

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

Навчально-науковий інститут електронних та інформаційних технологій  
Кафедра інформаційних та комп'ютерних систем

Допущено до захисту

Завідувач кафедри

к.т.н., доцент Роговенко А.І.



« 14 » Червня 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

ВЕБ-ДОДАТОК ДЛЯ ВЕДЕННЯ НОТАТОК ІЗ МОЖЛИВІСТЮ  
КАТЕГОРИЗАЦІЇ І НАВІГАЦІЇ

Спеціальність 123 – Комп'ютерна інженерія  
Галузь знань 12 – Інформаційні технології

Виконавець:

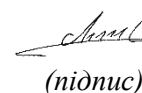
студент гр. КІ – 212

Ніконов Олександр Олексійович

  
(підпис)

Керівник:

Лисенко Дмитро Едуардович

  
(підпис)

Професор

Доктор. техн. наук

(посада)

(науковий ступінь, вчене звання)

Чернігів 2025

Я, Ніконов Олександр Олексійович, підтверджую, що дана робота є моєю власною письмовою роботою, оформленою з дотриманням цінностей та принципів етики і академічної доброчесності відповідно до Кодексу академічної доброчесності Національного університету «Чернігівська політехніка».

Я не використовував жодних джерел, крім процитованих, на які надано посилання в роботі.

\_\_\_\_\_12.06.2025\_\_\_\_\_

*Дата*

\_\_\_\_\_ 

*Підпис*

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

Навчально-науковий інститут електронних та інформаційних технологій  
Кафедра інформаційних та комп'ютерних систем



ЗАТВЕРДЖУЮ  
Завідувач кафедри  
к.т.н., доцент Роговенко А.І.

"\_06\_" \_\_Травня\_\_ 2025 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ  
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ**

Ніконов Олександр Олексійович

Тема роботи: ВЕБ -ДОДАТОК ДЛЯ ВЕДЕННЯ НОТАТОК ІЗ МОЖЛИВІСТЮ  
КАТЕГОРИЗАЦІЇ ТА НАВІГАЦІЇ

Тему затверджено наказом ректора  
від " 21 " Березня 2025 р. № 177 С/ВС

**1. Вхідні дані до роботи:**

Кількість створених облікових записів користувачів, категорій нотаток і нагадувань – без обмежень

Кількість UI-шаблонів та компонентів – до 50.

Операційна система Windows.

**2. Зміст розрахунково-пояснювальної записки:**

Кваліфікаційна робота складається з вступу, основної частини та висновків. Основна частина складається з трьох розділів: «Аналіз задачі створення системи», «Розробка системи» та «Реалізація системи».


**3. Демонстраційні матеріали:**

10 слайдів для презентації роботи

#### 4.Календарний план

№	Назва етапів роботи	Термін виконання	Примітки
1.	Визначитися з тематикою роботи	28.03.2025	
2.	Знайомство з проблемою, інформаційними джерелами, аналіз існуючих рішень	05.05.2025	
3.	Аналіз задачі створення системи	09.05.2025	Звіт
4.	Розробка бази даних та задач обробки даних	13.05.2025	
5.	Залік з переддипломної практики	16.05.2025	
6.	Розробка front-end проекту	18.05.2025	Звіт
7.	Реалізація системи	.05.2025	
8.	Підготовка текстової частини і слайдів	.05.2025	Звіт
9.	Попередній захист дипломної роботи	.06.2025	
10.	Завершення оформлення, рецензування, перевірка на плагіат	13.06.2025	
11.	Захист дипломної роботи	16.06.2025	

Завдання підготував:  
керівник

  
(підпис)

ПІБ Лисенко Д. Е.

«\_02\_» \_\_Травня\_\_\_\_2025 р.

Завдання одержав:  
студент

  
(підпис)

ПІБ Ніконов О. О.

«\_02\_» \_\_Травня\_\_\_\_2025 р.

## РЕФЕРАТ

Пояснювальна записка містить 66 стор., 41 рисунок, 8 джерел.

Об'єктом розробки є веб-додаток для персонального ведення нотаток.

Метою даної роботи є створення зручної та безпечної інформаційної системи у вигляді односторінкового веб-застосунку з функціями категоризації, повнотекстового пошуку, встановлення нагадувань. Результат розробки представлено у вигляді сукупності програмних документів, наведених у додатках до проекту.

Результатом роботи є реалізація таких можливостей додатку:

- повний цикл CRUD-операцій над нотатками;
- повнотекстовий пошук і комбінована фільтрація за категоріями й датою створення;
- налаштування push-нагадувань із cron-задачею на сервері та відображенням сповіщень у браузері;

Програма має сучасний, інтуїтивно зрозумілий адаптивний інтерфейс, реалізований за допомогою React (Vite) на клієнті та Node.js (Express) на сервері; застосовано CSS Grid і Flexbox для побудови макету. Для розробки використовували Visual Studio Code, мови JavaScript (ES6+) і SQL, фреймворки React, Express; бібліотеки bcrypt, jsonwebtoken, multer, nodemailer, axios; СУБД PostgreSQL; а також інструменти Vite і nodemon.

Робота інформаційної системи можлива на будь-яких операційних системах із сучасними веб-браузерами (Chrome, Firefox, Edge тощо).

Подальший розвиток передбачає розширення функціоналу через модульну архітектуру (плагіни для календарних сервісів, інтеграція з мобільними клієнтами, автоматична класифікація нотаток на основі машинного навчання), покращення інтерфейсу та реалізацію офлайн-синхронізації з хмарними сховищами.

Розробка має практичну цінність як універсальний інструмент для особистого використання, що сприяє підвищенню продуктивності та збереженню важливої інформації.

Ключові технології й інструменти: JavaScript, React, Node.js, Express, PostgreSQL, JSON Web Token, Vite, CSS Grid, Flexbox.

## ABSTRACT

The explanatory note comprises 67 pages, 41 figures, 8 references.

The subject of development is a web application for personal note-taking.

The goal of this work is to create a convenient and secure information system in the form of a single-page web application with features for categorization, full-text search, and setting reminders. The deliverable is presented as a set of software documents attached to the project appendices.

The application implements the following capabilities:

- Full CRUD cycle for notes (Create, Read, Update, Delete).
- Full-text search and combined filtering by category and creation date.
- Configuration of push reminders via a server-side cron job and display of notifications in the browser.

The program features a modern, intuitive, responsive interface, implemented using React (with Vite) on the client side and Node.js (Express) on the server side; CSS Grid and Flexbox are used for layout. Development tools and technologies include Visual Studio Code; JavaScript (ES6+) and SQL; the React and Express frameworks; the bcrypt, jsonwebtoken, multer, nodemailer, and axios libraries; a PostgreSQL DBMS; and the Vite and nodemon utilities.

The system can run on any operating system with a modern web browser (Chrome, Firefox, Edge, etc.).

Future development envisions expanding functionality via a modular architecture (calendar-service plugins, integration with mobile clients, automatic note classification using machine learning), improving the interface, and implementing offline synchronization with cloud storage.

This development has practical value as a universal tool for personal use, helping to boost productivity and preserve important information.

Key technologies and tools: JavaScript, React, Node.js, Express, PostgreSQL, JSON Web Token, Vite, CSS Grid, Flexbox.

## ЗМІСТ

Вступ .....	9
Розділ 1 .....	11
<b>АНАЛІЗ ЗАДАЧІ СТВОРЕННЯ ВЕБ-ДОДАТКУ ДЛЯ ВВЕДЕННЯ ОСОБИСТИХ НОТАТОК</b> .....	<b>11</b>
1.1.    Аналіз предметної області.....	11
1.1.1.    Актуальність проблеми.....	11
1.1.2.    Аналіз існуючих рішень .....	12
1.3.    Постановка задачі на розробку Веб-додатку .....	18
1.3.1.    Функціональні вимоги .....	20
Розділ 2 .....	22
<b>ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ ВВЕДЕННЯ ОСОБИСТИХ НОТАТОК</b> .....	<b>22</b>
2.1.    Вибір технічних засобів для реалізації додатку .....	22
2.1.1.    Вибір мови програмування та супутніх технологій.....	22
2.1.1.1.    Мова програмування JavaScript.....	22
2.1.1.2.    Фреймворк Express.js для серверної частини .....	24
2.1.1.3.    Фреймворк для клієнтської частини React .....	25
2.1.1.4.    Вибір середовища розробки.....	26
2.1.1.5.    Вибір СУБД.....	27
2.2.    Архітектура системи .....	28
2.2.1.    Логічна схема взаємодії компонентів .....	29
2.2.1.1.    Контекстна діаграма .....	29
2.2.1.2.    Компонентна діаграма.....	30
2.3.    Опис основних компонентів та сервісів.....	31
2.3.1.    Клієнтська частина .....	31
2.3.1.1.    Екран входу та відновлення паролю .....	31
2.3.1.3.    Сторінка нагадувань.....	35
2.3.1.4.    Послідовність дій користувача .....	36
2.3.1.5.    Профіль користувача.....	41
Розділ 3 .....	42
Реалізація системи.....	42
3.1.    Загальна архітектура та середовище розробки .....	42
3.1.1.    Огляд стеку технологій .....	42
3.1.2.    Структура проекту .....	43
3.1.3.    Конфігурація та запуск .....	45
3.2.    Реалізація бази даних.....	46
3.2.1.    Логічна модель даних .....	46

3.2.2.	Структура таблиць.....	48
3.2.3.	Первинні/зовнішні ключи та індекси .....	53
3.3.	Реалізація серверної частини .....	54
3.4.	Реалізація клієнтської частини .....	56
3.4.1.	Екран входу .....	56
3.4.2.	Загальний інтерфейс.....	57
3.4.3.	Створення нотатки .....	58
3.4.4.	Категоризація нотаток .....	60
3.4.5.	Пошук і фільтрація .....	61
3.4.6.	Нагадування .....	64
3.4.7.	Оформлення та адаптивність.....	66



## ВСТУП

Сфера цифрової організації особистої інформації стрімко розвивається. Користувачі в усьому світі щоденно зіштовхуються з потребою зберігати і впорядковувати особисті думки, завдання, ідеї та нагадування. У зв'язку з цим актуальності набувають цифрові засоби для ведення особистих нотаток, що забезпечують зручний доступ до даних у будь-який час і з будь-якого пристрою. Проте існуючі рішення часто не задовольняють усіх потреб користувачів — вони або надто складні, або не мають функціоналу нагадувань, або не підтримують категоризацію та пошук.

З розвитком інформаційних технологій та повсюдною доступністю інтернету виникає потреба у створенні зручних, функціональних і безпечних веб-платформ для ведення особистих записів. Багато сучасних додатків мають обмежену функціональність або не гарантують належний захист персональних даних. Наявність розрізнених сервісів, які реалізують лише частину потрібного функціоналу, спонукає користувачів шукати універсальні рішення.

Актуальність дослідження полягає у створенні веб-додатку, який би об'єднував в собі можливість зручного створення, редагування, категоризації та пошуку нотаток, а також встановлення нагадувань і захищеної аутентифікації користувачів. Така система повинна враховувати сучасні потреби користувача, тенденції в UI/UX-дизайні, принципи безпеки та ефективну архітектуру клієнт-серверної взаємодії.

Метою роботи є розробка та впровадження веб-додатку для особистих нотаток з можливістю категоризації, пошуку, нагадувань і аутентифікації користувачів, що забезпечить високий рівень зручності, надійності та функціональності.

Для досягнення поставленої мети слід виконати такі основні завдання: аналіз існуючих веб-сервісів для створення та ведення нотаток; визначення набору функціональних та технічних вимог до розроблюваної системи; розробка архітектурної схеми додатку та структури бази даних; вибір найбільш підходящих технологій для реалізації фронтенд та бекенд частин; реалізація функціоналу реєстрації користувачів; CRUD операцій із нотатками; керування категоріями пошуку та налаштування нагадувань; проведення тестування системи на відповідність функціональним вимогам; перевірка продуктивності та безпеки; підготовка рекомендацій щодо подальшого розвитку проєкту зокрема впровадження мобільної версії та інтеграцію з календарними сервісами. Предметом дослідження є розробка та реалізація повноцінного веб-додатку для управління особистими нотатками з використанням сучасних веб-технологій.

Задача полягає в поєднанні кількох функціональних напрямів (категоризація, пошук, нагадування, безпечна аутентифікація) в одному універсальному рішенні, яке базується на відкритому стеку технологій (React, Node.js, PostgreSQL, JWT) і відповідає сучасним вимогам щодо захисту персональних даних.

Практична значимість роботи полягає у створенні зручного, масштабованого і безпечного веб-додатку, який може використовуватися як приватними користувачами, так і в організаціях для особистих або робочих потреб. Розроблений додаток є гнучкою основою для подальшого розвитку сервісів цифрової організації інформації.

## Розділ 1

# АНАЛІЗ ЗАДАЧІ СТВОРЕННЯ ВЕБ-ДОДАТКУ ДЛЯ ВВЕДЕННЯ ОСОБИСТИХ НОТАТОК

## 1.1. Аналіз предметної області

### 1.1.1. Актуальність проблеми

У сучасному світі все частіше виникає потреба швидко та зручно впорядковувати особисті справи, ідеї, нагадування й списки завдань за допомогою цифрових інструментів. Паперові блокноти поступово відходять на другий план, адже веб- та мобільні додатки дозволяють синхронізувати інформацію між різними пристроями, здійснювати пошук за ключовими словами та оперативно створювати нагадування. Проте далеко не всі існуючі рішення одночасно поєднують зручність створення й редагування нотаток, гнучку систему категоризації, швидкий текстовий пошук, надійний механізм нагадувань і доступну, водночас безпечну авторизацію. Саме тому розробка власного веб-додатку для особистих нотаток із підтримкою категорій, пошуку й нагадувань сьогодні є не лише актуальною, а й доволі практичною задачею, особливо з огляду на швидкий розвиток сучасних фреймворків і зростаючі вимоги користувачів до індивідуальних інструментів підвищення продуктивності. Ключовим елементом комфортного робочого процесу є гнучка система категоризації: можливість призначати теги нотаткам, створювати вкладені папки та використовувати розумні фільтри дозволяє користувачам швидко звузити область пошуку. Наприклад, сортування за кольорами тегів або статусом виконання завдання допомагає миттєво зорієнтуватися у списку з десятків або сотень записів та зосередитися на найважливішому.

Розширена функція пошуку – ще один незамінний компонент: підтримка фразових запитів, логічних операторів (І, АБО, НЕ), виділення ключових слів у результатах і навіть пошук за змінами тексту (версіонування нотаток) робить роботу з великими обсягами інформації справді зручною. Завдяки цьому користувачі можуть виконувати складні запити, такі як «проект І бюджет НЕ затримка», та отримувати лише найактуальніші нотатки, не витрачаючи час на ручне переглядання списку.

Нагадування та планування. Інтегрований календар, який синхронізується з Календарем Google або Outlook, може перетворити статичні нотатки на активні завдання: просто встановіть дату та час для сповіщення, і програма сповістить вас, коли наближається термін виконання. Це особливо цінно для тих, хто поєднує особисті плани з робочими зустрічами або керує кількома проектами

одночасно. Не можна забувати про безпеку: користувачі часто зберігають не лише ідеї та списки справ, але й паролі, фінансові розрахунки чи приватні плани. Тому сучасний додаток повинен підтримувати надійні протоколи шифрування як у стані спокою, так і під час передачі. Для тих, хто потребує повного контролю над своїми даними, опції локального шифрування або бази коду з відкритим вихідним кодом можуть ще більше підвищити довіру до платформи. Кросплатформність означає, що будь-які зміни, внесені на одному пристрої, миттєво відображаються на всіх інших.

Хоча готові сервіси, такі як Evernote або OneNote, залишаються популярними, все більше користувачів шукають рішення, ідеально адаптовані до їхніх унікальних робочих процесів. Деяким потрібна глибока інтеграція з диспетчерами завдань, іншим потрібна підтримка сценаріїв для автоматизації рутинних операцій, а деяким критично потрібна LDAP або інша автентифікація корпоративного рівня. Впровадження системи плагінів або модульної архітектури дозволяє розширювати функціональність без зміни основної програми: від інтеграції чат-ботів для голосового ведення нотаток до модулів на базі штучного інтелекту, які автоматично підсумовують довгі тексти або пропонують теги на основі машинного навчання. Такий підхід не тільки знижує витрати на обслуговування, але й сприяє розвитку спільноти розробників, яка може постійно вдосконалювати платформу. Нарешті, продуктивність та зручність користування мають першочергове значення: інтерфейс повинен залишатися легким навіть за умови зберігання тисяч нотаток, а дизайн має бути інтуїтивно зрозумілим для користувачів будь-якого віку чи технічного досвіду. Використання сучасних фреймворків, таких як React або Vue, разом із оптимізованими запитами до серверної частини гарантує швидку роботу та плавну анімацію — функції, які особливо цінують користувачі мобільних браузерів.

Підсумовуючи, створення власної веб-програми для особистого ведення нотаток залишається не тільки актуальним, але й абсолютно необхідним. Тільки створивши рішення, яке відображає індивідуальні звички та робочі процеси кожного користувача, можна досягти справжньої ефективності та перетворити цифрові нотатки на незамінний щоденний інструмент.

### **1.1.2. Аналіз існуючих рішень**

На ринку доступно кілька поширених сервісів для ведення нотаток, кожен із яких має свої переваги та обмеження.

«Evernote» та «OneNote» пропонують широкий набір функцій: підтримку різноманітних медіафайлів, систему тегів, повнотекстовий пошук і синхронізацію між пристроями. Однак їхній інтерфейс часто здається складним і перевантаженим зайвими опціями, які не завжди потрібні тим, хто просто хоче швидко записати й організувати текстові замітки.

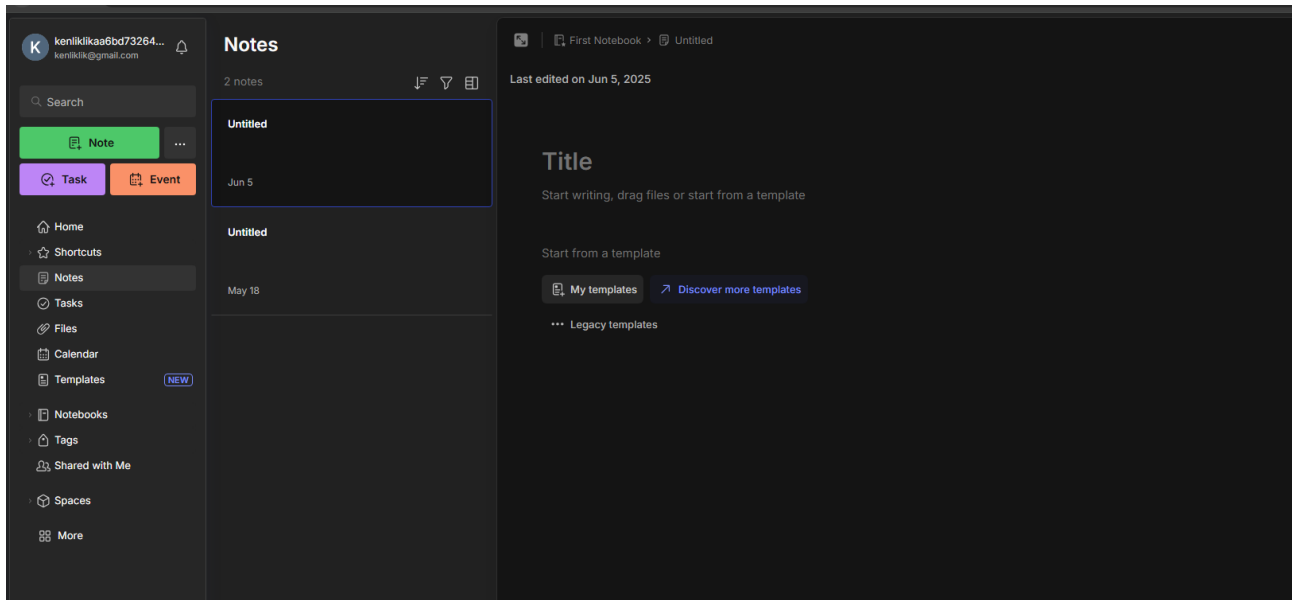


Рисунок 1.1 - Evernote

«Google Кеер» має дуже зрозумілий і спрощений інтерфейс у вигляді карток для нотаток. У ньому можна додавати нагадування й позначати записи кольоровими ярликами, але його система категоризації досить обмежена: доступні лише базові мітки та кольори. Крім того, гнучких фільтрів для пошуку майже немає, а потужну систему нагадувань реалізовано через інтеграцію з Google Календарем, що не завжди зручно.

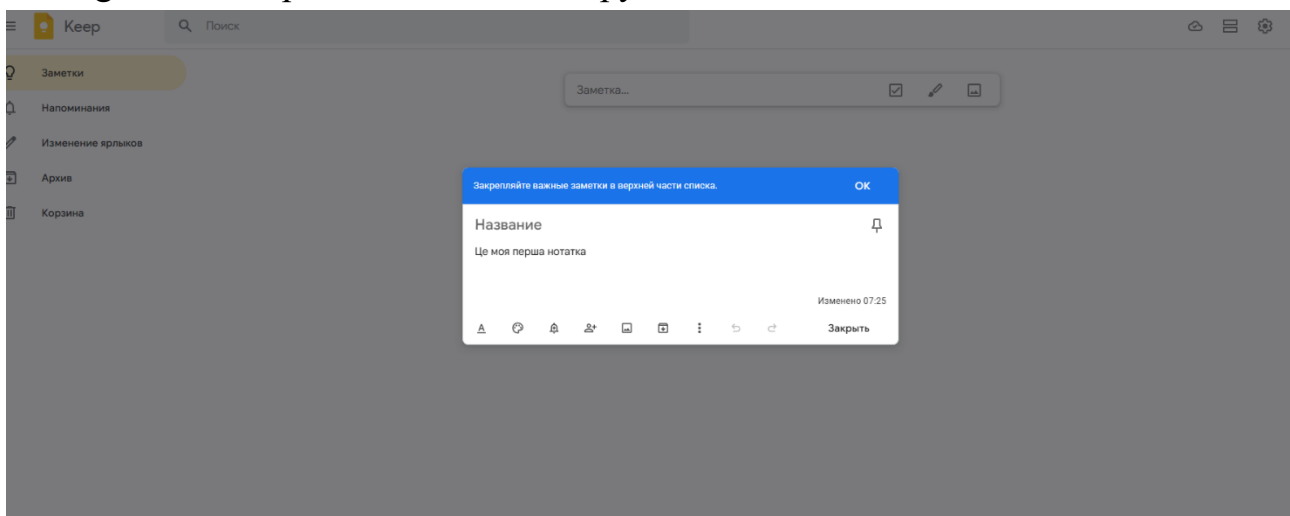


Рисунок 1.2 – Google Keep

«Simplenote» робить ставку на мінімалізм і швидку синхронізацію. Він підтримує прості теги й базовий пошук, але не має власного механізму нагадувань: неможливо прямо під час створення або редагування нотатки

встановити конкретну дату й час сповіщення. Це може бути критичною недоліком для тих, кому важливо отримувати вбудовані нагадування.

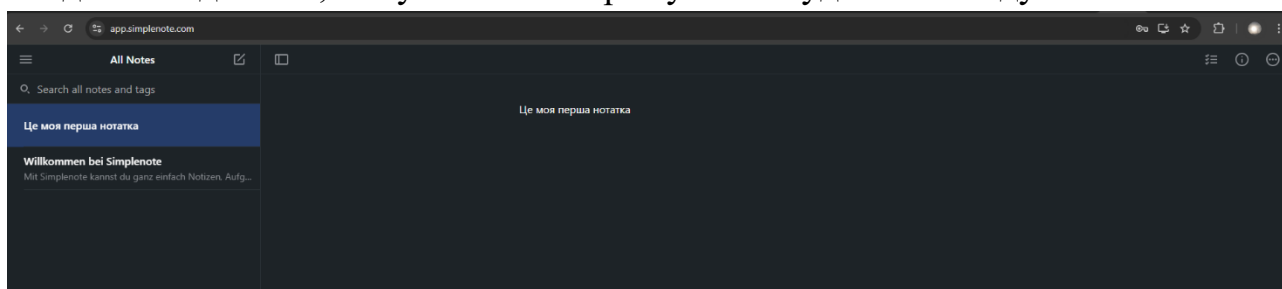


Рисунок 1.3 - Simplenote

«Notion» поєднує в собі нотатки, бази даних і простий таск-менеджер. Це справді універсальна система, але для новачка вона здається досить громіздкою: потрібно витратити чимало часу на налаштування сторінок, таблиць і зв'язків. Хоча в Notion можна створювати нагадування через поле дати в базі даних, автоматичного пуш-сповіщення там немає — доводиться самому стежити за наближенням дедлайнів.

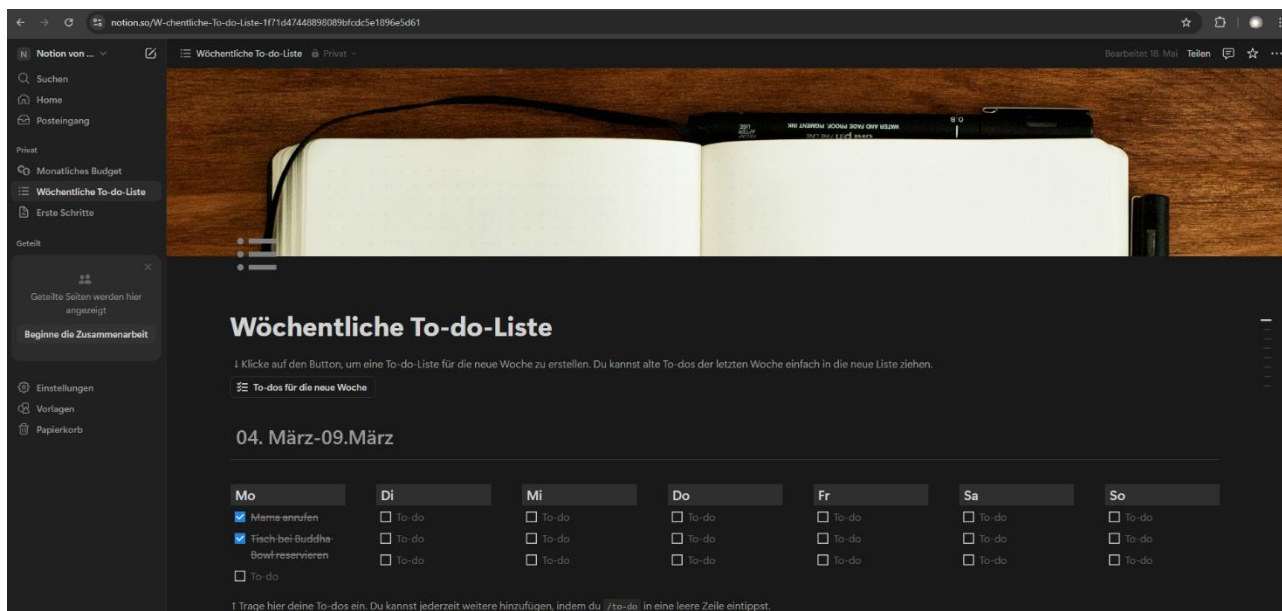
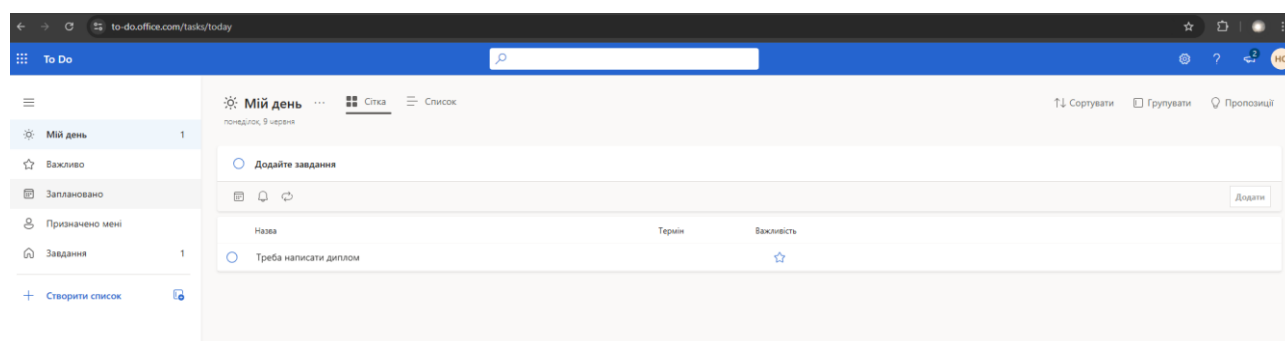


Рисунок 1.4 - Notion

«Microsoft To Do» більше орієнтований саме на завдання з дедлайнами, ніж на вільний формат нотаток. Категорії й дедлайни для тасків у нього є, але



неможливо створити довільно структуровану текстову нотатку так, як це дозволяють інші сервіси.

Рисунок 1.5 – Microsoft To Do

Таким чином, кожний із цих інструментів має як сильні сторони, так і обмеження:

- Evernote і OneNote добре підходять для універсального зберігання контенту з мультимедіа, але можуть відлякати тих, хто хоче просто робити текстові замітки без зайвого функціоналу.
- Google Keep вирізняється простотою, проте його можливості категоризації й пошуку доволі базові, а нагадування залежать від стороннього календаря.
- Simplenote приваблює мінімалізмом, але не дозволяє ставити нагадування безпосередньо в нотатці.
- Notion є надзвичайно гнучким, але через це може бути складним у налаштуванні, й автоматичних пуш-сповіщень у ньому немає.
- Microsoft To Do більше фокусується на тасках із дедлайнами, а не на довільних текстових нотатках.

Окрім згаданих раніше слабких сторін конкретних сервісів, можна також визначити більш загальні проблеми, що стосуються багатьох сучасних рішень для ведення нотаток:

#### 1. Фрагментація функцій та контекстів використання.

Багато програм зосереджені лише на одному аспекті – зберіганні тексту, списках справ або мультимедіа. Користувачеві доводиться перемикатися між різними інструментами, втрачаючи час та зосереджуючись на ручній синхронізації. В ідеалі всі основні функції – нотатки, завдання, нагадування, мультимедіа – повинні бути доступні в одному інтерфейсі, без постійного перемикавання між вкладками чи зовнішніми сервісами.

#### 2. Відсутність гнучкості в налаштуваннях інтерфейсу.

Багато програм пропонують лише кілька шаблонів або тем дизайну, які не завжди відповідають індивідуальним звичкам користувача. Для одних важлива компактна макетна схема «картки», для інших – лінійний список з датами та часом. Тому спеціалізований веб-додаток повинен дозволяти вибір макета, налаштовувати відображення полів і навіть підтримувати власні стилі CSS.

#### 3. Обмежена взаємодія з іншими інструментами та платформами.

Потужна екосистема продуктивності безперешкодно інтегрується з поштовими клієнтами, календарями, месенджерами та навіть менеджерами завдань. Якщо користувачі не можуть надсилати електронні листи, створювати зустрічі або ділитися записами з колегами безпосередньо з нотатки, ефективність страждає. Новий додаток має пропонувати відкритий API та інтегровані конектори до найпоширеніших сервісів.

#### 4. Проблеми зі зберіганням та відновленням даних.

Деякі сервіси дозволяють експорт, але лише у власних форматах, що ускладнює міграцію. Важливо ввімкнути імпорт та експорт у поширених форматах (Markdown, JSON, CSV) та забезпечити надійну систему резервного копіювання з історією версій.

#### 5. Відсутність адаптації до різних ролей користувачів.

Студенти, керівники проектів або фрілансери мають різні вимоги та права доступу. Система ролей та дозволів дозволяє створювати спільні блокноти з колегами в одному обліковому записі, тоді як приватні нотатки залишаються прихованими від інших учасників.

#### 6. Недостатня увага до доступності та локалізації.

Не всі сервіси підтримують доступний дизайн для людей з вадами зору або мобільності, а також часто бракує можливості змінювати мову інтерфейсу на льоту. Власний додаток повинен відповідати стандартам WCAG та бути готовим з самого початку до швидкої інтеграції додаткових мов. Це створює прогалину на ринку: сучасні інструменти або занадто захаращені та уповільнюють робочий процес, або вони занадто прості та не мають базових опцій. Розробка власного веб-додатку для особистих нотаток дозволяє:

- Поєднувати основні функції в масштабованому середовищі.
- Впроваджувати гнучку систему налаштувань та теми, що адаптуються до потреб кожного користувача.
- Забезпечувати надійну синхронізацію та відкриті конектори до популярних сервісів.
- Створювати розширювану систему плагінів, яка дозволяє швидко додавати нові модулі без зміни ядра.
- Гарантувати високий рівень безпеки та захисту даних, що відповідає як корпоративним, так і індивідуальним вимогам.
- Зосереджуватися на доступності та багатомовності з самого початку.

Ретельно проаналізувавши існуючі рішення та виявивши їхні спільні вразливості, можна сформулювати чітку основу для вимог до власного веб-додатку. Він має поєднувати легкість та мінімалізм простих програм для нотаток з потужністю професійних менеджерів завдань, а також відкритістю та адаптивністю корпоративних систем. Такий підхід забезпечує конкурентоспроможність розробленого продукту та робить його справді корисним для широкої бази користувачів.

Нище я розробив таблицю порівняння цих сервісів:

Таблиця 1.1 Аналіз сервісів



Критерії	Evernote	Google Keep	Simplenote	Notion	Розроблюваний проект <sup>17</sup>
Створення/редагування простих нотаток	+	+	+	+	+
Категоризація із кількома рівнями тегів	+	+	+	лише ручна перевірка полів	+(Папки/категорії)
Система нагадувань	-(Лише через сторонні інтеграції)	+( Через гугл календар )	-	+	+
Пошук/Фільтрація	+	+	+	+	+(повнота, категорії )
Аутентифікація користувачів	+	+	+	+	+(локальний реєстр + JWT )
Синхронізація між пристроями	+ мобільний десктоп	+ мобільний десктоп	+ мобільний десктоп	+ мобільний десктоп	(Орієнтовано на веб )
Простота інтерфейсу	-(Інтерфейс занадто перевантажений )	+( Дуже простий )	+	( дуже гнучкий але складний )	+

Підводячи підсумок, аналіз конкурентів показує, що більшість сервісів або перевантажені зайвими опціями (Evernote, Notion), або взагалі не мають вбудованих нагадувань (Simplenote), або пропонують лише примітивну категоризацію (Google Keep). Тож мені видається логічним зробити власний веб-додаток, де можна було б поєднати просту роботу з нотатками та необхідний мінімум “фіч”: текстові нотатки з розподілом за категоріями, швидкий повнотекстовий пошук і негайні сповіщення.

## 1.2. Аналіз вимог до інтерфейсу системи

Веб-додаток призначений здебільшого для звичайних людей, які ведуть свої особисті замітки: студентів, фахівців чи фрілансерів, тобто тих, хто записує ідеї, плани або щоденні списки справ. Інтерфейс має відповідати таким умовам:

- Лаконічність і читабельність: дизайн повинен бути настільки простим, щоб користувач міг зробити будь-яку базову операцію (перегляд списку нотаток, пошук, створення) максимум у два кліки.
- Адаптивність: сторінки обов'язково підлаштовуються під різні розміри екранів – як на десктопі, так і на мобільному, бо дуже часто користувачі додають нотатку “на ходу” зі смартфона.
- Мінімум зайвих елементів: замість складного меню повинна бути лише бічна панель із переліком категорій і помітна кнопка «Нова нотатка».
- Усі дії (CRUD нотатків, категорії) мають проходити в межах однієї сторінки або у модальному вікні, щоб не губити контекст і не відкривати багато вкладок.
- Режим пошуку: тут потрібна строка зверху, яка фільтрує нотатки у реальному часі, без додаткового натискання кнопки.

### **1.3. Постановка задачі на розробку Веб-додатку**

Розробити веб-додаток для введення та управління особистими нотатками з функціоналом категоризації, пошуку й нагадувань. Ключова мета — створити простір для текстових нотаток, доступний у будь-якому браузері, що забезпечує високу продуктивність та гарну роботу на різних пристроях.

Основні функції:

- Реєстрація/вхід (e-mail + пароль)
- CRUD-операції з нотатками
- Категоризація (категорії, теги, кольори)
- Пошук нотатки
- Нагадування (push-сповіщення)

Додатком може користуватись будь-який користувач для своїх цілей, все що потрібно зробити це зареєструватись. Функціонал абсолютно зрозумілий і простий. Для того щоб користуватись додатком не потрібно знання якоїсь системи чи деось шукати, тільки зареєструватись або зайти під своїм email.

Діаграма процесу заходу на веб-додаток представлена на рисунку 1.6



Рисунок 1.6 процес аутентифікації користувача

Після успішної аутентифікації користувача, коли користувач написав правильно свою електронну пошту, без зайвих символів в форматі @mail.com і звісно свій пароль, його данні заносяться в базу даних де обробляються і запам'ятовуються для подальших можливих входів.

На рисунку 1.7 показана ще одна діаграма з повними можливими діями користувача у додатку

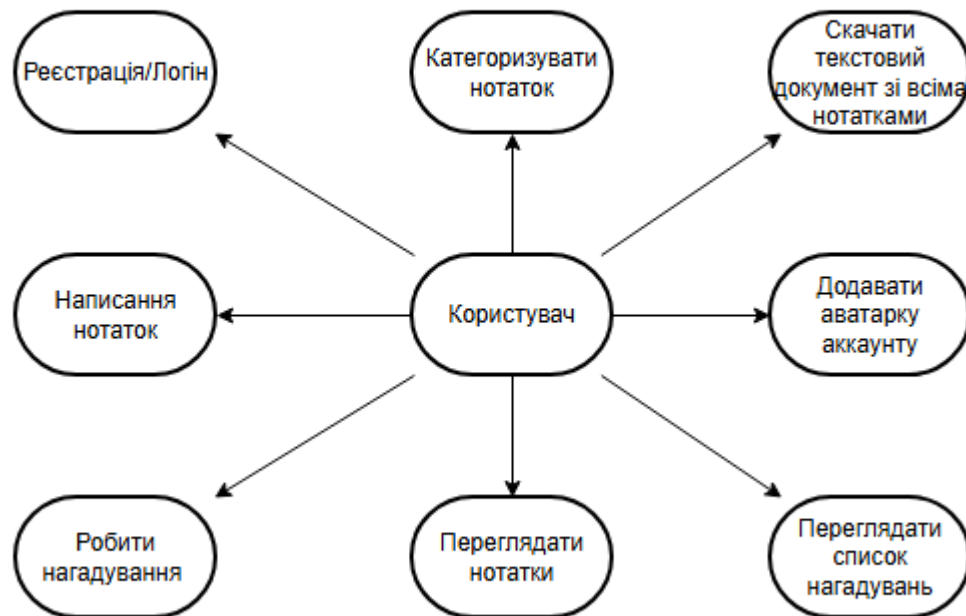


Рисунок 1.7 – діаграма всіх можливих дій користувача

### 1.3.1. Функціональні вимоги

1) Аутентифікація та реєстрація користувачів;

1. Реєстрація користувача за допомогою email і password;
2. Вхід у систему за допомогою JSON Web Token (JWT);
3. Можливість відновлення пароля.

2) Керування нотатками (CRUD)

1. Створення нотатки з обов'язковим полем «Заголовок» та обов'язковим полем «Зміст»;
2. Редагування нотаток зі зміненням заголовку, тексту, категорії та часу нагадування;

3. Видалення нотаток

3) Категоризація нотаток

1. Можливість створювати власні категорії;
2. Створювати назву й колір категорії;
3. Видалення категорії.
4. Категорії відображаються у вигляді кольорових назв

4) Пошук і фільтрація

1. Поле «Пошук» над нотатками має робити фільтрацію за ключовими словами у заголовку ;

2. Кліком на категорію в нижній панелі фільтруються нотатки тільки тієї категорії, яку обрали;
3. Можна комбінувати пошук і фільтрацію.
- 5) Нагадування
  1. У формі редагування нотатки є поле «Нагадати»;
  2. Якщо час нагадування вже пройшов, то під час наступного відкриття сторінки нотаток у браузері виводиться push-сповіщення (якщо користувач дозволив) ;
  3. На сервері працює cron-завдання яке щохвилини перевіряє всі нотатки з remindAt ;
  4. Після відправлення сповіщення автоматично встановлюється прапорець isReminded = true, щоб не надсилати повторно.
- 6) Особистий кабінет і захист даних
  1. Кожен користувач бачить лише свої нотатки;
  2. Якщо хтось спробує зайти на нотатку іншого користувача (введе URL із чужим id), сервер повертає 403 Forbidden;
  3. Сторінки “Вхід”/“Реєстрація” доступні всім, а всі інші маршрути — тільки для користувачів із дійсним JWT.

### **1.3.2. Нефункціональні вимоги**

- 1) Безпека
  - 1.1. Всі паролі зберігаються в базі даних у вигляді хешів;
  - 1.2. JWT діє 1 годину, можна додати refresh-токен на 7 днів;
  - 1.3. HTTPS має бути на рівні production, CORS налаштований тільки для домену фронтенда;
  - 1.4. Захист від SQL-ін'єкцій через параметризовані запити або ORM;
- 2) Продуктивність і масштабованість
  - 1.1. Час відповіді API під час пошуку й отримання нотаток не повинен перевищувати 200 мс при навантаженні до 10 000 нотаток;
  - 1.2. У базі даних використовуються індекси для полів title, content та remindAt;
- 3) У майбутньому можливо розбити на мікросервіси Юзабіліті
  - 3.1. Проєкт спроектувати так щоб в майбутньому можна було розбити його на мікросервіси
- 4) Юзабельність
  - 4.1. Лаконічний дизайн
  - 4.2. Підтримка доступності (a11y)
  - 4.3. Локалізація – Українська

## Розділ 2

# ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ ВВЕДЕННЯ ОСОБИСТИХ НОТАТОК

В цьому розділі буде показано процес вибору технологій і середовища розробки додатку. А саме вибір однієї мови програмування та інші інструменти. Надалі будуть конкретні фреймворки й сервіси.

### 2.1. Вибір технічних засобів для реалізації додатку

#### 2.1.1. Вибір мови програмування та супутніх технологій

##### 2.1.1.1. Мова програмування JavaScript

JavaScript — мова програмування з динамічною типізацією та автоматичним керуванням пам'яттю, спочатку розроблена для розширення можливостей веб-браузерів. Його синтаксис багато в чому успадковує конструкції C, Java, при цьому JavaScript підтримує як процедурний, так і об'єктно-орієнтований, а також функціональний стилі кодування.

На відміну від класичних клієнтських скриптів, JavaScript виконується не тільки в браузері, але і на сервері (за допомогою середовища Node.js), а також застосовується для створення утиліт командного рядка, десктоп- та мобільних додатків. Це дозволяє використовувати єдину мову у повному циклі розробки: від інтерфейсу до логіки та автоматизації фонових завдань.

Основні області застосування JavaScript:

Клієнтська сторона (інтерфейс користувача) - JavaScript безпосередньо контролює поведінку в браузері: він змінює DOM, реагує на кліки або введення з клавіатури та завантажує нові дані через Fetch без повного перезавантаження сторінки. Це створює плавний, інтерактивний досвід. Фреймворки, такі як React, або Angular, структурують цей процес за допомогою повторно використовуваних компонентів, маршрутів та керування станом, що полегшує розробку складних односторінкових застосунків.

Серверна сторона - За допомогою Node.js JavaScript також працює на сервері в керованій моделі. Це дозволяє ефективно реалізовувати HTTP-запити, доступ до файлів та операції з базою даних однією мовою. Популярні бібліотеки, такі як Express, Кoa або Нарі, надають проміжне програмне забезпечення для

маршрутизації, автентифікації та обробки помилок, що значно пришвидшує розробку REST або GraphQL API.

Автоматизація та командний рядок - JavaScript ідеально підходить для скриптів, що виконуються в терміналі: процеси збірки, тести, міграції баз даних, чи звичайні завдання cron – достатньо одного виклику, такого як `'node script.js'`. Цей набір інструментів дозволяє уникнути перебирання між кількома мовами та дозволяє легко налаштовувати конвеєри CI/CD.

Настільні додатки - Electron та NW.js поєднують Chromium та Node.js в окремий додаток. Розробники використовують HTML, CSS та JavaScript для створення кросплатформних графічних інтерфейсів – приклади включають Visual Studio Code та Slack. Це дозволяє виконувати швидкі ітерації та тісно інтегруватися з веб-технологіями.

Розробка мобільних пристроїв - Фреймворки, такі як React Native, Ionic або NativeScript, перетворюють код JavaScript на нативні елементи інтерфейсу користувача або запускаються у WebViews. Це дозволяє реалізовувати додатки для iOS та Android зі спільною логікою та компонентами, що можна використовувати повторно. Хоча вони не завжди досягають продуктивності чисто нативних додатків, вони пропонують величезні переваги економії часу та спільного використання коду.

Безсерверні та мікросервісні системи - У таких середовищах, як AWS Lambda, Google Cloud Functions або Azure Functions, достатньо завантажити окремі функції JavaScript. Вони автоматично масштабуються залежно від навантаження викликів і запускаються лише за потреби. Такий підхід сприяє слабкому зв'язку, зменшує накладні витрати на інфраструктуру. JavaScript підтримується на основних операційних системах (Windows, MacOS, Linux) і спокійно інтегрується з популярними веб-серверами (Apache, Nginx, IIS) через механізм зворотного “проху”. Оскільки формат даних JSON виник усередині екосистеми JavaScript, мова особливо зручна для створення WebSocket-API.

JavaScript було обрано, оскільки він дозволяє використовувати єдиний технологічний стек на стороні клієнта, так і на стороні сервера. Це скорочує час розробки та спрощує підтримку коду. Він також може похвалитися величезною екосистемою — тисячі попередньо створених бібліотек та утиліт у npm/yarn дозволяють швидко вирішувати практично будь-яку проблему. Сучасний синтаксис та стандарти (ES6+) і можливість інтеграції з TypeScript для статичної типізації роблять код більш читабельним та надійним. Сильна спільнота розробників та велика документація також сприяють швидкій розробці рішень та обміну досвідом.

Таким чином, застосування JavaScript у проекті дозволяє суттєво скоротити час розробки, спрощує підтримку коду та відкриває доступ до різноманітних інструментів та практик індустрії.

### 2.1.1.2. Фреймворк Express.js для серверної частини

Express.js — це простий, компонентний фреймворк для Node.js, спеціально розроблений для швидкого та легкого розгортання сучасних серверних додатків. Замість комплексної кодової бази, Express пропонує лише найнеобхідніше для маршрутизації та обробки HTTP — усі додаткові функції, такі як парсинг запитів, автентифікація або ведення журналу, можна додавати за бажанням за допомогою модулів проміжного програмного забезпечення. Ця модульна структура дозволяє точно налаштувати фреймворк відповідно до вимог без зайвого використання.

Підтримується добре відомий шаблон MVC: моделі даних можна реалізувати за допомогою встановлених ORM або ODM, таких як Sequelize та Mongoose, тоді як логіка контролера визначена у вигляді чітко структурованих маршрутів. Рівень представлення можна підключити через механізми шаблонів, такі як Pug або Handlebars, що дозволяє генерувати HTML на стороні сервера та одночасно надавати інтерфейси для фронтенд-фреймворків. Завдяки цій гнучкості Express підходить як для традиційних багатосторінкових веб-додатків, так і для чистих API-бекендів.

Однією з головних переваг Express.js є її швидкодія: оскільки фреймворк майже без змін користується рушієм V8 та неблокуючим event-loop у Node.js, надбавка до затримки виходить майже непомітною. Навіть коли надходить велика кількість запитів, Express справляється з навантаженням без проблем — кожен middleware працює як окрема функція й не створює зайвого «вагового» навантаження. А завдяки тисячам пакетів у npm, від модулів для безпеки до WebSocket-бібліотек, можна швидко додавати потрібний функціонал навіть для найскладніших завдань.

Завдяки активній спільноті Express.js постійно оновлюється відповідно до найновіших стандартів ECMAScript. Різноманітні навчальні матеріали, готові шаблони та рекомендації з кращих практик значно полегшують його освоєння, а регулярні оновлення підвищують безпеку, стабільність і сумісність. В результаті



Express.js залишається надійною, масштабованою та водночас легкою платформою для реалізації найрізноманітніших веб-проектів.

### **2.1.1.3. Фреймворк для клієнтської частини React**

Вибираючи фронтенд-фреймворк, я враховував не лише його популярність, але й особистий досвід і вимоги до проекту. Vue вражає своїм мінімалістичним синтаксисом і вбудованою реактивністю – компоненти з одного файлу можна легко вставляти в існуючі шаблони HTML, а прототипи створюються в найкоротші терміни. Однак, більші програми часто не мають обов'язкового структурного стандарту, тому проект може швидко стати заплутаним, якщо кожен розробник впроваджує власні концепції маршрутизації та управління станом.

Angular, з іншого боку, виглядає як комплексний архітектурний фреймворк: одразу після встановлення він пропонує типізацію TypeScript, вбудований HTTP-клієнт, ін'єкцію залежностей і реактивні потоки даних через RxJS. Однак на практиці початкове налаштування вимагає глибокого розуміння численних концепцій – від NgModule до Zone.js – і навіть незначні коригування можуть призвести до значних змін у багатьох файлах.

React знаходиться посередині: не універсальний фреймворк, а легка бібліотека з віртуальним DOM, яка гарантує максимальну продуктивність рендерингу. У моєму університетському проекті з React, Redux та React Router я на власні очі побачив, як компоненти як окремі модулі з чітко визначеними властивостями та локальним станом прискорюють розробку. Замість того, щоб переписувати код відповідно до нових вимог, мені потрібно було лише передавати змінені дані компоненту, і React оновив лише ті елементи DOM, які були фактично зачеплені.

Зрештою, я обрав React, тому що був знайомий з його екосистемою, а підхід «бібліотека плюс спеціалізовані пакети» ідеально відповідав гнучким вимогам мого проекту. Мої знання з React дозволив швидко налаштувати React

Router для маршрутизації та організувати локальний стан компонентів за допомогою хуків (`useState`, `useEffect`). Форми я перевіряю вручну без додаткових бібліотек, а стилі підключаю через звичайні CSS-файли. Завдяки великій та активній спільноті я завжди знаходжу відповіді на нестандартні помилки чи оптимізацію рендерингу на мобільних пристроях. Це дало змогу мені повністю зосередитися на реалізації бізнес-логіки замість вивчення зайвих абстракцій.

#### **2.1.1.4. Вибір середовища розробки**

Visual Studio Code - безкоштовне, кросплатформне інтегроване середовище розробки від Microsoft, засноване на платформі Electron. VS Code – це легкий, але сильний редактор для JavaScript, TypeScript, HTML, CSS та багатьох інших мов. Він пропонує IntelliSense з контекстно-залежним автодоповненням, аналіз коду на льоту з повідомленнями про помилки та попередження, а також швидкий доступ до переходів між визначеннями одним клацанням миші. Інтегрований налагоджувач дозволяє запускати та налагоджувати програми Node.js безпосередньо в редакторі, включаючи точки зупинки та перегляд стеку викликів.

Завдяки власній інтеграції Git та вбудованому терміналу команди контролю версій можна виконувати, не виходячи з редактора. Функціональність можна розширити за бажанням за допомогою тисяч розширень з Marketplace – від плагінів ESLint та Prettier для автоматичного зв'язування та форматування до інтеграцій Docker, Kubernetes або Unity.

Модульна архітектура дозволяє ідеально адаптувати IDE до відповідного технологічного стеку: людина може ввімкнути підсвічування синтаксису React або GraphQL, додати фрагменти коду для Express або налаштувати автоматизовані робочі процеси розгортання. Усі налаштування зберігаються у текстових файлах конфігурації (JSON/YAML), що спрощує комунікацію в команді.

За допомогою Visual Studio Code я можу швидко розпочати роботу без зайвих витрат, отримати гнучкі можливості налаштування та скористатися практичними функціями, які значно оптимізують мій щоденний робочий процес розробки. Також не забуваючи про свій особистий досвід з цією програмою. На протязі всього навчального процесу за часту я використовував саме Visual Studio Code для розробки багатьох програм. Це також дало великий вплив на вибір середовища розробки.

#### **2.1.1.5. Вибір СУБД**

Вибираючи між двома системами баз даних (PostgreSQL та MySQL), я враховував такі критерії, як структура даних, підтримка транзакцій та мій особистий досвід. Хоча я знайомий з обома движками, маю найбільший практичний досвід роботи з PostgreSQL, тому обрав саме цю систему. Реляційна модель забезпечує чітку організацію таблиць (користувачі, нотатки, категорії, нагадування) зі суворою цілісністю даних через зовнішні ключі, а також повну підтримку транзакцій ACID та складних запитів.

Схема бази даних ініціалізується за допомогою SQL-скрипта “db\\_init.sql”, який створює таблиці з типами даних ‘SERIAL’, ‘TEXT’, ‘TIMESTAMP’ та ‘UUID’, а також визначає зовнішні ключі та індекси для пришвидшення запитів та забезпечення цілісності.

Контролери для кожної сутності (Користувач, Примітка, Категорія, Нагадування) використовують параметризовані SQL-запити, які надійно запобігають SQL-ін'єкціям та забезпечують максимально прозору бізнес-логіку. Відмова від ORM на користь «чистих» SQL-інструкцій мінімізує накладні витрати та забезпечує негайне розуміння всіх операцій з базою даних.

Такий підхід дозволяє виконувати складні запити з кількома операціями ‘JOIN’ без додаткового рівня абстракції, надійно керувати транзакціями під час зміни пов'язаних записів та бачити систему горизонтально за допомогою реплікації та індексів у полях ‘created\_at’ та ‘remind\_at’.

Завдяки моєму великому досвіду роботи з PostgreSQL, я впевнений у надійності обраної бази даних та можу ефективно підтримувати та розвивати структуру даних протягом усього життєвого циклу проекту.

## 2.2. Архітектура системи

В цьому пункті описано те, як виглядає архітектура. Нижче на рисунку 2.1 наведено схему логіки взаємодії компонентів додатку від клієнта до бази даних

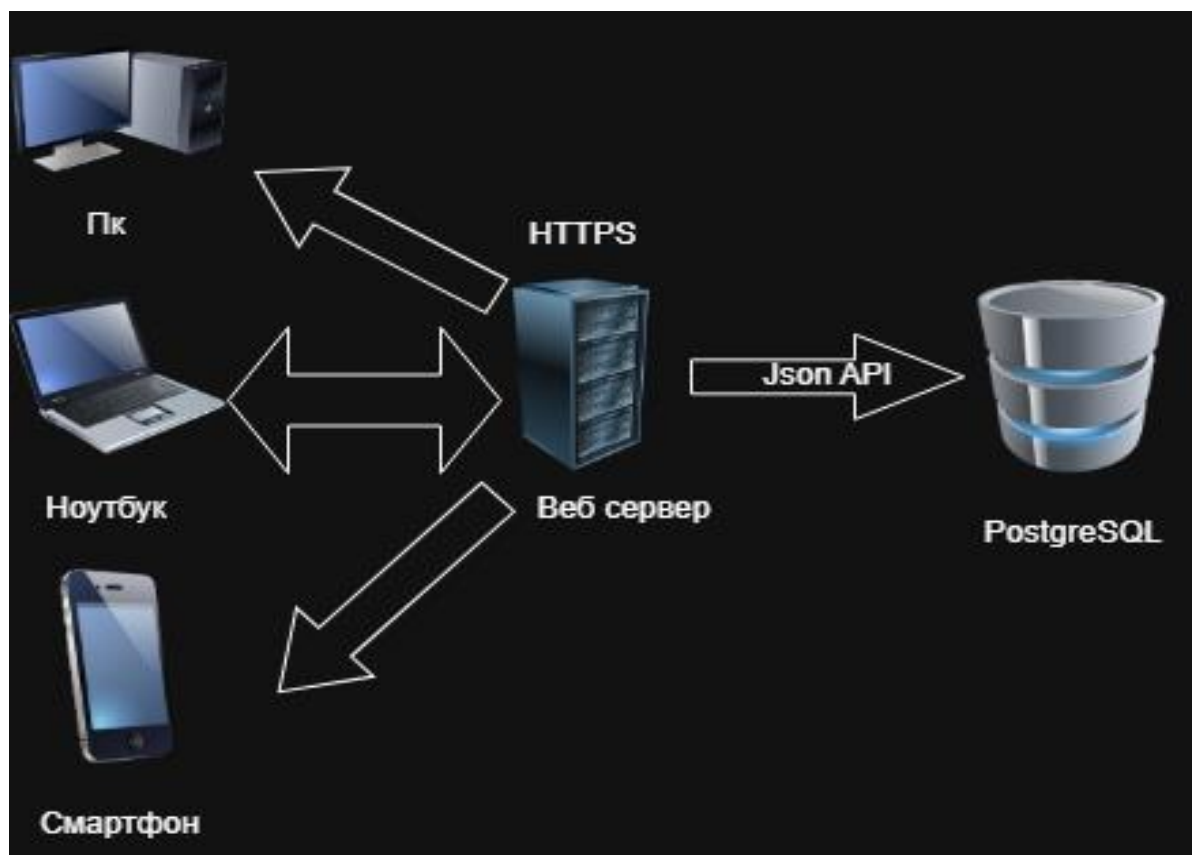


Рисунок 2.1 – Архітектура системи

На схемі видно, що клієнт (React SPA у браузері) звертається по захищеному HTTPS до сервера на Node.js + Express. Сервер спочатку перевіряє CORS та виконує JWT-автентифікацію, потім передає запит у відповідний контролер. Контролер обробляє запит і звертається до PostgreSQL через пул підключень pg, використовуючи параметризовані SQL-запити для гарантування цілісності даних.

Цей поділ на рівні клієнта, сервера, логіки контролерів та бази даних робить систему прозорою та гнучкою. HTTPS та reverse-proxy відповідають за безпечну доставку запитів, CORS і JWT контролюють доступ, Node.js + Express реалізують прикладну логіку, а PostgreSQL забезпечує надійне зберігання й швидке виконання складних запитів. Така архітектура спрощує масштабування та підтримку додатку, дозволяючи швидко додавати нові функції без переробок у суміжних компонентах.

## 2.2.1. Логічна схема взаємодії компонентів

### 2.2.1.1. Контекстна діаграма

У підпункті 2.2.1.1 наведено глобальну контекстну діаграму, яка демонструє межі системи, основних зовнішніх акторів і канали обміну даними між ними. Ця діаграма покликана надати швидкий огляд того, з чого складається додаток і як його частини взаємодіють на найвищому рівні.

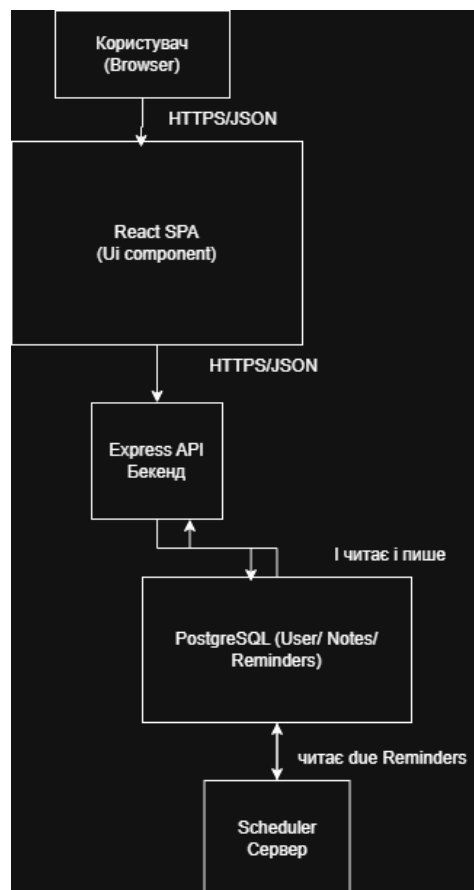


Рисунок 2.2 – Глобальна схема взаємодії компонентів системи

Після входу користувач потрапляє до головного вікна програми, де він бачить список своїх нотаток і може створювати нові записи або встановлювати

нагадування одним кліком миші. За кілька кроків відкривається форма, де потрібно ввести текст нотатки та вибрати потрібний час нагадування. Після надсилання всі дані негайно зберігаються на сервері.

На сервері у фоновому режимі працює не помітний процес, який відстежує майбутні нагадування. Щохвилини він перевіряє, які зустрічі заплановані і позначає їх як «готові до відображення». Це відбувається абсолютно непомітно, але гарантує, що нічого важливого не буде втрачено.

Клієнтська програма регулярно запитує сервер щодо нових нагадувань. Щойно з'являється нове нагадування, на екрані з'являється сповіщення або банер з текстом нотатки – як висновок не пропуститься жодного завдання.

Завдяки зрозумілому інтерфейсу, фоновому плануванню та миттєвим оновленням на стороні клієнта, користувач отримує надійні та зрозумілі нагадування, не турбуючись про технічні деталі.

#### 2.2.1.2. Компонентна діаграма

В цьому пункті зображена так звана Компонентна діаграма внутрішньої архітектури. Вона зображена на рисунку 2.3

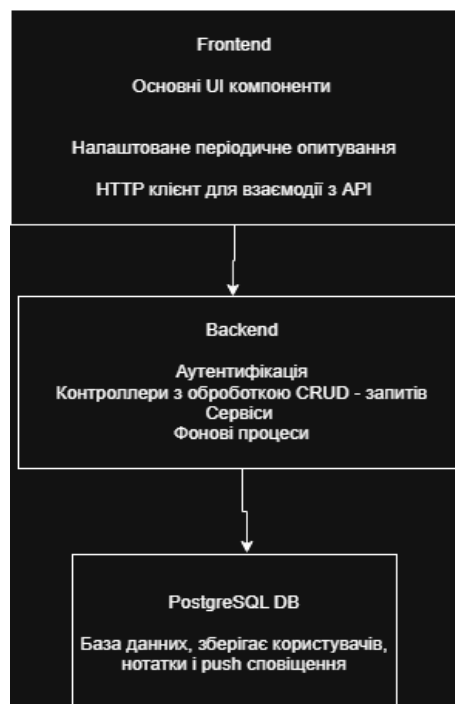


Рисунок 2.3 – Компонентна діаграма

В рамках дизайну проєкту, фронтенд формує єдину інтерактивну сторінку, на якій нотатки представлені у вигляді карток із заголовком, коротким текстом та візуальною категоризацією, і їх можна редагувати або створювати за допомогою інтуїтивно зрозумілої форми. Для кожної нотатки вводиться заголовок, пишеться багаторядковий текст та вибирається категорія. У заголовку інтерфейсу поєднуються кнопка огляду нагадування, відображення електронної пошти користувача, круглий значок аватара з функцією кліку для зміни зображення профілю та функція виходу. Всі запити до сервера спрямовуються через центральний HTTP-компонент, який автоматично додає заголовки автентифікації та забезпечує послідовну обробку помилок.

На стороні сервера REST API відповідає на вхідні HTTPS-запити. Спочатку він перевіряє їх на наявність дійсних веб-токенів JSON та визначених політик CORS за допомогою проміжного програмного забезпечення. Потім він передає їх спеціалізованим контролерам, які делегують завдання сервісам з ізольованою системною логікою. Ці сервіси обробляють усі необхідні перевірки, перевірки дозволів та транзакції. Вони зберігають дані в нормалізованій базі даних PostgreSQL, що містить окремі таблиці для користувачів, нотаток, категорій та нагадувань, а параметризовані запити та транзакції гарантують цілісність. Паралельно, окремий фоновий процес, що ініціалізується під час запуску сервера, гарантує, що всі нагадування з перевищенням часу спрацьовування вибираються з бази даних через регулярні проміжки часу, позначаються як оброблені та таким чином стають доступними для запитів фронтендом без впливу на веб-трафік. Це гарантує, що як запити API, так і фонові обробки обробляються через один і той самий доступ до бази даних, зберігаючи архітектуру спрощеною, високомасштабованою та легко розширюваною за допомогою додаткових функцій.

## **2.3. Опис основних компонентів та сервісів**

### **2.3.1. Клієнтська частина**

#### **2.3.1.1. Екран входу та відновлення паролю**

Початковий дизайн інтерфейсу користувача передбачає, що відвідувачів, які не ввійшли в систему, буде перенаправлено на один екран входу, що містить два поля введення електронної пошти та пароля, а також кнопки «Увійти» та «Зареєструватися». Якщо користувач вводить пароль менше ніж шість символів або якщо електронна адреса не відповідає базовому формату, безпосередньо над відповідним полем з'явиться повідомлення про помилку з коротким поясненням причини недійсності введення.

Поруч із цими елементами знаходиться посилання «Забули пароль?», яке веде на окрему сторінку лише з полем введення електронної адреси та кнопкою «Надіслати». Після підтвердження електронної адреси на вказану адресу надсилається одноразове посилання або токен для скидання пароля. Якщо посилання недейсне або термін дії минув, користувачеві відображається відповідне повідомлення про помилку та пропонується повторно запросити скидання пароля або звернутися до служби підтримки.

Таким чином, зручний зворотний зв'язок про помилки реалізовано на ранній стадії проекту, надаючи чіткі пояснення без необхідності розкриття технічних деталей.

Приблизний інтерфейс зображений на рисунку 2.4 у вигляді діаграми

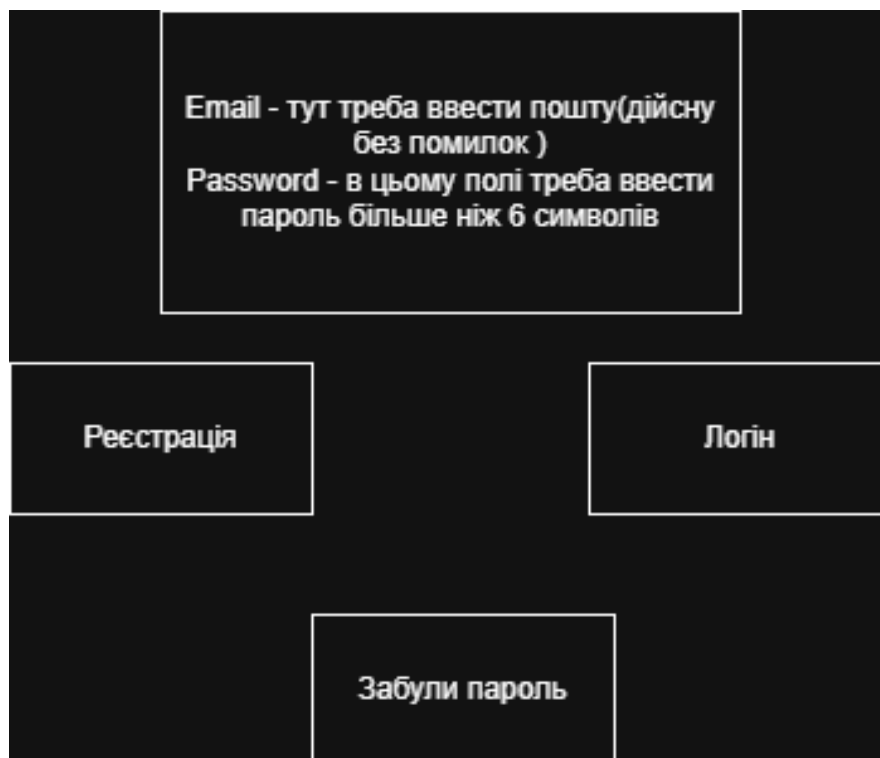


Рисунок 2.4 – приблизний інтерфейс авторизації/ реєстрації



У випадку якщо користувач забуде свій пароль, нажав на кнопку «Забули пароль» користувача перекине на сторінку з відновленням паролю, куди треба буде вписати свою пошту на яку прийде посилання з вікном на зміну пароля. Схему як буде виглядати процес зображено на рисунку 2.5



Рисунок 2.5 – схема відновлення паролю

### 2.3.1.2. Головний екран нотатки

Головний екран нотаток розділений на три функціональні області, розташовані одна над одною. Угорі розташована форма для створення нових нотаток, яка складається з двох полів введення для заголовка та основного тексту, а також випадаючого меню для вибору категорії. Кнопка «Створити нотатку» завжди видима, тому записи можна завантажувати на сервер одразу після введення.

Під нею розташована область для керування категоріями. Поле введення дозволяє додати нову категорію, а невеликий кольоровий індикатор поруч із ним

відкриває палітру для вибору кольору. Після підтвердження назви та вибору кольору нова мітка одразу з'являється у списку, а кнопка «Видалити», розташована безпосередньо поруч із нею, дозволяє видаляти категорії, які більше не потрібні.

У нижній частині динамічна сітка відображатиме всі існуючі нотатки у вигляді карток із заголовком, коротким текстом та назвою категорії. Під коротким текстом будуть відображатись значки «Редагувати» та «Видалити», а також кнопку «Нагадати», якщо для нотатки встановлено час. Для пошуку та фільтрації над сіткою доступне текстове поле та випадаюче меню «Усі категорії». Зміна фільтра гарантує відображення лише відповідних карток. Праворуч від елементів фільтра розташована кнопка «Скинути» для видалення всіх фільтрів та кнопка «Експортувати .txt» для завантаження нотаток у текстовому файлі.

У верхньому правому куті головного екрану передбачено реалізацію панелі користувача: відображення електронної пошти поточного акаунту поруч із круглим аватаром (завантажене фото), який при кліці відкриває папку з вибором файлів для зміни зображення профілю, а також кнопку «Вийти». Біля цих елементів розміщено посилання на сторінку «Нагадування», що забезпечує швидкий перехід до повного списку запланованих та виконаних подій. Така панель робить навігацію та керування профілем максимально зручними та інтуїтивними.

Усі компоненти мають узгоджений дизайн з м'якими тінями, точними інтервалами та кнопками високої контрастності. Плавна анімація під час нажимання на кнопки, виділення карток та відображення значків забезпечує адаптивний, сучасний користувацький інтерфейс, а адаптивний макет гарантує правильне відображення на екранах усіх розмірів.

На рисунку 2.6 зображений ескіз майбутнього інтерфейсу головної сторінки

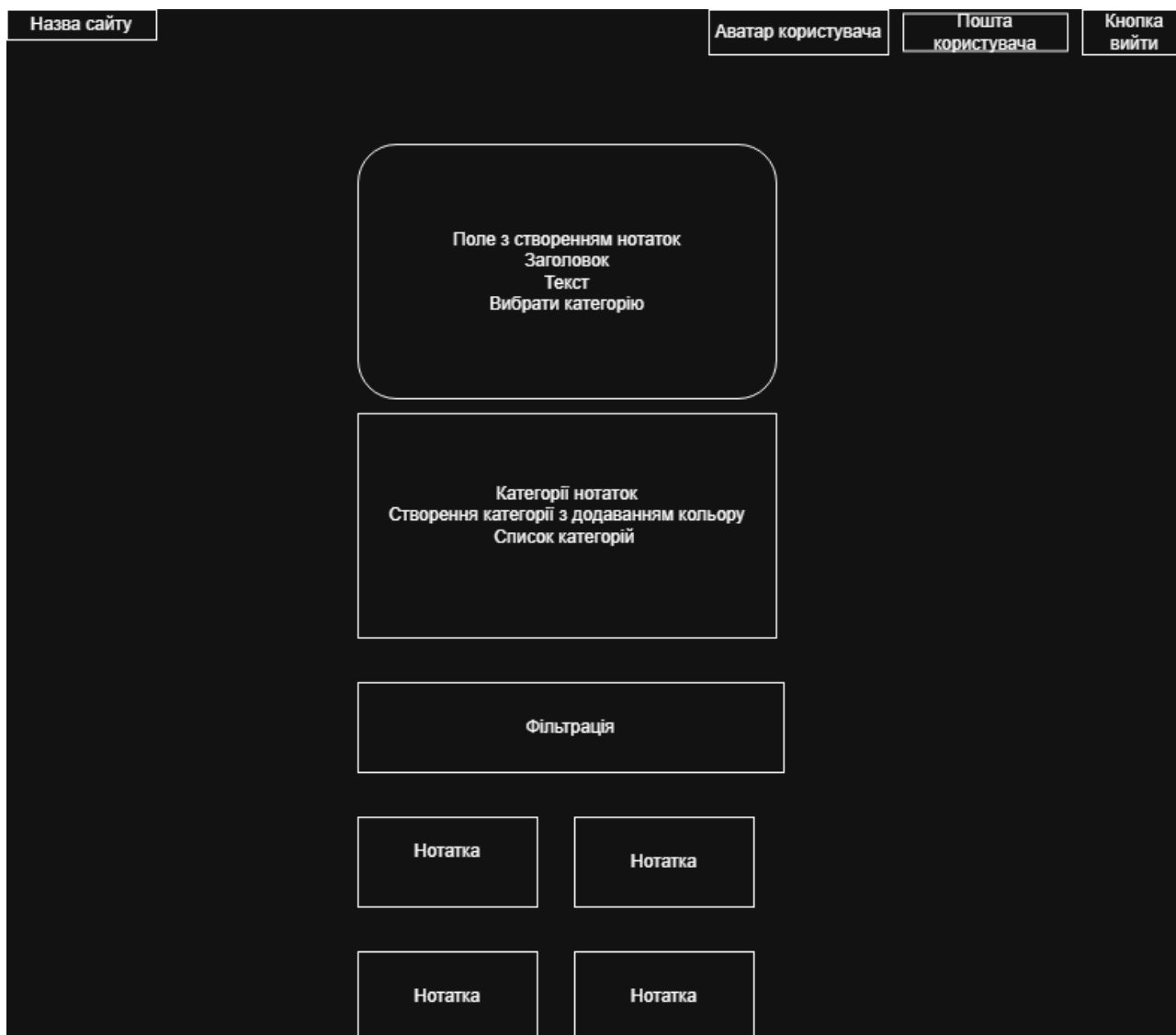


Рисунок 2.6 ескіз головної сторінки

### 2.3.1.3. Сторінка нагадувань

Головним елементом сторінки нагадувань буде область із двома вкладками, які дозволяють перемикатися між Активними та Виконаними нагадуваннями. Кожна вкладка відображає порядковий індекс та кількість елементів у дужках, що дає миттєве уявлення про обсяг завдань. Безпосередньо під вкладками розташована кнопка «Видалити все», яка призначена для одночасного очищення списку нагадувань без необхідності видаляти їх поодиночці.

Якщо на поточний момент є нагадування, кожне з них представлено у вигляді компактної картки з чітко відформатованим часовим штампом (дата й

час спрацювання). Під часовою міткою розташовані дві кнопки дій: «Редагувати», що відкриває рядок вводу нового часу у вигляді текстового поля з календарем, та «Видалити», яка дозволяє швидко прибрати одиничну подію. У режимі редагування з'являється панель із полем для зміни дати й часу, а також кнопки «Скасувати» та «Зберегти», що надають користувачеві контроль над відкладеним плануванням.

У випадку, коли жодного нагадування не існує в вибраному режимі, під кнопкою масових дій відображається повідомлення «Нагадувань немає», виконане у приглушеному відтінку, щоб воно не відволікало увагу, але водночас інформувало про відсутність записів.

Ескіз сторінки з нагадуваннями зображено на рисунку 2.7

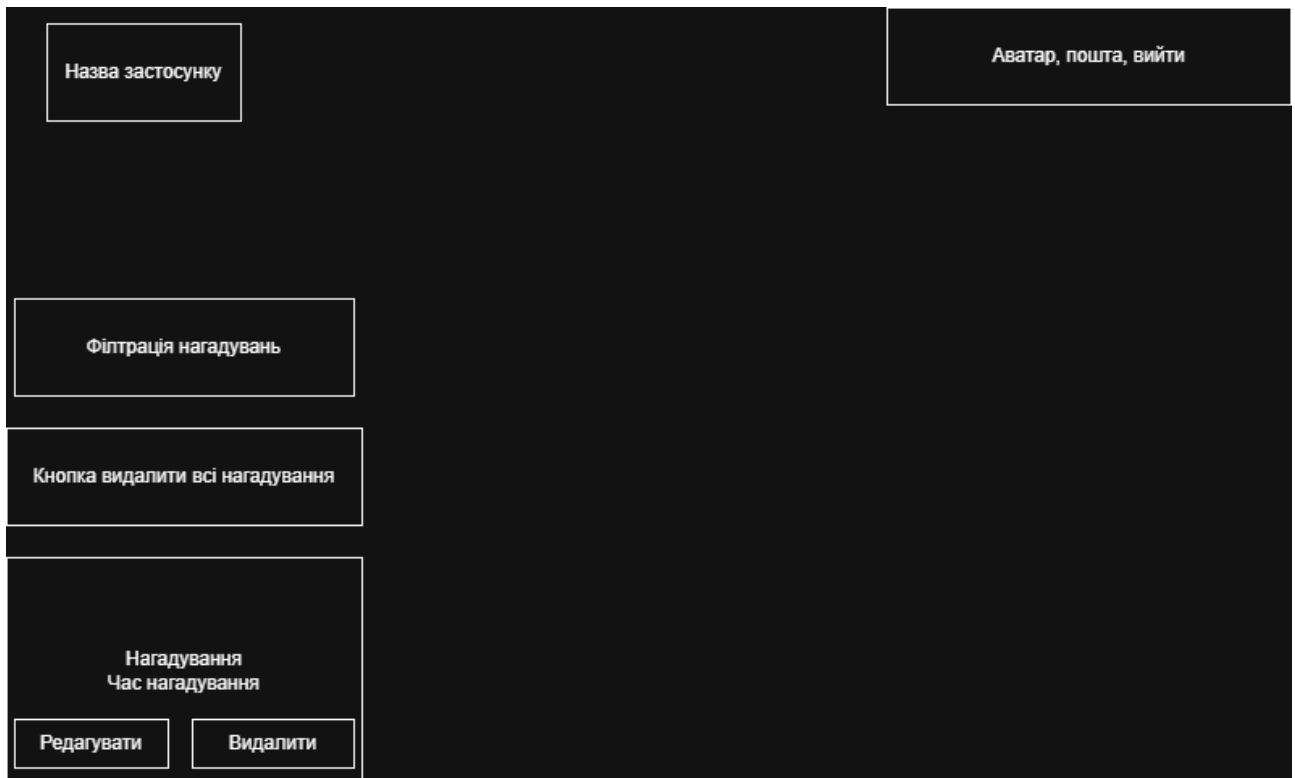


Рисунок 2.7 – ескіз сторінки з нагадуваннями

#### 2.3.1.4. Послідовність дій користувача

У цьому підпункті наведено детальні схеми основних послідовностей дій користувача під час роботи з веб-додатком. Кожна із діаграм ілюструє окремий

сценарій й показує, як за допомогою інтерфейсу можна фільтрувати й шукати нотатки, керувати створенням, редагуванням і видаленням записів та категорій, а також налаштовувати нагадування:

На рисунку 2.8 зображена схема створення нотатки та якщо потрібно категорії

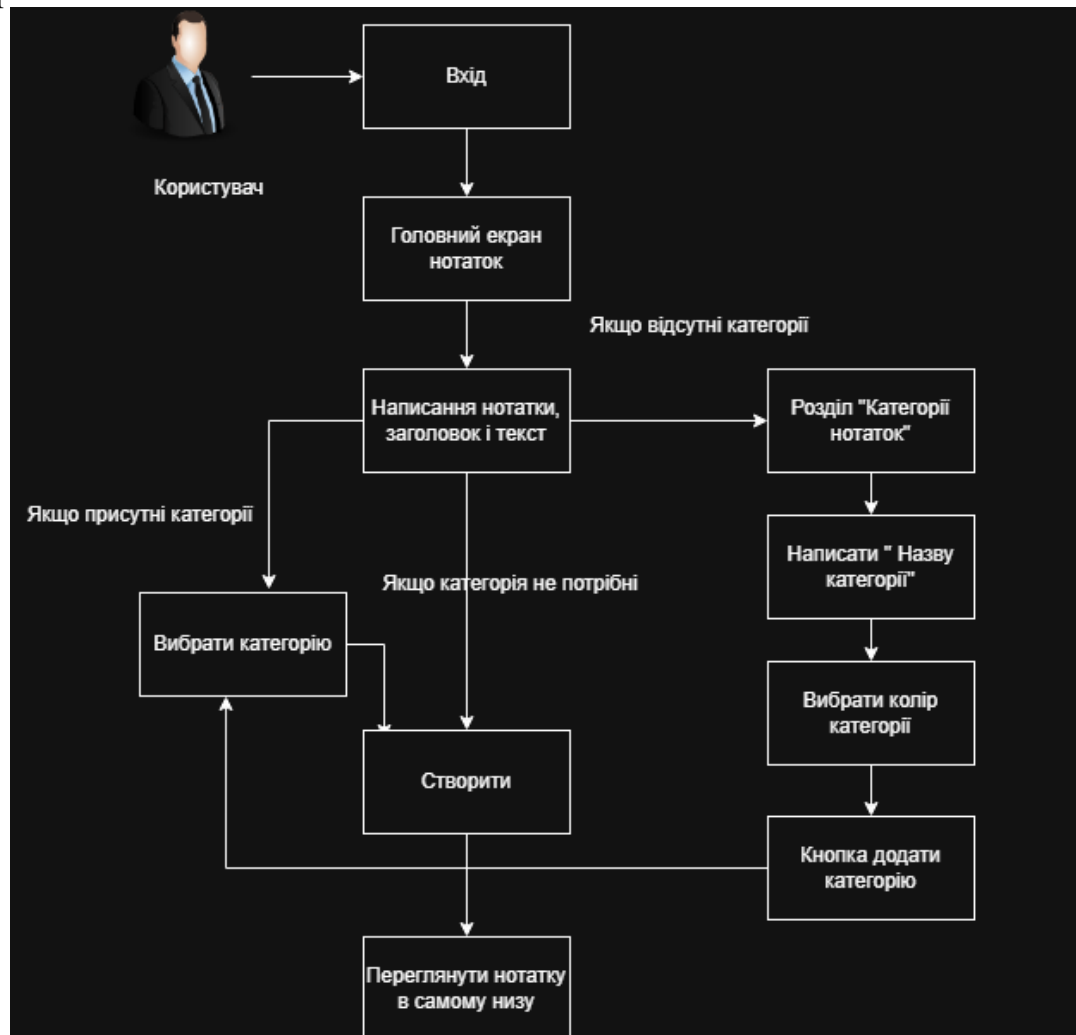


Рисунок 2.8 – створення нотатки, за потребою категорії

Користувач після входу переходить до головного екрану й натискає «Створити нотатку». Якщо в списку категорій є потрібна мітка, він просто обирає її та зберігає новий запис. Якщо потрібних категорій немає, переходить у розділ «Категорії нотаток», вводить назву, вибирає колір і натискає «Додати категорію», після чого повертається до форми нотатки та завершує створення.

На рисунку 2.9 зображена схема фільтрації та пошуку нотатки



Рисунок 2.9 – фільтрація/пошук нотатки

Після того, як користувач на головному екрані хоче переглянути або знайти нотатку він спускається вниз. Над списком нотаток буде поле Пошуку, Фільтрації по категоріям і даті нотатки. Введення ключового слова та/або вибір категорії одразу звужує перелік карток до тих, що відповідають запиту, без перезавантаження сторінки.

Користувач зможе робити дії з вже готовими нотатками, схема зображена на рисунку 2.10

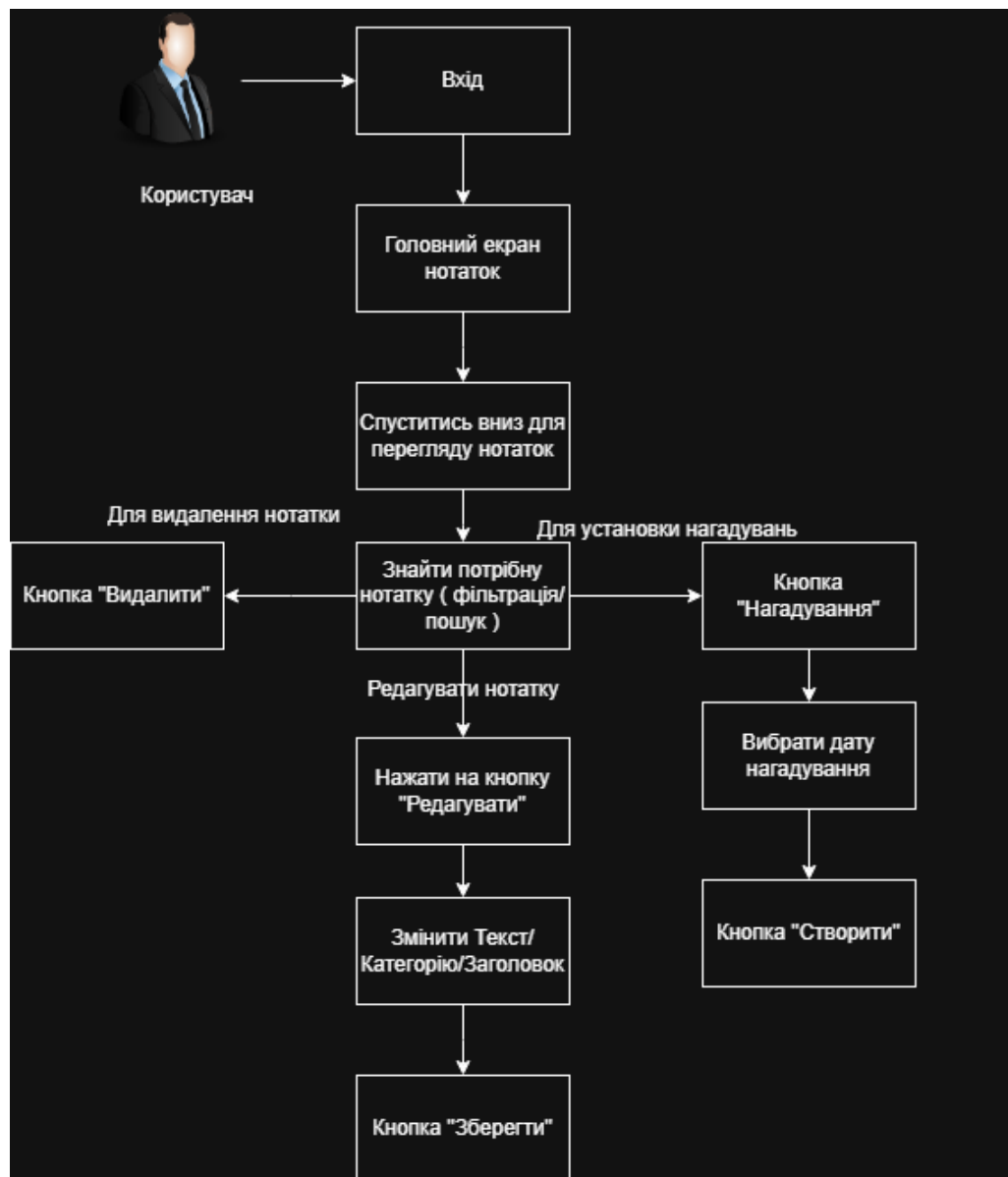


Рисунок 2.10 – робота з кнопками в полі нотаток

Для кожної картки нотатки при наведенні будуть доступні три кнопки: «Редагувати» відкриває форму з поточними даними, «Видалити» видаляє запис із підтвердженням, а «Нагадати» викликає інтерфейс вибору дати й часу нагадування. Тобто користувач в будь який момент при перегляді нотатки може її редагувати, змінити тіло нотатки ( Зміст ) змінити заголовок нотатки, або саму категорію нотатки. Нагадувань до однієї нотатки можна робити безліч.

На рисунку 2.11 зображено схему дій користувача для редагування і видалення нагадувань

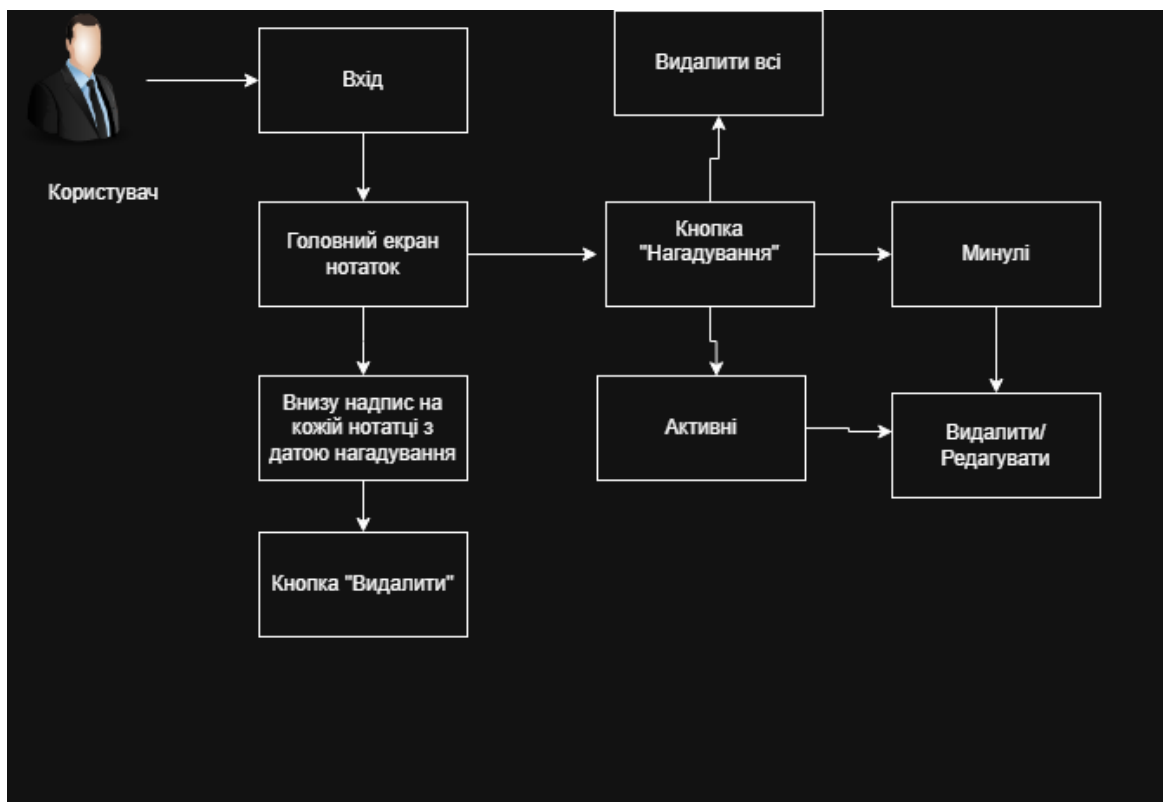


Рисунок 2.11 – редагування/видалення нагадувань

На сторінці «Нагадування» користувач може переключатися між активними та сплившими записами, редагувати час існуючих нагадувань або видаляти їх по одному. Масова кнопка «Видалити все» дозволяє одним кліком очистити усі записи у поточній вкладці. Також завжди буде можливість просто видалити нагадування спустившись до списку нотаток, там будуть розташовані нагадування з датою і кнопкою «Видалити»

На рисунку 2.12 зображена схема видалення категорій



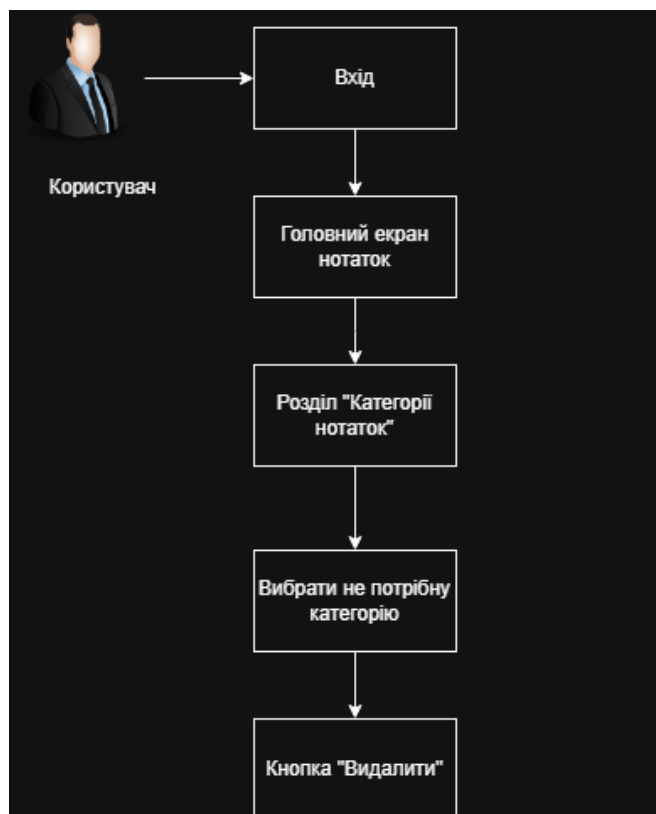


Рисунок 2.12 – схема видалення категорій

У розділі «Категорії нотаток» біля кожного елемента буде кнопка «Видалити». Користувач обирає непотрібну категорію та видаляє її, після чого всі нотатки, які були до неї прив’язані, залишаються доступними без кольорового маркера і підпису з назвою категорії до повторного розподілу.

### 2.3.1.5. Профіль користувача

У цьому проєкті керування власним профілем буде не просто технічною необхідністю, а центральним елементом для емоційного благополуччя користувача. Тому я планую розмістити круглий аватар у верхній частині інтерфейсу користувача, спочатку відображаючи ініціали користувача. Натискання на нього відкриває просте діалогове вікно для вибору файлу: користувачі можуть завантажити потрібне зображення за допомогою файлового менеджера. Після підтвердження зображення профілю оновлюється негайно без перезавантаження сторінки, таким чином передаючи відчуття особистого простору та індивідуальності.

Поруч із ним знаходиться кнопка «Вийти», яка є єдиним і чітко зрозумілим механізмом завершення сеансу. Натискання на неї очищає всі дані сеансу та повертає користувача до екрана входу. Цей простий крок — «ключ від дверей» програми — забезпечує впевненість у тому, що користувач завжди контролює, хто і коли має доступ до системи, а також зміцнює впевненість у безпеці персональних даних.

Особливо важливо, щоб сповіщення керувалися виключно за допомогою вбудованих функцій браузера. З першої ж нагадування користувачеві відображається діалогове вікно, в якому він може дозволити або заборонити показ веб-push-сповіщень. Такий підхід дозволяє уникнути зайвих меню налаштувань у додатку та зберігає дизайн чітким та лаконічним.

Можливість змінити зображення профілю та безпосередньо вийти з облікового запису – це не лише технічні функції, а й важливі емоційні елементи. Персоналізоване зображення профілю створює відчуття зв'язку з додатком, а швидкий та надійний механізм виходу забезпечує безпеку. Ця взаємодія індивідуальності та контролю гарантує, що користувачі почуватимуться бажаними гостями та захищеними з самого початку.

## **Розділ 3**

### **РЕАЛІЗАЦІЯ СИСТЕМИ**

#### **3.1. Загальна архітектура та середовище розробки**

##### **3.1.1. Огляд стеку технологій**

Для зберігання даних була обрана СУБД PostgreSQL, яка пропонує високу надійність, складні механізми індексації та оптимізацію запитів – це є критично важливим для швидкого пошуку та сортування великих обсягів нотаток.

Клієнтська частина базується на React у поєднанні з Vite. Таке поєднання забезпечує миттєве оновлення модулів під час розробки та створює

мінімальні пакети для робочого середовища. Компонентна структура React спрощує створення та підтримку елементів інтерфейсу користувача, які можна використовувати повторно, тоді як Vite автоматично оптимізує код відповідно до чинних стандартів.

Проект використовує кілька допоміжних бібліотек: `bcrypt` забезпечує безпеку облікових записів завдяки надійному хешуванню паролів, `jsonwebtoken` генерує та перевіряє токени JWT, `multer` обробляє завантаження файлів (наприклад, аватари користувачів), а `nodemailer` забезпечує надсилання електронної пошти. `Axios` виконує клієнтські запити до REST API, а `cors` керує політиками доступу. Змінні конфігурації завантажуються з файлу `.env` через `dotenv` для централізації підключень до бази даних та інших налаштувань. У середовищі розробки `nodemon` автоматизує перезапуск сервера, коли відбуваються зміни коду, мінімізуючи час між впровадженням та тестуванням. Такий вибір технологій та бібліотек створює збалансоване поєднання продуктивності, безпеки та зручності розробки.

### **3.1.2. Структура проекту**

Кореневий каталог проекту містить два основні каталоги, які чітко позначають розділення між клієнтським та серверним кодом. Каталог фронтенду містить усі файли фронтенду: підпапка `src/components` містить компоненти інтерфейсу користувача, які можна повторно використовувати, такі як `Button.jsx`, `CategoriesList.jsx`, `Header.jsx`, `Input.jsx`, `ReminderChecker.jsx`, `ReminderForm.jsx` та `RemindersList.jsx`. Ці елементи використовуються в компонентах сторінки, що зберігаються в папці `src/pages`, включаючи `Login.jsx`, `ForgotPassword.jsx`, `Notes.jsx` та `RemindersPage.jsx`. Файл `src/api.js` підсумовує всі виклики AJAX через `Axios` та знаходиться безпосередньо в каталозі `src`. Глобальні стилі знаходяться в `App.css` та `index.css`, тоді як `main.jsx` та `App.jsx` утворюють точку входу для застосунку React. Файли конфігурації, такі як `package.json` та `vite.config.js`, розташовані в кореневому каталозі проекту та керують залежностями та налаштуваннями пакетування.

Каталог backend організовує логіку на стороні сервера. З'єднання з PostgreSQL встановлюється в `src/config/db.js`, а змінні середовища завантажуються з `.env`. Структура бази даних відображається в моделях ORM в `src/models`, а саме `userModel.js`, `categoryModel.js`, `noteModel.js` та `reminderModel.js`. Обробка запитів відбувається в контролерах у `src/controllers` – до них належать `authController.js`, `categoryController.js`, `notesController.js` та `reminderController.js`. Маршрути (`auth.js`, `categories.js`, `notes.js`, `reminders.js`, `dbtest.js`, `ping.js`), які пересилають кожен HTTP-запит до відповідного контролера, визначені в `src/routes`. Допоміжні функції, такі як планувальник нагадувань (`reminderScheduler.js`) та поштовий сервер (`mail.js`), зберігаються в спільній папці служби. Нарешті, скрипт ініціалізації SQL `db_init.sql`, який автоматично створює схеми бази даних, розташований у кореневому каталозі проекту. На рисунку 3.1 я зобразив структуру проекту. На рисунку 3.1 видно всю структуру мого проекту, папки фронтенду зліва та бекенду справа.

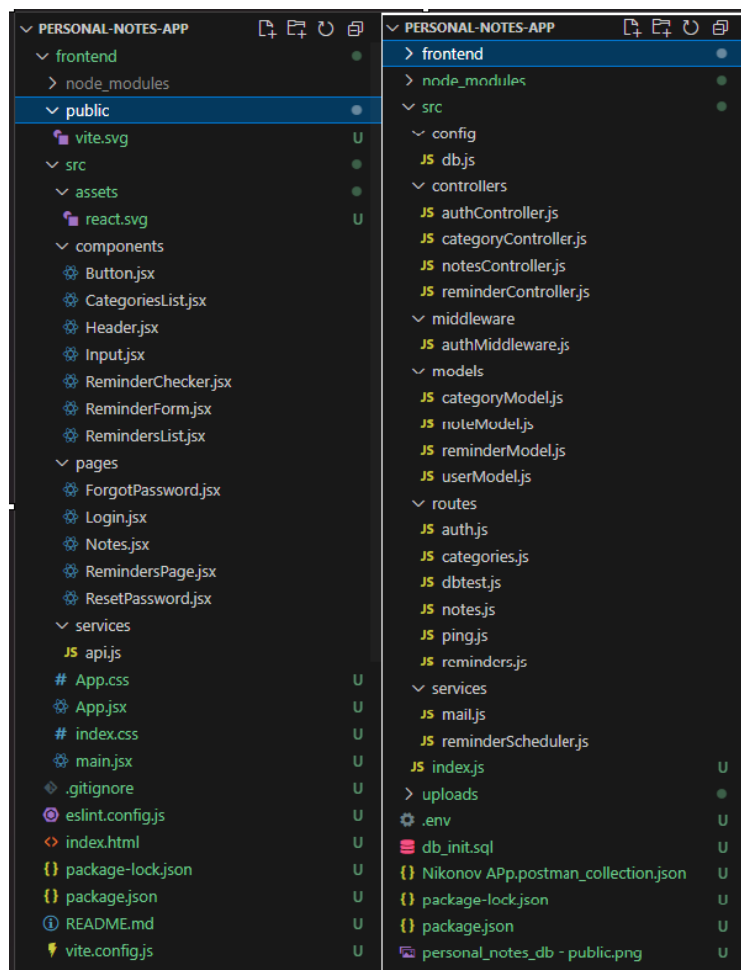


Рисунок 3.1 – Структура проекту ( дерево каталогів )

### 3.1.3. Конфігурація та запуск

Кореневий каталог проєкту містить кілька файлів конфігурації, які контролюють розробку та роботу. Файл `package.json` містить залежності клієнта, такі як `React`, `axios` та `react-router-dom`, а також серверні пакети (`Express`, `pg` тощо). `Nodemon` знаходиться в розділі `devDependencies`, який автоматично перезапускає сервер у режимі розробки, коли файли змінюються.

Файл `vite.config.js` контролює процес збірки фронтенду: він визначає точку входу, керує обробкою модулів, створює оптимізовані пакети для виробничого середовища та налаштовує проксі-сервер, через який запити API пересилаються від клієнта до сервера – таким чином усуваючи проблеми CORS під час локальної розробки.

Конфіденційні налаштування, такі як дані підключення до бази даних, підписання токенів JWT, порт сервера та режим роботи (`NODE_ENV`), зберігаються у файлі `.env`. Перемикання між середовищем розробки та виробничим середовищем здійснюється простим обміном цими змінними, без зміни коду програми.

Нижче наведено огляд центральних скриптів `npm` та важливих змінних середовища:

Таблиця 3.1 – основні `npm` скрипти

<code>package.json</code>	Призначення
<code>Npm instal</code>	Встановлення всіх залежностей з <code>package.json</code>
<code>npm run dev</code>	Запуск сервера в режимі розробки
<code>npm run build</code>	Створення збірки фронтенда
<code>npm start</code>	Запуск серверної частини в режимі використання

Таблиця 3.2 – основні змінні середовища

Змінна середовища (.env)	Призначення
DB_HOST, DB_PORT	Параметри підключення PostgreSQL
DB_USER, DB_PASS	Дані для доступу до бази
DB_NAME	Назва бази даних
JWT_SECRET	Секретний ключ для підпису та перевірки JWT-токенів
PORT	Порт HTTP-запитів
NODE_ENV	Режим виконання

### 3.2. Реалізація бази даних

#### 3.2.1. Логічна модель даних

PostgreSQL було обрано як реляційну базу даних для проекту та змодельовано за допомогою DBeaver як графічного інструменту адміністрування, оскільки це поєднання забезпечує зручну візуалізацію схеми та ефективне керування базою даних, забезпечуючи при цьому максимальну надійність. PostgreSQL вражає своєю архітектурою, сумісною з ACID, та розширеними механізмами індексування, які забезпечують видатну продуктивність, особливо для пошуку великих обсягів тексту та складних запитів у нотатках.

Таблиця `users` утворює центр і зберігає такі поля, як електронна пошта, хеш пароля, позначка часу вкладення та, за бажанням, URL-адресу аватара. Нотатки в таблиці `notes` пов'язані з їхніми відповідними авторами через посилання «один до багатьох» через `user_id`. Кожна нотатка містить заголовок, вміст, час створення та модифікації, а також, за бажанням, зовнішній ключ `category_id` для основної категорії.

Таблиця `categories` забезпечує гнучку категоризацію. Окрім `id` та `name`, вона також містить поля `color` (для кольорового кодування) та `parent_id` (для створення вкладених категорій). Проміжний інструмент `note_categories` також

пов'язує нотатки та категорії у зв'язку «багато до багатьох», що дозволяє призначати одну нотатку кільком категоріям і навпаки.

Нагадування зберігаються в таблиці нагадувань, яка пов'язана з нотатками через зовнішній ключ `note\_id`. Ця таблиця зберігає дату, час, текст і прапорець `is\_sent`, щоб увімкнути кілька сповіщень для кожної нотатки та відстежувати їх статус надсилання.

Таблиця `password\_reset\_tokens` підтримує відновлення пароля, зберігаючи тимчасові токени з терміном їх дії та пов'язаним з ними `user\_id`. Полі та права доступу організовані в таблицях `roles` та `user\_roles`, причому `user\_roles` діє як проміжна таблиця для представлення зв'язку "багато до багатьох" між користувачами та ролями.

Усі зв'язки зовнішнього ключа позначені стрілками на діаграмі та визначені таким чином, що коли основний запис видаляється або оновлюється, записи, на які посилаються, автоматично коригуються, щоб запобігти появі одиночних записів. На рисунку 3.2 показано цю структуру бази даних.

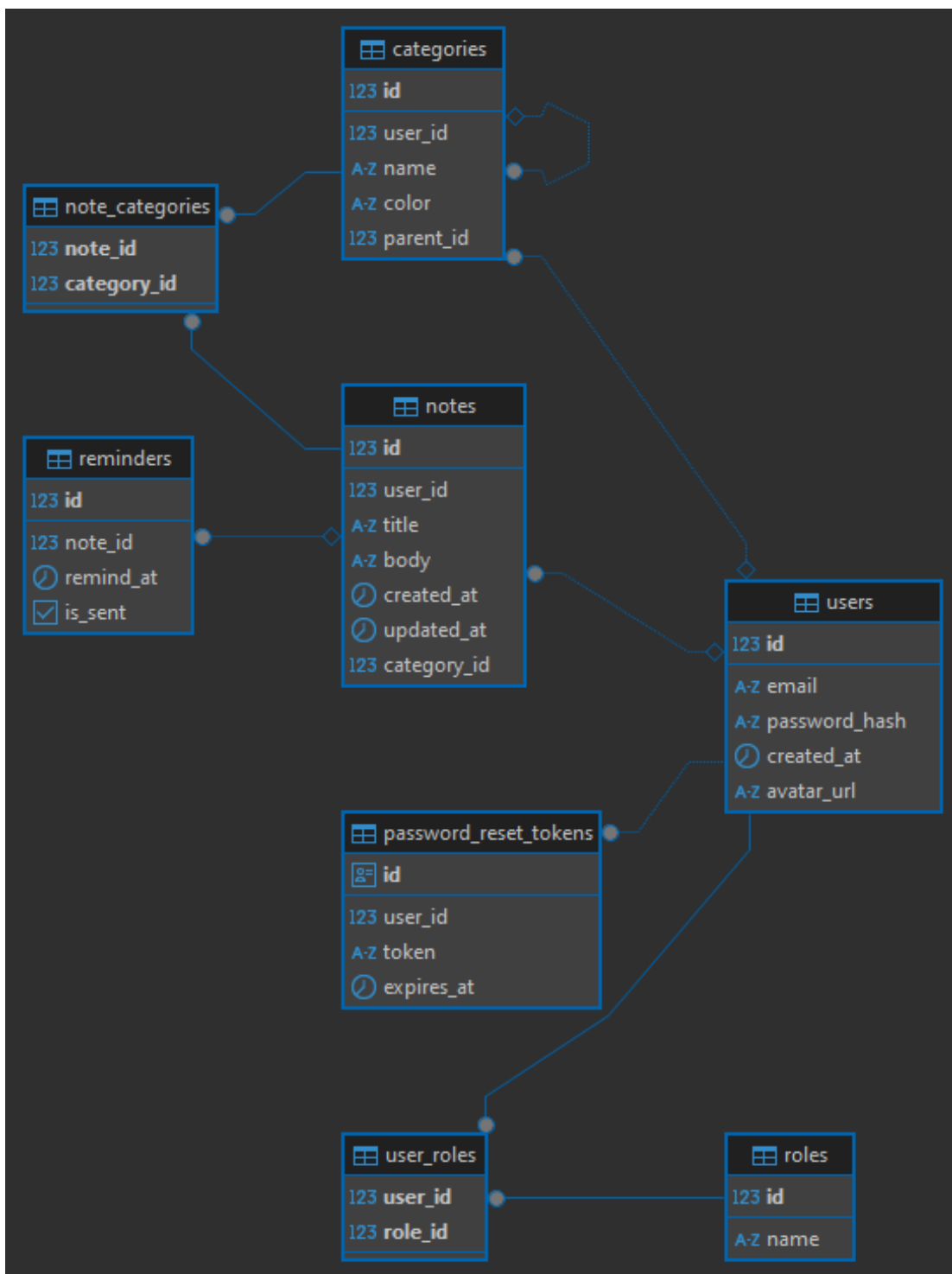


Рисунок 3.2 – структура бази даних

### 3.2.2. Структура таблиць

У базі даних реалізовано вісім основних таблиць, які забезпечують зберігання користувачів, нотаток, категорій, нагадувань, ролей і допоміжних сутностей. Кожну з них наведено нижче в табличному вигляді з позначенням первинних та зовнішніх ключів.



Таблиця «users» призначена для зберігання облікових записів користувачів системи. Кожен запис містить інформацію про електронну пошту (використовується як логін), захешований пароль, опційне посилання на аватар та часові мітки створення й оновлення профілю. Ця таблиця є відправною точкою для всіх операцій із нотатками, нагадуваннями та ролями та не змінюється автоматично під час роботи програми.

Таблиця 3.3 – структура таблиці <user>

Назва колонки	Тип даних	PK/FF	Опис
id	serial	PK	айди
email	text	UNIQUE	Логін
password_hash	text		Хеш пароля
avatar_url	text		Url аватара
created_at	timestamp		Дата створення запису

Таблиця «roles» зберігає перелік можливих ролей у системі. Кожна роль має унікальне ім'я (наприклад, “user” чи “admin”) і створюється адміністратором під час налаштування системи. Записи в цій таблиці рідко змінюються після початкової ініціалізації.

Таблиця 3.4 – структура таблиці <roles >

Назва колонки	Тип даних	PK/FF	Опис
id	serial	PK	id
name	text	UNIQUE	Назва ролі

Таблиця «user\_roles» служить зв'язком «багато-до-багатьох» між користувачами та їхніми ролями. Для кожного поєднання user\_id + role\_id указується статус призначення (активний/неактивний) та час надання права. Завдяки такій схемі один користувач може мати декілька ролей, а роль може бути присвоєна багатьом користувачам.

Таблиця 3.5 – структура таблиці <user\_roles>

Назва колонки	Тип даних	PK/FF	Опис
user_id	integer	FK-users(id)	id користувача
role_id	integer	FR-roles(id)	id ролі
PRIMARY KEY		user_id, role_id	Композитний первинний ключ

Таблиця «notes» відповідає за зберігання нотаток: кожна нотатка містить заголовок, текстовий вміст, часові мітки створення й оновлення та зовнішній ключ на автора (user\_id). За потреби в полі category\_id можна вказати основну категорію, хоча додаткові категорії реалізовано через проміжну таблицю note\_categories.

Таблиця 3.6 – структура таблиці <notes>

Назва колонки	Тип даних	PK/FF	Опис
id	serial	PK	id нотатки
user_id	integer	FK-users(id)	Автор нотатки
title	text		Заголовок нотатки
body	text		Текстова частина

created_at	timestamp		Дата й час створення
updated_at	timestamp		Дата й час last оновлення

Таблиця «categories» призначена для організації класифікації нотаток за темами. Кожна категорія має унікальне ім'я, колір (для підсвічування) та опційний зв'язок parent\_id із батьківською категорією, що дозволяє будувати ієрархічні структури. Додатково поле user\_id гарантує, що кожен користувач бачить тільки власні категорії.

Таблиця 3.7 – структура таблиці <categories>

Назва колонки	Тип даних	PK/FF	Опис
id	serial	PK	id категорії
user_id	integer	FK-users(id)	Власник категорії
name	text		Назва
color	char(7)		Колір, формат HEX
parent_id	integer	FK-categories(id)	Категорія для ієрархії

Таблиця «note\_categories» реалізує зв'язок «багато-до-багатьох» між нотатками й категоріями. У ній зберігаються пари note\_id + category\_id, що дозволяє одній нотатці належати до кількох категорій, а категорії використовуватися в багатьох нотатках.

Таблиця 3.8 – структура таблиці <note\_categories>

Назва колонки	Тип даних	PK/FF	Опис
---------------	-----------	-------	------

note_id	integer	FK-notes(id)	id нотатки
category_id	integer	FK-categories(id)	id категорій
PRIMARY KEY		note_id,category_id	Композитний первинний ключ

Таблиця «reminders» містить заплановані сповіщення для нотаток. Для кожного нагадування фіксуються часові позначки remind\_at, текст повідомлення та прапорець is\_sent, який інформує про статус відправлення. Зовнішній ключ note\_id пов'язує нагадування з конкретною нотаткою.

Таблиця 3.9 – структура таблиці <reminders>

Назва колонки	Тип даних	PK/FF	Опис
id	serial	PK	id нагадування
note_id	integer	FK-notes(id)	Нотатка з нагадув.
remind_at	timestamp		Дата й час сповіщ.
is_sent	boolean		Статус

Таблиця «password\_reset\_tokens» призначена для роботи з відновленням доступу. У ній фіксуються унікальні токени, пов'язані з конкретним користувачем через user\_id, а також термін їхньої дії expires\_at. Після закінчення терміну токени автоматично втрачають силу.

Таблиця 3.10 – структура таблиці <password\_reset\_tokens>

Назва колонки	Тип даних	PK/FF	Опис
id	uuid	PK	id токена
user_id	integer	Fk-users(id)	Користувач

token	varchar(255)	UNIQUE	Токен
expires_at	timestamp		Час дії

Усі поля налаштовані з каскадними діями ON DELETE CASCADE або ON DELETE SET NULL для FK, що дозволяє підтримувати цілісність даних під час видалення пов'язаних записів.

### 3.2.3. Первинні/зовнішні ключі та індекси

У структурі бази даних кожна таблиця має чітко визначений первинний ключ, який забезпечує унікальну ідентифікацію записів, а також серію зовнішніх ключів, які відображають зв'язки між сутностями та підтримують цілісність даних.

У таблицях users, roles, notes, categories, reminders та password\_reset\_tokens поле id служить первинним ключем та автоматично індексується для оптимізації швидкості запитів. У допоміжних таблицях user\_roles та note\_categories первинний ключ складається з двох стовпців (user\_id + role\_id або note\_id + category\_id), так що кожен зв'язок «багато-до-багатьох» однозначно ідентифікований.

Зовнішні ключі пов'язують сутності між собою: notes.user\_id посиляється на users.id, categories.user\_id посиляється на відповідного користувача в users, а categories.parent\_id посиляється на батьківську категорію для представлення ієрархічних структур. Таблиця note\_categories пов'язує нотатки та категорії через note\_id та category\_id, тоді як reminders.note\_id встановлює зв'язок з нотаткою в notes. У password\_reset\_tokens, user\_id посиляється на users.id і гарантує, що для неіснуючого користувача не буде створено токен. Під час видалення залежні записи автоматично видаляються або – у випадку вкладених категорій – встановлюються на NULL, щоб уникнути осиротілих записів.

До стандартних індексів за первинними та унікальними ключами (наприклад, users.email, roles.name або password\_reset\_tokens.token), було

створено додаткові індекси. Наприклад, є індекс за `notes.created\_at` для швидкого отримання найновіших нотаток, індекс за `reminders.remind\_at` для ефективних запитів щодо майбутніх нагадувань та індекс за `categories.user\_id` для пришвидшення фільтрації категорій для кожного користувача. Таке поєднання механізмів ключів та індексів забезпечує високу продуктивність та надійність, навіть з великими обсягами даних.

### 3.3. Реалізація серверної частини

Основна логіка розташована в модулі, який встановлює з'єднання з PostgreSQL та завантажує сценарій конфігурації для створення схеми бази даних. Зв'язок з базою даних потім встановлюється за допомогою пулу з'єднань перед реєстрацією маршрутів, контролерів та проміжного програмного забезпечення.

Контролери обробляють вхідні HTTP-запити та викликають відповідні моделі або служби. Модуль автентифікації обробляє реєстрацію, вхід та скидання пароля, генеруючи токени JWT, тоді як контролер Notes охоплює весь цикл CRUD: створення нової нотатки, отримання нотаток зареєстрованого користувача, їх оновлення та видалення. Категорії керуються окремим контролером, який дозволяє вкладені категорії та гарантує, що доступні лише власні категорії користувача. Служба нагадувань працює в окремому планувальнику, але нагадування можна створювати, отримувати, редагувати та видаляти в інтерфейсі REST через маршрути.

Для захисту кінцевих точок усі маршрути оснащені проміжним програмним рівнем, який перевіряє існування та дійсність токена JWT та забороняє доступ, якщо токен відсутній або недійсний. Цей механізм гарантує, що лише автентифіковані користувачі можуть взаємодіяти з приватними даними, такими як нотатки, категорії та нагадування.

Кожен маршрут має префікс у вигляді рядка, такого як `/auth`, `/notes`, `/categories` або `/reminders`, щоб чітко розрізняти різні функціональні області. Після обробки запитів контролери повертають JSON-відповіді, що містять щонайменше поля `status` та `data`, а у разі помилок – додаткове поле `message`.

В таблиці 3.11 наведено зведений перелік основних HTTP- ендпоінтів і їх опис

Таблиця 3.11 – Огляд REST-ендпоінтів

Метод	Шлях	Опис
POST	/auth/register	Реєстрація нового користувача
POST	/auth/login	Вхід у систему та отримання JWT-токена
POST	/auth/reset-request	Запит на скидання пароля
POST	/auth/reset	Скидання пароля з використанням токена
GET	/notes	Отримання всіх нотаток користувача
POST	/notes	Створення нової нотатки
PUT	/notes/:id	Оновлення існуючої нотатки
DELETE	/notes/:id	Видалення нотатки
GET	/categories	Отримання всіх категорій користувача
POST	/categories	Створення нової категорії
PUT	/categories/:id	Оновлення існуючої категорії
DELETE	/categories/:id	Видалення категорії

POST	/notes/:id/reminders	Додавання нового нагадування до нотатки
GET	/reminders	Отримання всіх нагадувань
PUT	/reminders/:id	Оновлення існуючого нагадування
DELETE	/reminders/:id	Видалення нагадування

Такий набір маршрутів забезпечує повноцінну взаємодію з даними через простий і зрозумілий REST-інтерфейс та дозволяє легко розширювати функціональність за необхідності.

### 3.4.Реалізація клієнтської частини

#### 3.4.1. Екран входу

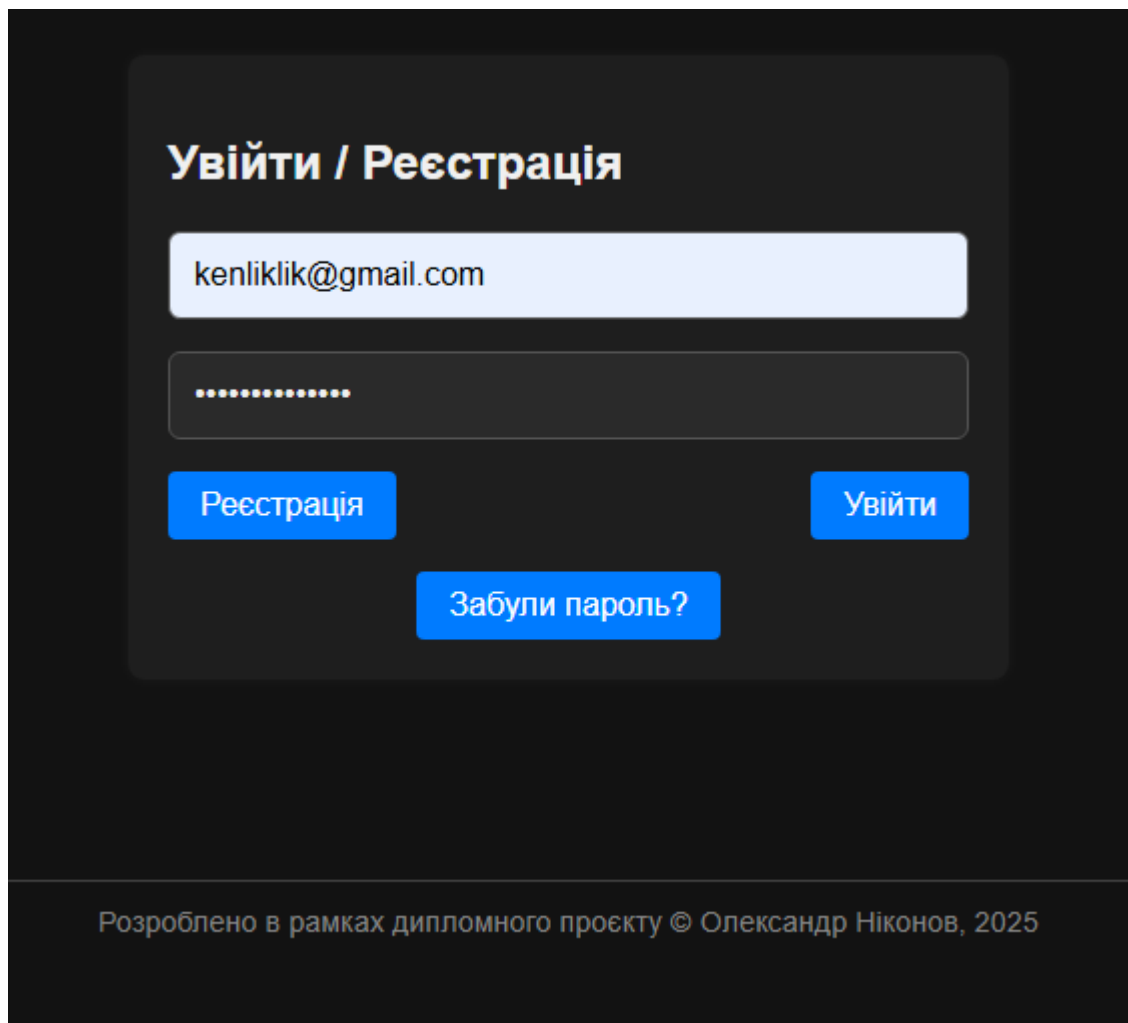
Перед тим як почати працювати з нотатками, користувача зустрічає єдина форма для входу або реєстрації, в якій два поля — для адреси електронної пошти та пароля — розташовані одне під одним у центрі екрану. Над ними великими літерами виведено заголовок «Увійти / Реєстрація», а під полями розміщені дві кнопки: «Реєстрація» ліворуч і «Увійти» праворуч. Якщо при введенні даних сталася помилка — наприклад, незареєстрована електронна адреса або невірний пароль — над заголовком форми з'являється рядок із повідомленням «Помилка», який миттєво інформує про невдачу спробу.

У самому низу блоку під кнопками знаходиться посилання «Забули пароль?», натискаючи її перенаправляє користувача на сторінку з відновленням паролю. Користувач вводить свою адресу, і система надсилає лист із посиланням для створення нового пароля. Таким чином, навіть у разі втрати облікових даних відновлення проходить без звернення до адміністратора.

Загальна стилістика форми витримана в темних тонах, поля вводу підсвічуються контрастною рамкою при фокусі, а кнопки мають однаковий

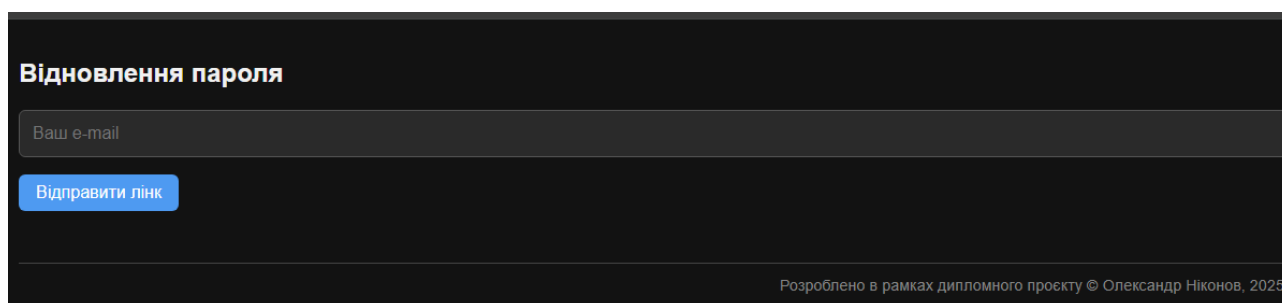


яскравий синій колір. Цей дизайн забезпечує чітке візуальне розмежування всіх елементів і зручність для користувача на етапі автентифікації.



The screenshot displays a dark-themed login and registration form. At the top, the title "Увійти / Реєстрація" is shown in a light blue font. Below the title, there are two input fields: the first contains the email address "kenliklik@gmail.com", and the second is a password field represented by a series of dots. Underneath the input fields, there are three buttons: "Реєстрація" (Registration) and "Увійти" (Login) are positioned side-by-side, while "Забули пароль?" (Forgot password?) is centered below them. All buttons are blue with white text. At the bottom of the form, a footer line reads "Розроблено в рамках дипломного проєкту © Олександр Ніконов, 2025".

Рисунок 3.3 - Інтерфейс сторінки входу та реєстрації користувача



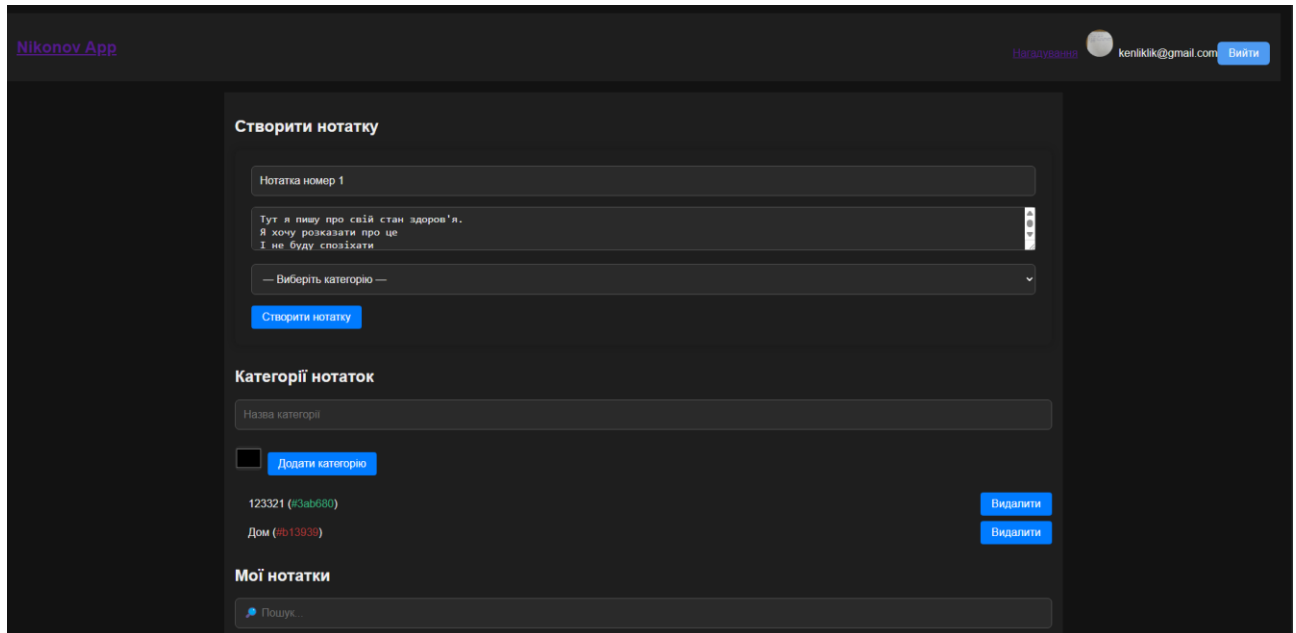
The screenshot shows a dark-themed password reset form. The title "Відновлення пароля" (Password Reset) is at the top in a light blue font. Below the title is a single input field labeled "Ваш e-mail" (Your email). Underneath the input field is a blue button with the text "Відправити лінк" (Send link). At the bottom right of the form, a footer line reads "Розроблено в рамках дипломного проєкту © Олександр Ніконов, 2025".

Рисунок 3.4 – Інтерфейс сторінки для відновлення паролю

### 3.4.2. Загальний інтерфейс

Після успішного входу користувач потрапляє на основний екран із власними нотатками. Інтерфейс який зустрічає користувач абсолютно повністю

зрозумілий і доступний. Немає ніяких лишніх функцій і навантаженості. Сама суть додатку це те щоб користувач писав нотатки і міг комфортно з ними працювати, а не перевантажуватися після складного робочого дня з розбиранням



інтерфейсу. На рисунку 3.5 зображений інтерфейс після успішної авторизації

Рисунок 3.5 – інтерфейс веб-додатку ( верхня частина )

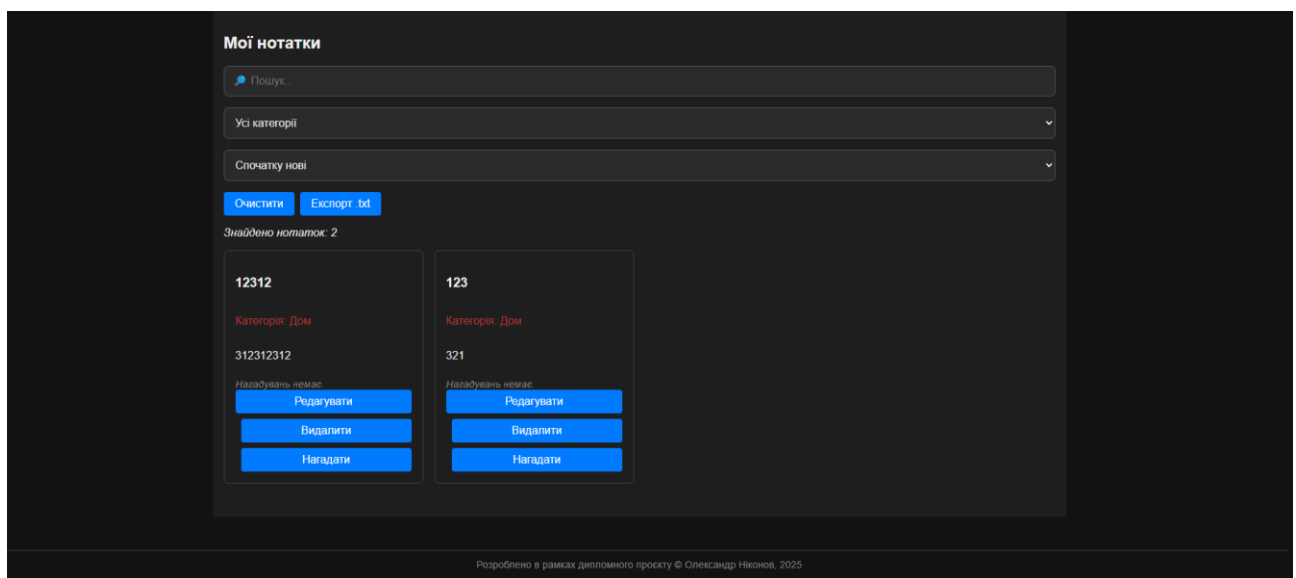


Рисунок 3.6 – інтерфейс веб-додатку ( нижня частина )

### 3.4.3. Створення нотатки

Перше що бачить перед собою користувач, це поле з створенням нотатки, в ньому він може ввести все що завгодно. Є два обов'язкових поля «Заголовок»

і «Тло нотатки» їх користувач має заповнити для того щоб створити свою нотатку, після цього нажав на кнопку «Створити нотаку» створиться абсолютно аналогічна нотатка з всім що вписав користувач. До цього користувач може додавати категорії, але перед як додати каегорію користувач має її створити.

Рисунок 3.7 – Поле створення нотатки

В разі якщо у користувача створені категорії, він може вибрати категорію для нотатки в впливаючому списку «--Виберіть категорію--».

На рисунку 3.8 зображений впливаючий список вибору категорій

Рисунок 3.8 – Список вибору категорій

Якщо користувач не хоче додавати категорію, він просто нічого не вибирає в цьому списку і у нього буде нотатка без категорії.

Як тільки користувач нажимає на кнопку «Створити нотатку» вона створюється в самому низу екрану. Це зроблено щоб користувач міг спокійно їх передивлятись не звертаючи увагу на інші частини веб- додатку.

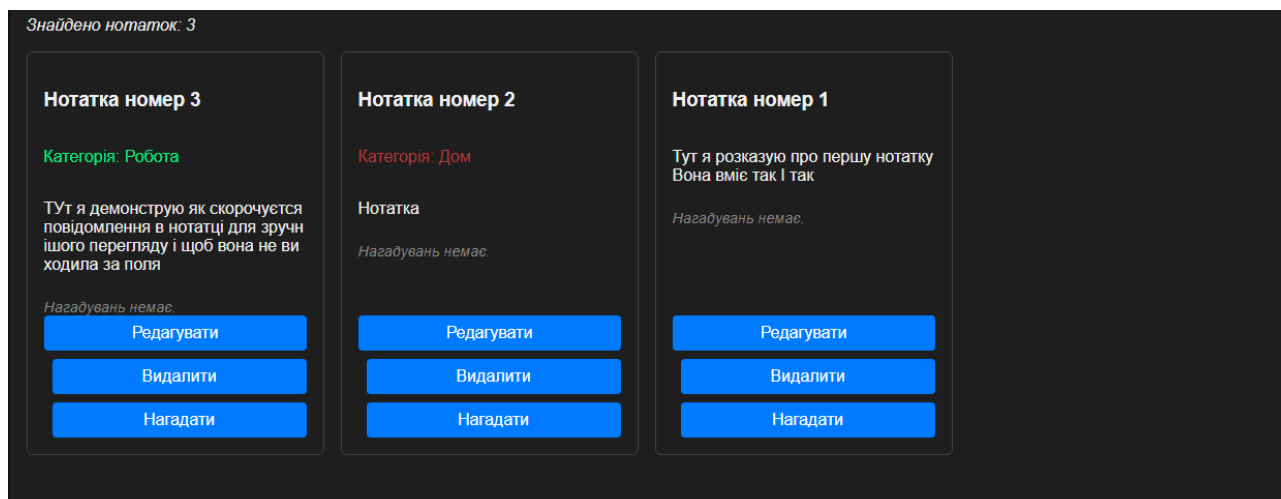


Рисунок 3.9 – Огляд нотаток

### 3.4.4. Категоризація нотаток

Створити категорію для нотатку ще легше і функціональніше. У нас є можливість створити категорію з окремим кольором і назвою, яка потім буде підсвічуватись в самій нотатці. Для цього розроблений блок «Категорії нотаток». Тут користувач створює, дає колір або видаляє категорії.

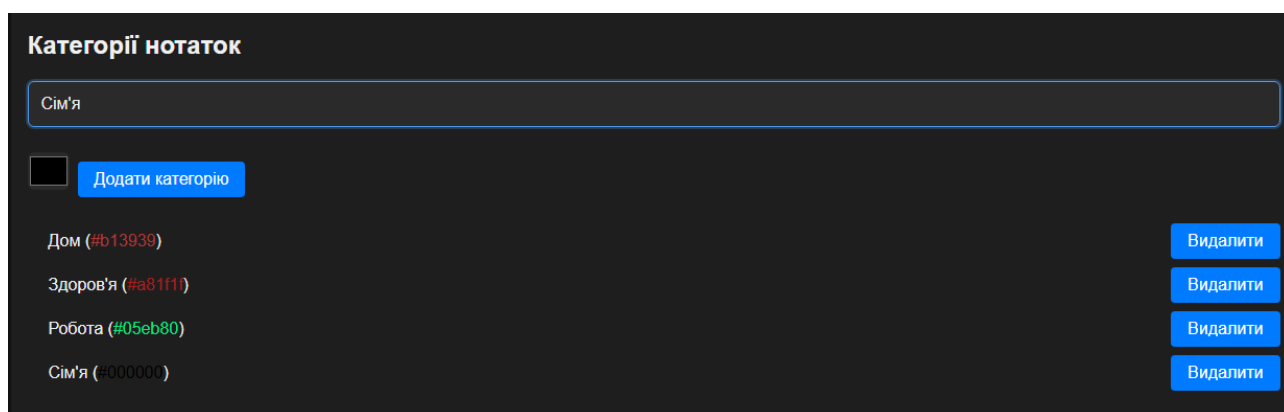


Рисунок 3.10 – Категорії нотаток

Для того щоб користувачу було легше вибирати колір, налаштована кольорова rgb сітка, для вибору кольору. Це зображено на рисунку 3.11

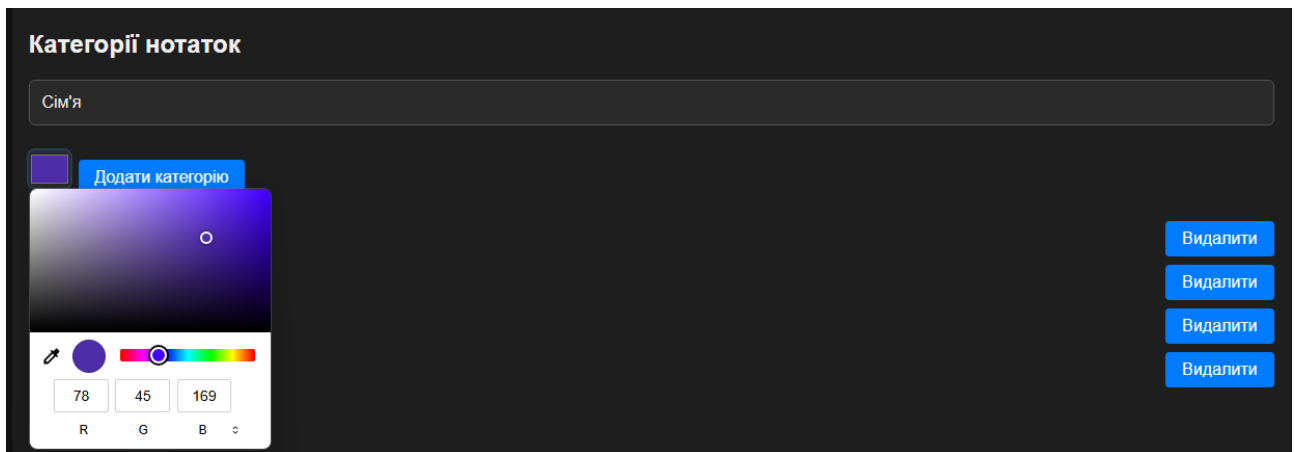


Рисунок 3.11 – Установка кольору категорії

Якщо користувач хоче «Видалити» категорію, йому висвічується вікно з запитанням, видалити ок чи відмінити

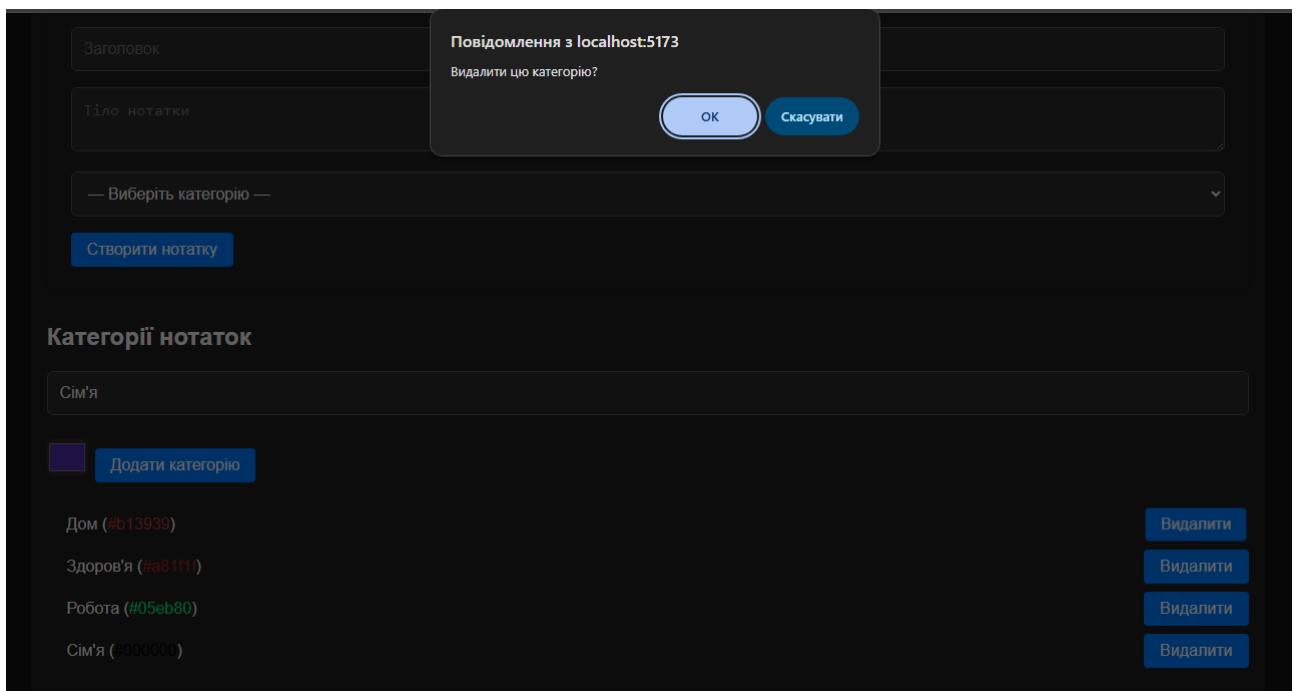


Рисунок 3.12 – Видалення категорії

### 3.4.5. Пошук і фільтрація

Щоб швидко знайти потрібні нотатки серед великої кількості, над нотатками передбачено інструменти пошуку та фільтрації. Спочатку розташоване поле з плейсхолдером «Пошук...», у яке можна вводити ключові слова з заголовку або тексту нотатки. Під час введення результати списку оновлюються в реальному часі, відображаючи лише ті нотатки, які містять введений фрагмент.

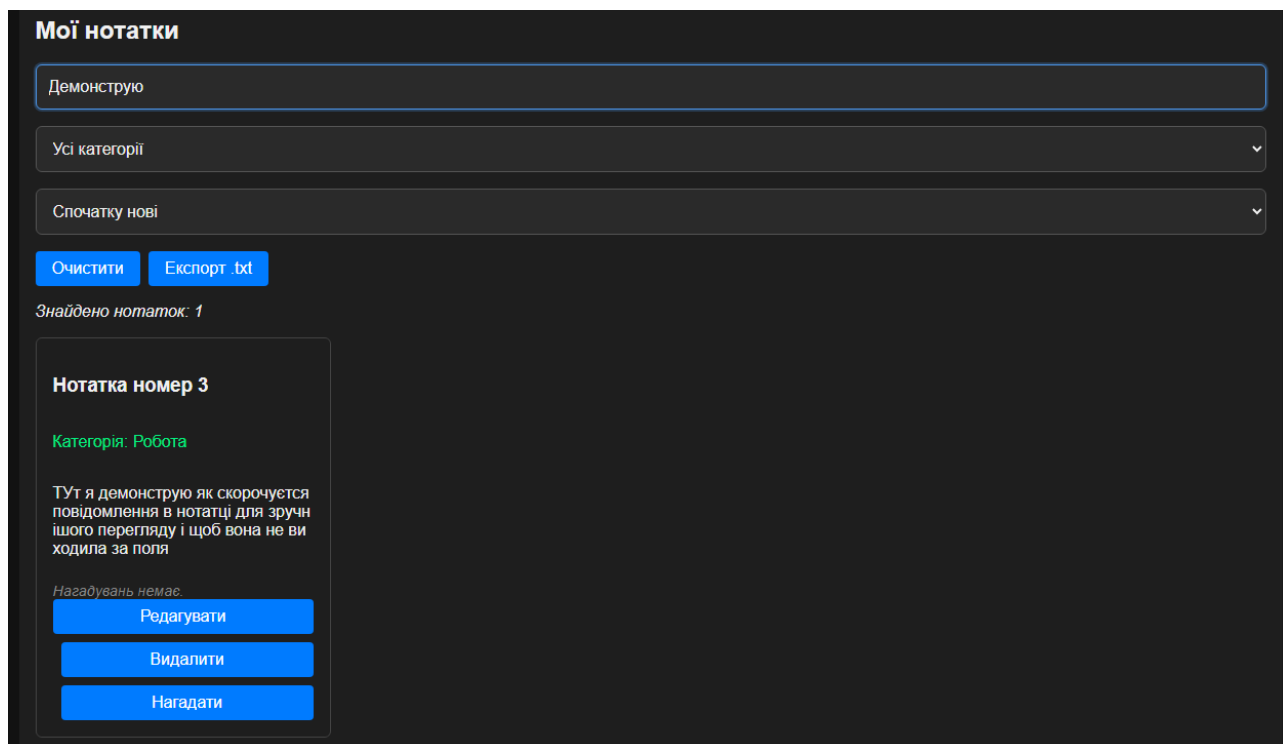


Рисунок 3.13 – Поле пошуку з динамічним оновленням результатів

Безпосередньо під полем пошуку знаходиться розкритий список категорій. За замовчуванням вибрано «Усі категорії», але розкриття меню дозволяє обрати конкретну категорію (наприклад «Дом» або «Робота») і відразу показати лише нотатки з цієї групи.

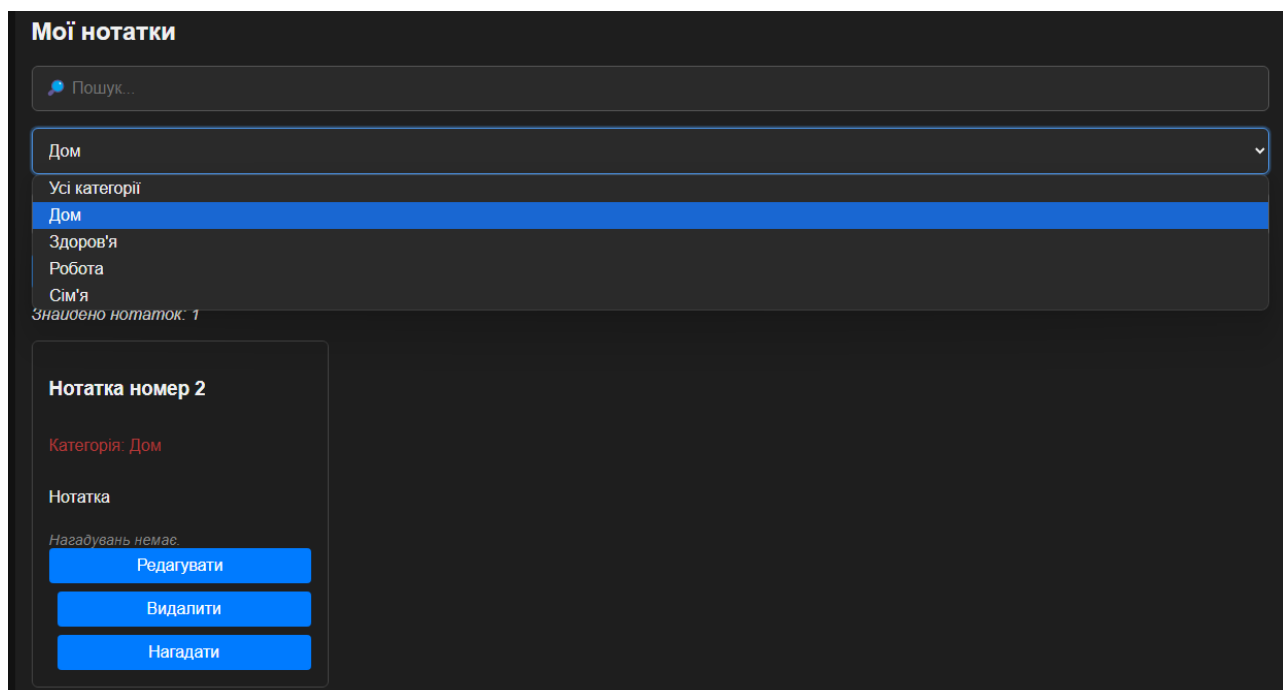


Рисунок 3.14 – Фільтрація за категоріями

Нижче розташований селектор сортування, який дає змогу вибирати порядок відображення нотаток за датою створення: «Спочатку нові» або «Спочатку старі». При зміні опції картки нотаток автоматично перепакуюються відповідно до вибраного порядку.

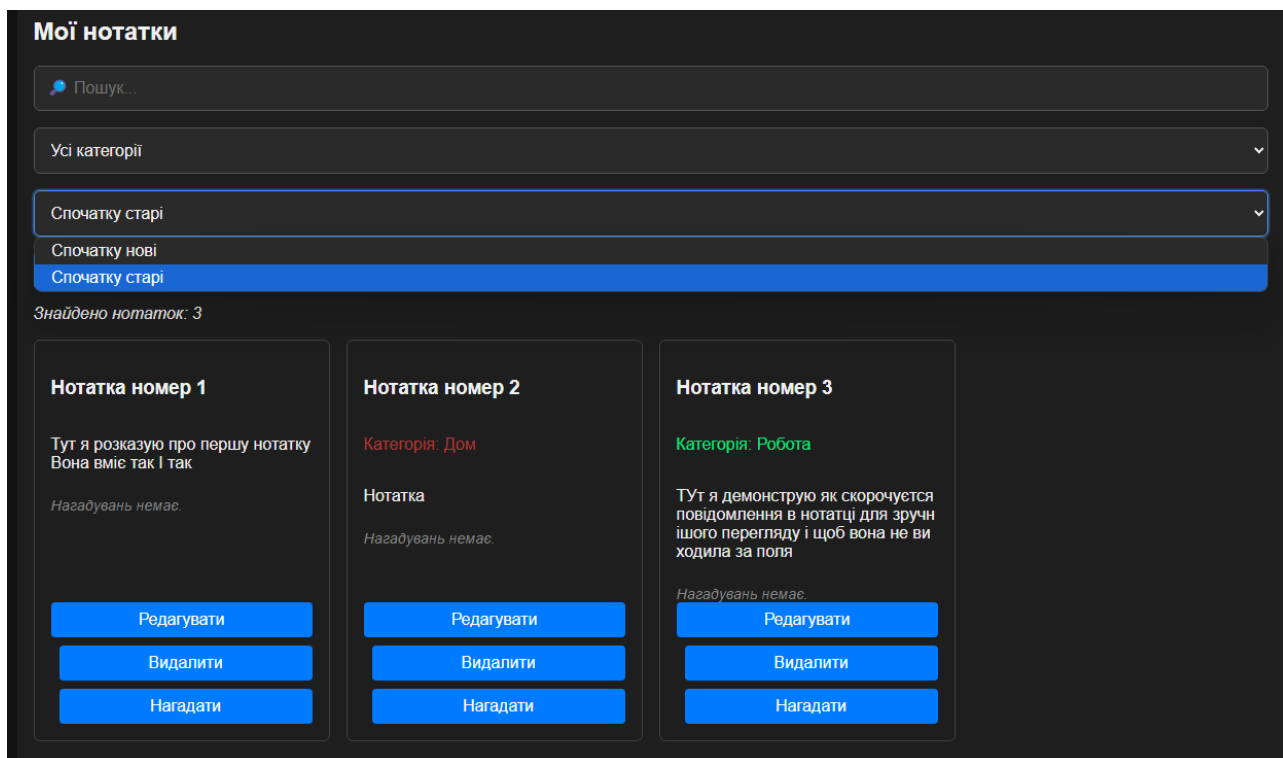


Рисунок 3.15 – Сортування нотаток за датою створення

Після вибору категорії та режиму сортування з'являється оновлена строка з кількістю знайдених нотаток — наприклад «Знайдено нотаток: 1». Якщо необхідно повернутися до початкового стану, досить натиснути кнопку «Очистити», яка скидає всі фільтри та пошуковий запит, відновлюючи загальний перелік нотаток.

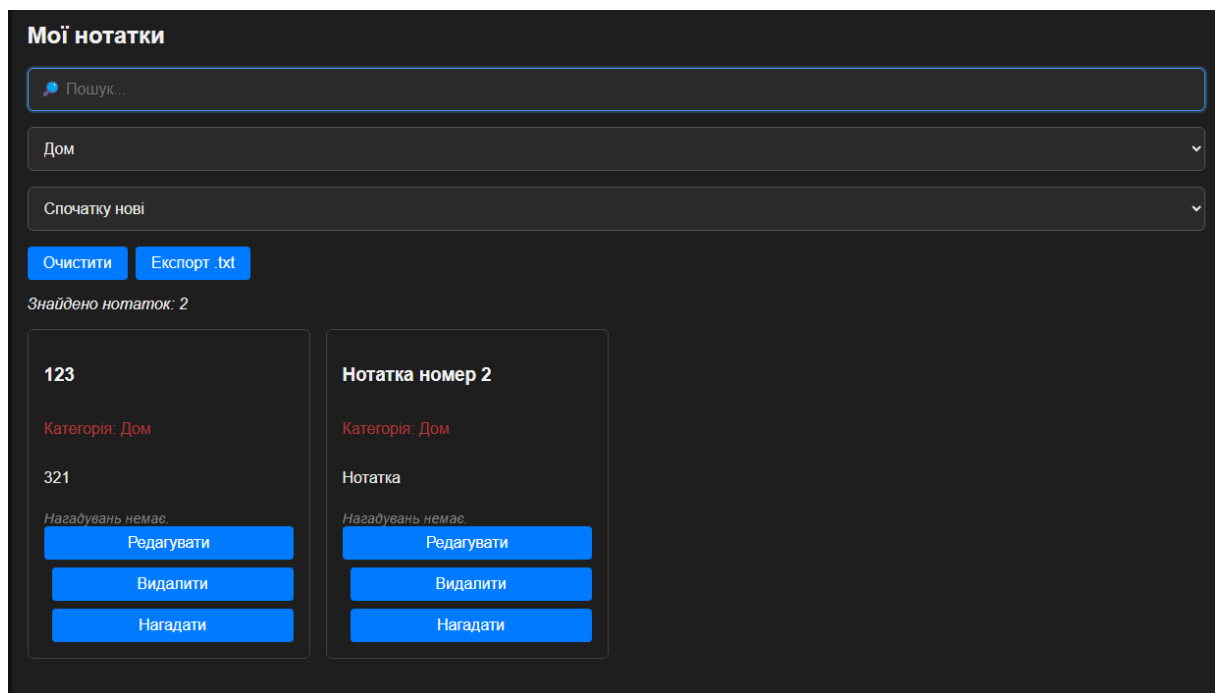


Рисунок 3.16 – Результат фільтрації за категорією «Дом» і часом

Ця комбінація пошуку, фільтрації за категоріями та сортування забезпечує гнучкий і швидкий механізм навігації по нотатках, дозволяючи користувачеві миттєво знаходити та сортувати потрібні записи без перевантаження інтерфейсу.

### 3.4.6. Нагадування

На панелі заголовка поруч із аватаркою й кнопкою виходу розташовано посилання «Нагадування», яке веде на окрему сторінку зі списком усіх запланованих сповіщень. Під заголовком сторінки відображаються дві кнопки: «Активні» та «Минулі», що фільтрують записи за поточним часом, а також кнопка «Видалити все» для миттєвого очищення списку.

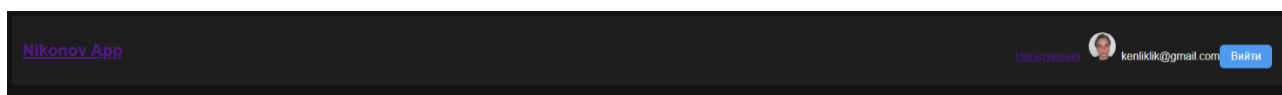


Рисунок 3.17 – Кнопка «Нагадування» перекидає нас на другу сторінку

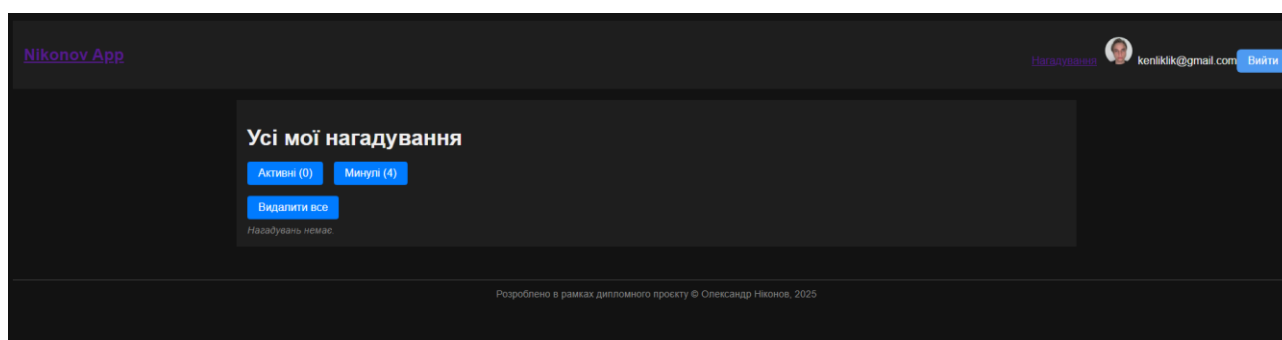


Рисунок 3.18 – сторінка «Усі мої нагадування»



Нижче показано активні сповіщення у вигляді карток із датою й часом, чотири з яких уже пройшли та переміщені в розділ «Минулі». Кожна картка містить текст дати й часу нагадування та кнопку «Видалити», що видаляє одне сповіщення.

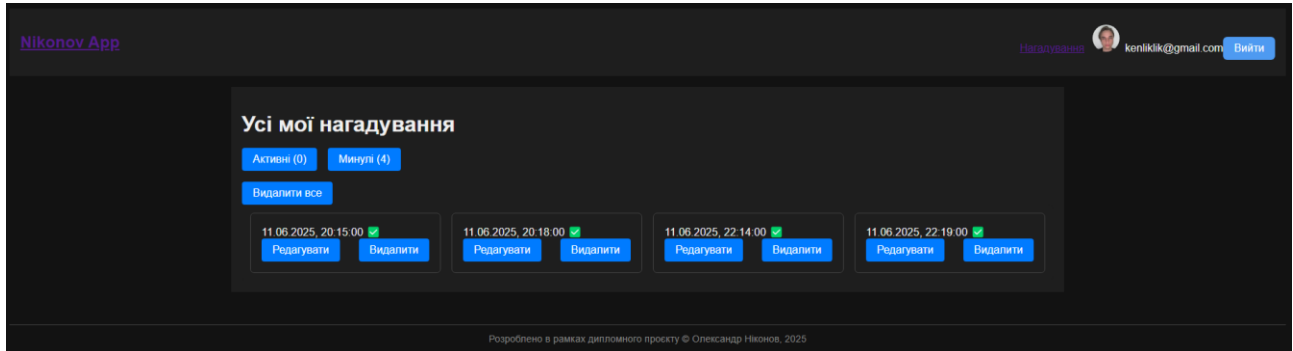


Рисунок 3.19 – Сторінка «Усі мої нагадування» з минулими сповіщеннями

Повернемося на головну сторінку, щоб додати нове нагадування, під кожною нотаткою натискають «Нагадати» – відкривається вбудований у форму селектор дати й часу. Календар дозволяє обрати день, а праворуч – точний час (години та хвилини).

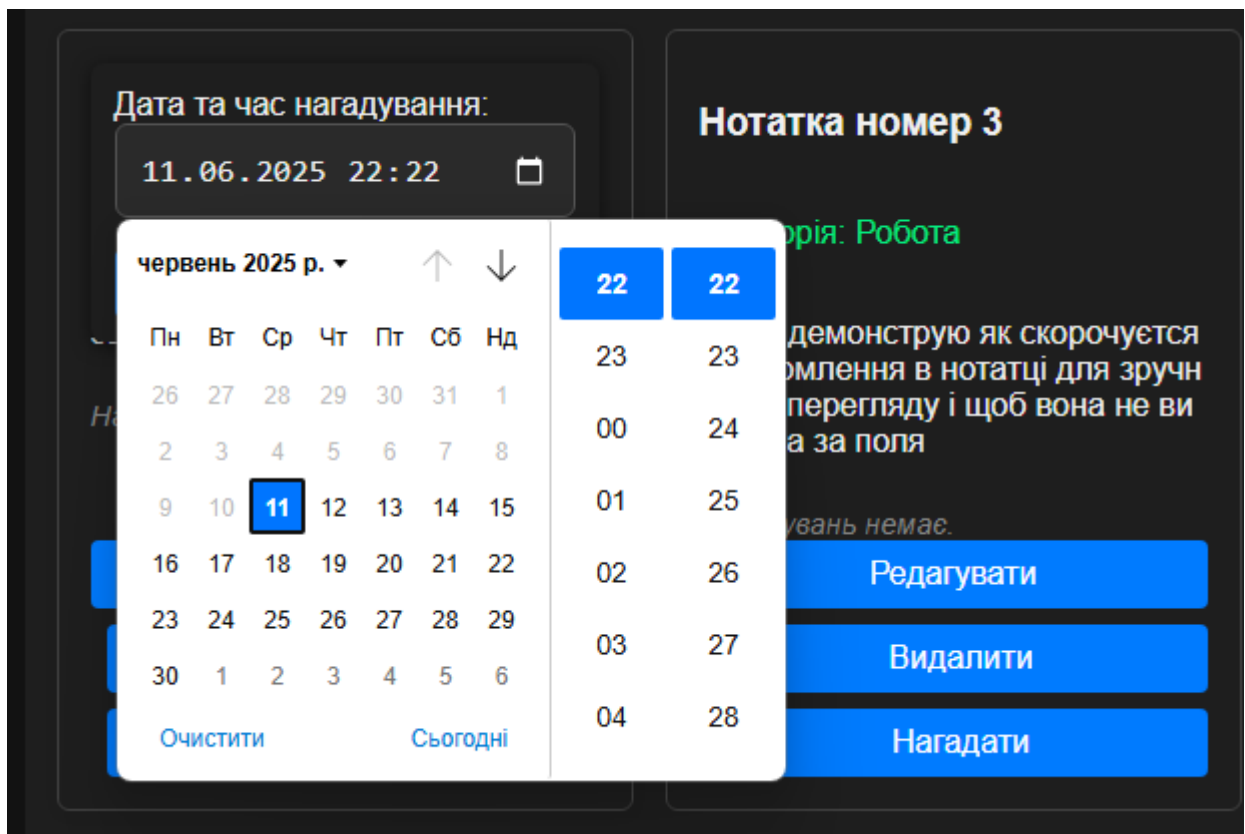


Рисунок 3.20 – Вибір часу й часу нагадування

Якщо обрати дату в минулому, під селектором з'являється повідомлення помилкою жовтим кольором, наприклад «Має бути 11.06.2025 22:15 або пізніша дата», і кнопку «Нагадати» заблоковано до коректного вибору.

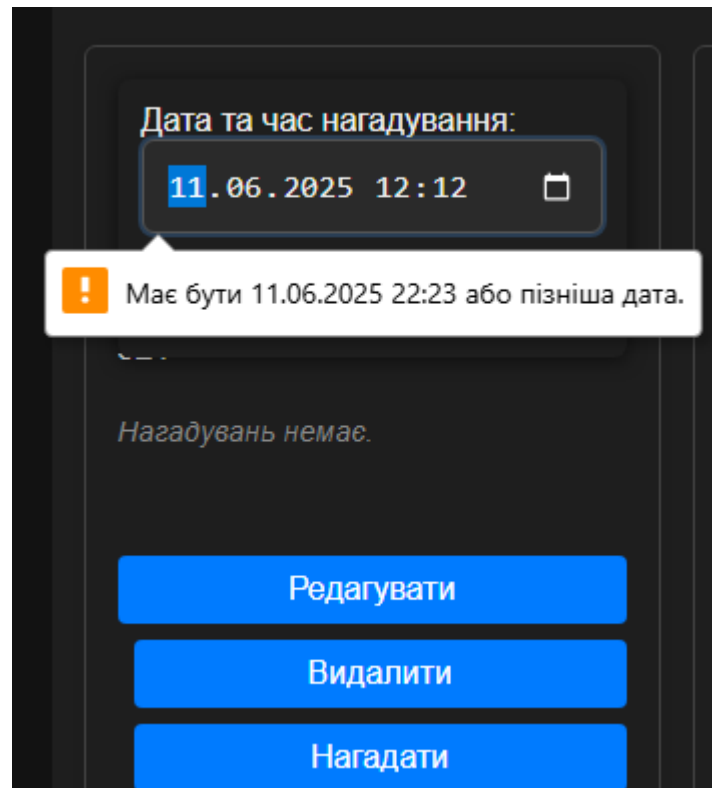


Рисунок 3.21 – Повідомлення про помилку при виборі минулого часу

Після успішного створення нагадування воно з'являється одночасно в інтерфейсі та надсилає системне сповіщення прямо на робочий стіл користувача, яке містить назву нотатки та джерело (наприклад, Google Chrome).

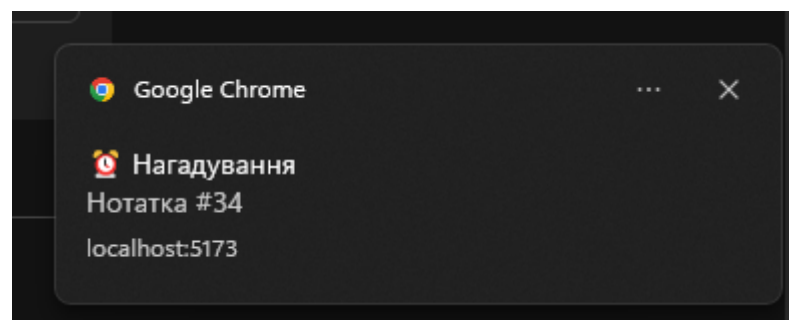


Рисунок 3.22 – Браузерне оповіщення

### 3.4.7. Оформлення та адаптивність

Інтерфейс застосунку автоматично підлаштовується під ширину екрана за допомогою CSS Grid і Flexbox. На великих екранах створюється двоколонковий макет із формою створення нотаток і списком нотаток поруч, тоді як на

мобільних пристроях усі блоки розташовуються один під одним для зручного вертикального скролу.

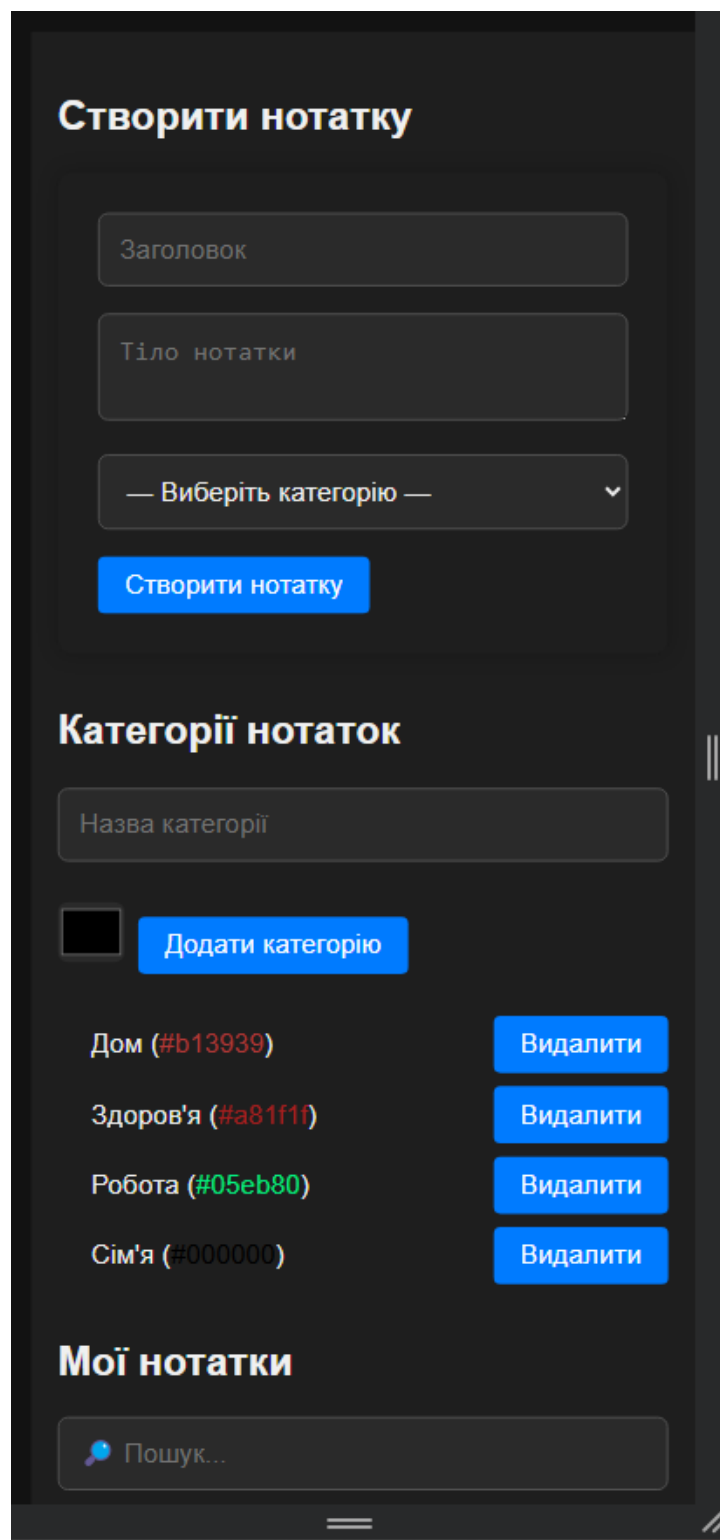


Рисунок 3.23 – Мобільний вигляд інтерфейсу

У мобільному режимі кнопки та поля залишаються достатньо великими для натискання пальцем, а між елементами збережено комфортні відступи. Панель пошуку та фільтр категорій ледь помітно приховується за невеликим

«гамбургер»-баром, який за натисканням розгортає фільтри зверху, щоб зберегти місце на екрані.

Завдяки такому адаптивному дизайну користувач може працювати з нотатками на будь-якому пристрої — від десктопа до смартфона — без втрати функціональності чи зручності.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. <https://evernote.com/>
2. <https://keep.google.com/>
3. <https://simplenote.com/>
4. <https://www.notion.com/>
5. <https://to-do.office.com/>
6. <https://react.dev/learn>
7. <https://expressjs.com/>
8. <https://www.postgresql.org/docs/>

### **Висновок**

Під час реалізації дипломного проекту було розроблено повнофункціональний веб-застосунок під назвою Personal Notes App. Він поєднує безпечну автентифікацію користувачів, інтуїтивно зрозумілий інтерфейс для створення, читання, оновлення та видалення нотаток, а також гнучку категоризацію та потужну функцію пошуку та фільтрації з інтегрованим механізмом нагадувань. Для фронтенду було обрано React у поєднанні з Vite, щоб забезпечити швидке завантаження та плавне оновлення компонентів; бекенд базується на Node.js та Express, що забезпечує модульний та легкий у обслуговуванні REST API. Дані зберігаються в базі даних PostgreSQL, продумана схема якої з чітко визначеними зв'язками та ретельно встановленими індексами забезпечує як цілісність даних, так і високу продуктивність. Реалізована REST-архітектура забезпечує всі кінцеві точки для керування користувачами, нотатками, категоріями та нагадуваннями. Для захисту маршрутів використовуються токени JWT та хешування паролів, а додаткове проміжне програмне забезпечення гарантує, що лише автентифіковані користувачі мають доступ до захищених ресурсів. Інтерфейс користувача розроблено як односторінковий застосунок та автоматично адаптується до різних розмірів екрана; Усі зміни відображаються в режимі реального часу без необхідності оновлення сторінки, що робить роботу з нотатками особливо ефективною та зручною. Особливу увагу було приділено впровадженню push-сповіщень та перевірці дат нагадувань, що гарантує, що важливі події завжди своєчасно надходять до користувача.

Практична перевага розробленого застосунку полягає в його універсальності: додаток Personal Notes може використовуватися як на смартфоні так і на персональному комп'ютері. Чіткий розподіл обов'язків між компонентами та модульна архітектура дозволяють легко масштабувати та розширювати функціональність.