla Developer Network.
URL: <https://developer.mozilla.org/en-US/docs/Web/CSS>.
- 26.W3C Selectors Level 4. World Wide Web Consortium (W3C). URL: <https://www.w3.org/TR/selectors-4/>

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижевотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

27.Font Awesome Icons Reference. Font Awesome. URL:
<https://fontawesome.com/docs>

28.MDN Web Docs: Document Object Model (DOM). Mozilla Developer Network.URL:https://developer.mozilla.org/enUS/docs/Web/API/Document_Object_Model

29.PHP: Error Handling - Manual. PHP.net. URL:
<https://www.php.net/manual/en/book.errorfunc.php>

30.PHP: Sessions - Manual. PHP.net. URL:
<https://www.php.net/manual/en/book.session.php>

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижевотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

ДОДАТКИ

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижевотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

Глосарій

Вебплатформа "Layttle" - розроблена соціальна вебплатформа, призначена для комунікації користувачів, ведення особистих профілів, публікації контенту, обміну повідомленнями у реальному часі та керування соціальними зв'язками.

Емодзі-вподобайки / Likes - інтерактивний елемент соціальної взаємодії, що дозволяє користувачам оцінювати публікації за допомогою емодзі-символів реалізовано через модулі на кшталт рісто.js та обробники лайків.

Користувач / User - зареєстрований користувач вебплатформи, який володіє унікальним обліковим акаунтом, профілем, може створювати дописи, додавати друзів, залишати коментарі та вести приватне листування.

Медіафайли - сукупність завантажених користувачем зображень та відеоматеріалів, які зберігаються у спеціальних директоріях сервера для ілюстрації публікацій і повідомлень.

Профіль / Profile - персональний профіль користувача, що містить його контактні дані, галерею медіаконтенту, а також стрічку його власних публікацій, репостів і збережених матеріалів.

Приватний чат / Chat - модуль обміну приватними повідомленнями між користувачами у реальному часі, що підтримує історію листування, редагування, видалення повідомлень та прикріплення медіафайлів.

Публікація / Post - контент створюваний користувачем у стрічці публікацій або на власній сторінці профілю, яка може містити текст, мультимедійні файли, дату створення, редагування та коментарі.

Стрічка контенту / Index - динамічно генерований інтерфейс головної сторінки сайту, де відображаються дописи користувачів із можливістю сортування за часовими періодами.

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

Система керування базами даних СУБД MySQL - реляційна база даних, що використовується для збереження облікових записів, інформації профілів, публікацій, коментарів та повідомлень через безпечні підключення PDO.

Система маршрутизації / Router - базовий компонент ядра платформи Core/Router.php, що відповідає за аналіз вхідних URL-запитів та їх перенаправлення до відповідних контролерів.

Шаблон / Template - структурний елемент інтерфейсу, що використовується для відображення вебсторінок за патерном MVC (Model-View-Controller).

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

ФРАГМЕНТИ ВИХІДНОГО КОДУ ПРОГРАМНОЇ СИСТЕМИ

Лістинг Б.1. Реалізація механізму маршрутизації та обробки запитів
(Router.php)

```

public function run() {
    $this->route = $this->getRoute();
    if ($this->route === 'site/index') {
        return (new SiteController())->actionIndex();
    }
    if ($this->route === 'profile/like') {
        return (new ProfileController())->actionLike();
    }
    if ($this->route === 'profile/comment') {
        return (new ProfileController())->actionComment();
    }
    if ($this->route === 'profile/addpost') {
        return (new ProfileController())->actionAddPost();
    }
    if ($this->route === 'profile/delete-post') {
        return (new \controllers\ProfileController())->actionDeletePost();
    }
    if ($this->route === 'profile/edit-post') {
        return (new \controllers\ProfileController())->actionEditPost();
    }
    if ($this->route === 'profile/add-friend') {
        return (new ProfileController())->actionAddFriend();
    }
    if ($this->route === 'profile/remove-friend') {
        return (new \controllers\ProfileController())->actionRemoveFriend();
    }
    if ($this->route === 'profile/update-profile-ajax') {
        return (new ProfileController())->actionUpdateProfileAjax();
    }
    if ($this->route === 'profile/repost') {
        return (new ProfileController())->actionRepost();
    }
    if ($this->route === 'profile/favorite') {
        return (new ProfileController())->actionFavorite();
    }
    if (preg_match('#^profile/reposts/([\w\d_-]+)$#', $this->route, $matches)) {
        return (new ProfileController())->reposts($matches[1]);
    }
    if (preg_match('#^profile/favorites/([\w\d_-]+)$#', $this->route, $matches)) {
        return (new ProfileController())->favorites($matches[1]);
    }
    if (preg_match('#^profile/([\w\d_-]+)$#', $this->route, $matches)) {
        return (new ProfileController())->view($matches[1]);
    }
}

```

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижемоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

        if (preg_match('#^search/([\w\d\p{L} _-]*)$#u', $this->route, $matches)) {
            $_GET['search'] = urldecode($matches[1]);
            return (new \controllers\SearchController())->actionIndex();
        }
        if ($this->route === 'users/login') {
            return (new UsersController())->actionLogin($_POST ?? []);
        } elseif ($this->route === 'users/register') {
            return (new UsersController())->actionRegister($_POST ?? []);
        }
        $this->error(404);
    }
}

```

Лістинг Б.2. Реєстрація маршрутів додатку в системі (Router.php)

```

$router = new Router(trim($_SERVER['REQUEST_URI'], '/'));
$router->addRoute('users/login', 'UsersController@actionLogin');
$router->addRoute('users/register', 'UsersController@actionRegister');
$router->addRoute('users/registersuccess', 'UsersController@actionRegistersuccess');
$router->addRoute('users/logout', 'UsersController@actionLogout');
$router->addRoute('profile/addpost', 'Profile@actionAddpost');
$router->addRoute('profile/edit', 'ProfileEditController@edit');
$router->addRoute('profile/me', 'ProfileController@view');
$router->addRoute('profile/update', 'Profile@actionUpdate');
$router->addRoute('profile', 'ProfileController@view');
$router->addRoute('profile/{username}', 'ProfileController@view');
$router->addRoute('chat', 'ChatController@index');
$router->addRoute('chat/chat/{id}', 'ChatController@show');
$router->addRoute('chat/editMessage', 'ChatController@editMessage');
$router->addRoute('chat/deleteMessage', 'ChatController@deleteMessage');
$router->addRoute('chat/show/{id}', 'ChatController@show');
$router->addRoute('chat/store/{id}', 'ChatController@store');
$router->addRoute('chat/messages/{id}', 'ChatController@messages');
$router->addRoute('profile/add-friend', 'ProfileController@actionAddFriend');
$router->addRoute('profile/remove-friend', 'ProfileController@actionRemoveFriend');
$router->addRoute('profile/update-profile-ajax', 'ProfileController@actionUpdateProfileAjax');
$router->addRoute('search', 'SearchController@index');
$router->addRoute('friends/{login}', 'SearchController@myFriends');
$router->addRoute('friends/{login}/search/{query}', 'SearchController@searchMyFriends');
$router->addRoute('profile/reposts/{username}', 'ProfileController@reposts');
$router->addRoute('profile/favorites/{username}', 'ProfileController@favorites');
$router->addRoute('profile/repost', 'ProfileController@actionRepost');
$router->addRoute('profile/favorite', 'ProfileController@actionFavorite');
$router->addRoute('', 'SiteController@actionIndex');
$router->addRoute('index', 'SiteController@actionIndex');
$router->run();

```

Лістинг Б.3. Головний клас ініціалізації середовища та конфігурації

(Core.php)

```
<?php
namespace core;
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
require_once __DIR__ . '/../config/AppConfig.php';
require_once __DIR__ . '/Template.php';
require_once __DIR__ . '/Session.php';
require_once __DIR__ . '/DB.php';
class Core
{
    public string $defaultLayoutPath = 'views/layouts/index.php';
    public $moduleName;
    public $actionName;
    public $router;
    public $template;
    public $db;
    public Controller $controllerObject;
    public $session;
    private static $instance;
    private function __construct($route)
    {
        $this->template = new Template(__DIR__ . '/../views/layouts/index.php');
        $host = AppConfig::get()->dbHost;
        $name = AppConfig::get()->dbName;
        $login = AppConfig::get()->dbLogin;
        $password = AppConfig::get()->dbPassword;
        $this->db = new DB($host, $name, $login, $password);
        $this->session = new Session();
    }
    public function run($route)
    {
        $this->router = new Router($route);
        $params = $this->router->run();
        if (!empty($params)) {
            $this->template->setParams($params);
        }
    }
    public function done()
    {
        $this->template->display();
        $this->router->done();
    }
    public static function get($route = null)
    {
        if (empty(self::$instance)) {
            if ($route === null) {
                throw new \InvalidArgumentException("Route parameter is required for Core::get()");
            }
        }
    }
}
```

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

```

    }
    self::$instance = new Core($route);
  }
  return self::$instance;
}
}

```

Лістинг Б.4. Базовий клас контролера для обробки запитів додатку (Controller.php)

```

<?php
namespace core;
require_once 'D:\wamp64\domains\Layttle\core\Session.php';
require_once __DIR__ . '/Template.php';
use JetBrains\PhpStorm\NoReturn;
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
class Controller
{
    protected ?Template $template = null;
    protected array $errorMessages = [];
    public bool $isPost = false;
    public bool $isGet = false;
    public Post $post;
    public Get $get;
    public function __construct()
    {
        error_log("Module: " . Core::get()->moduleName);
        error_log("Action: " . Core::get()->actionName);
        $session = new Session();
        $module = Core::get()->moduleName ?: 'users';
        $action = Core::get()->actionName ?: 'login';
        $path = __DIR__ . '/../views/{ $module}/{ $action}.php";
        if (file_exists($path)) {
            $this->template = new Template($path);
        } else {
            error_log("Template file does not exist: " . $path);
        }
        switch ($_SERVER['REQUEST_METHOD']) {
            case 'POST':
                $this->isPost = true;
                break;
            case 'GET':
                $this->isGet = true;
                break;
        }
        $this->post = new Post();
        $this->get = new Get();
    }
}
/**
 * @throws \Exception

```

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

*/
public function render($pathToView = null, $data = []): array
{
    if ($this->template !== null) {
        if (!empty($pathToView)) {
            $path = ltrim($pathToView, '/');
            $fullPath = __DIR__ . '/../views/' . $path;
            $this->template->setTemplateFilePath($fullPath);
        }
        $this->template->post = $this->post;
        foreach ($data as $key => $value) {
            $this->template->setParam($key, $value);
        }
        return [
            'Content' => $this->template->getHTML()
        ];
    } else {
        throw new \Exception("Template object is not initialized.");
    }
}
#[NoReturn] public function redirect($path): void
{
    header("Location:{$path}");
    die;
}
}

```

Лістинг Б.5. Компонент управління сесіями та авторизацією користувачів
(Session.php)

```

<?php
namespace core;
class Session
{
    public function __construct()
    {
        if (session_status() === PHP_SESSION_NONE) {
            session_start();
        }
    }
    public function set($name, $value)
    {
        if (is_array($value)) {
            $value = serialize($value);
        }
        $_SESSION[$name] = $value;
    }
    public function get($name)
    {
        if (isset($_SESSION[$name])) {
            $value = $_SESSION[$name];
            if ($this->isSerialized($value)) {

```

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

        $value = unserialize($value);
    }
    return $value;
}
return null;
}
public function remove($name)
{
    if (isset($_SESSION[$name])) {
        unset($_SESSION[$name]);
    }
}
private function isSerialized($data)
{
    return is_string($data) && ($data == serialize(false) || @unserialize($data) !== false);
}
}

```

		Дубров М.Ю.			ІПЗ.КР.Б. – 121 – 26 – ПЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		90