БАКАЛАВРСЬКА РОБОТА

БР.КІ-18.00.00.000 ПЗ

Група КІ-21-1

Савчук Ольга

2025

Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу
Інститут інформаційних технологій

Кафедра комп'ютерних систем і мереж

Савчук Ольга Анатоліївна

УДК 007

БАКАЛАВРСЬКА РОБОТА

Розробка web-застосунку для ведення особистих нотаток та нагадувань з використанням HTML, CSS, JavaScript, PHP

Комп'ютерна інженерія
(назва освітньої програми)

123 – Комп'ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач освітнього ступеня _____ *Савчук О.А*
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____ *Кропивницький Д.Р, доцент*
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри
д.т.н., професор _____ /С. І. Мельничук/
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025 рік

Івано-Франківський національний технічний університет нафти і газу

Інститут *Інформаційних технологій*

Кафедра *Комп'ютерних систем і мереж*

Освітньо-кваліфікаційний рівень *бакалавр*

Спеціальність *123 – Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ:

Зав. кафедрою КСМ

С.І. Мельничук

« » травня 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Савчук Ользі Анатоліївній

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи*) Розробка web-застосунку для ведення особистих нотаток та нагадувань з використанням HTML, CSS, JavaScript,PHP*

керівник проекту (роботи) *Кропивницький Дмитро Романович, доцент.*

затверджені наказом вищого навчального закладу від *05.05.2025 № 275/7*

2. Строк подання студентом проекту (роботи) *12 червня 2025р.*

3. Вихідні дані до роботи *Методичні вказівки, технічна література .*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *1. Теоретичні основи та аналіз предметної області_ 2. Проектування та архітектура системи . 3.Реалізація веб-застосунку.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) .

6. Консультанти розділів роботи

7. Дата видачі завдання *29 січня 2025 р.* .

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	*Збір інформації, вивчення літератури та пошук додаткової інформації*	*Лютий 2025р*	
2	*Теоретичні основи та аналіз предметної області*	*Березень 2025р*	
3	*Проектування та архітектура системи*	*Квітень 2025р*	
4	*Реалізація веб-застосунку*	*Травень 2025р*	
5	*Оформлення додатків, дипломної роботи*	*Червень 2025р*	

Студент _ *Савчук О.А*

Керівник роботи _____ *Кропивницький Д.Р*

АНОТАЦІЯ

Дипломна робота містить 77 сторінок, 45 рисунків, 35 використаних джерел.

Метою дипломної роботи є створення веб-застосунку для ведення особистих нотаток та нагадувань з використанням сучасних веб-технологій.

Об’єкт дослідження: процес створення, збереження та управління персональною інформацією користувача у веб-середовищі.

Предметом дослідження є методи, засоби та підходи до проєктування, реалізації й тестування інтерактивних веб-застосунків, що забезпечують зручну та інтуїтивну роботу з текстовими даними, нагадуваннями та мультимедійними вкладеннями.

У першому розділі представлено огляд і аналіз існуючих веб-застосунків для нотаток, визначено цільову аудиторію, сформульовано функціональні й нефункціональні вимоги до майбутньої системи.

Другий розділ присвячено проектуванню системи, включаючи вибір технологій, архітектурну побудову, структуру бази даних, діаграми взаємодії, а також моделювання інтерфейсу користувача.

У третьому розділі розглянуто реалізацію основних функціональних модулів: реєстрації та авторизації, створення, редагування й видалення нотаток, встановлення нагадувань, додавання вкладень, тегів, чеклістів, коментарів, а також можливість поширення нотаток між користувачами. Проведено тестування системи на коректність роботи та визначено напрями подальшого розвитку проєкту.

Отримані результати дослідження: функціональний застосунок для ведення особистих нотаток, адаптований до індивідуальних потреб користувача.

Ключові слова: веб-застосунок, нотатки, нагадування, бази даних, інтерфейс, тестування.

ANNOTATION

The thesis contains 77 pages, 45 figures, 35 used sources.

The aim of the thesis is to develop a web application for managing personal notes and reminders using modern web technologies.

Research object: he process of creating, storing, and managing personal user information in a web environment.

The subject of research s the methods, tools, and approaches for designing, implementing, and testing interactive web applications that provide convenient and intuitive work with text data, reminders, and multimedia attachments.

The first section provides an overview and analysis of existing web applications for note-taking, defines the target audience, and formulates the functional and non-functional requirements for the system.

The second chapter describes the system design, including the selection of development technologies, system architecture, database structure, interaction diagrams, and user interface modeling.

The third section is devoted to the implementation of the system, covering the main functional modules such as registration and authentication, creation, editing, and deletion of notes, setting reminders, attaching files, adding tags, checklists, and comments, as well as the ability to share notes between users. Testing of the system was conducted to verify its correctness, and directions for further development were outlined.

The conclusions summarize the achieved results, confirm the attainment of the goal, and demonstrate the compliance of the implementation with the defined requirements.

The obtained results of the research: functional web application for managing personal notes, adapted to individual user needs.

Keywords: web application, notes, reminders, databases, interface, testing.

ЗМІСТ

ВСТУП **4**

1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ **6**

1.1 Актуальність створення веб-застосунку для організації особистих нотаток ..... **Ошибка! Закладка не определена.**

1.2 Огляд і аналіз існуючих веб-застосунків для нотаток та нагадувань 7

1.3 Визначення основних функції та вимог системи. 13

2 ПРОЕКТУВАННЯ ТА АРХІТЕКТУРА СИСТЕМИ **17**

2.1 Вибір технологій та інструментів розробки 18

2.2 Use-case діаграма та діаграма послідовності 19

2.3 Архітектура системи 28

2.4 Проектування бази даних 31

2.5 Моделювання інтерфейсу користувача та UI/UX дизайн 42

3 РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ **47**

3.1 Підготовка середовища розробки 47

3.2 Реалізація програмних модулів 50

3.3 Інтерфейс користувача 54

3.4 Перевірка системи на працездатність 61

ВИСНОВКИ **73**

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ **75**

ДОДАТКИ

ВСТУП

Актуальність теми. З огляду на зростання обсягів інформації та потребу в ефективному управлінні особистими справами, особливої важливості набувають інструменти для фіксації, збереження та систематизації даних. Сучасні веб-застосунки, які не потребують встановлення та забезпечують доступ з будь-якого пристрою, є зручним рішенням для щоденного використання. Попри наявність великої кількості вже існуючих сервісів, не всі з них відповідають індивідуальним вимогам користувачів або є надто складними для повсякденного використання. Це відкриває можливості для створення власного веб-застосунку, який буде адаптований під конкретні потреби, матиме простий інтерфейс та забезпечуватиме швидкий доступ до важливої інформації. Особливо актуальним є створення простих, інтуїтивно зрозумілих та адаптивних рішень.

Об’єкт дослідження. Процеси створення, збереження, обробки та управління особистими інформаційними даними користувача, зокрема нотатками та нагадуваннями, у межах веб-інформаційних систем.

Предмет дослідження. методи та інструменти розробки веб-застосунків для організації особистих нотаток і нагадувань, а також програмні рішення на основі HTML, CSS, JavaScript і PHP, що забезпечують зручний інтерфейс, функціональність, персоналізацію та доступність з різних пристроїв.

Мета роботи – Розробка веб-застосунку для ведення особистих нотаток та нагадувань із використанням сучасних веб-технологій: HTML, CSS, JavaScript і PHP. Такий застосунок має забезпечити користувачеві базову функціональність з високою зручністю та швидкістю роботи, а також надати можливість персоналізації досвіду.

Завдання роботи: Для досягнення цієї мети поставлено наступні завдання:

- проаналізувати існуючі рішення у сфері цифрових нотатників;
- визначити вимоги до системи;

- спроєктувати архітектуру веб-застосунку та бази даних;
- реалізувати інтерфейс і функціональну частину застосунку;
- протестувати застосунок і оцінити перспективи його розвитку.

Методи дослідження. У процесі виконання роботи було використано такі методи: аналіз та порівняння аналогічних програмних рішень, моделювання архітектури та структури бази даних, проектування інтерфейсу користувача, розробка веб-сторінок з використанням HTML, CSS і JavaScript, серверна логіка на основі PHP, а також ручне тестування функціональних модулів.

Практичне значення роботи полягає у створенні реального працюючого прототипу веб-застосунку, який може використовуватись для особистих цілей або адаптований до ширшого кола користувачів. Результати дослідження можуть бути використані в подальших розробках застосунків для особистої організації.

1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні потреби в цифрових засобах для особистих записів

У сучасному інформаційно насиченому середовищі люди щодня мають справу з великою кількістю завдань, ідей, нагадувань і зобов'язань. Це створює постійну потребу в зручних інструментах для фіксації та впорядкування особистої інформації. Відсутність порядку в таких даних часто призводить до втрати важливої інформації або зниження продуктивності [1].

Цифрові технології суттєво розширили можливості особистого тайм-менеджменту. Вебзастосунки для створення нотаток дозволяють користувачам зберігати й структурувати інформацію в хмарному середовищі, забезпечуючи доступ з будь-якого пристрою. В епоху стрімкого розвитку дистанційної роботи та онлайн-навчання вебрішення мають перевагу над локальними застосунками завдяки гнучкості, мобільності й простоті використання [2].

Серед сучасних інструментів для створення особистих нотаток можна виокремити кілька основних типів застосунків.

Перший тип — просторові застосунки — побудовані за ієрархічною структурою блокнотів, розділів і сторінок. Ці сервіси дозволяють гнучко організовувати інформацію за темами, проєктами або напрямами. Вони ідеально підходять для навчання, досліджень або професійної діяльності, де необхідно структурувати великі обсяги даних.

Другий тип — застосунки у простому або мінімалістичному форматі — охоплює сервіси для швидких нотаток або списків справ. Такі інструменти зазвичай мають надзвичайно простий інтерфейс, не потребують спеціального навчання і чудово підходять для коротких щоденних записів, таких як списки покупок, нагадування або ідеї «на ходу».

Третя категорія — комбіновані рішення — поєднує базові функції створення нотаток з додатковими можливостями, такими як вставлення зображень,

тегування, чеклісти, нагадування й історія змін. Такі системи орієнтовані на широку аудиторію і можуть використовуватись як для особистих, так і для робочих потреб.

Четвертий тип — колаборативні або командні інструменти — дозволяє не лише створювати нотатки, а й ділитися ними з іншими користувачами, залишати коментарі та редагувати їх у реальному часі. Вони популярні у командній співпраці, управлінні проєктами та освітньому середовищі.

У такому контексті створення індивідуального вебзастосунку, який поєднує простий інтерфейс, адаптивність, базову структуру організації нотаток і можливість роботи в різних режимах, є надзвичайно актуальним. Такий підхід дозволяє задовольнити потреби різних цільових груп — від початківців до досвідчених користувачів, які працюють з великим обсягом інформації.

Розроблений веб-застосунок виступає як універсальна система для підтримки особистого інформаційного процесу — від фіксації думок і планів до нагадування про важливі події. Завдяки поєднанню зручного інтерфейсу, системи збереження даних і можливостей взаємодії в реальному часі він забезпечує безперервність і доступність особистої інформації.

Таким чином, розробка вебзастосунку для управління особистими нотатками є доцільним і своєчасним рішенням, що відповідає сучасним потребам користувачів і сприяє ефективнішому плануванню повсякденного життя.

1.2 Огляд і аналіз існуючих веб-застосунків для нотаток та нагадувань

На сьогодні ринок веб-застосунків для ведення особистих нотаток і нагадувань представлений широким спектром сервісів, що пропонують користувачам різні підходи до організації інформації. Найбільш популярними серед них є Google Keep, Microsoft OneNote, Evernote, Simplenote. Ці сервіси активно використовуються мільйонами людей по всьому світу завдяки своїй надійності, гну-

чкості та багатому функціоналу. Кожен із перелічених застосунків має власні особливості, які відображаються у способах створення та структурування записів, системах тегів, інтеграції з іншими платформами, підтримці форматування, спільного доступу до нотаток тощо [3]. Разом з тим, існують і певні обмеження, як-от перевантаженість інтерфейсу, обмеження безкоштовних версій або залежність від хмарної інфраструктури [4].

У цьому підрозділі було проведено детальний огляд та порівняльний аналіз зазначених сервісів, що стане основою для формування вимог до майбутнього програмного продукту.

Сервіс Google Keep (https://keep.google.com/) є одним із найпопулярніших сервісів для створення швидких нотаток та нагадувань. Сервіс орієнтований на швидке фіксування ідей, організацію простих завдань та збереження інформації для подальшого використання. Він дозволяє створювати нотатки у вигляді кольорових карток, додавати чек-листи, прикріплювати зображення та встановлювати нагадування, зокрема й за геолокацією [5]. Крім того, підтримується спільний доступ, що дозволяє кільком користувачам редагувати одну нотатку [6]. Водночас, сервіс має певні обмеження: він не підтримує папки або вкладені структури, немає повноцінного форматування тексту, і всі нотатки відображаються разом без гнучкої організації [7]. Тому для користувачів, які прагнуть простоти, але потребують трохи більшої структури, Google Keep може виявитись недостатнім. Основна функціональність включає: створення текстових нотаток та чек-лістів, додавання зображень, нагадування за часом та місцем, а також синхронізація з Google аккаунтом. Перевагами застосунку Google Keep є простий та інтуїтивний інтерфейс, який є зрозумілим для користувачів, можливість спільного доступу до нотаток, створення голосових нотаток, а також кольорове маркування. Недоліками системи є відсутність папок і вкладених структур, обмеження форматування тексту, відсутність категорій.

З технічного боку: Google Keep — це хмарний сервіс, який працює у браузері за рахунок клієнтської логіки на JavaScript. Інтерфейс реалізований із

використанням сучасних веб технологій (HTML5, CSS3), що дозволяє адаптувати вигляд під різні екрани та розміри пристроїв. Для зберігання та обміну даними використовується API Google, а інформація зберігається на серверах Google у структурованій формі, з ідентифікацією користувача через обліковий запис.

Microsoft OneNote (https://www.onenote.com/), на відміну від Google Keep, пропонує гнучку систему організації інформації, що включає блокноти, розділи та сторінки. Його функціонал дозволяє створювати складно структуровані нотатки з використанням повного форматування тексту, вставкою таблиць, зображень, рукописного вводу та аудіо [8]. OneNote імітує структуру традиційного паперового блокнота: нотатки можна зберігати в «блокнотах», які поділяються на розділи й сторінки. Така ієрархічна система дозволяє зручно організовувати великі обсяги інформації. Проте через велику кількість можливостей інтерфейс може бути перевантаженим для нових користувачів, що не шукають складних рішень. Основна функціональність системи включає: багаторівневу структуру нотаток, підтримка рукописного вводу , додавання файлів, зображень, таблиць. Серед переваг — гнучкість, глибоке форматування, розширена організація та синхронізація через OneDrive [9]. До недоліків системи можна віднести складний інтерфейс, собливо для нових користувачів, відсутність вбудованних нагадувань, а також складна навігація при великому обсязі контенту (нотаток, завдань, заміток).

З технічного боку OneNote — це хмарний сервіс, частина екосистеми Microsoft, з веб інтерфейсом, побудованим на HTML, CSS і JavaScript, а також із потужною серверною підтримкою. Дані зберігаються у структурованій формі в хмарі, синхронізуються через Microsoft Graph API, що дозволяє підтримувати складну ієрархію блокнотів.

Evernote (https://evernote.com/) — інструмент, який надає можливість створювати детальні та форматовані нотатки, організовані у блокноти з підтримкою тегів. Він поєднує простоту базових дій із широкими можливостями для

структурованого збереження інформації, що робить його корисним як для звичайних користувачів, так і для професіоналів у сфері освіти, бізнесу чи досліджень. Evernote надає багаті можливості структуризації нотаток: можна створювати блокноти, використовувати шаблони, формати, а також здійснювати повнотекстовий пошук навіть усередині зображень і PDF-файлів. Користувач може додавати зображення, документи, використовувати шаблони, а також зберігати сторінки з вебу за допомогою спеціального розширення. Попри багатий набір функцій, інтерфейс може здатися перевантаженим і не завжди інтуїтивно зрозумілим, особливо для нових користувачів [10]. Ще одним недоліком є обмежена функціональність нагадувань: сервіс не має централізованого механізму управління завданнями, що змушує користувачів шукати альтернативи або додаткові застосунки [11].

З технічного боку Evernote реалізований як хмарний сервіс із веб інтерфейсом на основі JavaScript, HTML5 та CSS. Всі дані зберігаються на серверах Evernote у вигляді структурованих документів, які синхронізуються через REST API.

Simplenote (https://simplenote.com/) призначений для користувачів, яким важлива швидкість і простота роботи. Основна функціональність Simplenote полягає у створенні звичайних текстових нотаток, а також надає користувачам низку зручних інструментів: тегування нотаток для організації вмісту, пошук за ключавими словами, автоматичне збереження змін, а також activity log, що дозволяє переглядати та відновлювати попередні редакції [12]. Однією з важливих особливостей є підтримка Markdown, що дозволяє форматувати текст без складного графічного інтерфейсу. Завдяки цьому Simplenote підходить для технічних користувачів, студентів, письменників або тих, хто веде щоденники в лаконічному стилі [13]. Разом із тим, застосунок має обмеження, які роблять його менш придатним для візуального або мультимедійного використання. Зокрема, відсутня підтримка зображень, файлів, нагадувань чи календаря**.** Також не передбачено спільного редагування нотаток у реальному часі [14]. Цей за-

стосунок ідеально підходить для текстових записів, але не задовольняє потреби користувачів, які шукають функцію нагадувань або базову візуальну організацію нотаток.

З технічного боку Simplenote є прикладом ефективного вебзастосунку на JavaScript з використанням фреймворку React. Усі нотатки зберігаються у вигляді простих текстових об’єктів на сервері компанії Automatic, а зміни синхронізуються через API.

Нижче наведенна порівняльна таблиця, яка дозволяє зіставити популярні додатки для створення нотаток за основною функціональністю, перевагами, недоліками та технічною реалізацією. Вона допоможе об’єктивно оцінити можливості кожного сервісу з точки зору зручності, функціоналу та практичного застосування.

Таблиця 1.1 - Порівняльна таблиця сервісів створення нотаток

Сервіс	**Функціональність**	**Переваги**	**Недоліки**	**Технічна реалізація**
Google Keep	Текстові нотатки, чеклісти, зображення, нагадування, кольорове маркування	Простота, інтуїтивність, швидкість, голосові нотатки, синхронізація	Немає вкладених структур, обмежене форматування, відсутність категорій	JavaScript, HTML5/CSS3, API Google, хмарне зберігання
Microsoft OneNote	Багаторівнева структура, форматування, таблиці, файли, рукопис	Гнучкість, потужна структура, інтеграція з Microsoft 365	Складний інтерфейс, відсутність зручних нагадувань	HTML/CSS/JS, Microsoft Graph API, інтеграція з OneDrive
Evernote	Текст, файли, шаблони, теги, пошук у PDF/зображеннях	Розширений пошук, шаблони, збереження сторінок з вебу	Обмеження безкоштовної версії, складність інтерфейсу	JavaScript, REST API, хмарна синхронізація
Simplenote	Markdown, теги, історія змін, пошук, швидке введення	Швидкодія, мінімалізм, безкоштовність, підтримка Markdown	Немає зображень, нагадувань, слабкий візуальний функціонал	React, API Automattic, зберігання простих текстових об'єктів

Аналіз зазначених веб-застосунків показує, що жоден з них не є одночасно простим і достатньо функціональним для базових потреб нотаток та нагаду-

вань. Одні сервіси перевантажені можливостями, інші, навпаки, надто обмежені.

Для успішної реалізації веб-застосунку важливим етапом було проведення аналізу цільової аудиторії, на яку буде орієнтовано продукт і виділено 3 сегмента потенційних користувачів:

Перша – люди, які активно планують свій час і справи в повсякденному житті. Це найбільша і найрізноманітніша група користувачів. До неї належать студенти, офісні працівники, фрілансери, молоді мами, — всі, хто прагне краще організовувати свій день, не забувати важливі справи й фіксувати особисті ідеї. Такі користувачі потребують простого, інтуїтивно зрозумілого інтерфейсу, швидкого створення нотаток і нагадувань, можливості візуального розділення записів за категоріями або кольорами. Особливу цінність для них мають функції нагадування за датою і часом, чеклісти для щоденних задач, а також можливість ділитись та/або створювати спільні нотатки.

Друга – користувачі, які працюють з великими обсягами інформації. Цей сегмент включає фахівців, які ведуть записи у рамках навчальної, дослідницької чи проєктної діяльності: викладачі, аналітики, студенти, журналісти, письменники, розробники. Вони використовують нотатки для збереження структурованої інформації, створення чернеток, планів, тез чи технічних завдань. Такі користувачі зацікавлені у більш гнучких інструментах: тегуванні, пошуку, додаванні вкладень, коментарях до нотаток, а також підтримці форматування. Вони також можуть потребувати можливості спільної роботи над нотатками.

Третя - Люди, які щойно починають використовувати цифрові інструменти для самоменеджменту. Це новачки в цифровому плануванні — користувачі, які раніше вели записи на папері або взагалі не використовували систематизовані способи зберігання інформації. Їм важливо мати максимально просту систему з підказками, яка не потребує складного навчання.

Розуміння потреб кожного з цих трьох сегментів дозволяє забезпечити збалансований функціонал, що буде однаково зручним як для звичайного кори-

стувача, який хоче записати список покупок, так і для професіонала, який структурує проєктну інформацію.

1.3 Визначення основних функції та вимог системи

Після проведеного аналізу існуючих аналогів та вивчення потреб користувачів, ми чітко бачимо і розуміємо, на що варто звернути особливу увагу під час розробки власного веб-застосунку. Такий аналіз дозволив сформувати уявлення про необхідні можливості системи та визначити основні функції, які повинні бути реалізовані для досягнення поставлених цілей. Процес створення програмного забезпечення передбачає чітке формулювання як функціональних, так і нефункціональних вимог.

Функціональні вимоги визначають, які конкретно дії має виконувати вебзастосунок для задоволення потреб користувачів.

Реєстрація та аутентифікація користувачів. Система забезпечує можливість реєстрації нового користувача шляхом введення електронної пошти та пароля. Після успішної реєстрації користувач отримує доступ до персонального кабінету. Під час реєстрації здійснюється перевірка правильності заповнення полів. Після створення облікового запису користувач має можливість авторизуватись у системі за допомогою введення своїх даних. Також реалізована функція виходу із системи, що забезпечує безпеку при завершенні роботи

Створення, редагування та видалення нотаток. Застосунок надає можливість створення нових нотаток з текстовим вмістом. Користувач може вводити довільний текст, вказувати заголовок, а також при потребі — теги. Після створення, нотатка зберігається у базі даних, і відображається у загальному списку. Також користувач має змогу редагувати вже існуючі нотатки — змінювати текст, заголовок, оновлювати теги. У разі необхідності передбачена функція видалення нотаток. При цьому реалізовано механізм попередження або підтвердження дії видалення для уникнення випадкової втрати інформації. Користу-

вацьке керування контентом є ключовим для цифрових записників. Як зазначає Nielsen Norman Group, «користувачі очікують миттєвої реакції на зміни контенту» [15].

Створення та керування нагадуваннями. Веб-застосунок дозволяє створювати нагадування, які можуть бути пов’язані з певними нотатками або мати окрему форму. Для кожного нагадування користувач має змогу встановлювати дату та час спрацювання. У момент досягнення встановленого часу система повідомляє користувача про нагадування — через відображення повідомлення в інтерфейсі. При створенні нагадування реалізована можливість додавати короткий опис або назву події. Користувач може переглядати список активних нагадувань, редагувати їх або видаляти при потребі.

Пошук, фільтрація та сортування записів. Веб-застосунок надає зручні засоби пошуку нотаток за ключовими словами, що зустрічаються в заголовку, основному тексті або тегах. Також реалізована фільтрація записів за категоріями, датою створення, наявністю нагадування тощо.

Інтерфейс взаємодії з нотатками. Інтерфейс є інтуїтивно зрозумілим і зручним. Для кожної нотатки має бути відображено її заголовок, текст, дата створення, теги та категорія. Також передбачений перегляд повного тексту нотатки та функції редагування й видалення. Для підвищення зручності важливі нотатки можна закріпити у верхній частині списку. Також корисною є можливість позначення нотаток як “виконаних” або “активних”. За даними ISO 9241, інтуїтивність інтерфейсу прямо пов’язана з рівнем задоволеності користувача [16].

Архівування та відновлення нотаток. Користувач має змогу архівувати нотатки, які не використовуються часто, але можуть знадобитися в майбутньому. Архівовані записи зберігаються окремо та не відображаються у загальному списку активних нотаток. При потребі користувач може переглянути архів, повернути нотатку до активного стану або остаточно її видалити.

Взаємодія з користувачем та підтвердження дій. Усі критичні дії, пов’язані зі зміною або видаленням даних (наприклад, видалення нотатки чи нагадування), повинні супроводжуватися відповідними діалоговими вікнами з підтвердженням. Це дозволяє зменшити ризик випадкових дій.

Таким чином, перелічені функціональні вимоги охоплюють повний цикл взаємодії користувача з системою: від реєстрації до створення, перегляду, редагування, фільтрації та архівування записів. Вони є основою для реалізації логіки роботи веб-застосунку, що дозволить забезпечити ефективне управління особистими нотатками й нагадуваннями.

Нефункціональні вимоги визначають загальні характеристики системи, які не стосуються конкретної функціональності, але мають вирішальне значення для забезпечення стабільної, зручної, безпечної та ефективної роботи веб-застосунку.

Продуктивність. Веб-застосунок забезпечує високу продуктивність, зокрема швидкий час відгуку при роботі з нотатками, нагадуваннями, авторизацією та іншими діями. Середній час відгуку інтерфейсу не перевищує 1–2 секунд, що є критичним для комфортного користування.

Масштабованість. Хоча веб-застосунок є орієнтованим на індивідуальне використання, система має архітектуру, здатну до масштабування. У разі розширення функціональності, архітектура повинна дозволяє додавання нових модулів без повної перебудови коду. Це забезпечується модульною структурою застосунку та чітким поділом між клієнтською та серверною частинами.

Доступність і надійність. Система є стабільною та доступною для користувача в будь-який час. Для цього забезпечено резервне копіювання даних користувача, щоб у разі збою дані можна було відновити без втрати. Надійність веб-застосунку передбачає обробку екстрених ситуацій, таких як втрата з’єднання, відсутність відповіді від бази даних або помилки введення даних.

Зручність використання. Зручність інтерфейсу є ключовим фактором для системи, орієнтованої на щоденне використання. Інтерфейс є інтуїтивно зрозу-

мілим, зі зручною навігацією, чіткими кнопками, підписами та підказками. Всі основні дії, такі як створення, редагування, видалення нотаток або налаштування нагадувань, повинні є доступними за кілька кліків. При розробці системи витриманий єдиний стиль оформлення на всіх сторінках. Додатково, реалізована функція зворотнього зв'язку від користувачів, що дозволить оперативно покращувати зручність системи. Згідно з дослідженнями Interaction Design Foundation, адаптивний інтерфейс і підтримка доступності (навігація з клавіатури, контрастність) є ключовими для щоденного використання [17].

Підтримка та обслуговування. Надійна підтримка й технічне обслуговування забезпечують стабільність роботи системи. Кодова база є добре структурована, що полегшить її модифікацію, оновлення або масштабування. Документаці містить опис основної архітектури, структури БД, опис API та логіки взаємодії між модулями. Це забезпечить швидке впровадження нових функцій та спростить супровід. Додатково, в застосунку реалізована форма для зворотного зв’язку, через яку користувачі зможуть надсилати побажання чи повідомляти про баги. Усі ці заходи спрямовані на забезпечення довготривалої стабільної роботи веб-застосунку.

Проведений аналіз функціональних і нефункціональних вимог, а також вивчення цільової аудиторії та конкурентних рішень дозволили сформувати перелік ключових функцій, реалізованих у веб-застосунку.

Таблиця 1.2 – Функції для реалізації у власному додатку

Функція	Опис
1	2
Створення текстових нотаток	Дозволяє користувачу записувати короткі чи довгі текстові записи.
Редагування нотаток	Можливість змінювати вміст існуючих нотаток.
Видалення нотаток	Функція для очищення або видалення непотрібної інформації.

Кінець таблиці 1.2

1	2
Додавання чек-листів	Списки справ із можливістю відмічати ви-

	конані пункти
Встановлення нагадувань	Нотифікації або повідомлення про події, пов'язані з нотаткою.
Теги / категорії	Організація нотаток за допомогою міток для фільтрації.
Прикріплення зображень	Можливість додавати картинки до нотаток.
Пошук по нотатках	Швидкий пошук за ключовими словами або тегами.
Форма зворотнього зв’язку	Зв’язок з користувачем у випадку виявлення багів, та формування бажань користувачів з приводу того що вони хочуть бачити в системі.

Реалізація наведених функцій забезпечує повноцінне використання веб-застосунку як у повсякденному житті, так і в професійній діяльності. Вони охоплюють базові потреби користувачів у збереженні, організації, пошуку та нагадуванні важливої інформації. Такий набір функцій забезпечує зручність, ефективність і гнучкість роботи із записами, що є ключовими чинниками для залучення й утримання користувачів.

2 ПРОЕКТУВАННЯ ТА АРХІТЕКТУРА СИСТЕМИ

2.1 Вибір технологій та інструментів розробки

У процесі розробки вебзастосунку для ведення особистих нотаток і нагадувань було використано набір сучасних вебтехнологій, що охоплюють як клієнтську, так і серверну частини. Обрані інструменти забезпечують взаємодію між користувачем, інтерфейсом, серверною логікою та базою даних, що дозволяє реалізувати повноцінну функціональність системи.

Основою для створення інтерфейсу став HTML5 — мова розмітки, яка дала змогу структурувати вміст сторінок за допомогою семантичних тегів, таких як <header>, <section>, <article> тощо. Це підвищує доступність інтерфейсу, спрощує SEO та покращує читабельність коду [18]. При розробці HTML5 використовувався для форм реєстрації, входу, додавання нотаток, перегляду нагадувань тощо.

Для візуального оформлення застосовано CSS3. Він дозволяє реалізувати адаптивний дизайн, стилізацію елементів, використання градієнтів, анімацій і медіа-запитів [19]. У проєкті CSS3 використовується для стилізації форм, кнопок, блоків нотаток, повідомлень і панелей.

Розмітка була значно спрощена завдяки Bootstrap 5 — фреймворку з готовими класами для створення макетів, форм, навігаційних панелей, карток, модальних вікон тощо. Це дозволило швидко реалізувати адаптивний інтерфейс із дотриманням сучасних стандартів доступності [20].

Динамічна поведінка та взаємодія інтерфейсу з користувачем реалізовані за допомогою JavaScript — мови програмування, що працює на стороні клієнта. JavaScript використовується для підтвердження дій, керування чеклістами, збереження локальних нагадувань через localStorage, а також для обробки подій без перезавантаження сторінки [21].

Серверна логіка побудована на основі PHP — скриптової мови, яка широко застосовується для створення динамічних вебсторінок. У застосунку PHP обробляє реєстрацію та авторизацію користувачів, форми, логіку створення й

редагування нотаток, а також зберігання файлів, коментарів, тегів і логів активності [22].

Для зберігання структурованих даних було використано MySQL — реляційну систему керування базами даних. Вона підтримує складні зв’язки між таблицями, ефективне виконання запитів і транзакції [23]. У проєкті використовуються таблиці для користувачів, нотаток, тегів, категорій, чеклістів, вкладень і зворотного зв’язку.

Як інструмент адміністрування бази даних було застосовано phpMyAdmin — вебінтерфейс для взаємодії з MySQL, який надає зручні засоби для створення таблиць, виконання SQL-запитів, перевірки зв’язків і тестування структури [24].

Для локального розгортання та тестування було використано XAMPP — програмний пакет, що включає Apache, PHP, MySQL і phpMyAdmin. Він дозволяє запускати вебзастосунок локально без потреби у підключенні до віддаленого сервера [25].

Додатково для покращення інтерфейсу використовуються Google Fonts — онлайн-бібліотека шрифтів, та Font Awesome — набір іконок, які інтегруються у кнопки, заголовки та панелі, підвищуючи зручність взаємодії [26].

Усі ці технології були підібрані не лише з міркувань популярності чи зручності, а й через їхню практичну придатність до завдань цього проєкту. Завдяки цьому вдалося створити функціональний, масштабований і доступний вебзастосунок, який можна розвивати та адаптувати відповідно до потреб користувачів.

2.2 Use-case діаграма та діаграма послідовності

Побудова функціонального каркасу програмного забезпечення неможлива без попереднього моделювання поведінки системи з урахуванням очікуваної

взаємодії з користувачем. Одним із найбільш ефективних способів представлення цієї взаємодії є графічне зображення ключових сценаріїв, які дозволяють окреслити межі системи, визначити ролі користувачів і можливості, що надаються кожному з них [27].

Змоделюємо діаграму варіантів використання для застосунку, що дозволяє вести особисті нотатки та нагадування. До системи доступ мають два основних типи користувачів — звичайний користувач і адміністратор. Звичайний користувач виконує типові дії: реєстрація, вхід, створення, редагування та видалення нотаток. Крім того, реалізовано можливість налаштування нагадувань, додавання вкладень, тегів, чеклістів, категорій. Передбачено функціонал надсилання зворотного зв'язку, перегляду історії дій, поширення нотаток та додавання подій.

Для представлення залежностей між основними та допоміжними сценаріями використовується механізм включення (включені сценарії). Наприклад, створення нотатки може включати встановлення дати або часу нагадування, додавання категорії, вкладень чи тегів. Це дозволяє формалізувати зв'язки між частинами функціональності, не ускладнюючи загальну структуру діаграми.
На рисунку 2.1 зображено діаграму варіантів використання для користувача, яка ілюструє основні дії, доступні користувачу: реєстрація, вхід у систему, створення нотаток, редагування, видалення, додавання вкладень, поширення нотаток, коментування, додавання тегів та перегляд журналу активності.

На рисунку 2.3 зображено діаграму варіантів використання для адміністратора**,** яка відображає основні функції, що доступні користувачу з розширеними повноваженнями яка має розширені повноваження: перегляд відгуків користувачів, доступ до загального журналу активності, можливість модерації нотаток та управління обліковими записами. Діаграма ілюструє залежності між базовими та допоміжними сценаріями, зокрема такі дії, як блокування або видалення користувача, є похідними від загальної функції адміністрування. Модерація передбачає перегляд змісту нотатки, її оцінку на відповідність умовам ви-

користання, а в разі порушення — можливість її видалення. Аналогічно, у разі порушень з боку користувача передбачені функції перегляду профілю, блокування або повного видалення облікового запису [28].

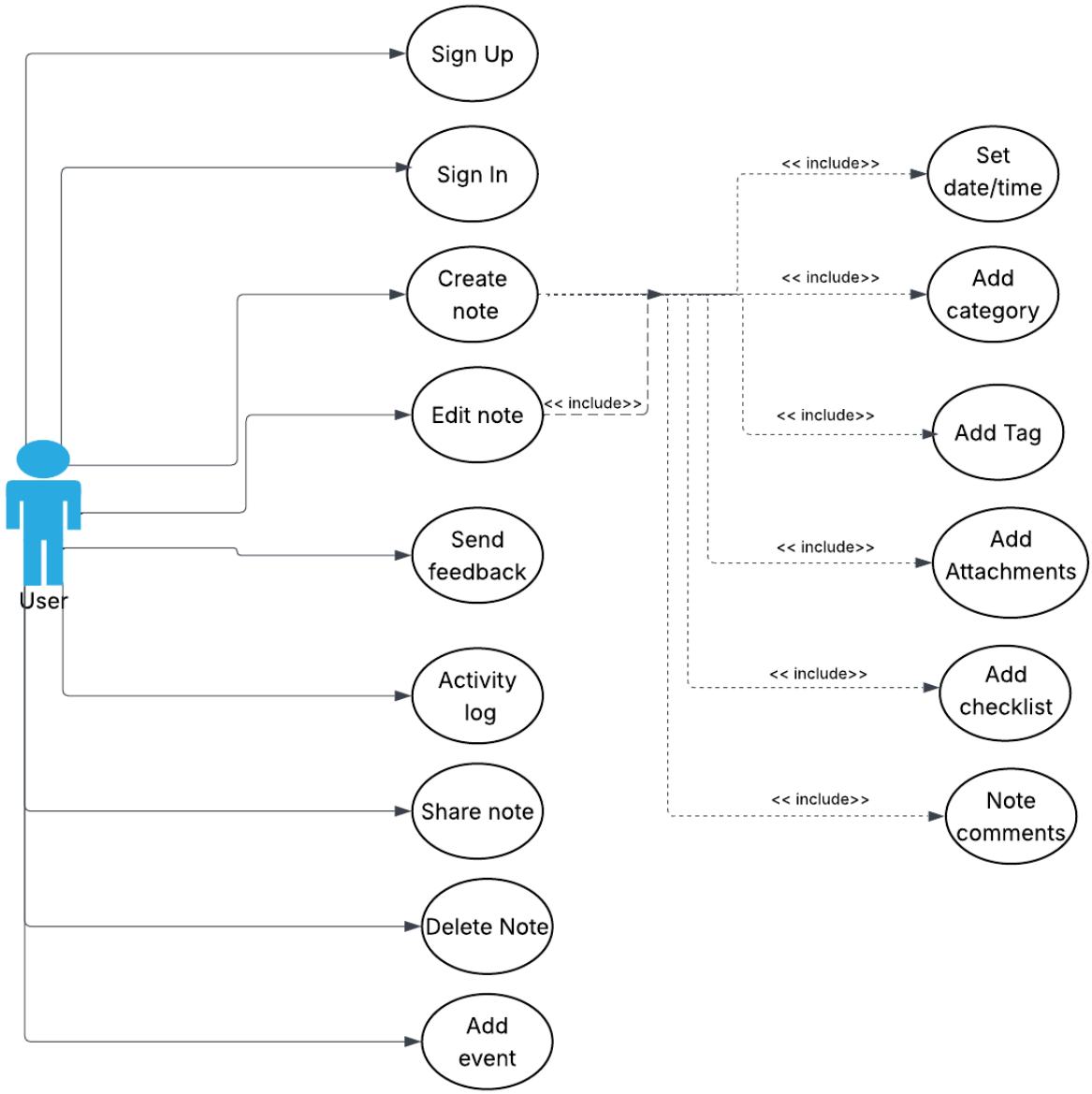


Рисунок 2.1 – Use case діаграма взаємодії користувача з системою

Така структура дозволяє забезпечити контроль за дотриманням правил користування системою та підтримувати її стабільну роботу.

Варіанти використання не описують технічних деталей реалізації — вони визначають лише, які дії доступні в системі і хто може їх виконувати. Для того щоб деталізувати логіку обміну даними між користувачем, інтерфейсом і внут-

рішніми модулями системи у межах конкретного сценарію на рівні окремих компонентів, була створенна діаграма послідовності. Це дозволяє не лише відстежити етапи обробки інформації, а й знайти потенційні вузькі місця в логіці системи ще до її реалізації [29].

На рисунку 2.2 зображено діаграму послідовності, що ілюструє два ключові сценарії взаємодії користувача з вебзастосунком: реєстрацію нового облікового запису та вхід до системи.

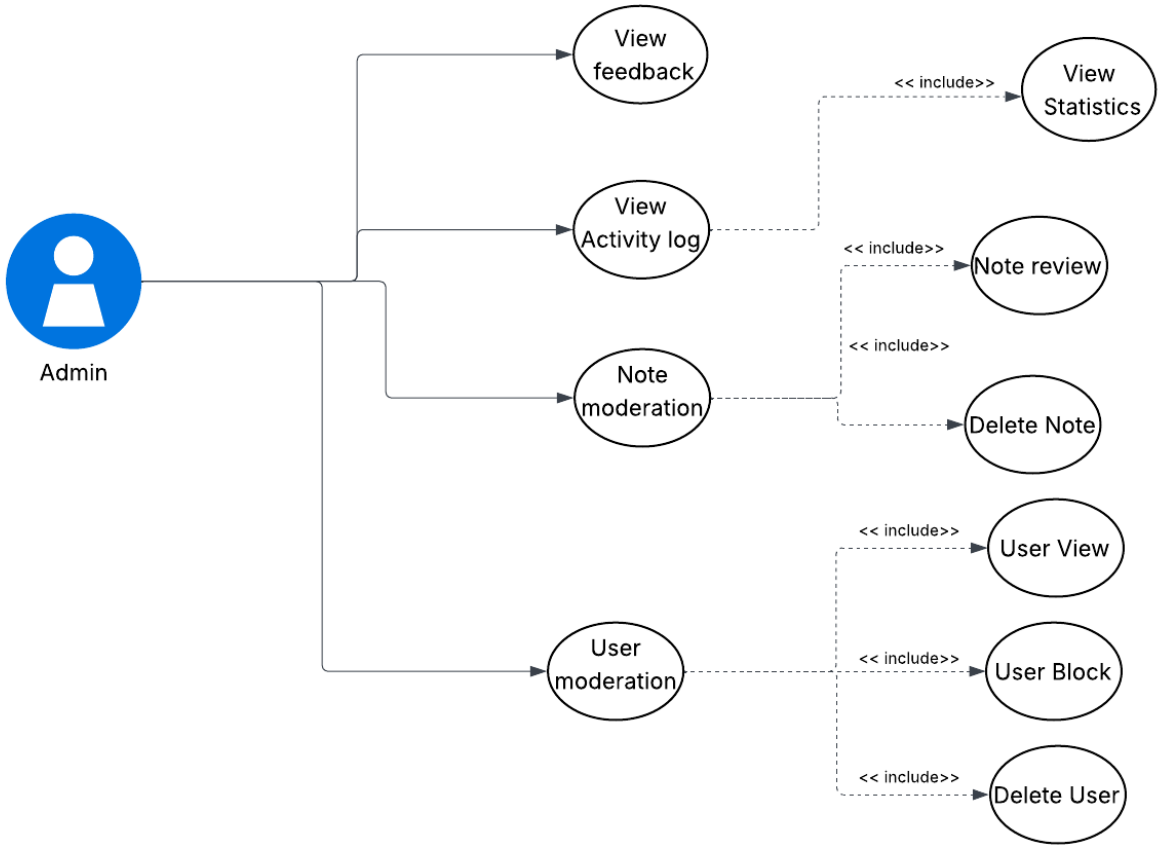


Рисунок 2.2 – Use case діаграма взаємодії адміністратора з системою

У межах реєстрації користувач вводить особисті дані на сторінці реєстрації, після чого система перевіряє унікальність email або імені користувача у базі даних. У разі виявлення дубліката користувач отримує повідомлення про помилку, інакше — його облікові дані зберігаються, після чого відбувається перенаправлення на головну сторінку (dashboard).Сценарій входу передбачає перевірку облікових даних користувача, що надсилаються зі сторінки входу до контро-

лера автентифікації, який, у свою чергу, звертається до бази даних. Якщо дані коректні, користувач успішно авторизується і переходить до системи, в іншому випадку — отримує відповідне повідомлення про помилку. Діаграма відображає послідовність обміну повідомленнями між інтерфейсом, логікою та сховищем даних, а також варіативність перебігу процесу залежно від дій користувача.

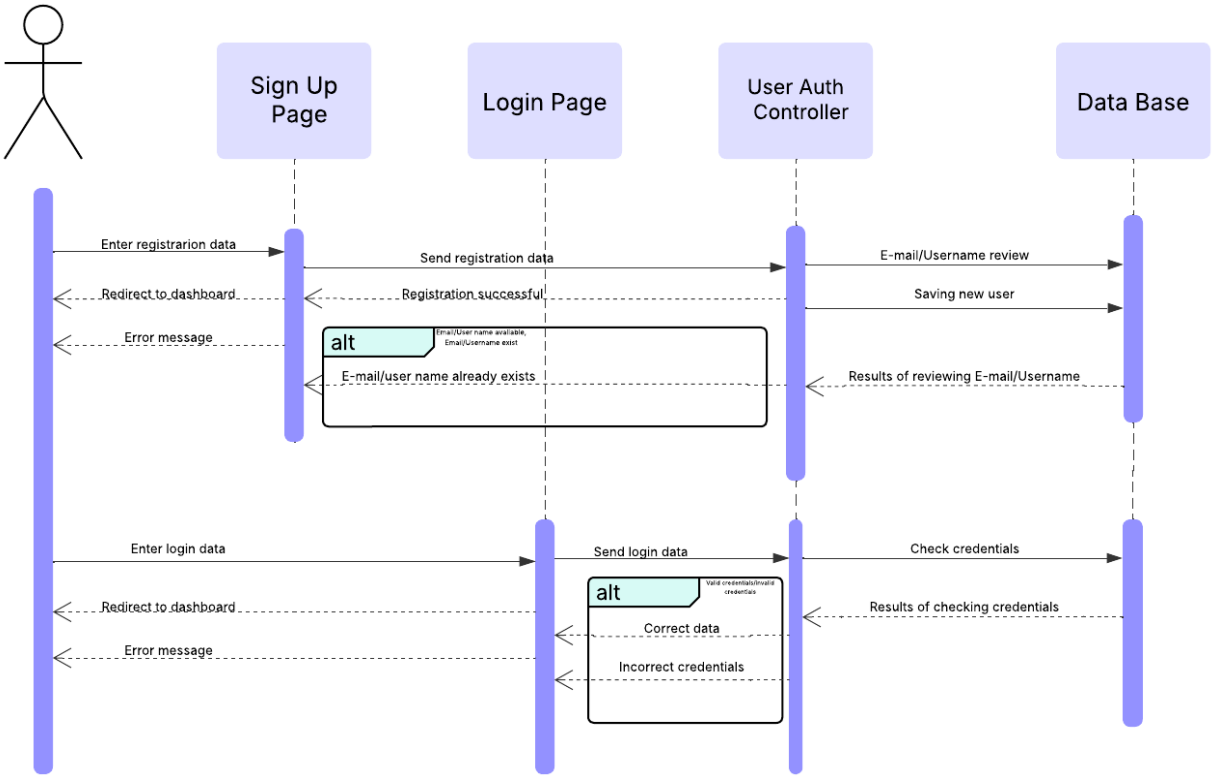


Рисунок 2.3 - Діаграма послідовності для реєстрації та входу користувача

На рисунку 2.4 зображено блок - схему процесу реєстрації користувача. Процес починається з відображення форми, потім йде перевірка введених даних, у випадку якщо одне з полів заповненно не коректно з’являється повідомлення з помилкою. У випадку успішного проходження всіх перевірок відбувається хешування та збереження користувача в таблиці Users (таблиця 2.1).

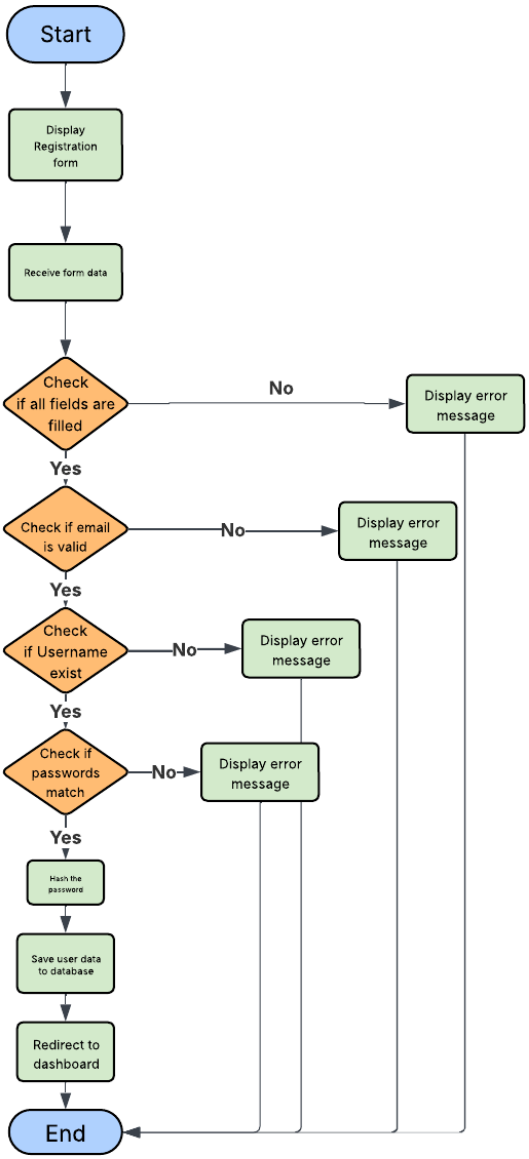


Рисунок 2.4 – Блок – схема реєстрації користувача

Рисунок 2.5 ілюструє блок-схему авторизації користувача в систему. Спочатку відбувається перевірка чи залогінений користувач в системі, якщо так, відображається повідомлення "You already logged in" і автоматичний перехід на головну сторінку. Якщо користувач не авторизований продовжуються наступні перевірки: чи заповненні всі поля форми, чи існує користувач в базі данних та відбувається верифікація паролю, за умови успішного проходження всіх кроків користувач авторизується в системі.

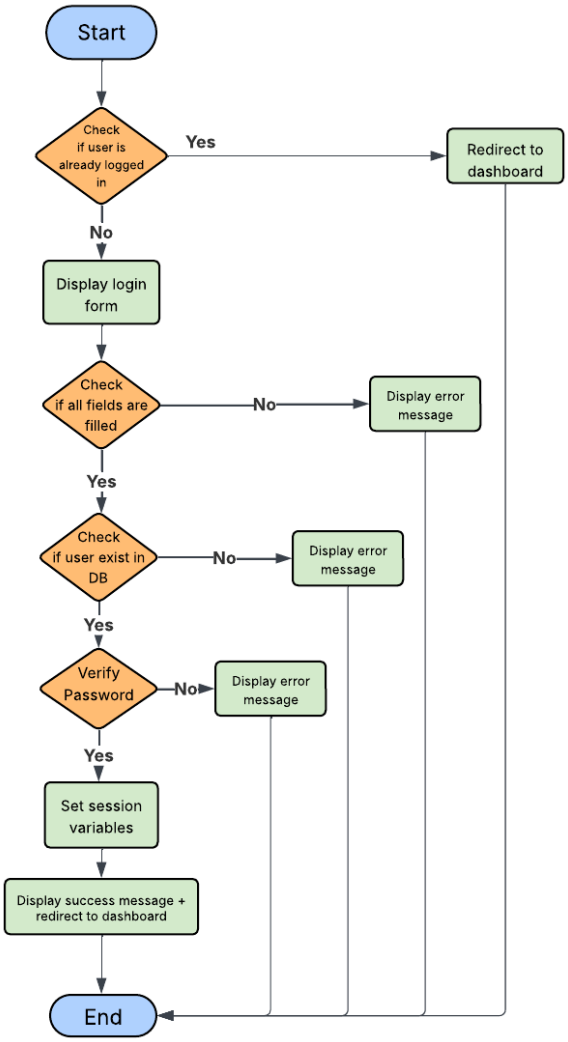


Рисунок 2.5 – Блок – схема авторизації користувача

На рисунку 2.6 зображено діаграму послідовності, що ілюструє процес створення або редагування нотатки користувачем у вебзастосунку. Діаграма демонструє взаємодію між користувачем, формою введення нотатки, контролером нотаток, низкою допоміжних сервісів та базою даних. Після введення даних (заголовка, тексту, нагадування, категорій, тегів, вкладень або чеклісту), форма надсилає їх до контролера, який у свою чергу поетапно звертається до відповідних сервісів. У разі наявності певних елементів — наприклад, дати нагадування або тегів — відповідні дії виконуються умовно, що відображено через блок opt.

Після обробки всіх додаткових даних виконується основна операція збереження: або створюється нова нотатка, або оновлюється вже існуюча. Для

цього в діаграмі використано блок alt, який чітко розділяє ці два сценарії. Після завершення операції система повертає повідомлення про успішне створення чи редагування, або повідомлення про помилку. Таким чином, діаграма охоплює повний цикл роботи з нотаткою — від введення даних до підтвердження результату, що дозволяє наочно продемонструвати логіку роботи застосунку та взаємодію його компонентів

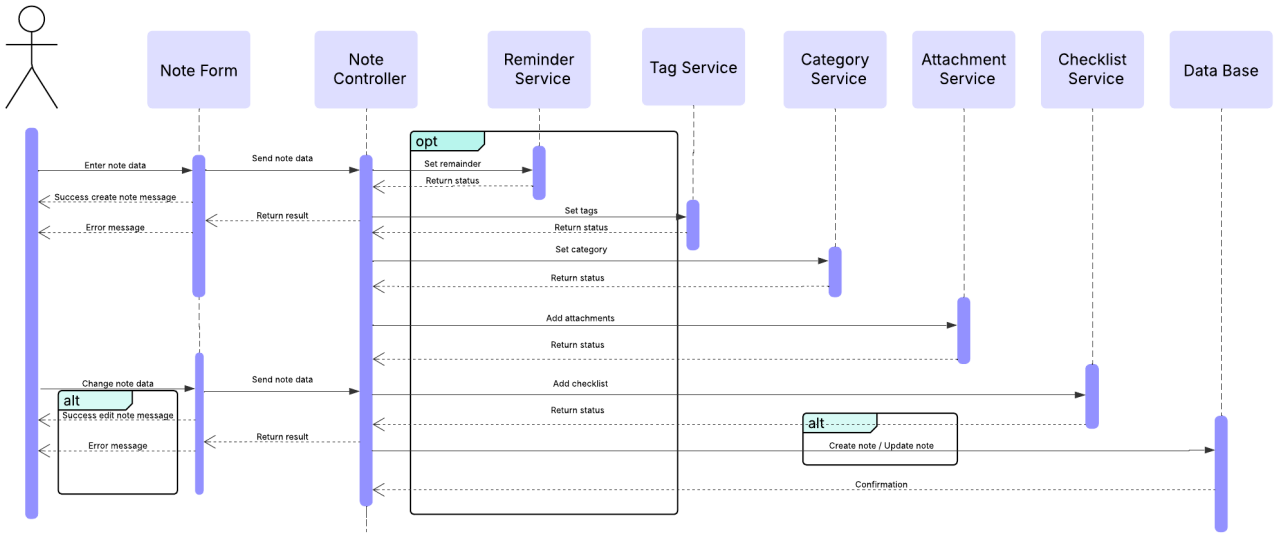


Рисунок 2.6 - Діаграма послідовності для сценаріїв створення нотатки та редагування нотатки

На рисунку 2.7 проілюстровано блок-схему процесу створення нотатки. Після відображення форми та отримання введених даних від користувача система перевіряє, чи заповнено обов’язкові поля — заголовок та вміст. Якщо хоча б одне з них порожнє, виводиться повідомлення про помилку. У разі коректного введення даних відбувається спроба збереження нотатки до бази даних. Якщо збереження проходить успішно — користувач перенаправляється на сторінку підтвердження; у протилежному випадку — повертається до форми додавання нотатки з відповідним повідомленням.

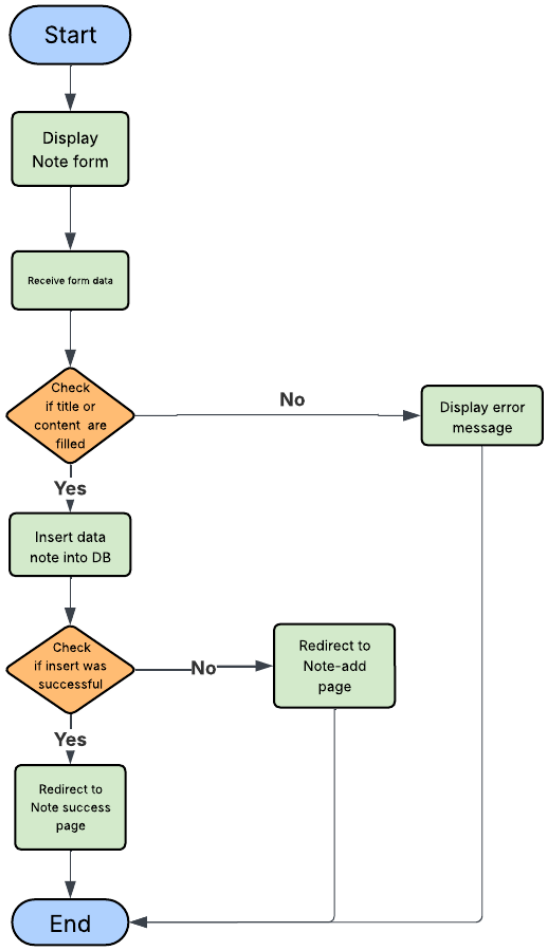


Рисунок 2.7 – Блок – схема процесу створення нотатки

Побудова діаграм варіантів використання та послідовності, а також блок-схем процесів дає змогу створити загальну концептуальну модель поведінки системи. Це дозволяє краще розуміти потреби користувача, оптимізувати структуру застосунку, передбачити можливі труднощі в реалізації та уніфікувати процес розробки. Крім того, така візуалізація є важливою складовою документації, яка використовується як розробниками, так і тестувальниками [30].

Важливо також зазначити, що діаграми використання дають змогу швидко ідентифікувати зв’язки між функціональними модулями, оцінити вплив змін у коді на інші частини системи, та забезпечити цілісність і логічну узгодженість усіх компонентів. Діаграми послідовності, своєю чергою, забезпечують прозорість обміну даними та полегшують виявлення помилок або неефективної взаємодії між модулями [31].

2.3 Архітектура системи

Архітектура системи була розроблена з урахуванням принципів масштабованості, розширюваності та чіткої модульності, що дозволяє ефективно управляти окремими компонентами системи, зберігаючи при цьому цілісність і стабільність функціонування [32]. У цьому контексті найбільш відповідною є модульна архітектура, яка дозволяє ефективно розділяти обов'язки між частинами системи та організовувати її компоненти в логічні блоки.

Архітектура умовно поділяється на чотири основні рівні: інтерфейс користувача, бізнес-логіка, сховище даних та сервіси (рис. 2.8) Кожен з них виконує окрему роль, забезпечуючи взаємозв'язок між клієнтською частиною, логікою застосунку та базою даних.

Користувацький інтерфейс - візуальна частина застосунку забезпечує доступ до основних функцій: реєстрація, логін, створення та редагування нотаток, управління нагадуваннями, категоріями, тегами, вкладеннями, перегляд спільних нотаток, додавання коментарів, використання чеклістів. Всі сторінки мають адаптивний дизайн і побудовані за принципами інтуїтивної навігації. Серед основних реалізованих сторінок:

- сторінка аунтефікації: Вхід в системуфо, реєстрація нових користувачів.
- головна сторінка користувача: форма додавання швидкої нотатки, нещодавно додані нотатки, лист з нотатками та прикріпленні нотатки;
- профіль користувача: особиста інформація;
- сторінка спільного доступу;
- сторінка календаря з нагадуваннями;
- журнал активності користувача;
- сторінки додавання та редагування нотаток;
- форма зворотного зв'язку.

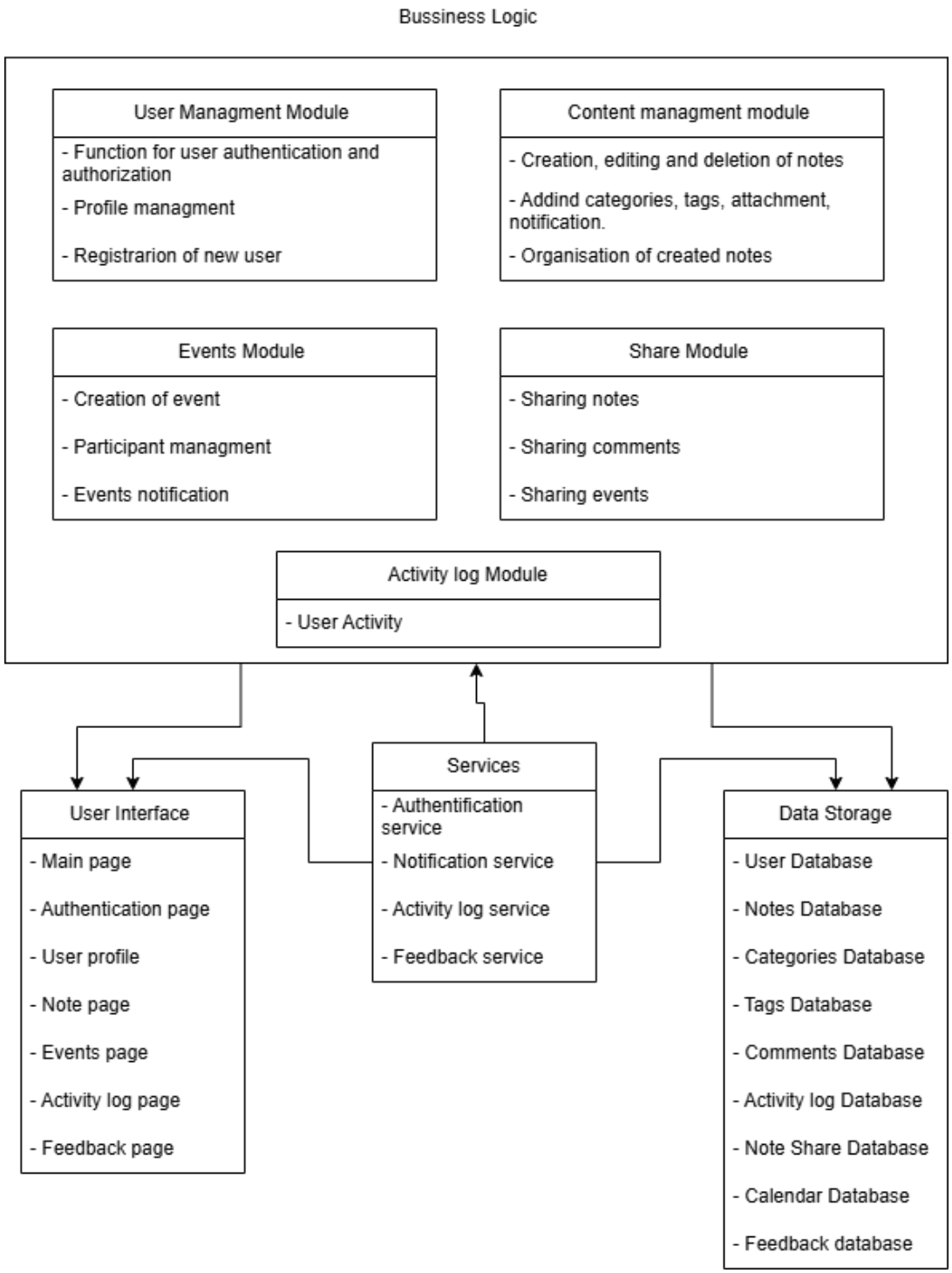


Рисунок. 2.8 – Схема архітектури

Бізнес-логіка визначає, які дії виконуються при взаємодії з користувачем та які дані обробляються. У вебзастосунку Notespace вона реалізована мовою PHP із дотриманням принципу MVC (Model-View-Controller), що дозволяє чіт-

ко розмежувати представлення, логіку та роботу з даними. Основні функціональні модулі бізнес-логіки включають:

– модуль аутентифікації та управління користувачами: забезпечує авторизацію, реєстрацію, валідацію облікових даних та контроль доступу до захищених частин системи;

– модуль управління нотатками: дозволяє створювати, редагувати, видаляти нотатки, додавати заголовки, текстовий вміст, категорії, теги, вкладення та чеклісти;

– модуль нагадувань: забезпечує обробку та збереження дати й часу нагадування, а також перевірку їх на клієнтському рівні через JavaScript;

– модуль коментарів: відповідає за додавання, збереження й виведення коментарів до нотаток, у тому числі спільних;

– модуль спільного доступу: реалізує функцію поширення нотаток між користувачами за допомогою відповідної таблиці в базі даних;

– модуль логування активності: фіксує дії користувачів у спеціальному журналі подій (створення, редагування, видалення, поширення нотаток тощо);

– модуль взаємодії з категоріями та тегами: дозволяє прив’язати відповідні метадані до нотаток і забезпечити фільтрацію та пошук;

– модуль вкладень: відповідає за завантаження, збереження та відображення прикріплених файлів у нотатках.

Сховище даних для роботи вебзастосунку було реалізовано за допомогою реляційної базу даних MySQL, яка забезпечує високу продуктивність, масштабованість і підтримку складних SQL-запитів. Сервіси у проєкті виконують допоміжні функції, які підтримують основну логіку застосунку та покращують взаємодію з користувачем. До таких сервісів належать:

– сервіс автентифікації, що відповідає за валідацію облікових даних і забезпечення безпеки доступу до ресурсу;

– сервіс сповіщень, який інформує користувача про майбутні події або нагадування відповідно до заданого часу;

– сервіс логування, що фіксує всі ключові дії користувача в системі у спеціальному журналі активності для подальшого аналізу або моніторингу;

– сервіс роботи з файлами, який дозволяє додавати вкладення до нотаток і зберігати їх у безпечному місці;

– сервіс зворотного зв’язку, що дає можливість користувачам залишати свої пропозиції чи повідомлення для розробників.

Отже, як видно з рисунку 2.8, архітектура системи побудована на основі модульного підходу, що забезпечує гнучкість, масштабованість і легкість супроводу проєкту. Завдяки чіткому розмежуванню між інтерфейсом користувача, бізнес-логікою, сховищем даних і допоміжними сервісами досягається стабільна взаємодія всіх компонентів. Така структура дозволяє ефективно керувати даними, забезпечує безпечну роботу з нотатками та нагадуваннями, а також створює позитивний користувацький досвід

2.4 Проектування бази даних

Проєктування бази даних є ключовим етапом розробки веб-застосунку для ведення особистих нотаток і нагадувань. Раціонально сформована структура забезпечує ефективне зберігання, організацію, оновлення та взаємозв’язок даних, що сприяє стабільній роботі системи, її масштабованості та підтримці цілісності інформації.

База даних включає 13 таблиць, які охоплюють основні функціональні сутності системи, зокрема:

Користувачі (users) - дана таблиця містить в собі облікові записи зареєстрованих користувачів. Зберігає логін, пароль, email та дату створення акаунта.

Таблиця users є базовою для функціонування системи, оскільки зберігає ключову інформацію про користувачів, необхідну для аутентифікації, авторизації та розмежування прав доступу. Завдяки структурованості цієї таблиці за-

безпечується зручне керування як звичайними користувачами, так і адміністраторами.

Таблиця 2.1. – Опис таблиці Users

Поле	Розмір поля	Тип даних	Значення
id	4	Int	унікальний ідентифікатор користувача (PRIMARY KEY)
id	4	Int	унікальний ідентифікатор користувача (PRIMARY KEY)
username	100	Varchar	логін для входу в систему
password	255	Varchar	хешований пароль
email	255	Varchar	електронна пошта
created_at	8	Datetime	дата реєстрації, Формат дати YYYY-MM-DD hh:mm:ss
is_admin	1	Tinyint	статус користувача: 0 — звичайний, 1 — адміністратор (за замовчуванням 0)

Нотатки (notes) - основна таблиця, в якій зберігаються особисті записи користувача. Містить текстовий вміст, заголовок, дату нагадування, категорію, пріоритет і статус виконання.

Таблиця notes має повноцінну структуру для зберігання всіх необхідних атрибутів нотатки. Вона пов’язана з користувачем через зовнішній ключ user_id і дає змогу зберігати як заголовок (title), так і повний вміст нотатки (content). Поле remind_at відповідає за встановлення дати й часу нагадування, priority дозволяє користувачеві встановити важливість запису, а category_id забезпечує зв’язок із таблицею категорій. Для реалізації логіки архівації та видалення передбачено відповідні прапорці (is_archived, deleted_at), а is_done визначає, чи завершено виконання завдання. Дати створення та останнього редагування (created_at, updated_at) дають змогу відстежувати зміни у часі, що є важливим для реалізації історії змін або сортування.

Таблиця 2.2. – Опис таблиці Notes

Поле	Розмір поля	Тип даних	Значення
id	4	Int	Унікальний ідентифікатор нотатки
user_id	4	Int	Зовнішній ключ на користувача (таблиця users)
title	255	Varchar	Заголовок нотатки
content	до 65 535	Text	Основний вміст нотатки
remind_at	8	Datetime	Дата й час нагадування
priority	1	Enum('low','medium','high')	Пріоритет нотатки
category_id	4	Int	Зовнішній ключ на категорію (таблиця categories)
created_at	8	Datetime	Дата створення нотатки
updated_at	8	Datetime	Дата останнього редагування
deleted_at	8	Datetime	Мітка про видалення
is_done	1	Tinyint	Позначення виконаності нотатки
is_archived	1	Tinyint	Позначення, чи нотатка архівована

Категорії (categories) - таблиця дозволяє групувати нотатки за тематиками (наприклад: робота, особисте, фінанси). Категорії можуть бути унікальними для кожного користувача.

Поле id є первинним ключем і забезпечує унікальність кожної категорії. Поле user_id пов’язує категорію з конкретним користувачем. Поле name містить назву категорії та використовується для її виводу у формі створення або редагування нотатки. Це дозволяє реалізувати логічну фільтрацію нотаток за тематиками й забезпечити персоналізоване впорядкування даних для кожного користувача.

Таблиця 2.3 – Опис таблиці Categories

Поле	Розмір поля	Тип даних	Значення
id	4	Int	Унікальний ідентифікатор категорії
user_id	4	Int	Зовнішній ключ на користувача (таблиця users) Вказує хто створив категорію
name	100	Varchar	назва категорії

Теги (tags) таблиця містить мітки, які додаються до нотаток для гнучкої фільтрації або пошуку. Один тег може бути прив'язаний до багатьох нотаток.

2.4 – Опис таблиці Tags

Поле	Розмір поля	Тип даних	Значення
id	4	Int	Унікальний ідентифікатор тегу
name	50	Varchar	назва тегу
user_id	4	Int	Зовнішній ключ на користувача (таблиця users) Вказує що тег належить певному користувачу.

Таблиця tags служить для зберігання міток, які дозволяють класифікувати нотатки за вмістом, темою чи іншим критерієм. Поле id є унікальним ідентифікатором кожного тега. Назва тега зберігається в полі name, яке має тип varchar і обмежене довжиною у 50 символів. Поле user_id виступає як зовнішній ключ і вказує, якому користувачу належить тег. Завдяки цій структурі система підтримує багато до багатьох зв'язок між тегами та нотатками, що забезпечує гнучку фільтрацію, пошук та персоналізоване використання міток у межах профілю кожного користувача.

Файли (attachments): таблиця містить файли (зображення, документи), які користувач прикріплює до своїх нотаток. Файли зберігаються як шлях у файловій системі.

Таблиця attachments реалізує механізм зберігання файлів, які користувач прикріплює до нотаток у вигляді зображень або документів. Поле id є унікальним ідентифікатором вкладення. Через поле note_id, що виступає зовнішнім ключем, забезпечується зв'язок з відповідною нотаткою. Шлях до файлу збері-

гається в полі file_path, що дозволяє системі звертатися до файлів, розміщених у файловій структурі проєкту. Додатково, поле remind_at може фіксувати дату і час додавання вкладення, що особливо корисно для автоматичних нагадувань або організації матеріалів у хронологічному порядку. Така структура дає змогу зручно інтегрувати мультимедійний контент у нотатки користувача.

2.5 – Опис таблиці Attachments

Поле	**Розмір поля**	**Тип даних**	**Значення**
id	4	Int	Унікальний ідентифікатор вкладення
note_id	4	Int	Зовнішній ключ на користувача (таблиця notes)Вказує до якої нотатки прикріплений файл
file_path	255	Varchar	Шлях до прикріпленного файлу.
remind_at	8	Datetime	Дата й час завантаження файлу

Календар подій (calendar_events) - таблиця зберігає події користувача, які були створенні в календарі.

2.6 – Опис таблиці Calendar_events

Поле	**Розмір поля**	**Тип даних**	**Значення**
id	4	Int	Унікальний ідентифікатор події
user_id	4	Int	Зовнішній ключ на користувача, який створив подію (таблиця users)
title	255	Varchar	Назва події
description	до 65 535	Text	Опис або додаткова інформація про подію
start	8	Datetime	Дата і час початку
end	8	Datetime	Дата і час завершення
color	20	Varchar	Колір відображення у події
created_at	8	Datetime	Дата створення запису події

Таблиця calendar_events відповідає за зберігання подій, створених користувачем у календарі. Вона містить усю необхідну інформацію для відображення подій у часовому контексті. Поле id є унікальним ідентифікатором кожної події, а user_id вказує на користувача, якому вона належить. Назва події зберігається в полі title, а докладний опис — у полі description. Для відображення часових меж події передбачені поля start і end, які зберігають дату та час початку й завершення відповідно. Поле color дозволяє візуально маркувати події в інтерфейсі, підвищуючи зручність користування календарем. Дата створення події фіксується у полі created_at, що дозволяє впорядковувати або відстежувати хронологію дій користувача. Така структура таблиці забезпечує гнучке управління персональними подіями в межах вебзастосунку.

Чеклісти (checklist_items) - таблиця використовується для збереження пунктів списку справ, які додаються до конкретної нотатки.

Таблиця 2.7 – Опис таблиці Checklist_items

Поле	**Розмір поля**	**Тип даних**	**Значення**
id	4	Int	Унікальний ідентифікатор пункту в чеклісті
note_id	4	Int	Зовнішній ключ на користувача, який створив подію (таблиця notes)
content	255	Varchar	Текст пункту (елементу) списку справ
is_checked	1	Tinyint	Ознака виконаності

Таблиця checklist_items призначена для зберігання елементів списків справ, що додаються до певної нотатки користувача. Кожен запис у цій таблиці відповідає окремому пункту чекліста. Поле id є унікальним ідентифікатором елемента, а note_id — зовнішнім ключем, що вказує на нотатку, до якої він належить. Текстовий зміст кожного пункту зберігається у полі content, яке обмежене 255 символами. Поле is_checked вказує на стан виконання пункту: 0 — не виконано, 1 — виконано. Така структура дозволяє реалізувати зручний функці-

онал списку завдань безпосередньо в межах нотатки, з можливістю позначати виконані пункти.

Коментарі до нотаток (note_comments) - таблиця використовується для збереження коментарів, які користувачі залишають до власних нотаток.

Таблиця 2.8 – Опис таблиці Note_comments

Поле	**Розмір поля**	**Тип даних**	**Значення**
id	4	Int	Унікальний ідентифікатор коментаря
user_id	4	Int	Зовнішній ключ на користувача (таблиця users) Користувач, який залишив коментар.
note_id	4	Int	Зовнішній ключ на нотатку (таблиця notes). Ідентифікатор нотатки, до якої належить коментар.
content	до 65 535	Text	Текстовий вміст коментаря
created_at	8	Datetime	Дата й час створення коментаря.

Таблиця note_comments використовується для зберігання коментарів, які користувачі залишають до власних або поширених нотаток. Кожен запис містить унікальний ідентифікатор коментаря (id), посилання на автора (user_id, зовнішній ключ на таблицю users) та ідентифікатор нотатки, до якої цей коментар належить (note_id, зовнішній ключ на таблицю notes). Текст самого коментаря зберігається в полі content, а час його створення — у полі created_at. Така структура дозволяє гнучко реалізувати функціональність коментування й упорядковувати повідомлення за хронологією.

Закріплені нотатки (note_pins) - таблиця містить інформацію про нотатки, які були закріплені користувачем для швидкого доступу.

Таблиця note_pins використовується для реалізації функціоналу швидкого доступу до важливих нотаток, які були вручну закріплені користувачем. Як зазначено у вступі перед таблицею, вона зберігає зв’язок між користувачем і нотаткою, що дозволяє системі визначити, які записи мають бути виведені в пріо-

ритетному порядку. Поля user_id та note_id виступають зовнішніми ключами, що вказують відповідно на користувача та відповідну нотатку, а поле pinned_at фіксує точний момент закріплення. Завдяки цій структурі, система може забезпечити інтуїтивний доступ до закріпленого контенту без потреби в додатковій обробці або пошуку.

Таблиця 2.9 – Опис таблиці Note_pins

Поле	Розмір поля	Тип даних	Значення
user_id	4	Int	Зовнішній ключ. Ідентифікатор користувача, який закріпив нотатку.
note_id	4	Int	Зовнішній ключ. Ідентифікатор закріпленої нотатки.
pinned_at	8	Datetime	Дата й час коли нотатку було закріплено.

Спільні нотатки (shared_notes) - таблиця зберігає інформацію про реалізацію спільного доступу до нотаток між користувачами.

Таблиця 2.10 – Опис таблиці Shared_notes

Поле	Розмір поля	Тип даних	Значення
id	4	Int	Унікальний ідентифікатор запису про спільний доступ
note_id	4	Int	Зовнішній ключ на нотатку (таблиця notes). Нотатка, яку пошерено
shared_with_user_id	4	Int	Зовнішній ключ. (таблиця users). Ідентифікатор користувача з яким поширено нотатку.
can_edit	1	Tinyint	Права доступу
shared_at	8	Datetime	Дата й час надання спільного доступу

Таблиця shared_notes призначена для збереження інформації про спільний доступ до нотаток між користувачами. Як зазначено у вступі перед таблицею, вона дозволяє реалізувати механізм поширення окремих нотаток іншим зареєстрованим користувачам системи. Поле note_id вказує, яку саме нотатку поширено, а shared_with_user_id ви-

значає користувача, який отримав доступ. Додатково поле can_edit встановлює права доступу (тільки для перегляду або з можливістю редагування), a shared_at фіксує дату та час поширення. Така структура забезпечує гнучке керування спільним використанням контенту та підтримує контроль за рівнем доступу до кожної нотатки.

Журнал дій (activity_log) таблиця використовується для фіксації ключових дії користувача: створення, редагування, видалення нотаток тощо.

Таблиця 2.11 – Опис таблиці Activity_log

Поле	**Розмір поля**	**Тип даних**	**Значення**
id	4	Int	Унікальний ідентифікатор запису активності
user_id	4	Int	Зовнішній ключ. Ідентифікатор користувача,виконував дію
action	100	Varchar	Опис дії користувача
created_at	8	Datetime	Дата й час коли було виконано дію

Таблиця activity_log містить чотири основні поля, кожне з яких виконує важливу роль у збереженні інформації про дії користувачів. Поле id є унікальним ідентифікатором кожного запису активності. user_id — зовнішній ключ, що вказує, який саме користувач виконав дію. У полі action зберігається текстовий опис події (наприклад, створення або редагування нотатки), а поле created_at фіксує точний час, коли ця дія була здійснена. Така структура дає змогу зберігати хронологічну історію змін і контролювати поведінку користувачів у системі.

Відгуки користувачів (feedback): Містить повідомлення, які користувачі залишають як зворотний зв’язок. Може використовуватись для покращення функціональності системи. Таблиця feedback складається з чотирьох полів, що забезпечують збереження інформації про зворотний зв’язок від користувачів. Таблиця

2.12 – Опис таблиці Feedback

Поле	Розмір поля	Тип даних	Значення
id	4	Int	Унікальний ідентифікатор відгуку
user_id	4	Int	Зовнішній ключ на користувача (таблиця users) Користувач, який залишив відгук.
message	до 65 535	Text	Текстовий вміст повідомлення або відгуку
submitted_at	8	Datetime	Дата й час надсилання.

Поле id є унікальним ідентифікатором кожного повідомлення. user_id — це зовнішній ключ, що вказує, який саме користувач залишив відгук. У полі message зберігається сам текст відгуку або пропозиції, тоді як submitted_at фіксує дату та час надсилання повідомлення. Така структура дозволяє легко ідентифікувати автора, зміст та час надходження кожного відгуку.

Таблиця Note_tags- таблиця реалізує проміжну таблицю (junction table), яка дозволяє кожній нотатці (note) мати кілька тегів (tag) і кожному тегу бути прив’язаним до багатьох нотаток

Таблиця 2.13 – Опис таблиці Note_tags

Поле	Розмір поля	Тип даних	Значення
note_id	4	Int	Зовнішній ключ, який посилається на note_id
tag_id	4	Int	Зовнішній ключ, який посилається на tag_id

Таблиця note_tags складається з двох полів — note_id та tag_id, обидва типу Int. Вони виконують роль зовнішніх ключів і забезпечують встановлення зв’язку між нотатками та тегами. Така структура дозволяє реалізувати зв’язок "багато-до-багатьох", коли одна нотатка може мати кілька тегів, а один тег — застосовуватись до кількох нотаток. Це забезпечує гнучку систему класифікації та фільтрації даних у додатку.

На рисунку 2.9 представлена схема зв’язків між таблицями бази даних, що використовуються у вебзастосунку для ведення особистих нотаток. Діаграма демонструє логічну структуру системи з урахуванням основних сутностей

(користувачі, нотатки, категорії, теги тощо) та їхніх взаємозв'язків, зокрема зв'язків типу "один-до-багатьох" і "багато-до-багатьох". Така організація дозволяє ефективно зберігати, обробляти й пов'язувати інформацію між різними модулями системи.

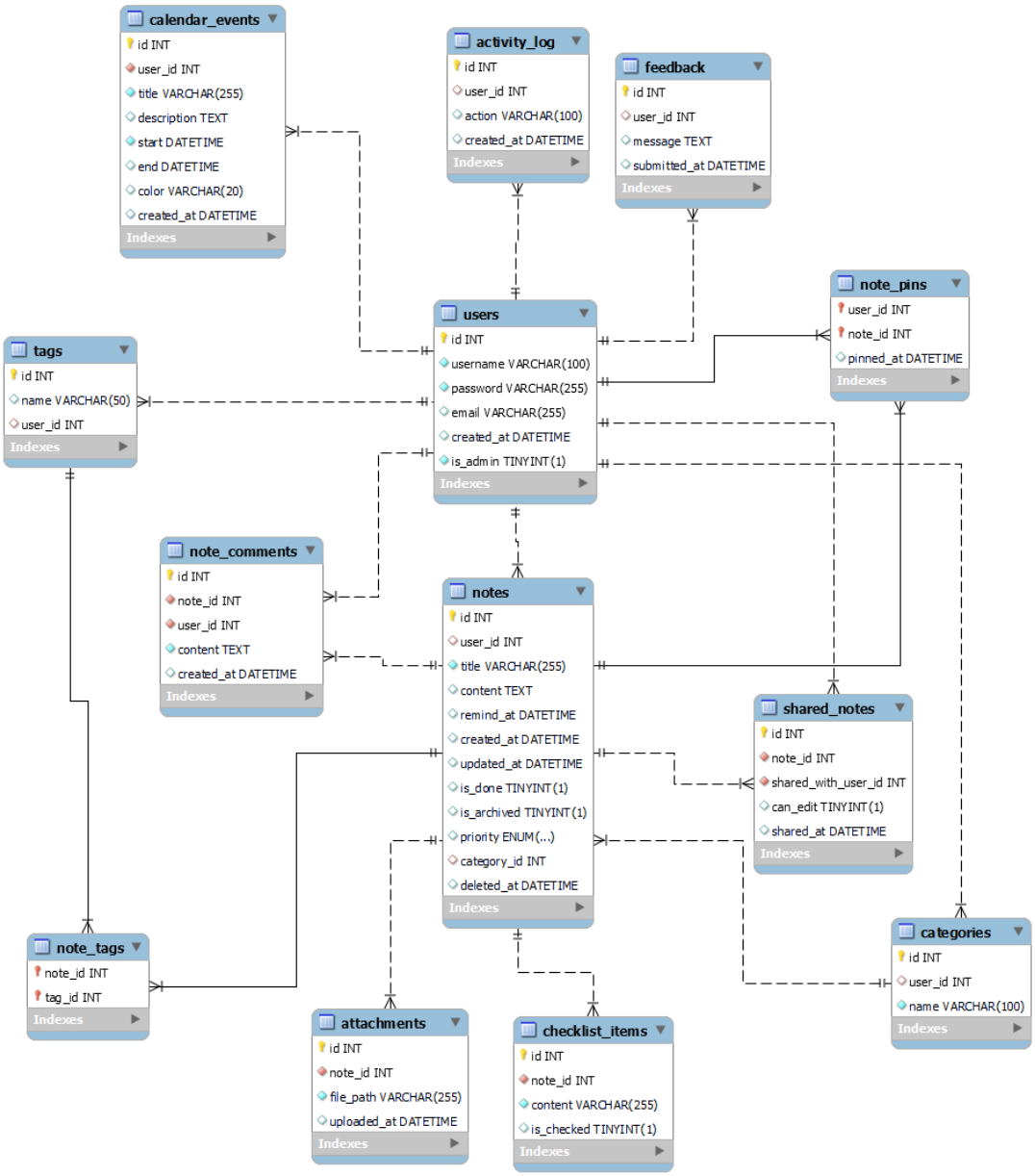


Рисунок. 2.9**.** – Структура бази даних веб-застосунку для особистих нотаток і нагадувань.

База даних вебзастосунку реалізована у вигляді реляційної структури з чітко визначеними зв'язками між таблицями. Центральним елементом є таблиця

нотаток, яка пов’язана з більшістю інших таблиць через зовнішні ключі. Зокрема, кожна нотатка пов’язується з конкретним користувачем через поле user_id, а також може належати до певної категорії завдяки зв’язку з таблицею categories. Додаткові можливості нотаток реалізуються через зв’язок із таблицею вкладень (attachments), пунктів чеклісту (checklist_items) та коментарів (note_comments), які зберігають додаткові елементи, прив’язані до конкретної нотатки. Також для реалізації класифікації за тегами використовується зв’язок багато до багатьох із таблицею tags, що опосередковується проміжною таблицею note_tags. Спільний доступ до нотаток реалізується через таблицю shared_notes, яка дозволяє користувачам ділитися своїми нотатками з іншими, з можливістю керування правами доступу. Закріплення нотаток реалізується через таблицю note_pins, яка дає змогу зберігати їх у пріоритетному вигляді. Крім того, дії, пов’язані з нотатками, фіксуються в журналі активності (activity_log), що забезпечує простежуваність подій у системі.

2.5 Моделювання інтерфейсу користувача та UI/UX дизайн

Інтерфейс користувача (UI) та користувацький досвід (UX) є ключовими компонентами успішного вебзастосунку, особливо у сфері ведення особистих нотаток та нагадувань. Основною метою UI/UX дизайну в рамках розробки системи Notespace стало створення інтуїтивно зрозумілого, візуально привабливого та функціонального середовища для взаємодії користувача з платформою [33]. Основні принципи проєктування інтерфейсу базуються на простоті, логічності навігації, адаптивності й доступності, що дозволяє ефективно використовувати систему.

Інтерфейс було розроблено відповідно до сучасних стандартів — із використанням мінімалістичного підходу, чистих форм і зрозумілих елементів управління. Користувацький досвід забезпечується через швидку реакцію інтерфейсу, зрозумілу ієрархію сторінок, адаптивне відображення контенту [34].

Дизайн усіх сторінок — від авторизації та реєстрації до панелі нотаток і створення нагадувань — був виконаний з урахуванням інтуїтивності. Сторінки побудовані за принципом двоколонкового або одноколонкового макету, що забезпечує зручність взаємодії з формами.

Таким чином, UI/UX дизайн у даному вебзастосунку не лише виконує роль візуального представлення функціоналу, а й створює повноцінний досвід користувача, який сприяє легкому зануренню в роботу з системою, знижує поріг входу для нових користувачів і стимулює регулярне використання платформи [35]. У подальших підрозділах подано детальний опис основних сторінок застосунку та дизайнерських рішень, що були реалізовані на практиці.

З метою візуалізації інтерфейсу були створені макети основних сторінок веб-застосунку, що наведено на рисунках нижче.

Сторінки реєстрації (рис. 2.10) та входу (рис. 2.11) є першим етапом взаємодії користувача з платформою. Вони мають двоколонкову структуру: зліва – мотиваційна панель із брендуванням, справа – форма входу або реєстрації. Для користувачів передбачено простий механізм відновлення паролю та швидкий перехід між формами.

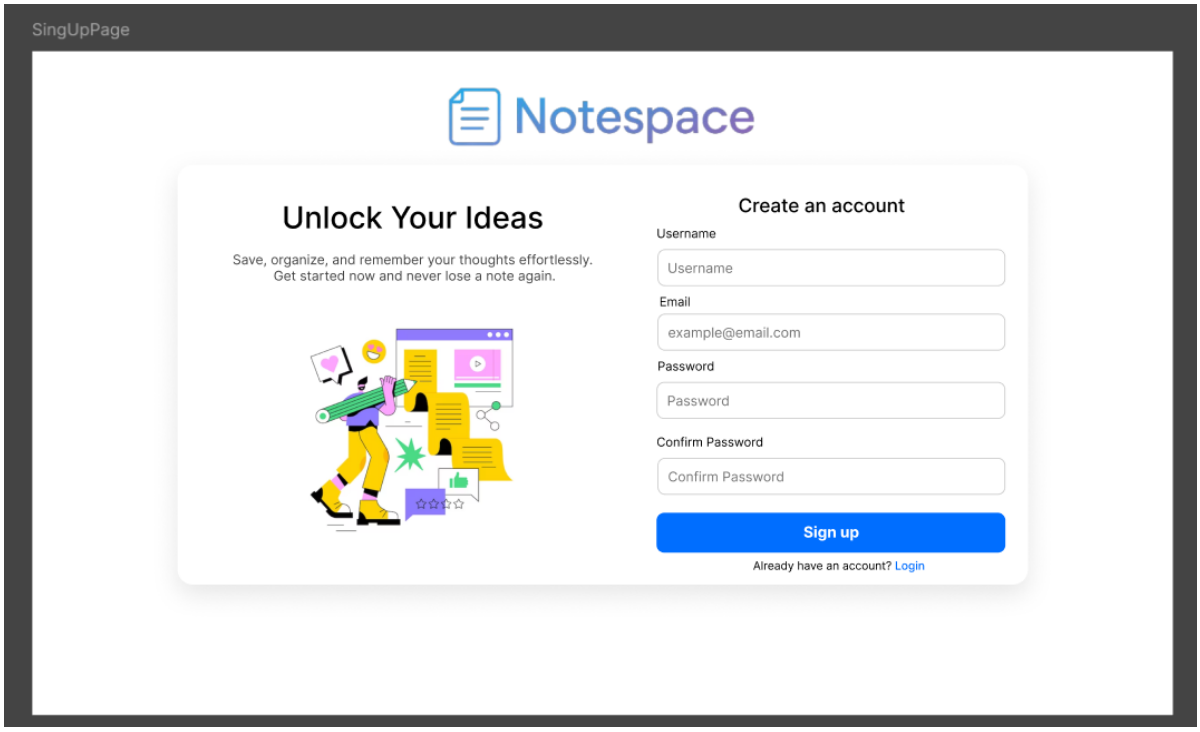


Рисунок. 2.10 – Макети сторінки реєстрації

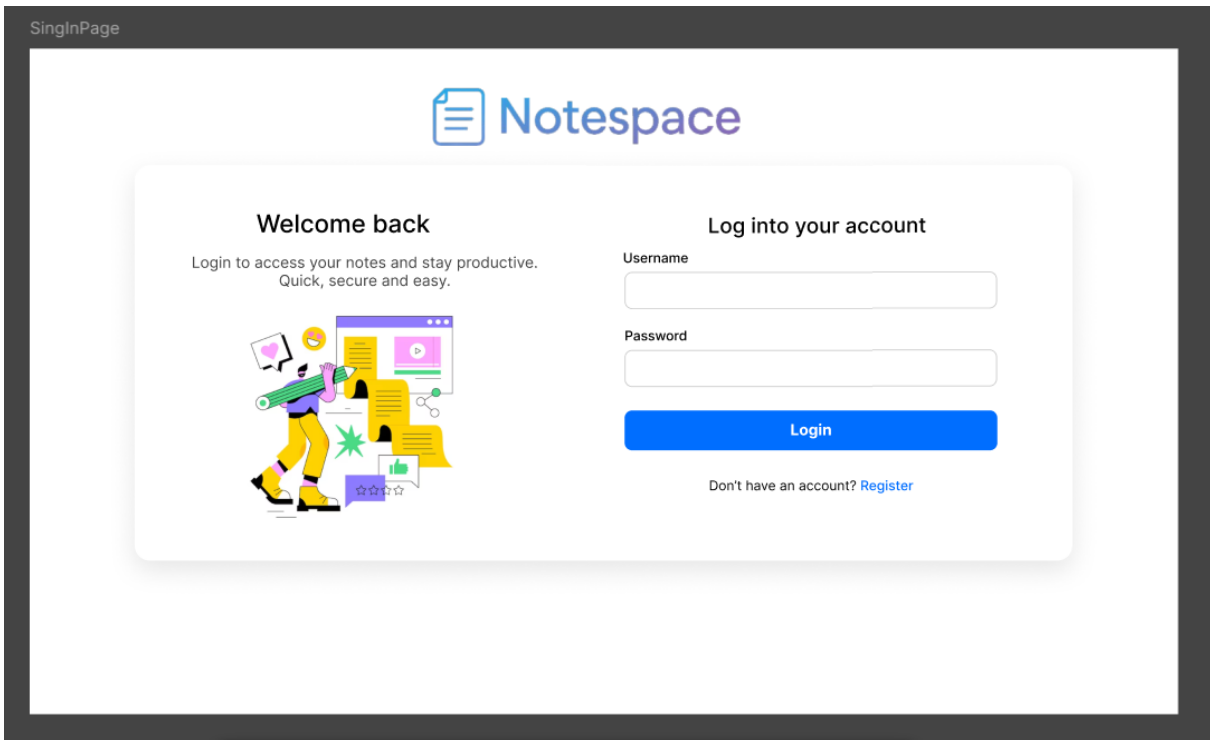


Рисунок. 2.11 – Макети сторінки входу

Головна сторінка (рис. 2.12) — це центральна панель взаємодії користувача з вебзастосунком Notespace, де зосереджено основні функції для роботи з особистими нотатками. У верхній частині розміщено навігаційну панель із логотипом, кнопкою переходу до календаря подій, а також випадаючим меню користувача з пунктами профілю, журналу активності, зворотного зв'язку та виходу із системи. Ліва частина інтерфейсу містить функціональну бічну панель, яка дозволяє швидко створити нову нотатку, додати подію або перейти до списку нотаток, якими поділились інші користувачі. Також тут реалізовано зручний пошук нотаток і відображення повного списку створених користувачем записів, відсортованих за датою створення — останні розміщуються вище.

У центральній частині сторінки розміщено привітальне повідомлення з іменем користувача, поле для швидкого додавання нотатки та блок "Recent Notes", де показано останні створені нотатки з коротким описом і датою. Кожна нотатка супроводжується кнопками швидких дій — редагування, закріплення, видалення та поширення. Інтерфейс адаптований для швидкої взаємодії та візуально розділений на функціональні блоки для кращого користувацького сприйняття. У разі виконання певних дій, як-от створення або видалення нотатки, у

застосунку з’являються спливаючі повідомлення (toast), які інформують користувача про успіх або помилку відповідної операції. Такий підхід сприяє інтуїтивності, ефективності та зручності користування системою.

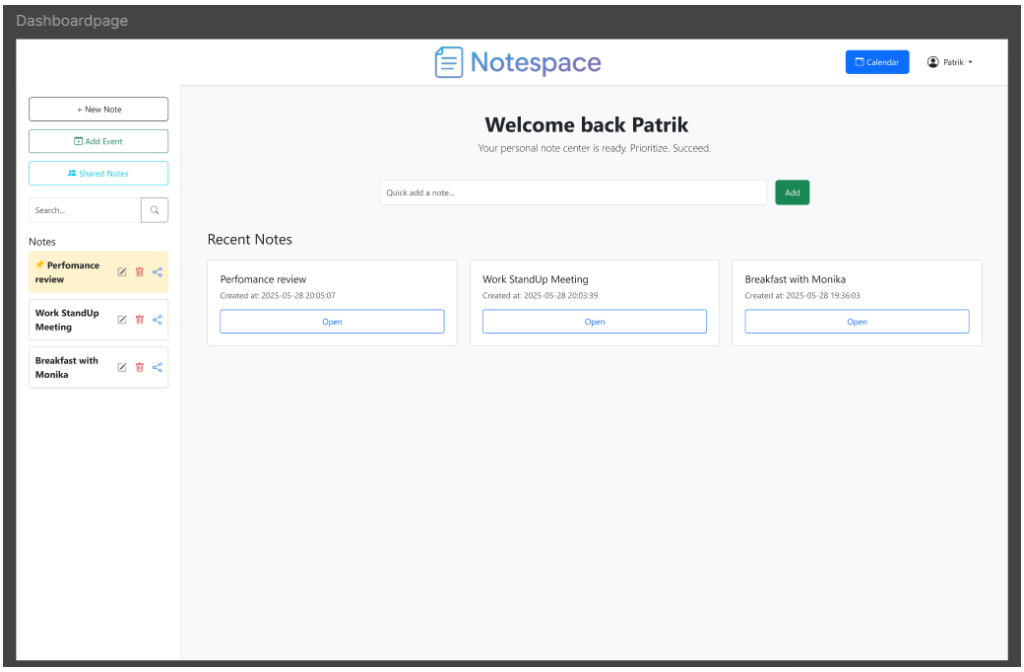
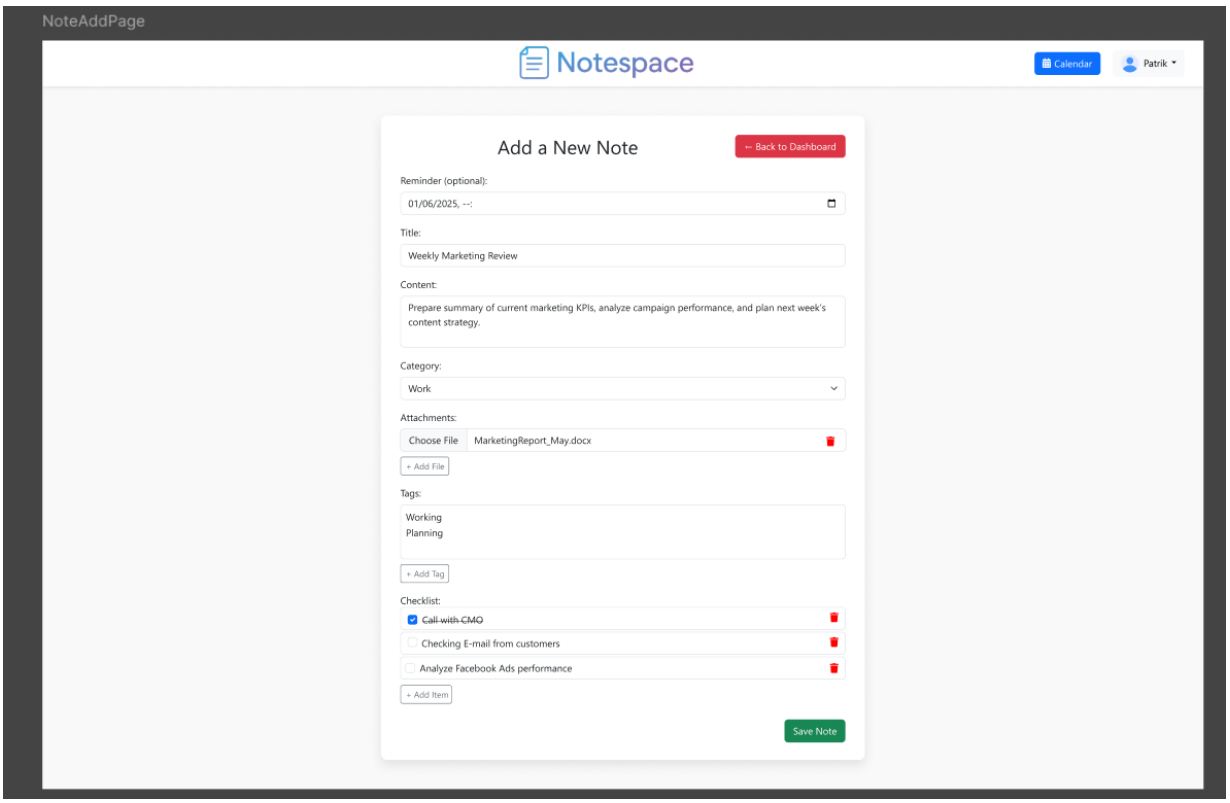


Рисунок. 2.12 – Макети головної сторінки дашборд

Сторінка створення нотатки (рис. 2.13) є ключовим елементом взаємодії користувача з вебзастосунком. Її інтерфейс розроблено так, щоб процес створення нової нотатки був максимально простим і зручним. У центрі розміщена форма з полями для заголовка, основного тексту, вибору категорії, тегів, встановлення дати й часу нагадування, а також додавання вкладень.

Інтерфейс форми побудований за принципом мінімалізму — поля логічно структуровані, супроводжуються підказками та візуально відокремлені. Це дозволяє швидко зорієнтуватися і створити нотатку без зайвих кроків.

Система тегів допомагає краще організувати записи, а нагадування дозволяють вчасно повертатися до важливих справ. Функція додавання вкладень забезпечує можливість прикріпити документи або зображення, які будуть виведені у зручному форматі з можливістю їх видалення до збереження. дозволяючи користувачам зосередитися на змісті та швидко зберігати потрібну інформацію.



Рисуно. 2.13 – Макети головної сторінки дашборд

Сторінка профілю (рис.2.14) відображає аватар та ім'я користувача, біографію, публікації у вигляді сітки або стрічки, а також кількість підписників та підписок. Вона дозволяє користувачу переглядати свою активність, редагувати інформацію та налаштування.

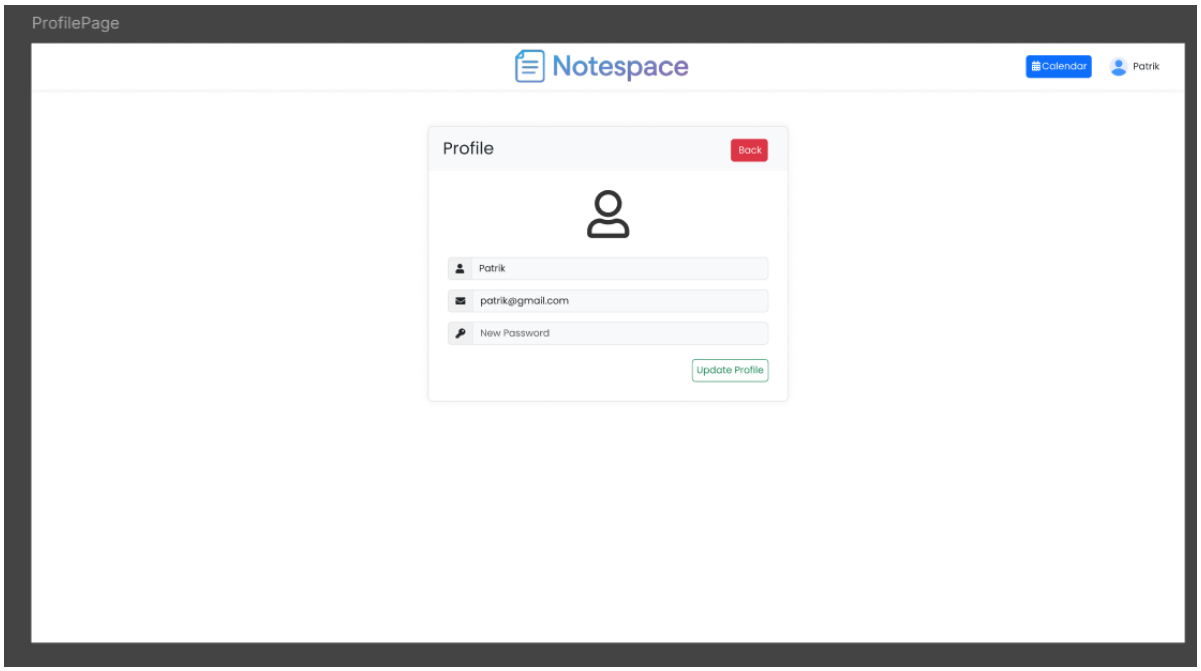


Рисунок. 2.14 – Макети головної сторінки профайл

Отже, розробка UI/UX дизайну вебзастосунку була спрямована на створення зручного, привабливого та інтуїтивно зрозумілого інтерфейсу, що відповідає потребам сучасного користувача. Продумана структура сторінок, адаптивність макетів, мінімалістичне оформлення та послідовна візуальна мова сприяють ефективній взаємодії з системою. Завдяки гармонійному поєднанню естетики та функціональності користувачі можуть швидко орієнтуватися в інтерфейсі, легко створювати й редагувати нотатки, керувати подіями, а також взаємодіяти з іншими користувачами, що загалом підвищує якість користувацького досвіду та задоволення від роботи з платформою.

3 РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

3.1 Підготовка середовища розробки

У ході розробки веб-застосунку для ведення особистих нотаток було реалізовано 8 модулів таких як: модуль аутентифікації та управління користувачами, модуль управління нотатками, модуль тегів, модуль чеклістів, модуль нагадувань, модуль поширення нотаток, модуль коментарів, модуль зворотнього зв’язку.

Розробка веб-застосунку розпочалась з налаштування локального середовища. Для цього було обрано програмний пакет XAMPP v3.3.0**,** до складу якого входять вебсервер Apache**,** система управління базами даних MySQL та графіч-

на оболонка phpMyAdmin. Усі компоненти були встановлені на операційну систему Windows 11.

Після успішного запуску серверів Apache і MySQL було створено директорію проєкту в папці htdocs, що входить до структури XAMPP. Надалі вона слугувала основним простором для розміщення всіх файлів застосунку.

Для розробки інтерфейсу та серверної логіки було обрано середовище Visual Studio Code, у якому велась вся розробка. Код писався з використанням HTML5, CSS3, JavaScript, PHP та MySQL. Додатково застосовувався фреймворк Bootstrap 5, що значно полегшив створення адаптивного і сучасного зовнішнього вигляду сторінок.

На початковому етапі було створено базу даних notes_app у phpMyAdmin. Вона включає таблиці для зберігання інформації про користувачів, нотатки, категорії, теги, чеклісти, коментарі, файли-вкладення, події календаря, лог активності та зворотний зв'язок. SQL-структура бази була збережена у файлі notes_app.sql.

Після створення бази даних розробка продовжилась із формування конфігураційного файлу config.php, в якому було прописано параметри з'єднання з MySQL: хост, логін, пароль та назва бази. Цей файл став базовим для всіх скриптів, що взаємодіють із БД, забезпечуючи централізовану конфігурацію.

Далі була створена базова структура директорій проєкту:

- auth/ — для обробки автентифікації та реєстрації;
- components/ — спільні шаблони (header, footer);
- assets/ — стилі та зображення;
- assetsjs/ — окремі JavaScript-файли;
- dashboardIncludes/ — частини панелі керування;
- uploads/ — папка для збереження прикріплених до нотаток файлів.

Після цього було розпочато розробку основних функціональних модулів.

Спочатку реалізовано модулі:

- аутентифікації та управління користувачами (login.php, profile.php, profile-success.php);
- управління нотатками (note-add.php, note-edit.php, note-view.php, update-note.php, delete-note.php);
- тегів та категорій (tag-add.php, search.php);
- чеклістів, як частина нотаток;
- системи нагадувань (reminder-checker.php);
- функції спільного доступу до нотаток (note-share.php, note-share-handler.php, shared-notes.php);
- коментарів до нотаток (add-comment.php, note-comment-handler.php);
- форми зворотного зв’язку (feedback.php, message.php);
- модулю подій календаря (add-event.php, calendar.php, delete-event.php, get-events.php, get-calendar-events.php);
- журналу активності (log-activity.php, activity-log.php).

Для реалізації повідомлень, підтвердження дій та взаємодії з чекбоксами в чеклістах застосовувався JavaScript**.** Також використовувались toast-повідомлення та localStorage для локального зберігання нагадувань.

Щоб уникнути дублювання коду та забезпечити зручне масштабування, повторювані елементи інтерфейсу — такі як заголовок (header.html), нижній колонтитул (footer.html) і бічне меню — були винесені у окремі шаблонні файли, які підключаються через PHP-функцію include().

Розробка кожного модуля проходила поетапно: створення інтерфейсу, реалізація логіки у PHP, перевірка запитів до бази даних, тестування у браузері через localhost**.** Тестування проводилось вручну — через симуляцію різних сценаріїв користувача (вхід, створення, редагування, пошук, видалення тощо). Це дозволило виявити та виправити помилки ще на етапі реалізації без потреби в зовнішньому сервері або деплойменті.

Таким чином, середовище розробки було налаштоване повністю локально, із використанням сучасних веб-технологій, що дозволило ефективно реалі-

зувати всі необхідні модулі та забезпечити зручне тестування і супровід проєкту.

3.2 Реалізація програмних модулів

Для реалізації модулю авторизації та аунтефікації було створено окрему папку auth/, яка містить сторінки signUp.php, login.php, logout.php.

Сторінка реєстрації (signUp.php) реалізована за допомогою PHP, HTML5, CSS та фреймворку Bootstrap 5. У процесі реєстрації відбувається перевірка обов'язкових полів форми (ім'я користувача, email, пароль, підтвердження пароля). Дані проходять валідацію на заповненість і співпадіння пароля з підтвердженням. Далі здійснюється перевірка унікальності імені користувача в базі даних. У разі успіху пароль хешується за допомогою функції password_hash(), і дані нового користувача додаються до таблиці users. Після успішної реєстрації створюється сесія ($_SESSION['user'], $_SESSION['userId'], $_SESSION['log'] = "1"), і користувач автоматично перенаправляється на головну сторінку (dashboard.php).

Фрагмент коду файлу signUp.php :

```
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);
$insert = "INSERT INTO users (username, email, password)
VALUES    (?, ?, ?)";
$stmt = $conn->prepare($insert);
$stmt->bind_param("sss", $username, $email, $hashedPassword);
```

Сторінка входу (login.php) також реалізована з використанням PHP, HTML5, Bootstrap 5 та стилів CSS. Користувач вводить ім'я користувача та пароль у форму. Після надсилання виконується перевірка на заповненість полів. Далі система звертається до бази даних і отримує відповідний запис користувача за введеним ім'ям. За допомогою функції password_verify() перевіряється відповідність пароля. У разі успішного входу ініціалізується сесія ($_SESSION['user'], $_SESSION['userId'], $_SESSION['log'] = "1"), після чого ко-

ристувач автоматично перенаправляється на сторінку dashboard.php. У разі помилки виводяться відповідні повідомлення про помилку.

Фрагмент коду файлу login.php :

```
if (password_verify($post5, $hashedPassword)) {
  $_SESSION['user'] = $username;
  $_SESSION['userId'] = $Rowsql['id'];
  $_SESSION['log'] = "1";
}
```

Файл виходу (logOut.php) відповідає за завершення сесії користувача. Перед знищенням сесії викликається функція log_activity(), яка фіксує подію виходу в журналі активності. Потім усі змінні сесії ($_SESSION) очищуються за допомогою unset() і session_destroy(). Після цього користувач бачить повідомлення про вихід та автоматично перенаправляється на сторінку входу (login.php).

Фрагмент коду файлу logOut.php:

```
unset($_SESSION['user']);
unset($_SESSION['log']);
unset($_SESSION['userId']);
session_destroy();
```

Сторінка профілю користувача (profile.php) дозволяє редагувати особисту інформацію: ім’я користувача, email та пароль. Дані попередньо завантажуються з бази даних на основі userId, що зберігається в сесії. Користувач може змінити свої дані, які після відправки форми оновлюються в базі даних. Пароль при цьому знову хешується через password_hash(). У разі успішного оновлення користувач перенаправляється на сторінку підтвердження profile-success.php

Фрагмент коду файлу profile.php :

```
$hashedPassword = password_hash($newPassword,
PASSWORD_DEFAULT);
$updateQuery = "UPDATE users SET username = ?, email = ?,
password = ? WHERE id = ?";
$stmtUpdate = mysqli_prepare($conn, $updateQuery);
```

```
    mysqli_stmt_bind_param($stmtUpdate, 'sssi', $username,
$email, $hashedPassword, $user_id);
```

Сторінка успішного оновлення профілю (profile-success.php) реалізована у вигляді повідомлення з кнопкою повернення на головну сторінку. Вона підтверджує, що зміни були застосовані, та забезпечує зворотний перехід до dashboard.php. Також вона містить інтеграцію з повідомленням (toast), яке підтверджує дію у верхній частині екрана.

Фрагмент коду файлу profile-success.php:

```
<div class="card p-4 text-center">
  <h4 class="mb-3 text-success">
    ✓ Your profile has been updated successfully!
  </h4>
  <a href="dashboard.php" class="btn btn-outline-primary btn-
icon">
    <span>🔙</span> Back to Dashboard
  </a>
</div>
<script>
showNotification("Profile updated successfully!", "success");
</script>
```

Модуль управління нотатками є одним із ключових у застосунку, адже саме він відповідає за створення, редагування, перегляд, швидке додавання та видалення нотаток. Основна інформація зберігається у таблиці notes, яка містить такі поля, як title, content, remind_at, category_id, priority, а також службові поля для логіки архівації, статусу виконання і дати створення.

Форма створення нової нотатки реалізована у файлі note-add.php. Вона дозволяє вводити заголовок, зміст, обирати категорію, додавати нагадування, прикріплювати файли, теги, а також створювати чеклісти. Після заповнення форми дані передаються на обробку до скрипта create-note.php, де відбувається запис у базу

даних. Після успішного додавання нотатки зберігаються додаткові дані: файли (attachments), теги (note_tags) та лог активності.

Фрагмент коду файлу note-add.php:

```
$query = "INSERT INTO notes (user_id, title, content,
remind_at, category_id)
VALUES (?, ?, ?, ?, ?)";
$stmt = $conn->prepare($query);
$stmt->bind_param("issss", $user_id, $title, $content,
$remind_at, $category_id);
```

Для зручності користувачів реалізовано функцію швидкого створення нотатки без переходу на окрему сторінку. Цей функціонал реалізовано у файлі quick-add-note.php. Достатньо ввести короткий заголовок — і він буде збережений до таблиці notes.

Фрагмент коду файлу quick-add-note.php:

```
$insert = $conn->prepare("INSERT INTO notes (user_id, title)
VALUES (?, ?)");
$insert->bind_param("is", $user_id, $title);
$insert->execute();
```

Файл note-view.php дозволяє переглядати повну інформацію про обрану нотатку: заголовок, зміст, дату нагадування, категорію, список тегів, прикріплені файли та коментарі. Тут також реалізовано можливість залишити коментар через модальне вікно.

Редагування нотатки реалізовано через файли note-edit.php (форма) та update-note.php (обробка). При завантаженні сторінки завантажуються дані нотатки та пов’язані з нею теги, вкладення й категорія. Після редагування виконується оновлення даних у базі. Також оновлюються вкладення та теги. Подія редагування записується в журнал активності.

Фрагмент коду файлу update-note.php:

```
$query = "UPDATE notes SET title = ?, content = ?,
category_id = ?, remind_at = ? WHERE id = ?";
$stmt = mysqli_prepare($conn, $query);
```

```
    mysqli_stmt_bind_param($stmt, 'ssssi', $title, $content,
$category_id, $remind_at, $note_id);
```

Сторінка delete-note.php відповідає за повне видалення нотатки з бази. Перед цим проводиться перевірка, чи існує запис у таблиці, а також зчитується заголовок нотатки для фіксації в журналі. Успішне видалення супроводжується повідомленням для користувача та записом події у таблицю логів.

Фрагмент коду файлу delete-note.php:

```
$delete_query = "DELETE FROM notes WHERE id = ?";
$delete_stmt = mysqli_prepare($conn, $delete_query);
mysqli_stmt_bind_param($delete_stmt, "i", $note_id);
```

У процесі реалізації програмних модулів було створено функціональну архітектуру, що охоплює всі ключові потреби кінцевого користувача: збереження, редагування, перегляд, сортування, організацію та спільне використання особистих нотаток. Модулі розроблені з урахуванням вимог до зручності, доступності та розширюваності системи. Реалізовані механізми авторизації, нагадувань, тегування, прикріплення файлів, чеклістів, коментарів і календарного планування забезпечують комплексне ведення нотаток. Додаткові модулі, такі як система повідомлень, пріоритети нотаток, теми інтерфейсу та розмежування доступу, розширюють функціональність і покращують користувацький досвід. Таким чином, програмна реалізація веб-застосунку є гнучкою, масштабованою та готовою до подальшого розвитку.

3.3 Інтерфейс користувача

Система авторизації є першим кроком взаємодії користувача з веб-застосунком. Після переходу за посиланням на сторінку входу користувач бачить зручну форму, яка дозволяє ввести логін та пароль. У разі успішної автентифікації відбувається перенаправлення на головну сторінку (dashboard.php), де відображаються всі нотатки, створені або отримані користувачем.

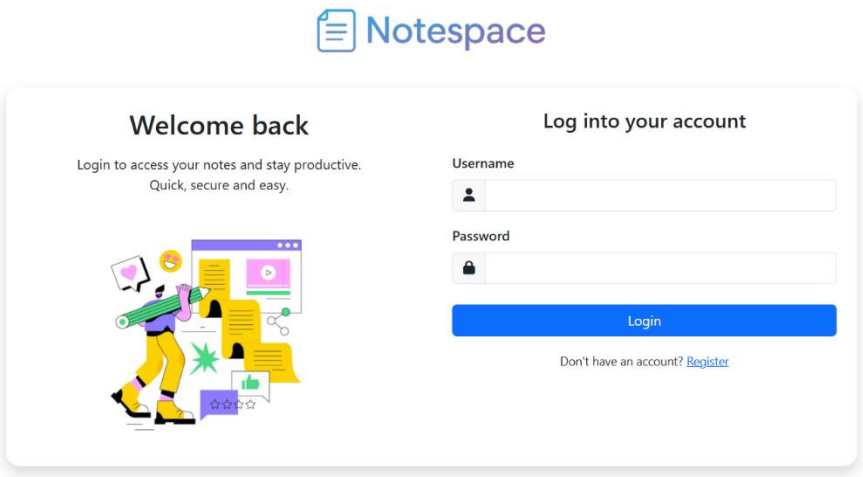


Рисунок. 3.1 – Сторінка входу до системи

Якщо користувач ще не має облікового запису, він може перейти на сторінку реєстрації (signUp.php), скориставшись відповідним посиланням. Реєстраційна форма передбачає заповнення таких полів, як ім'я користувача, адреса електронної пошти, пароль і підтвердження пароля. Після валідації введених даних новий користувач автоматично потрапляє на головну сторінку та отримує доступ до персонального простору для нотаток.

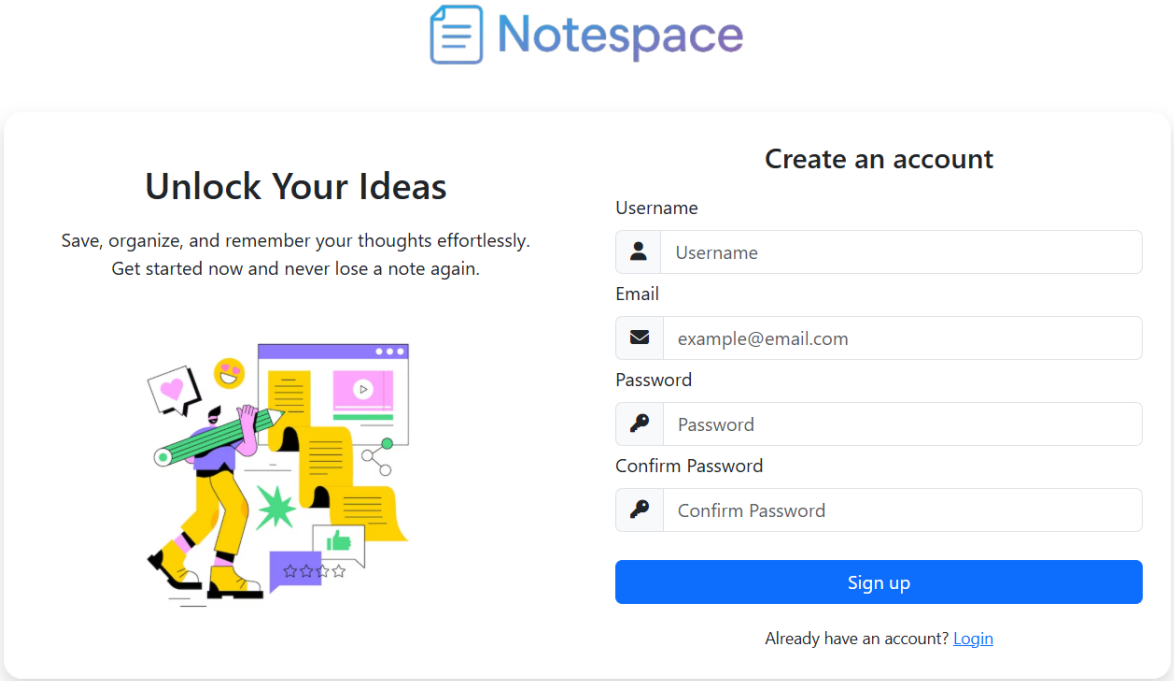


Рисунок. 3.2 – Сторінка реєстрації нового користувача

Головна сторінка є центральним елементом інтерфейсу користувача, де зосереджено основні інструменти взаємодії із нотатками. У верхній частині ро-

зміщено привітальне повідомлення з іменем користувача, логотип застосунку та кнопку переходу до календаря. Це створює позитивний настрій і забезпечує швидкий доступ до основних функцій. У центрі екрана відображається блок Recent Notes, де нотатки виводяться у вигляді карток із назвою та кнопкою відкриття. Користувач може швидко переглянути останні записи, що сприяє зручній навігації та фокусу на поточних завданнях. Над списком нотаток розміщено поле «Quick add a note…», за допомогою якого можна створити нотатку без переходу на окрему сторінку. Цей функціонал забезпечує надзвичайно швидке додавання ідей або важливої інформації «на ходу», що є особливо зручним у динамічному ритмі роботи. У лівій частині екрана знаходиться бічна панель, яка виконує роль навігаційного меню. Тут розміщено кнопки для створення нової нотатки (New Note), додавання події до календаря (Add Event), а також переходу до розділу Shared Notes. Нижче — форма пошуку за ключовими словами, яка дає змогу оперативно знаходити потрібні нотатки. У блоці Notes представлено всі створені користувачем нотатки у вигляді компактного списку з іконками для редагування, видалення, прикріплення.

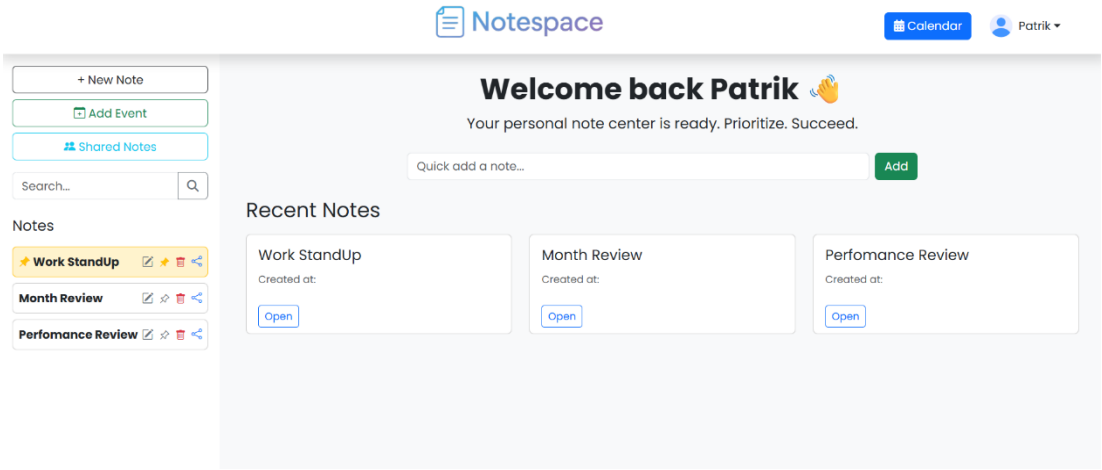


Рисунок. 3.3 – Головна сторінка після авторизації (Dashboard)

Щоб перейти до профілю, користувач натискає на своє ім’я або аватар у верхньому правому куті сторінки та обирає пункт «Profile». На сторінці профілю користувач може переглянути своє ім’я, адресу електронної пошти та, за потреби, змінити пароль. Після внесення змін достатньо натиснути кнопку

«Update Profile», щоб зберегти оновлені дані. Для повернення на попередню сторінку передбачена кнопка «Back».

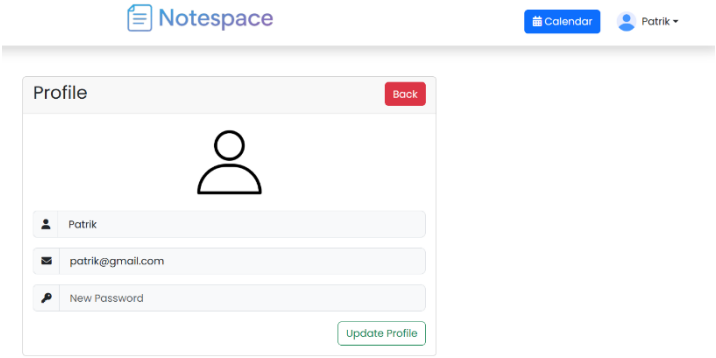


Рисунок. 3.4 – Сторінка створення нової нотатки

Щоб створити нову нотатку, користувач натискає кнопку «New Note» або відповідне посилання у меню. Форма створення нотатки містить обов'язкові поля (заголовок, зміст), а також додаткова опція — встановлення нагадування.

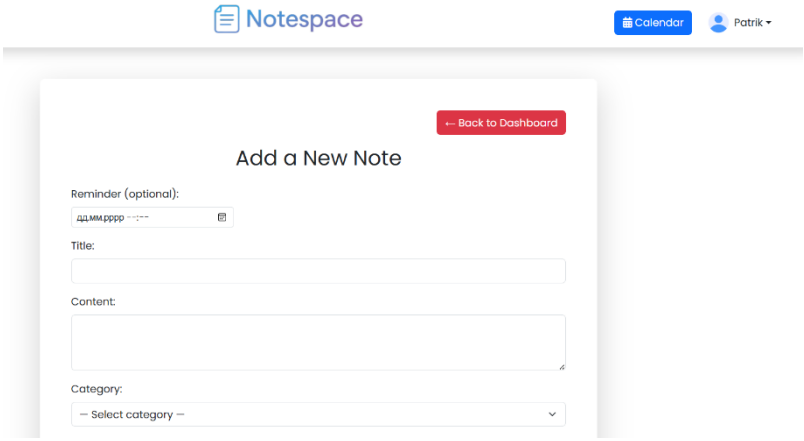


Рисунок. 3.5 – Сторінка створення нової нотатки

Якщо нотатка містить список завдань, користувач може додати чекліст. Кожен елемент містить чекбокс і текстове поле. Виконані пункти можна позначати галочкою, що автоматично закреслює відповідний текст. Крім того, користувач може прикріпити файли будь-якого типу — наприклад, документи, звіти чи зображення — через кнопку «+ Add File». Прикріплений файл відображається з назвою та кнопкою для видалення, що забезпечує зручне управління вкладеннями. Для швидкої фільтрації нотаток можна додати теги, які вводяться вручну та зберігаються для подальшого використання. Також передбачено вибір категорії із випадаючого списку - це дозволяє групувати нотатки за темами.

Таке поєднання можливостей робить нотатку не просто текстовим записом, а гнучким інструментом для організації задач, ідей або проєктів. Після внесення всіх даних нотатку можна зберегти, натиснувши кнопку «Save Note».

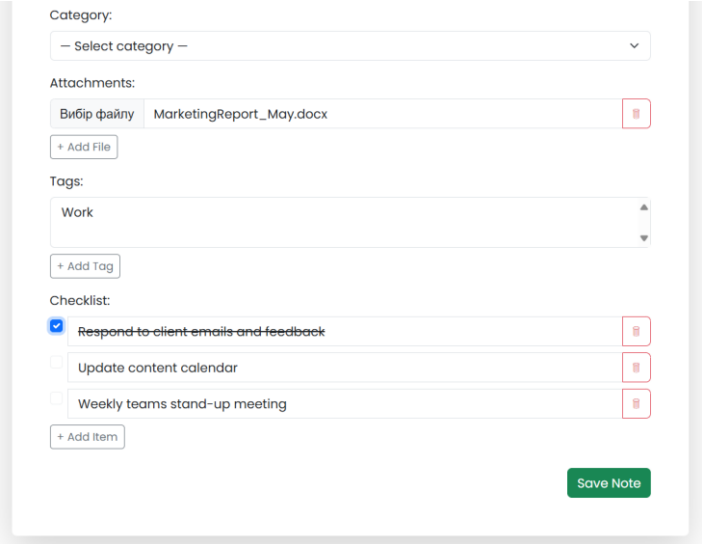


Рисунок. 3.6 – Приклад чеклісту в нотатці

Вкладка «Календар» забезпечує візуальне представлення усіх подій, які користувач створив у застосунку. Інтерфейс реалізовано у вигляді класичного календаря з можливістю перегляду розкладу за місяць, тиждень або у форматі списку. За замовчуванням відкривається режим перегляду на місяць. Події відображаються у відповідних днях із зазначенням часу та кольоровими мітками, які відповідають вибраним кольорам під час створення події. Це дозволяє швидко орієнтуватися у щоденному або тижневому плані. Для зручності навігації передбачені кнопки переходу між місяцями та кнопка «today» для швидкого повернення до поточної дати.

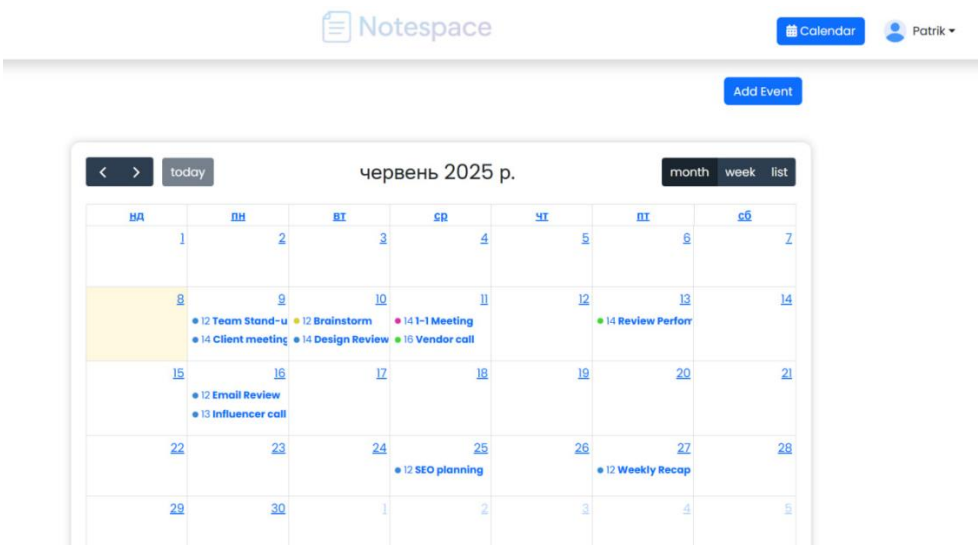


Рисунок. 3.7 – Сторінка Календар

Щоб створити нову подію, користувач натискає кнопку «Add Event» на головній сторінці. Після цього відкривається спливаюче вікно з формою, у якій потрібно вказати основні параметри майбутнього івенту. Користувач заповнює назву події та детальний опис. Далі задається точний час початку і завершення події за допомогою зручного вибору дати та години. Для візуального розмежування подій можна обрати колір у полі Color, який буде відображатися у календарі. Після заповнення всіх полів подія зберігається в інтерфейсі календарю, а вона автоматично з’являється у календарному інтерфейсі. Форма створення подій є простою та функціональною, що дозволяє швидко додавати зустрічі, нагадування або інші важливі події без потреби переходити на окремі сторінки.

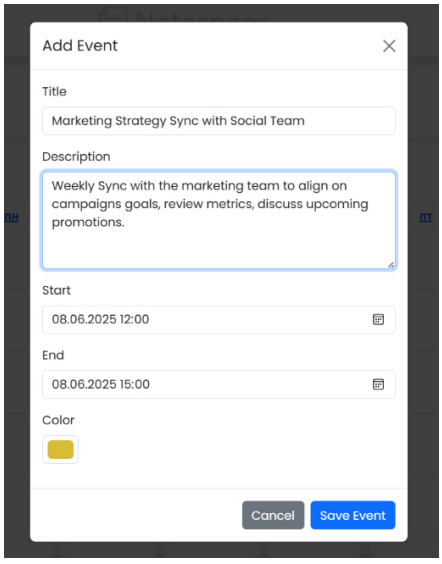


Рисунок. 3.8 – Приклад додавання події

Також у користувача є можливість поширювати нотатки з іншими користувачами. У спеціальному розділі відображаються усі нотатки, якими з користувачем поділилися інші.

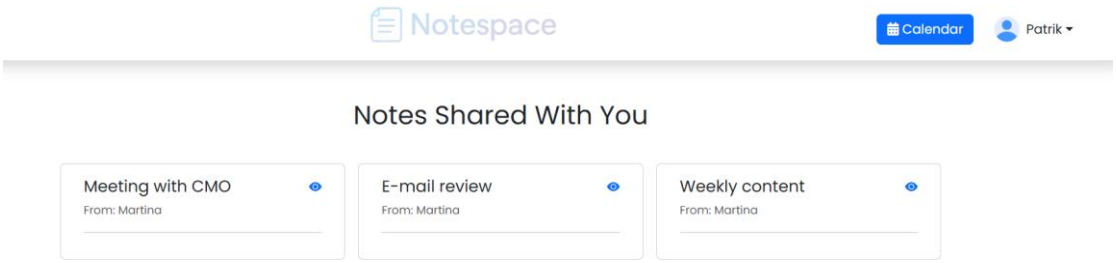


Рисунок. 3.9 – Вкладка зі спільними нотатками

Для контролю активності в системі реалізовано журнал дій. Користувач бачить свої дії (створення, редагування, видалення нотаток тощо), а адміністратор — усі записи. Це забезпечує прозорість та зручний аудит змін.

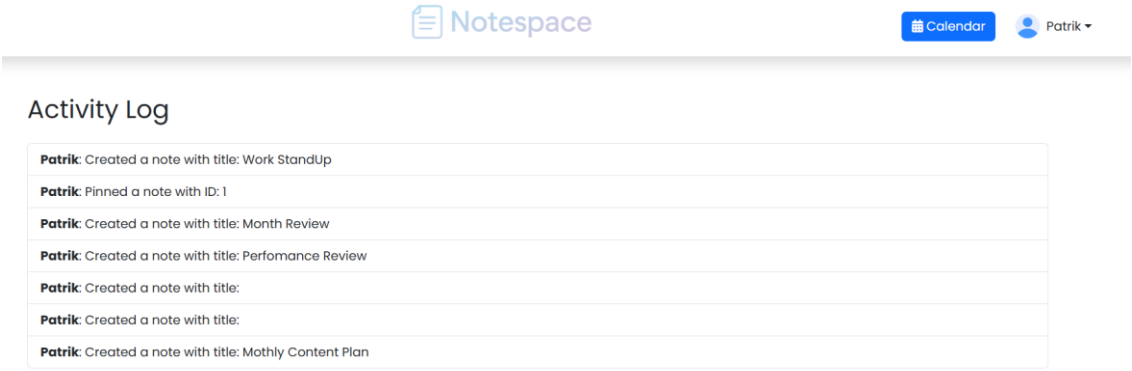


Рисунок. 3.10 – Сторінка журналу дій користувача

Щоб надіслати відгук, користувач переходить на сторінку Feedback через головне меню. На цій сторінці можна поділитися ідеями, пропозиціями або повідомити про помилки в роботі системи. Інтерфейс складається з одного поля для введення повідомлення, де користувач вводить текст звернення. Після заповнення поля достатньо натиснути кнопку «Submit», щоб повідомлення було передано розробникам. Така форма зворотного зв'язку дозволяє оперативно реагувати на потреби користувачів і враховувати їхні побажання під час вдосконалення.

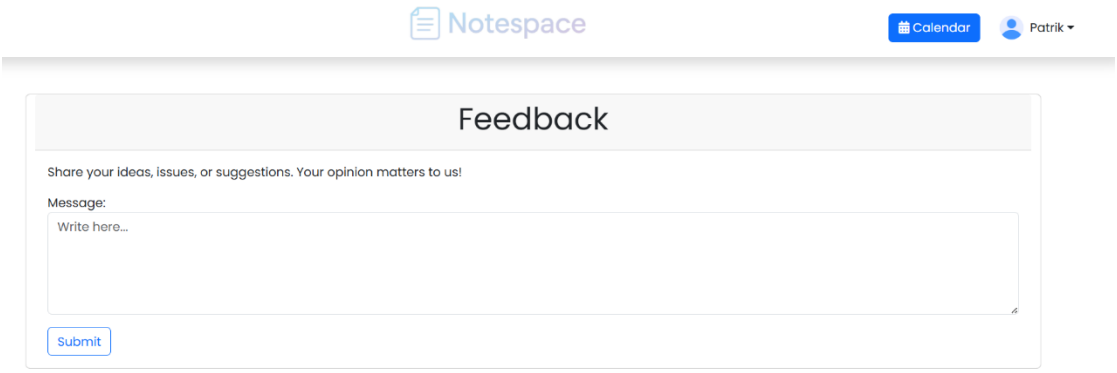


Рисунок. 3.11 – Сторінка зворотнього зв’язку

Таким чином, у межах реалізації веб-застосунку було впроваджено широкий спектр функціональних можливостей, що охоплюють автентифікацію користувачів, створення та управління нотатками, роботу з вкладеннями, тегами, категоріями, чеклістами, коментарями та спільним доступом. Кожен модуль інтегровано у єдину систему з інтуїтивно зрозумілим інтерфейсом. Така архітектура дозволяє користувачам ефективно організовувати особисту інформацію, взаємодіяти з іншими та зберігати дані у структурованому вигляді.

3.4 Перевірка системи на працездатність

Перевірка працездатності є обов’язковим етапом розробки веб-застосунку, що дозволяє переконатися у відповідності реалізованої функціональності заявленим вимогам. Основною метою тестування стало підтвердження коректної роботи всіх модулів системи : від авторизації до створення та перегляду нотаток, взаємодії з вкладеннями, нагадуваннями, коментарями, а також журналом активності.

Тестування проводилось вручну в середовищі розробки Windows 11 за допомогою локального сервера XAMPP. Усі функції перевірялись через браузер Microsoft Edge на стаціонарному комп’ютері. Система запускалась локально через адресу http://localhost, без використання мобільних пристроїв, хостингу або сторонніх сервісів. Автоматизовані інструменти тестування, дебагери або зовнішні модулі при перевірці не застосовувались.

Складено дев’ять основних тест-кейсів, які охоплюють ключові операції, включно з реєстрацією, авторизацією, створенням, редагуванням і видаленням нотаток, прикріпленням файлів, додаванням тегів, коментарів, а також перевіркою журналу активності.

Нижче наведено таблицю 3.1, що містить перелік тест-кейсів, які були використані для перевірки працездатності веб-застосунку..

Таблиця 3.1 – Тест-кейси для системи

№	Назва тесту	Опис дії	Вхідні дані	Очікуваний результат	Статус
1	2	3	4	5	6
1	Реєстрація нового користувача	Користувач вводить дані для створення акаунта	Username, email, пароль, повторне введення паролю	Акаунт створено, перенаправлення на головну сторінку	Пройдено
2	Реєстрація паролями, які не збігаються	Введеня паролів, які не збігаються	Username, Email, паролі, які не збігаються	Виводиться повідомлення про помилку: «Passwords do not match!»	Пройдено
3	Авторизація з правильними даними	Вхід у систему через email і пароль	Username, пароль	Вхід успішний, відкривається сторінка нотаток	Пройдено
4	Авторизація з неправильним паролем	Спроба входу з хибним паролем	Username, неправильний пароль	Повідомлення про помилку «Wrong password!»	Пройдено
5	Створення нотатки	Створення текстової нотатки з тегами	Заголовок, текст, теги	Нотатку створено, вона відображається в списку	Пройдено

Кінець таблиці 3.1

1	2	3	4	5	6
6	Видалення нотатки	Користувач натискає "видалити"	ID нотатки	Нотатку видалено зі списку	Пройдено
7	Надання спільного доступу	Введення email іншого користувача	Email	Доступ надано, інший користувач бачить нотатку	Пройдено
8	Додавання коментаря	Написання коментаря під нотаткою	Текст коментаря	Коментар відображається під нотаткою	Пройдено
9	Перегляд журналу активності	Перехід у розділ "Історія дій"	–	Відображається список дій із часовими мітками	Пройдено

Тест-кейс №1: Одним із ключових етапів тестування стало перевірення модуля реєстрації нового користувача, що є критично важливою частиною будь-якого веб-застосунку з авторизацією. Сценарій передбачав заповнення всіх обов’язкових полів форми реєстрації — імені користувача (Username), електронної пошти (Email), пароля (Password) та його підтвердження (Confirm Password). Для тестування були використані валідні тестові дані.

На рисунку 3.14 зображено заповнену форму реєстрації, в якій користувач вводить відповідну інформацію. Після натискання кнопки «Sign up» відбувається перевірка:

- чи всі поля заповнені;
- чи збігаються паролі;
- чи не зайнятий логін;
- якщо всі умови виконані, система формує новий запис у базі даних і запускає сесію.

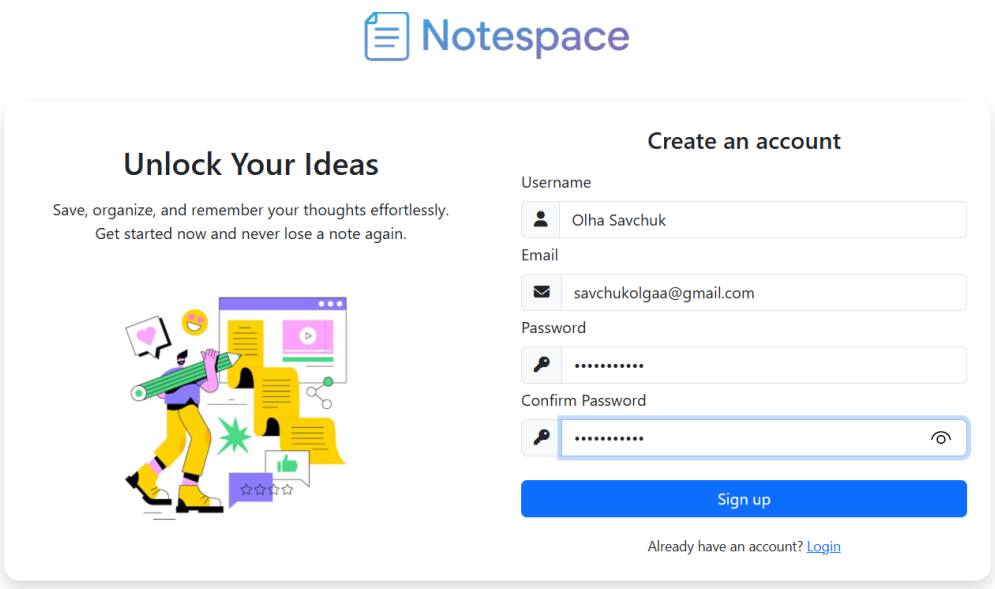


Рисунок. 3.14 – Форма реєстрації нового користувача

У разі успішного завершення процесу, користувачу відображається повідомлення «Successfully Signed Up!», що видно на рисунку 3.15. Крім того, реалізовано автоматичне перенаправлення на головну сторінку панелі керування (dashboard.php), що також підтверджує правильне виконання сценарію.

Весь процес відбувся без збоїв. Результат відповідав очікуванням: акаунт було створено, і користувача перенаправило до робочого середовища. Тест-кейс було позначено як «Пройдено».

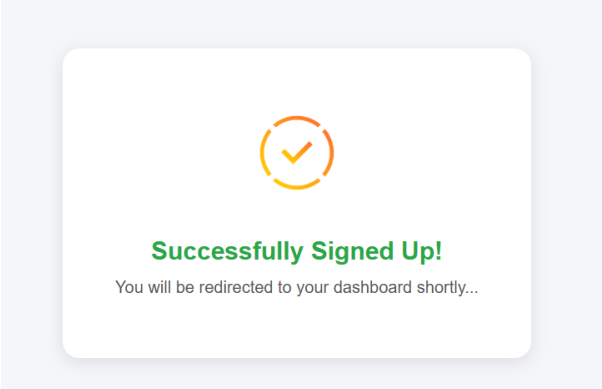


Рисунок. 3.15 – Повідомлення про успішну реєстрацію

Тест-кейс №2: Одним із поширених випадків є ситуація, коли при реєстрації користувач заповнює поле «Password» та «Confirm Password» різними значеннями. У таких випадках перед надсиланням форми до бази даних виконується перевірка на збіг паролів. Як тільки система виявляє, що значення полів

password і confirm password не збігаються, вона виводить повідомлення рисунок 3.16. Жодних змін у базі не відбувається, користувач залишається на сторінці та може виправити введені дані.

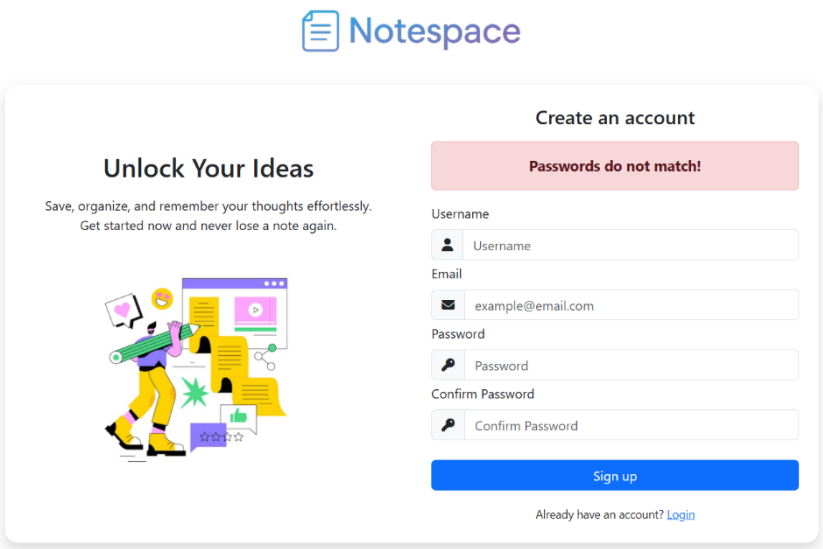


Рисунок 3.16 – Повідомлення про помилку при різних паролях у формі реєстрації

Тест-кейс №3: У ході тестування перевірено процес авторизації з коректно введеними обліковими даними. На рисунку 3.17 показано форму входу, де користувач вводить логін та пароль, отримані під час реєстрації. Після натискання кнопки Login система успішно проводить автентифікацію та ініціалізує сесію.

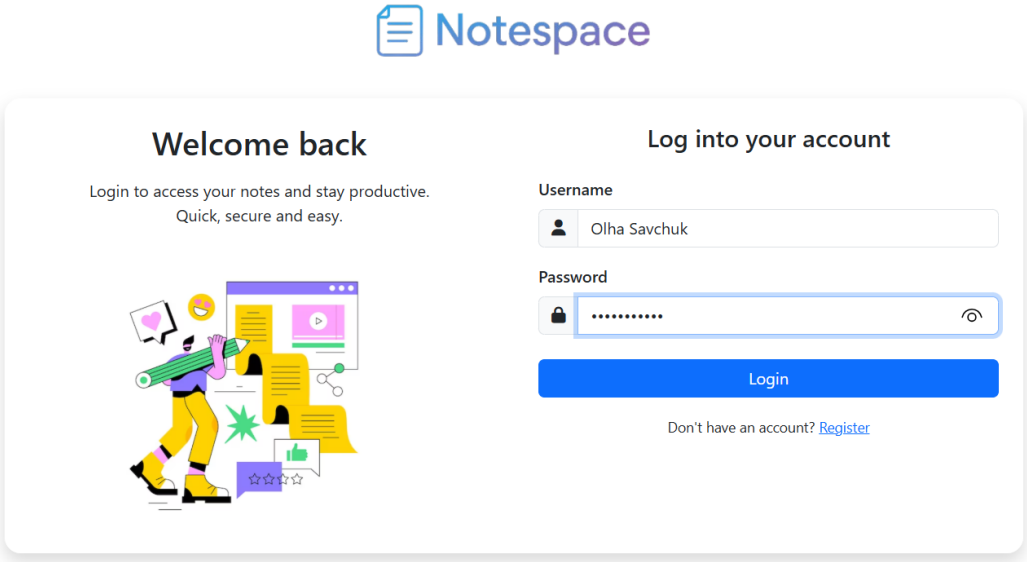


Рисунок 3.17 – Форма авторизації користувача

Як видно на рисунку 3.18, відображається повідомлення "Successfully Logged In!", що супроводжується автоматичним перенаправленням на сторінку користувача. Вхід у систему здійснюється без помилок, функціональність підтверджена.

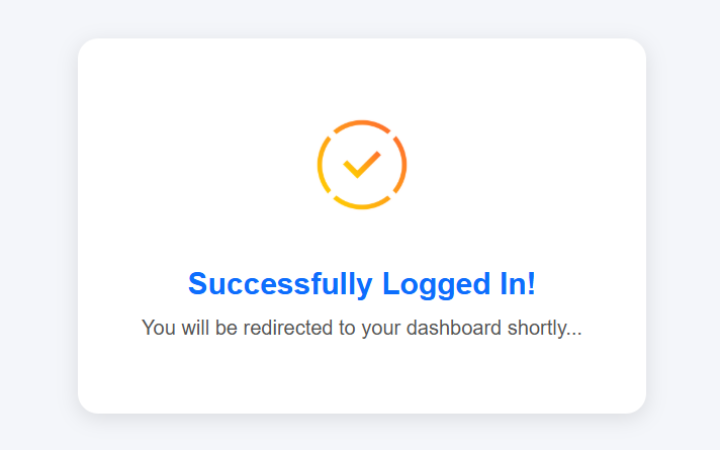


Рисунок 3.18 – Повідомлення про успішний вхід

Тест-кейс №4: У цьому тесті перевірялась реакція системи на спробу входу з неправильним паролем. У разі некоректного пароля або відсутності такого користувача система блокує доступ і виводить відповідне повідомлення. Як показано на рисунку 3.20, після натискання кнопки Login система миттєво вивела повідомлення про помилку — "Wrong password!", і не допустила вхід до особистого кабінету. Це свідчить про надійно реалізований механізм валідації облікових даних, який виключає несанкціонований доступ.

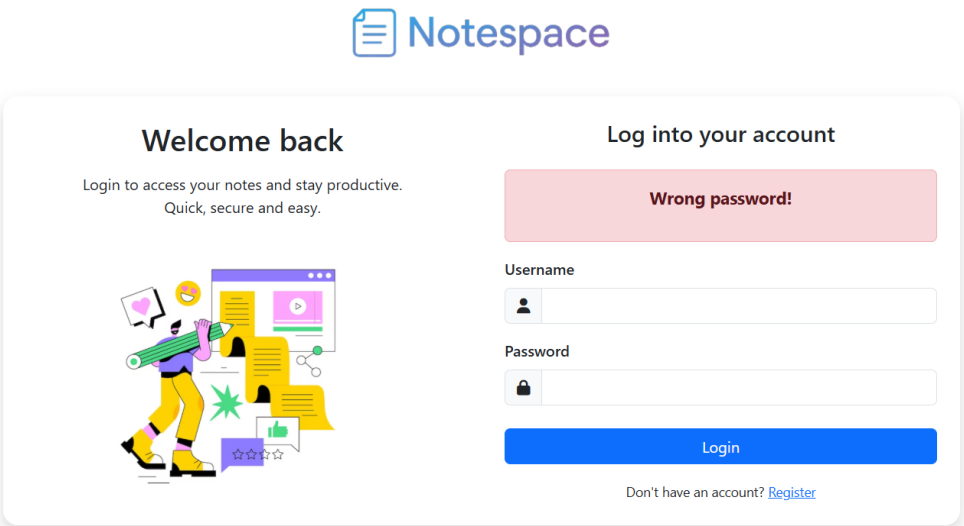


Рисунок 3.20 – Повідомлення про помилку при введенні неправильного пароля

Тест-кейс №5: Під час тестування була перевірена базова функція системи — створення нової нотатки. Користувач переходить на сторінку створення нотатки та заповнює форму: вказує заголовок, вміст, додає дату нагадування, теги, прикріплює файл та вводить пункти чекліста — рисунки 3.21, 3.22. Після натискання кнопки Save Note, система зберігає всі введені дані у відповідні таблиці бази даних, включаючи супутні записи (теги, вкладення, нагадування тощо).

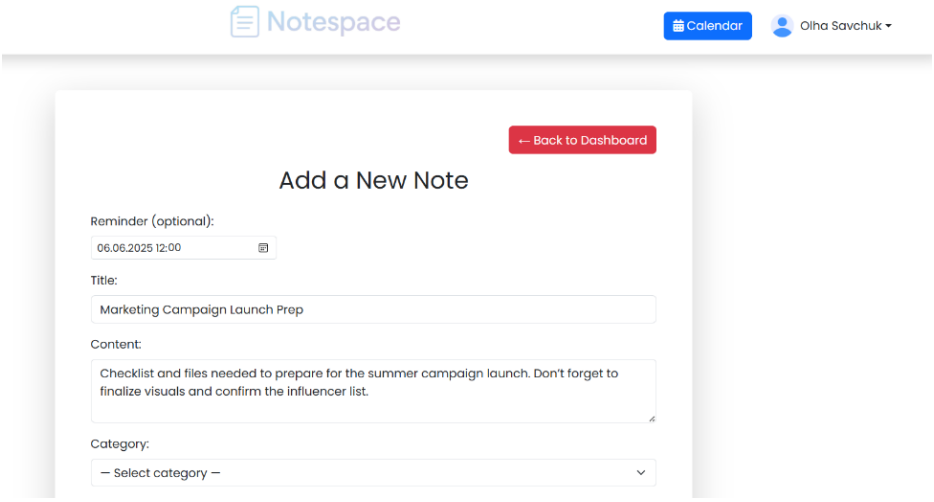


Рисунок 3.21 – Форма створення нової нотатки

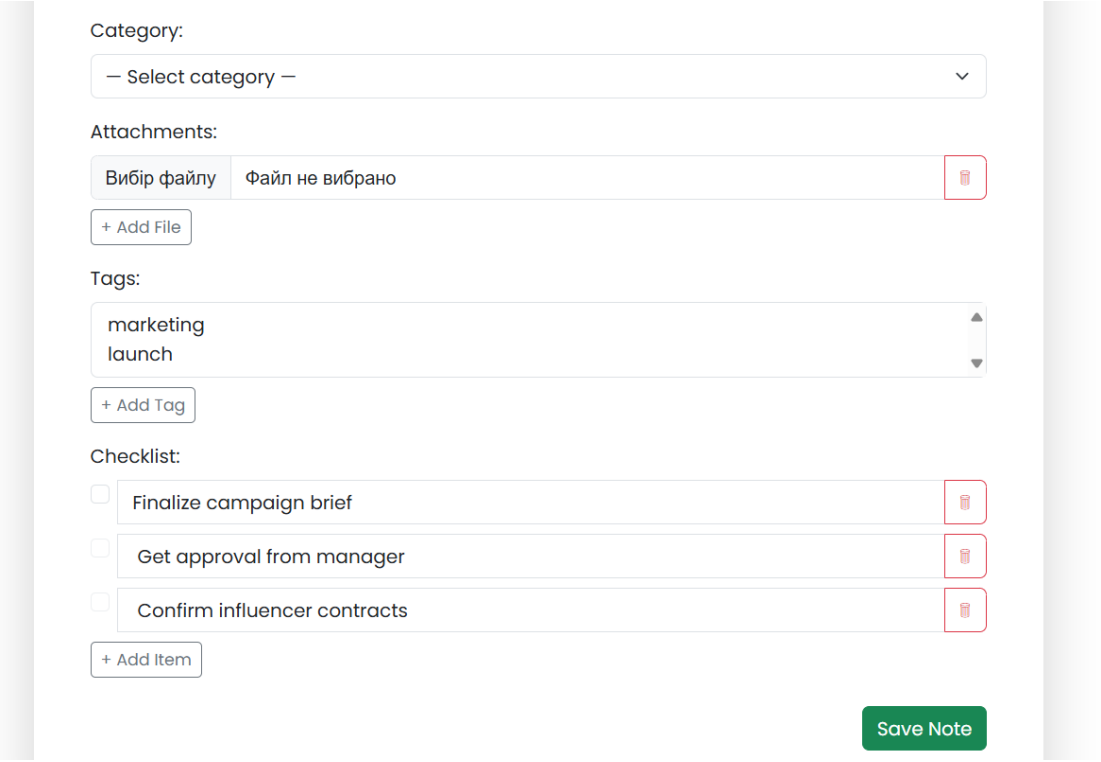


Рисунок 3.22 – Заповнення тегів, вкладень та чеклісту

На головній сторінці користувач одразу бачить новостворену нотатку в блоці «Recent Notes» та у загальному списку — рисунок 3.23. Таким чином, система підтвердила правильність обробки введених даних, візуалізацію результату та коректну роботу інтегрованих модулів.

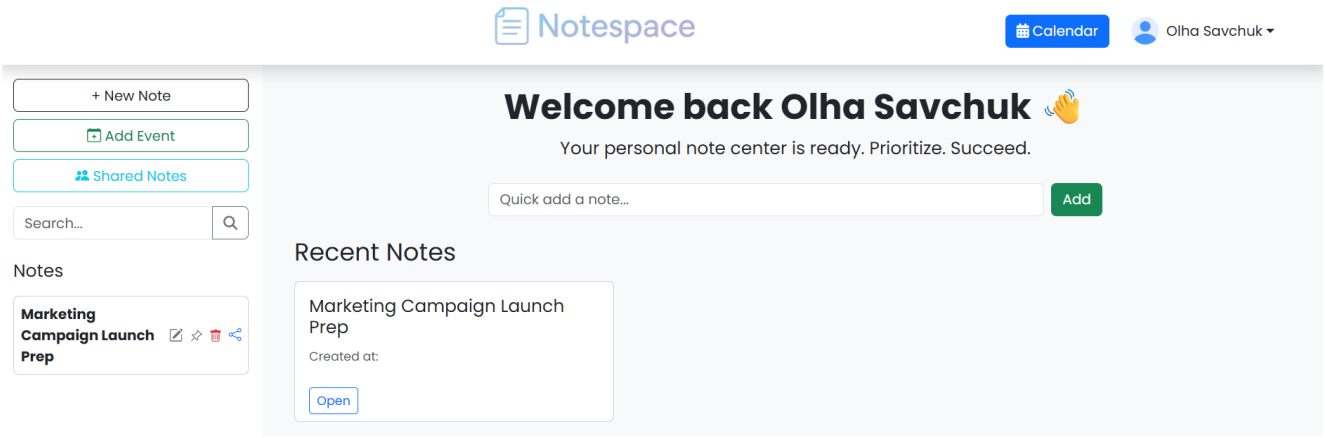


Рисунок 3.23 – Відображення створеної нотатки на головній сторінці

Тест-кейс №6: У цьому тесті перевірено коректність роботи функції видалення нотатки в інтерфейсі застосунку. Користувач обирає відповідну нотатку зі списку, натискає іконку видалення та виконує підтвердження дії. Як видно на рисунку 3.24**,** кнопка видалення розташована безпосередньо біля назви кожної нотатки у списку.

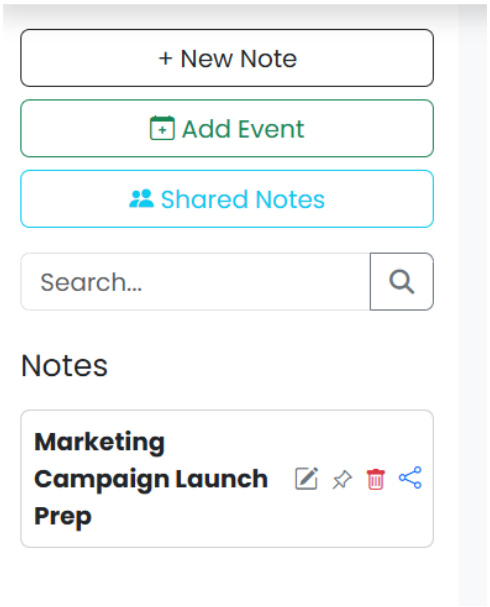


Рисунок 3.24– Вигляд нотатки зі списку перед видаленням

Після натискання кнопки, система виводить вбудоване повідомлення браузера з підтвердженням дії рисунок 3.25**.** Користувач може скасувати видалення або підтвердити його, натиснувши кнопку OK**.** Така реалізація допомагає уникнути випадкових видалень.

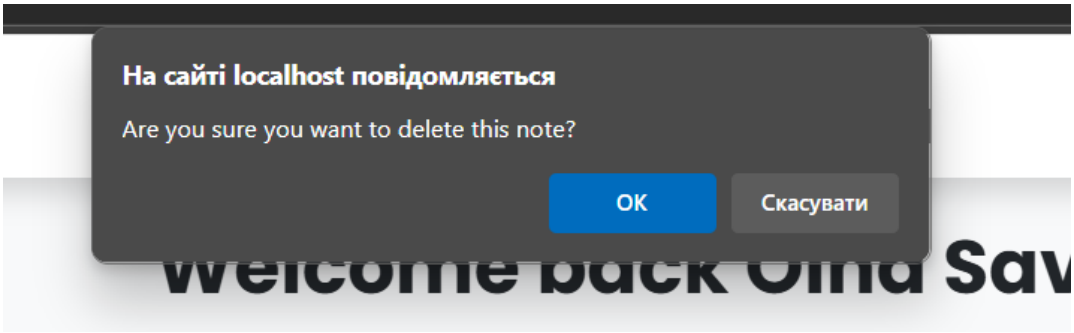


Рисунок 3.25 – Повідомлення підтвердження дії видалення

Після підтвердження, нотатка миттєво зникає зі списку — без перезавантаження сторінки. На рисунку 3.26 видно, що поле зі списком нотаток після видалення стало порожнім, а запис було фізично прибрано з бази даних.

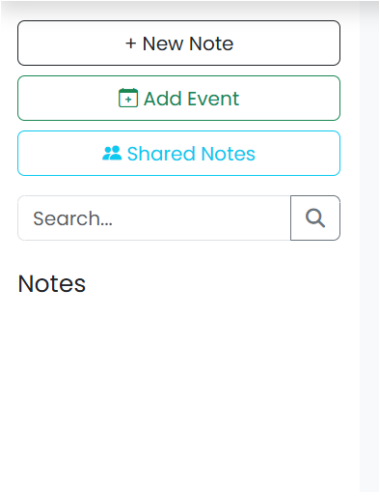


Рисунок 3.26 – Список нотаток після успішного видалення

Тест-кейс №7: У цьому тесті перевірено реалізацію функції поширення нотатки іншому користувачу. На головній сторінці поруч із кожною нотаткою розміщено кнопку "Share", натискання якої відкриває спливаюче вікно для вве-

дення імені користувача (рис. 3.27). Додатково можна увімкнути опцію "Allow editing", що передбачає редагування нотатки іншим користувачем.

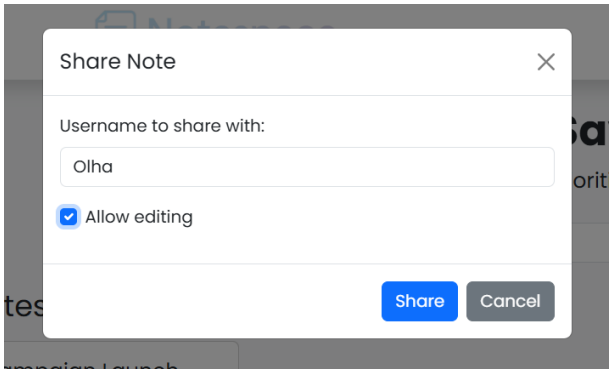


Рисунок 3.27 – Форма поширення нотатки

Після підтвердження доступу, система зберігає відповідний запис у таблицю shared_notes. При вході в систему другий користувач бачить поширену нотатку в розділі "Notes Shared With You", як показано на рисунку 3.28.

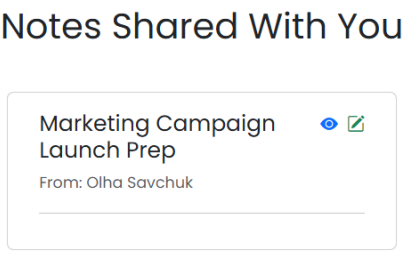


Рисунок 3.28 – Вигляд поширеної нотатки в інтерфейсі іншого користувача

Тест-кейс №8: У цьому тесті перевіряється можливість користувача залишати коментарі до нотаток Після натискання кнопки «Add Comment» відкривається модальне вікно, де користувач вводить текст рисунок. 3.29.

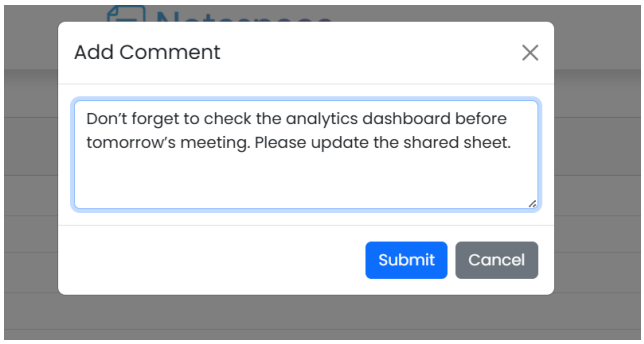


Рисунок 3.29 – Модальне вікно додавання коментаря

Після підтвердження, коментар зберігається в таблиці note_comments і виводиться під нотаткою на сторінці перегляду рисунок. 3.30.

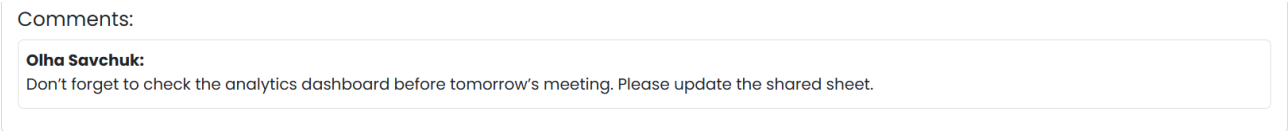


Рисунок 3.30 – Коментар відображається під нотаткою

Тест-кейс №9: Даний тест перевіряє можливість користувача переглядати історію власних дій у системі. Сторінка журналу активності містить перелік виконаних дій — створення, редагування, видалення нотаток, поширення тощо. Записи зберігаються у таблиці activity_log та виводяться з урахуванням прав доступу: звичайний користувач бачить лише свої дії, тоді як адміністратор — всі. Результат тесту представлено на рисунку 3.31 — у вигляді списку дій користувача Olha Savchuk.

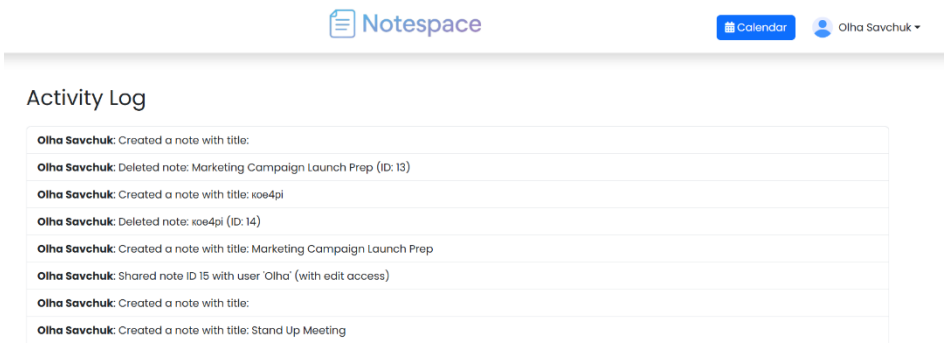


Рисунок 3.31 – Сторінка журналу активності користувача

Проведене тестування веб-застосунку дозволило переконатися у працездатності основних функціональних модулів системи. Було реалізовано 9 тест-кейсів, які охоплюють ключові сценарії взаємодії користувача із системою: реєстрацію, авторизацію, створення, редагування та видалення нотаток, роботу з коментарями, на також надання спільного доступу до нотаток і перегляд журналу активності. У процесі перевірки жодних критичних помилок не виявлено.

Функціональність відповідає вимогам, інтерфейс коректно реагує на введення даних, повідомлення про помилки відображаються у відповідних ситуаціях. Застосунок стабільно зберігає, обробляє та відображає дані, забезпечуючи базову безпеку, включно з хешуванням паролів. Таким чином, система показала себе надійною у межах заявленого функціоналу та готовою до подальшого розгортання або масштабування. Надалі тестування може бути доповнене автоматизованими скриптами, перевіркою на мобільних пристроях, а також навантажувальним тестуванням для оцінки стабільності при високій активності користувачів.

ВИСНОВКИ

У процесі написання дипломної роботи було створено функціональний веб-застосунок для ведення особистих нотаток, нагадувань, чеклістів, коментарів та інших корисних функцій. Метою проєкту було забезпечити зручний, швидкий та інтуїтивно зрозумілий інструмент для організації особистої інформації користувача в межах веб-середовища.

У ході роботи було виконано всі поставлені завдання:

- Обґрунтовано актуальність створення подібного застосунку з огляду на зростаючі потреби користувачів у збереженні й систематизації особистої інформації.
- Проаналізовано існуючі рішення, визначено їхні недоліки й можливості вдосконалення, сформульовано основні переваги власного проєкту.
- Сформульовано функціональні та нефункціональні вимоги до майбутньої системи, включно з простотою використання, доступністю, надійністю та захистом даних.
- Обрано сучасні веб-технології для реалізації: HTML5, CSS3, JavaScript, PHP, MySQL, Bootstrap 5.
- Спроєктовано архітектуру системи та структуру бази даних, реалізовано всі необхідні зовнішні ключі та логічні зв’язки між таблицями.
- Реалізовано всі основні функціональні модулі: реєстрацію, авторизацію, створення, редагування, видалення нотаток, встановлення нагадувань, додавання вкладень, тегів, чеклістів, коментарів, а також функцію поширення нотаток між користувачами.
- Проведено тестування працездатності системи за допомогою ручних тест-кейсів, виявлено та усунуто недоліки.

В результаті виконання дипломного проєкту створено повноцінний веб-застосунок, адаптований під індивідуальні потреби користувачів. Таким чином, розроблений веб-застосунок відповідає поставленим вимогам, демонструє практичне застосування сучасних технологій веб-розробки, а також має потенціал до подальшого розвитку та використання в реальному середовищі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Міссюк Н. В. Організація інформації в умовах цифрового суспільства. Інформаційні технології в освіті. 2021. № 39. С. 25–31.

2. Руденко Т. А. Переваги хмарних застосунків у порівнянні з десктопними рішеннями. Вісник Київського університету. 2020. № 2. С. 43–47.

3. LeClair, J. Best Note-Taking Apps in 2024. PCMag, 2024. URL: https://www.pcmag.com/picks/the-best-note-taking-apps (дата звернення: 27 березня 2025 р.).

4. Ulanoff, L. Why Google Keep is too simple for serious use. Mashable, 2023. URL: https://mashable.com/article/google-keep-limits (дата звернення: 9 червня 2025 р.).

5. Google Support. Set reminders in Google Keep. URL: https://support.google.com/keep/answer/6105533 (дата звернення: 9 червня 2025 р.).

6. TechRadae. Google Keep Review. URL: https://www.techradar.com/reviews/google-keep (дата звернення: 9 червня 2025 р.).

7. Google Keep Review. URL: https://www.techradar.com/reviews/google-keep (дата звернення: 9 червня 2025 р.).

8. Microsoft Support. Basic tasks in OneNote. URL: https://support.microsoft.com/en-us/office/basic-tasks-in-onenote (дата звернення: 9 червня 2025 р.).

9. OneNote Blog. Using OneNote for Study and Work. URL: https://techcommunity.microsoft.com/t5/education-blog/using-onenote-in-education (дата звернення: 9 червня 2025 р.)

10. Evernote. Plans Comparison. URL: https://evernote.com/compare-plans (дата звернення: 9 червня 2025 р.).

11. CNET Reviews. Evernote Review 2024: Great Tools with Usability Hurdles. URL: https://www.cnet.com/reviews/evernote-review/ (дата звернення: 9 червня 2025 р.).

12. Anderson, C. Why Evernote may not be your to-do list. Productivity Today, 2023.

13. Simplenote Help Center. Version history & backups. URL: https://help.simplenote.com/hc/en-us/articles/360021394831 (дата звернення: 9 червня 2025 р.).

14. Simplenote Docs. Using Markdown in Simplenote. URL: https://help.simplenote.com/hc/en-us/articles/360021397011 (дата звернення: 9 червня 2025 р.).

15. Nielsen, J. Designing Web Usability. New Riders, 2000.

16. ISO 9241-210. Human-centred design for interactive systems.

17. Interaction Design Foundation. Accessibility and Inclusive Design. 2022.

18. W3C. HTML5: A vocabulary and associated APIs for HTML and XHTML. URL: https://www.w3.org/TR/html5/ (дата звернення: 9 червня 2025 р.).

19. MDN Web Docs. CSS3 Introduction. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 9 червня 2025 р.).

20. Bootstrap. Introduction to Bootstrap 5. URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 9 червня 2025 р.).

21. Bootstrap. JavaScript components. URL: https://getbootstrap.com/docs/5.0/components/ (дата звернення: 9 червня 2025 р.).

22. Mozilla Developer Network. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 9 червня 2025 р.).

23. PHP.net. What is PHP? URL: https://www.php.net/manual/en/intro-whatis.php (дата звернення: 9 червня 2025 р.).

24. MySQL Reference Manual. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 9 червня 2025 р.)

25. phpMyAdmin Official Site. URL: https://www.phpmyadmin.net/ (дата звернення: 9 червня 2025 р.).

26. Google Fonts. URL: https://fonts.google.com/ (дата звернення: 9 червня 2025 р.).

27. Гавриленко О. В. Проєктування інформаційних систем: навчальний посібник. Харків: ХНУРЕ, 2019. 121 с.

28. Макаров Ю. В. Адміністрування веб-систем: функціональна безпека. Львів: Видавництво ЛНУ, 2021. 76 с.

29. Нікітін О. В. Архітектура клієнт-серверних застосунків. Дніпро: ДНУ, 2018. 64 с.

30. Бочаров С. В. Основи документування програмних рішень. К.: Ліра-К, 2020. 93 с.

31. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. P. 150–151.

32. Литвиненко С. Ю. Модульна архітектура програмних систем: принципи побудови та ефективність. — Харків: ХНУРЕ, 2020. — С. 102.

33. Nielsen, J. (2000). *Designing Web Usability: The Practice of Simplicity*. New Riders.

34. Interaction Design Foundation. *User Experience Design*. – https://www.interaction-design.org/literature/topics/ux-design (дата звернення: 9 червня 2025 р.).

35. Norman, D. (2013). *The Design of Everyday Things*. MIT Pr

ДОДАТКИ

ДОДАТОК А

Програмний код для реалізації веб – застосунку для ведення особистих нотаток та нагадувань

SignUp.php

```
<?php
session_start();

// Переадресація після успішної реєстрації
if (isset($_SESSION['log']) && $_SESSION['log'] === '1') {
  echo <<<HTML
  <!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Registration Success</title>
    <meta            http-equiv="refresh"            content="2;
url=../dashboard.php">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/boot
strap.min.css" rel="stylesheet" />
    <link rel="stylesheet" href="../assets/css/style.css" />
    <style>
      body {
        font-family: "Poppins", sans-serif;
        background-color: #f4f6fa;
        display: flex;
        justify-content: center;
        align-items: center;
        min-height: 100vh;
        margin: 0;
      }
      .confirmation-box {
        background: #fff;
        border-radius: 16px;
        padding: 40px 30px;
        box-shadow: 0 4px 16px rgba(0,0,0,0.1);
        text-align: center;
        max-width: 450px;
        width: 100%;
      }
      .confirmation-box img {
        width: 120px;
        margin-bottom: 20px;
      }
      .confirmation-box h4 {
        font-weight: 600;
        color: #28a745;
      }
      .confirmation-box p {
```

```
        color: #555;
        font-size: 16px;
      }
    </style>
  </head>
  <body>
    <div class="confirmation-box">
      <img src="../assets/tick4.webp" alt="Success">
      <h4>Successfully Signed Up!</h4>
      <p>You will be redirected to your dashboard shortly...</p>
    </div>
  </body>
  </html>
  HTML;
  exit();
}

$error = "";
$conn = new mysqli("localhost", "root", "", "notes_app", 3307);
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$username = $_POST['username'] ?? '';
$email = $_POST['email'] ?? '';
$password = $_POST['password'] ?? '';
$confirm = $_POST['confirm'] ?? '';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  if (empty($username) || empty($email) || empty($password) ||
empty($confirm)) {
    $error = "Please fill in all fields!";
  } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $error = "Invalid email format.";
  } elseif ($password !== $confirm) {
    $error = "Passwords do not match!";
  } else {
    $stmt = $conn->prepare("SELECT id FROM users WHERE username
= ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $stmt->store_result();

    if ($stmt->num_rows > 0) {
      $error = "Username is already taken.";
    } else {
      $hashedPassword          =          password_hash($password,
PASSWORD_DEFAULT);
      $stmt   =   $conn->prepare("INSERT   INTO   users   (username,
email, password) VALUES (?, ?, ?)");
```

```
$stmt->bind_param("sss", $username, $email, $hashedPassword);

      if ($stmt->execute()) {
        $_SESSION['user'] = $username;
        $_SESSION['userId'] = $conn->insert_id;
        $_SESSION['log'] = '1';
        header("Location: " . $_SERVER['PHP_SELF']);
        exit();
      } else {
        $error = "Registration failed: " . $stmt->error;
      }
    }
  }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
  <title>Sign Up</title>
  <link rel="stylesheet" href="../assets/css/style.css" />
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/boot
strap.min.css" rel="stylesheet" />
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.0/css/all.min.css" rel="stylesheet" />
</head>
<body class="body background-container">
  <div class="text-center py-4">
    <img src="../assets/logo.png" alt="logo" style="width:
270px;" class="mx-auto d-block">
  </div>

  <div class="container signup-container d-flex min-vh-90
justify-content-center align-items-center">
    <div class="row shadow rounded-4 overflow-hidden w-100"
style="max-width: 1000px;">
      <div class="col-md-6 d-flex flex-column justify-content-
center align-items-center bg-white p-4">
        <h2 class="mb-3 text-dark text-center fw-
semibold">Unlock Your Ideas</h2>
        <p class="mb-4 text-dark text-center">Save, organize,
and remember your thoughts effortlessly.<br>Get started now and
never lose a note again.</p>
        <img src="../assets/note3.png" alt="Illustration"
class="img-fluid" style="max-height: 300px;" />
      </div>
```

```
        <div class="col-md-6 bg-white p-4">
        <form method="POST" action="">
          <h4  class="text-center  mb-3  fw-semibold">Create  an
account</h4>

          <?php if (!empty($error)): ?>
            <div  class="alert  alert-danger  text-center  fw-bold"
role="alert">
              <?= htmlspecialchars($error) ?>
            </div>
          <?php endif; ?>

          <div class="mb-1">
            <label          for="username"          class="form-
label">Username</label>
            <div class="input-group">
              <span  class="input-group-text"><i  class="fa-solid
fa-user"></i></span>
              <input  type="text"  name="username"  id="username"
class="form-control" placeholder="Username" required />
            </div>
          </div>

          <div class="mb-1">
            <label for="email" class="form-label">Email</label>
            <div class="input-group">
              <span  class="input-group-text"><i  class="fa-solid
fa-envelope"></i></span>
              <input    type="email"    name="email"    id="email"
class="form-control" placeholder="example@email.com" required />
            </div>
          </div>

          <div class="mb-1">
            <label          for="password"          class="form-
label">Password</label>
            <div class="input-group">
              <span  class="input-group-text"><i  class="fa-solid
fa-key"></i></span>
              <input       type="password"       name="password"
id="password"     class="form-control"     placeholder="Password"
required />
            </div>
          </div>

          <div class="mb-4">
            <label    for="confirm"    class="form-label">Confirm
Password</label>
            <div class="input-group">
              <span  class="input-group-text"><i  class="fa-solid
fa-key"></i></span>
```

```
          <input   type="password"   name="confirm"   id="confirm"
class="form-control" placeholder="Confirm Password" required />
            </div>
          </div>

          <button type="submit" class="btn btn-primary w-100 mb-
3">Sign up</button>
          <div class="text-center">
            <small>Already     have     an     account?     <a
href="login.php">Login</a></small>
          </div>
        </form>
      </div>
    </div>
  </div>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootst
rap.bundle.min.js"></script>
</body>
</html>
```

Dashboard.php

```
<?php
include('dashboardIncludes/header.html');
require 'config.php';

if (!isset($_SESSION['userId'])) {
  header("Location: ../auth/login.php");
  exit();
}

$username = $_SESSION['user'];
$get_user_id = $conn->prepare("SELECT id FROM users WHERE
username = ?");
$get_user_id->bind_param("s", $username);
$get_user_id->execute();
$user_result = $get_user_id->get_result();
$user = $user_result->fetch_assoc();
$user_id = $user['id'];
?>

<div class="d-flex" style="min-height: 100vh;">
  </main>
  <!-- Sidebar -->
  <div class="bg-white shadow p-3" style="width: 300px;">
    <div class="d-grid gap-2 mb-3">
      <a href="note-add.php" class="btn btn-outline-dark w-
100">+ New Note</a>
```

```
<a href="calendar.php" class="btn btn-outline-success w-100"><i
class="bi bi-calendar-plus"></i> Add Event</a>
      <a href="shared-notes.php" class="btn btn-outline-info w-
100"><i class="bi bi-people-fill"></i> Shared Notes</a>
    </div>

    <form action="search.php" method="get" class="input-group
mb-4">
      <input type="search" name="query" class="form-control"
placeholder="Search...">
      <button class="btn btn-outline-secondary" type="submit"><i
class="fa fa-search"></i></button>
    </form>

    <h5 class="mb-3">Notes</h5>

    <?php
    $query = "
      SELECT notes.*, categories.name AS category_name,
             IF(note_pins.note_id IS NOT NULL, 1, 0) AS
is_pinned
      FROM notes
      LEFT JOIN categories ON notes.category_id = categories.id
      LEFT JOIN note_pins ON notes.id = note_pins.note_id AND
note_pins.user_id = ?
      WHERE notes.user_id = ?
      ORDER BY is_pinned DESC, notes.created_at DESC
    ";

    $stmt = $conn->prepare($query);
    $stmt->bind_param("ii", $user_id, $user_id);
    $stmt->execute();
    $result = $stmt->get_result();

    while ($note = $result->fetch_assoc()):
    ?>
      <div class="card mb-2 <?= $note['is_pinned'] ? 'bg-
warning-subtle border-warning' : '' ?>" style="max-width:
100%;">
        <div class="card-body p-2 d-flex justify-content-between
align-items-center">
          <div>
            <a href="note-view.php?id=<?= $note['id']; ?>"
class="fw-bold text-decoration-none text-dark">
              <?= $note['is_pinned'] ? '<i class="bi bi-pin-
angle-fill text-warning"></i>' : '' ?>
              <?= htmlspecialchars($note['title']); ?>
            </a>
            <?php if (!empty($note['category_name'])): ?>
              <div><small class="text-muted"><?=
htmlspecialchars($note['category_name']); ?></small></div>
```

```
          <?php endif; ?>
        </div>
        <div class="d-flex gap-2">
          <a href="note-edit.php?id=<?= $note['id']; ?>">
            <i class="bi bi-pencil-square text-secondary"></i>
          </a>
          <form action="pin-toggle.php" method="POST"
class="d-inline">
            <input type="hidden" name="note_id" value="<?=
$note['id']; ?>">
            <button type="submit" class="btn btn-sm p-0
border-0 bg-transparent">
              <i class="bi <?= $note['is_pinned'] ? 'bi-pin-
angle-fill text-warning' : 'bi-pin-angle text-secondary'
?>"></i>
            </button>
          </form>
          <form action="delete-note.php" method="POST"
class="d-inline" onsubmit="return confirmDelete();">
            <input type="hidden" name="note_id" value="<?=
$note['id']; ?>">
            <button type="submit" name="delete_note"
class="btn btn-sm p-0 border-0 bg-transparent">
              <i class="bi bi-trash text-danger"></i>
            </button>
          </form>
          <button class="btn btn-sm p-0 border-0 bg-
transparent" data-bs-toggle="modal" data-bs-target="#shareModal"
data-note-id="<?= $note['id']; ?>">
            <i class="bi bi-share text-primary"></i>
          </button>
        </div>
      </div>
    </div>
  <?php endwhile; ?>
</div>

<!-- Main content -->
<div class="flex-grow-1 p-4 bg-light">
  <div class="text-center mb-4">
    <h1 class="fw-bold">Welcome back <?=
htmlspecialchars($username); ?> 👋</h1>
    <p class="lead">Your personal note center is ready.
Prioritize. Succeed.</p>
  </div>

  <div class="d-flex justify-content-center mb-4">
    <form action="quick-add-note.php" method="POST" class="d-
flex" style="width: 60%;">
      <input type="text" name="quick_title" class="form-
control me-2" placeholder="Quick add a note..." required>
```

```
        <button type="submit" class="btn btn-
success">Add</button>
      </form>
    </div>

    <div class="container">
      <h3 class="mb-3">Recent Notes</h3>
      <div class="row row-cols-1 row-cols-md-3 g-4">
        <?php
        $recent_query = "SELECT id, title, created_at FROM notes
WHERE user_id = ? ORDER BY created_at DESC LIMIT 3";
        $stmt = $conn->prepare($recent_query);
        $stmt->bind_param("i", $user_id);
        $stmt->execute();
        $recent_result = $stmt->get_result();
        while ($recent = $recent_result->fetch_assoc()):
        ?>
          <div class="col">
            <div class="card h-100">
              <div class="card-body">
                <h5 class="card-title"><?=
htmlspecialchars($recent['title']); ?></h5>
                <p class="card-text"><small class="text-
muted">Created at: <?= $recent['created_at']; ?></small></p>
              </div>
              <div class="card-footer bg-transparent border-top-
0">
                <a href="note-view.php?id=<?= $recent['id']; ?>"
class="btn btn-outline-primary btn-sm">Open</a>
              </div>
            </div>
          </div>
        <?php endwhile; ?>
      </div>
    </div>
  </div>
</div>

<!-- Share Note Modal -->
<div class="modal fade" id="shareModal" tabindex="-1" aria-
labelledby="shareModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <form method="POST" action="note-share-handler.php">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="shareModalLabel">Share
Note</h5>
          <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
        </div>
        <div class="modal-body">
```

```
          <input type="hidden" name="note_id" id="modalNoteId">
          <div class="mb-3">
            <label for="share_with" class="form-label">Username
to share with:</label>
            <input type="text" class="form-control"
name="share_with" id="share_with" required>
          </div>
          <div class="form-check mb-3">
            <input class="form-check-input" type="checkbox"
name="can_edit" id="can_edit">
            <label class="form-check-label" for="can_edit">Allow
editing</label>
          </div>
        </div>
        <div class="modal-footer">
          <button type="submit" name="share_note" class="btn
btn-primary">Share</button>
          <button type="button" class="btn btn-secondary" data-
bs-dismiss="modal">Cancel</button>
        </div>
      </div>
    </form>
  </div>
</div>

<?php include('dashboardIncludes/footer.html'); ?>
```

Create-note.php

```
<?php
session_start();
require 'config.php';

if (!isset($_SESSION['userId'])) {
  header("Location: ../auth/login.php");
  exit();
}

if (isset($_POST['save_note'])) {
  $title = trim($_POST['title']);
  $content = trim($_POST['content']);
  $remind_at_raw = $_POST['remind_at'] ?? '';
  $category_id = !empty($_POST['category_id']) ?
(int)$_POST['category_id'] : null;
  $user_id = $_SESSION['userId'];

  // Перевірка обов’язкових полів
  if (empty($title) || empty($content)) {
    header("Location: note-add.php?error=empty_fields");
    exit();
  }
```

```
  // Обробка дати
  $remind_at = $remind_at_raw ? str_replace("T", " ",
$remind_at_raw) . ':00' : null;

  // Підготовка та виконання запиту на створення нотатки
  $query = "INSERT INTO notes (user_id, title, content,
remind_at, category_id)
            VALUES (?, ?, ?, ?, ?)";

  $stmt = $conn->prepare($query);
  $stmt->bind_param("isssi", $user_id, $title, $content,
$remind_at, $category_id);

  if ($stmt->execute()) {
    $note_id = $stmt->insert_id;

    // Логування
    include 'log-activity.php';
    log_activity($conn, $user_id, "Created a note with title:
$title");

    // Обробка вкладень
    if (!empty($_FILES['attachments']['name'][0])) {
      $total = count($_FILES['attachments']['name']);
      for ($i = 0; $i < $total; $i++) {
        $tmpFilePath = $_FILES['attachments']['tmp_name'][$i];
        $originalName =
basename($_FILES['attachments']['name'][$i]);

        if ($tmpFilePath) {
          $uploadDir = 'uploads/';
          if (!file_exists($uploadDir)) {
            mkdir($uploadDir, 0777, true);
          }

          $newFileName = uniqid('note_', true) . '_' .
$originalName;
          $filePath = $uploadDir . $newFileName;

          if (move_uploaded_file($tmpFilePath, $filePath)) {
            $insertAttachment = "INSERT INTO attachments
(note_id, file_path) VALUES (?, ?)";
            $attachStmt = $conn->prepare($insertAttachment);
            $attachStmt->bind_param("is", $note_id, $filePath);
            $attachStmt->execute();
          }
        }
      }
    }
```

```
    // Збереження тегів
    if (!empty($_POST['tags']) && is_array($_POST['tags'])) {
      $tag_query = "INSERT INTO note_tags (note_id, tag_id)
VALUES (?, ?)";
      $tag_stmt = $conn->prepare($tag_query);
      foreach ($_POST['tags'] as $tag_id) {
        if (is_numeric($tag_id)) {
          $tag_id = (int)$tag_id;
          $tag_stmt->bind_param("ii", $note_id, $tag_id);
          $tag_stmt->execute();
        }
      }
    }

    header("Location: note-success.php");
    exit();
  } else {
    header("Location: note-add.php?error=save_failed");
    exit();
  }
}
?>
```

Note-edit.php

```
<?php
include('dashboardIncludes/header.html');
require 'config.php';

if (!isset($_SESSION['userId'])) {
  header("Location: ../auth/login.php");
  exit();
}

$location = isset($_SESSION['search']) && $_SESSION['search'] ?
"./search?query="     .     urlencode($_SESSION['query'])     :
"./dashboard.php";
$user_id = $_SESSION['userId'];
?>

<div class="container mt-5">
  <div class="row">
    <div class="col">
      <div class="card">
        <div  class="card-header  d-flex  justify-content-between
align-items-center">
          <h3>Update note</h3>
          <div>
            <a  href="<?=  $location;  ?>"  class="btn  btn-danger
me-2">Back</a>
```

```
            <button   type="button"   class="btn   btn-outline-
primary"             data-bs-toggle="modal"             data-bs-
target="#commentModal">Add Comment</button>
          </div>
        </div>

        <div class="card-body">
          <?php
          if (isset($_GET['id'])) {
            $note_id = (int) $_GET['id'];

            $note_stmt  =  $conn->prepare("SELECT  *  FROM  notes
WHERE id = ? AND user_id = ?");
            $note_stmt->bind_param("ii", $note_id, $user_id);
            $note_stmt->execute();
            $note_result = $note_stmt->get_result();

            if ($note_result->num_rows > 0) {
              $note = $note_result->fetch_assoc();

              // Selected tags
              $tag_stmt  =  $conn->prepare("SELECT  tag_id  FROM
note_tags WHERE note_id = ?");
              $tag_stmt->bind_param("i", $note_id);
              $tag_stmt->execute();
              $tag_result = $tag_stmt->get_result();
              $selected_tags     =     array_column($tag_result-
>fetch_all(MYSQLI_ASSOC), 'tag_id');
          ?>
              <form    action="update-note.php"    method="POST"
enctype="multipart/form-data">
                <input  type="hidden"  name="note_id"  value="<?=
$note_id; ?>">

                <div class="mb-3">
                  <label>Reminder (date and time):</label>
                  <input  type="datetime-local"  name="remind_at"
class="form-control"   value="<?=   isset($note['remind_at'])   ?
date('Y-m-d\TH:i', strtotime($note['remind_at'])) : ''; ?>">
                </div>

                <div class="mb-3">
                  <label>Title:</label>
                  <input  type="text"  name="title"  class="form-
control"  required  value="<?=  htmlspecialchars($note['title']);
?>">
                </div>

                <div class="mb-3">
                  <label>Content:</label>
```

```
          <textarea      class="form-control"      name="content"
rows="10"    required><?=    htmlspecialchars($note['content']);
?></textarea>
                </div>
                  <div class="mb-3">
                  <label for="category_id">Category:</label>
                  <select   name="category_id"   id="category_id"
class="form-control">
                    <option   value="">—   Select   category   —
</option>
                    <?php
                    $cat_stmt  =  $conn->prepare("SELECT  id,  name
FROM categories WHERE user_id = ?");
                    $cat_stmt->bind_param("i", $user_id);
                    $cat_stmt->execute();
                    $cat_result = $cat_stmt->get_result();
                    while      ($category      =      $cat_result-
>fetch_assoc()) {
                      $selected   =   ($note['category_id']   ==
$category['id']) ? "selected" : "";
                      echo   "<option   value='{$category['id']}'
$selected>" . htmlspecialchars($category['name']) . "</option>";
                    }
                    ?>
                  </select>
                </div>

                <div class="mb-3">
                  <label>Current Attachments:</label>
                  <ul class="list-group">
                    <?php
                    $attach_stmt = $conn->prepare("SELECT * FROM
attachments WHERE note_id = ?");
                    $attach_stmt->bind_param("i", $note_id);
                    $attach_stmt->execute();
                    $attachments = $attach_stmt->get_result();
                    while     ($attachment     =     $attachments-
>fetch_assoc()):
                    ?>
                      <li class="list-group-item d-flex justify-
content-between align-items-center">
                        <a                                href="<?=
htmlspecialchars($attachment['file_path'])                    ?>"
target="_blank"><?= basename($attachment['file_path']) ?></a>
                        <a     href="delete-attachment.php?id=<?=
$attachment['id']  ?>&note_id=<?=  $note_id  ?>"  class="btn btn-sm
btn-danger">🗑</a>
                      </li>
                    <?php endwhile; ?>
                  </ul>
                </div>
```

```
                <div class="mb-3">
                  <label>Add more attachments:</label>
                  <div id="attachment-container">
                    <div class="input-group mb-2">
                      <input   type="file"   name="attachments[]"
class="form-control">
                      <button   type="button"   class="btn   btn-
danger remove-attachment">🗑</button>
                    </div>
                  </div>
                  <button    type="button"    class="btn    btn-
secondary" id="add-attachment">+ Add File</button>
                </div>

                <div class="mb-3">
                  <label for="tags">Tags:</label>
                  <select  name="tags[]"  id="tags"  class="form-
control" multiple>
                    <?php
                    $tags_stmt = $conn->prepare("SELECT id, name
FROM tags WHERE user_id = ?");
                    $tags_stmt->bind_param("i", $user_id);
                    $tags_stmt->execute();
                    $tags_result = $tags_stmt->get_result();
                    while ($row = $tags_result->fetch_assoc()) {
                      $selected      =      in_array($row['id'],
$selected_tags) ? 'selected' : '';
                      echo     "<option     value='{$row['id']}'
$selected>" . htmlspecialchars($row['name']) . "</option>";
                    }
                    ?>
                  </select>
                  <a      href="tag-add.php"      target="_blank"
class="btn btn-sm btn-outline-secondary mt-2 mb-2">+ Add Tag</a>
                </div>

                <div class="mb-3">
                  <label class="form-label">Checklist:</label>
                  <div id="checklist-container">
                    <?php
                    $checklist_stmt  =  $conn->prepare("SELECT  *
FROM checklist_items WHERE note_id = ?");
                    $checklist_stmt->bind_param("i", $note_id);
                    $checklist_stmt->execute();
                    $checklist_result     =     $checklist_stmt-
>get_result();
                    while     ($item     =     $checklist_result-
>fetch_assoc()):
                    ?>
```

```
                    <div   class="input-group   mb-2   checklist-
item">
```

Кінець додатку А

```
                    <div class="form-check">
                        <input            class="form-check-input
checklist-checkbox"  type="checkbox"  <?=  $item['is_checked']  ?
'checked' : '' ?> disabled>
                      </div>
                      <input    type="text"    name="checklist[]"
value="<?=  htmlspecialchars($item['content'])  ?>"  class="form-
control checklist-text">
                      <button   type="button"   class="btn   btn-
outline-danger remove-checklist">🗑</button>
                    </div>
                    <?php endwhile; ?>
                  </div>
                  <button  type="button"  class="btn  btn-outline-
secondary btn-sm" id="add-checklist">+ Add Item</button>
                </div>

                <hr>
                <h5>Comments:</h5>
                <?php
                $comment_stmt      =      $conn->prepare("SELECT
note_comments.content,  users.username,  note_comments.created_at
FROM  note_comments  JOIN  users  ON  note_comments.user_id  =
users.id  WHERE  note_id  =  ?  ORDER  BY  note_comments.created_at
DESC");
                $comment_stmt->bind_param("i", $note_id);
                $comment_stmt->execute();
                $comments = $comment_stmt->get_result();
                while ($comment = $comments->fetch_assoc()):
                ?>
                  <div class="border rounded p-2 mb-2">
                    <strong><?=
htmlspecialchars($comment['username']) ?>:</strong>
                    <p                            class="mb-1"><?=
nl2br(htmlspecialchars($comment['content'])) ?></p>
                    <small                  class="text-muted"><?=
$comment['created_at'] ?></small>
                  </div>
                <?php endwhile; ?>

                <div class="mb-3 mt-3">
                  <button  class="btn  btn-outline-warning  float-
end" type="submit" name="update_note">UPDATE</button>
                </div>
              </form>
          <?php
            } else {
```

```
            echo "<p class='text-danger'>Note not found or
access denied.</p>";
          }
        } else {
```

Кінець додатку А

```
          echo "<p class='text-danger'>No note ID
provided.</p>";
        }
        ?>
      </div>
    </div>
  </div>
</div>

<!-- Comment Modal -->
<div class="modal fade" id="commentModal" tabindex="-1" aria-
labelledby="commentModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <form method="POST" action="add-comment.php">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="commentModalLabel">Add
Comment</h5>
          <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
        </div>
        <div class="modal-body">
          <input type="hidden" name="note_id" value="<?=
$note_id; ?>">
          <textarea name="content" class="form-control" rows="4"
required placeholder="Write your comment..."></textarea>
        </div>
        <div class="modal-footer">
          <button type="submit" name="submit_comment" class="btn
btn-primary">Submit</button>
          <button type="button" class="btn btn-secondary" data-
bs-dismiss="modal">Cancel</button>
        </div>
      </div>
    </form>
  </div>
</div>

<script src="assetsjs/attachments.js"></script>
<script src="assetsjs/checklist.js"></script>
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: ***Розробка web-застосунку для ведення особистих нотаток та нагадувань з використанням HTML, CSS, JavaScript, PHP***

Обсяг пояснювальної записки 77 аркушів:

16 таблиць;

45 рисунків;

1 додаток.

Дата завершення роботи: *10 червня 2025р*.

Підпис студента- __  ___ *Савчук О.А.*