

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБЗАСТОСУНКУ ІНФОРМАЦІЙНОЇ
ДОШКИ
DEVELOPMENT OF THE INFORMATION BOARD WEB
APPLICATION

Виконав: студент 4 курсу, групи КНТ-210сп
Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ДЕМ'ЯНЕНКО Є.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ГЛАДКОВА О.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки

(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки

(назва освітньої програми)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій СУББОТІН

“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДЕМ'ЯНЕНКА Євгена Віталійовича

(ПРИЗВИЩЕ, ім'я, по-батькові)

1. Тема проєкту (роботи) Розробка вебзастосунку інформаційної дошки.
Development of the Information Board Web Application

керівник проєкту (роботи) к.т.н., ГЛАДКОВА Ольга Миколаївна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по-батькові)

затверджені наказом закладу вищої освіти від “04” квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 01 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1 Аналіз предметної області. 2 Вибір методів і технологій
розробки. 3 Розробка програмного продукту. 4 Експлуатація та тестування
програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень,
кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю. ст.викладач		

7. Дата видачі завдання «04» квітня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів дипломного проєкту (роботи)	Строк виконання проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	1-2 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки	2-3 тиждень	Розділ 2
4	Розробка програми	3-4 тиждень	Розділ 3
5	Тестування програми	4-5 тиждень	Розділ 4
6	Оформлення пояснювальної записки	5-6 тиждень	
7	Нормконтроль та рецензування	7 тиждень	
8	Захист роботи	8 тиждень	

Студент(ка)

_____ Євген ДЕМ'ЯНЕНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Ольга ГЛАДКОВА
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 88 с., 4 табл., 20 рис., 3 дод., 31 джерело.

ІНФОРМАЦІЙНА ДОШКА, ЗАМІТКИ, JAVASCRIPT, REACT.

Об'єкт дослідження – програмне забезпечення ведення інформаційних дошок.

Предмет дослідження – методи та технології розробки програмного забезпечення для створення інформаційних дошок.

Мета роботи – розробка вебзастосунку для створення інформаційної дошки, що надає користувачам зручний інструмент для зберігання, організації та керування нотатками.

Матеріали, методи та технічні засоби: IntelliJ IDEA, JavaScript, React, MERN.

Актуальність дипломної роботи полягає у тому, що інформаційні дошки є популярними інструментами для організації та зберігання інформації в цифровому форматі. Вони забезпечують зручний спосіб зберігання нотаток, списків завдань, нагадувань та інших важливих даних. Розробка програмного засобу для інформаційної дошки має великий потенціал для полегшення повсякденних завдань користувачів і покращення їх продуктивності.

Очікувані результати роботи включатимуть розробку та реалізацію програмного засобу для створення інформаційної дошки, який буде мати зручний та зрозумілий інтерфейс та функціонал.

ABSTRACT

Explanatory note to the bachelor's thesis: 88 pages, 4 tables, 20 figure, 3 appendixes, 31 sources.

INFORMATION BOARD, NOTES, JAVASCRIPT, REACT.

The object of research: software for maintaining information boards.

The subject of research: methods and technologies of software development for creating an information board.

Purpose: To develop a whiteboard software tool that provides users with a convenient tool for storing, organizing, and managing notes.

Materials, methods and technical means: IntelliJ IDEA, JavaScript, React, MERN.

The relevance of the thesis lies in the fact that information boards are popular tools for organizing and storing information in digital format. They provide a convenient way to store notes, to-do lists, reminders, and other important data. Dashboard software development has great potential to ease users' daily tasks and improve their productivity.

The expected results of the work will include the development and implementation of a software tool for creating an information board, which will have a convenient and understandable interface and functionality.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Визначення та поняття веброзробки.....	10
1.1.1 Frontend розробка.....	10
1.1.2 Backend розробка	11
1.1.3 Вебдизайн	12
1.2 Етапи створення вебпродуктів та їх оптимізація.....	12
1.2.1 Етапи веброзробки.....	13
1.2.2 Оптимізація.....	14
1.3 Огляд інформаційних дошок для нотаток	17
1.3.1 «Блокнот Онлайн»	17
1.3.2 «Rapidtables».....	19
1.3.3 «Online Notepad»	20
1.3.4 «aNotepad»	21
1.4 Постановка завдання та вимоги до проекту	23
1.4.1 Мета.....	23
1.4.2 Функціональність.....	24
1.4.3 Інтерфейс	25
1.5 Висновки до розділу 1	25
2 Вибір методів і технологій розробки	26
2.1 Мови програмування	26
2.1.1 JavaScript.....	26
2.1.2 PHP	27
2.1.3 Python	28
2.2 Фреймворки та база даних	30
2.2.1 MERN-стек	31

2.2.2 MongoDB	32
2.2.3 TailwindCss	33
2.3 Середовище розробки та редактор коду	33
2.3.1 Visual Studio Code	34
2.3.2 Sublime Text	35
2.3.3 Atom.....	36
2.3.4 IntelliJ IDEA	37
2.4 Висновок до розділу 2	38
3 Розробка програмного продукту	40
3.1 Структурна схема розробки	40
3.2 Клієнтська частина.....	41
3.2.1 Головна сторінка додатку	42
3.2.2 Елемент «нотатка»	42
3.2.3 Створення «нотатки»	43
3.2.4. Бізнес-логіка	46
3.3 Серверна частина	47
3.3.1 Контролер записів.....	48
3.3.2 API сервер	49
3.3.3 Взаємодія з базою даних	50
3.4 Висновки за розділом 3	52
4 Експлуатація та тестування програми	53
4.1 Призначення й умови використання програми.....	53
4.2 Опис інтерфейсу та функціоналу програмного продукту	54
4.3 Тестування програми	57
4.4 Висновки за розділом 4	59
Висновки	61
Перелік джерел посилань	63
Додаток А Технічне завдання	66
Додаток Б Текст програми	69
Додаток В Слайди презентації.....	81

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CSS — каскадні таблиці стилів;

IDE — інтегроване середовище розробки;

HTTP — протокол передачі гіпертексту;

JS — JavaScript;

MERN — MongoDB, Express, React, Node.js;

MongoDB — нереляційна система управління базами даних;

Node.js — платформа для виконання JavaScript на сервері;

React — бібліотека для розробки інтерфейсів користувача на JavaScript;

БД — база даних;

ПЗ — програмне забезпечення;

ПП — програмний проєкт.

ВСТУП

Інформаційні дошки є популярними інструментами для організації та зберігання інформації в цифровому форматі. Вони надають користувачам зручний спосіб зберігання нотаток, списків завдань, нагадувань та інших важливих даних. Завдяки цим програмним продуктам, люди можуть швидко зафіксувати свої ідеї, організувати свої завдання, зберігати важливі відомості та керувати своїм робочим процесом.

Метою даної дипломної кваліфікаційної роботи є розробка програмного засобу для створення інформаційної дошки, що надає користувачам зручний інструмент для зберігання, організації та керування нотатками. Для досягнення цієї мети будуть використані методи та технології розробки програмного засобу, зокрема IntelliJ IDEA, JavaScript, React та MERN.

Актуальність дипломної роботи полягає у тому, що інформаційні дошки мають великий потенціал для полегшення повсякденних завдань користувачів і покращення їх продуктивності. Завдяки зручному інтерфейсу та функціоналу програмного засобу для інформаційної дошки, користувачі зможуть ефективно організовувати свою роботу, легко зберігати та швидко отримувати доступ до важливої інформації. Очікувані результати роботи включатимуть розробку та реалізацію програмного засобу для створення інформаційної дошки, який задовольнить потреби користувачів та сприятиме їхній продуктивності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення та поняття веброзробки

Веброзробка — процес створення вебсайтів або вебзастосунків. Основними етапами процесу є вебдизайн, верстка сторінок, програмування на стороні клієнта і сервера, а також конфігурування вебсервера[1].

Вебсайт, або просто сайт (англ. website, від web - павутина, «веб» і site – «місце») — це одна або сукупність вебсторінок, доступних в інтернеті через протоколи HTTP/HTTPS; Вебсайт — це місце в інтернеті, яке визначається своєю адресою (URL), має свого власника і складається з вебсторінки, які сприймаються як єдине ціле [2].

1.1.1 Frontend розробка

Frontend розробка — це процес створення вебзастосунків, що включає розробку імовірного вигляду і функціональності вебінтерфейсу. Основна задача frontend розробника полягає в створенні користувацького інтерфейсу, який буде максимально зручним і зрозумілим для користувачів.

Основними компонентами, що входять в процес Frontend розробки, є:

- HTML (HyperText Markup Language): мова розмітки, яка використовується для створення структури вебсторінки;
- CSS (Cascading Style Sheets): мова стилів, яка використовується для надання вебсторінкам зовнішнього вигляду, дизайну і форматування;
- JavaScript: мова програмування, що використовується для розробки функціональності вебсторінки, такої як анімація, обробка подій, робота з формами та інше;
- фреймворки (наприклад, React, Angular, Vue): це зручні інструменти для розробки вебдодатків, які містять набір готових компонентів і функціоналу, який значно спрощує процес розробки та зменшує час, необхідний для його виконання.

Також до завдань frontend розробника можуть входити тестування вебсторінок на різних пристроях та розмірах екрана, оптимізація сторінок для пошукових систем (SEO), підтримка вебсторінки в роботі на різних браузерах та пристроях [3].

1.1.2 Backend розробка

Backend розробка — це процес створення серверної частини вебдодатків або програмного забезпечення, яке відповідає за зберігання, обробку та передачу даних з сервера на клієнтську сторону. Основні компоненти backend розробки перелічені нижче.

Серверний фреймворк: це набір інструментів та бібліотек, які використовуються для створення серверного додатка. До популярних фреймворків мовою програмування Python відносяться Django та Flask, на мові програмування PHP - Laravel та Symfony, мовою програмування Java - Spring.

Бази даних: це програмне забезпечення для зберігання даних, які можуть бути використані на сервері. До популярних баз даних належать MySQL, PostgreSQL та MongoDB.

Мови програмування: це інструменти, які використовуються для створення серверних додатків. До популярних мов програмування в backend розробці відносяться Python, PHP, Java, Ruby та Node.js.

API: це інтерфейс, який надає доступ до функціональності серверного додатка через мережу Інтернет. API використовується для обміну даними між різними додатками.

Безпека: це заходи, які приймаються для захисту сервера від злоумисників. До заходів безпеки в backend розробці відносяться аутентифікація, авторизація, шифрування та захист від SQL-ін'єкцій та XSS-атак.

Інструменти розробки: це програми та інші інструменти, які використовуються для створення та тестування серверного додатка. До

популярних інструментів розробки в backend розробці відносяться Visual Studio Code, Sublime Text, PyCharm та IntelliJ IDEA [4].

1.1.3 Вебдизайн

Вебдизайн — це процес створення вебсайтів та інтерфейсів користувача для вебзастосунків. Вебдизайнер відповідає за створення дизайну вебсайту, включаючи вибір кольорової гама, шрифтів, графіки, розміщення елементів інтерфейсу та іншого контенту на сторінці.

Основні елементи вебдизайну включають:

- графіка та фотографії: вебдизайнер відповідає за вибір і розміщення графіки та фотографій на вебсторінці;
- кольорова гама: кольорова гама вебсайту повинна бути відповідною до цілей та цільової аудиторії, вебдизайнер відповідає за вибір кольорів та їх відтінків;
- шрифти: вибір шрифтів також відіграє важливу роль в вебдизайні, оскільки допомагає створити відчуття бренду та забезпечити зручне читання контенту;
- розміщення елементів інтерфейсу: вебдизайнер повинен враховувати важливість розміщення елементів на сторінці, таких як кнопки, меню, поля вводу та інше;
- взаємодія з користувачем: вебдизайнер відповідає за створення ефективного та зручного інтерфейсу для користувачів, зокрема за взаємодію з кнопками, посиланнями, формами введення даних та іншими елементами [5].

1.2 Етапи створення вебпродуктів та їх оптимізація

Розробка вебпродукту — це складний процес, незалежно від масштабів майбутнього проєкту, який має велике значення для успіху бізнесу. Створення вебсторінки, будь то односторінковий лендінг чи глобальний інтернет-

магазин, потребує великої уваги, часу, фінансових вкладень і залучення команди фахівців.

1.2.1 Етапи веброзробки

Для успішної розробки вебпродукту потрібно ретельно продумати всі етапи створення вебсторінки та дотримуватись наміченого плану, оскільки незалежно від розміру майбутнього проєкту виконання великого обсягу робіт є обов'язковим.

Важливо зазначити що етапи веброзробки можуть відрізнятись залежно від підходу та методології, але загалом можна виділити наступні етапи.

Планування: цей етап включає визначення цілей проєкту, аналіз цільової аудиторії, вибір технологій та інструментів, складання технічного завдання і планування бюджету.

Дизайн: на цьому етапі візуально створюється майбутній вебсайт або вебдодаток. Створюються макети, дизайн-концепції, кольорові схеми та інші важливі деталі.

Розробка: на цьому етапі розроблюється функціональність вебсайту або вебдодатку. Розробники створюють код, базу даних, забезпечують взаємодію з сервером, додають функціональні елементи та інші необхідні компоненти.

Тестування: на цьому етапі вебсайт або вебдодаток перевіряється на наявність помилок та на відповідність вимогам технічного завдання. Тестування може включати тестування взаємодії користувача з вебсайтом, перевірку швидкості завантаження, адаптивності та ін.

Реліз та підтримка: після успішного тестування, вебсайт або вебдодаток розгортається на сервері та запускається для користувачів. Після запуску важливо забезпечити підтримку вебсайту, забезпечення безпеки та регулярне оновлення [6].

1.2.2 Оптимізація

Оптимізація, в загальному розумінні — це комплекс заходів, направлених на надання будь-чому найвигідніших характеристик [7].

Оптимізація веброзробки — це процес покращення продуктивності та ефективності розробки вебпродуктів, що передбачає застосування різноманітних методів та інструментів, які дозволяють зменшити розмір файлів, прискорити написання коду, спростити процес розробки, скоротити час завантаження сторінок та поліпшити користувацький досвід.

CSS-методології — це набір правил та рекомендацій для структурування та організації CSS-коду. Вони допомагають забезпечити більшу чіткість, стабільність та легкість управління CSS-кодом, особливо в проєктах з великою кількістю CSS.

БЕМ (Block, Element, Modifier) — одна з найпопулярніших на даний момент, в основі якої лежить домовленість про те, як розбивати інтерфейс на певні незалежні блоки. Розшифровується аббревіатура як «Блок Елемент Модифікатор». Цей підхід до web-розробки дозволяє швидко створювати сайти з гнучкою архітектурою.

У всіх технологіях (CSS, JavaScript, HTML) одна і та ж БЕМ-сутність завжди повинна бути названа однаково. Основна ідея домовленості по іменуванню — вкласти сенс в імена і зробити їх максимально інформативними для розробника [8].

HTML-препроцесори. Популярність препроцесорів для CSS вплинула на створення таких технологій і для HTML. Їхня мета абсолютно ідентична: спростити написання коду і скоротити час на нього, врятувати програміста від багаторазових повторень рядків і однакових тегів.

Пакувальники JavaScript. Ці інструменти допомагають програмістам збирати та об'єднувати JavaScript-файли в один, щоб зменшити їх кількість та зменшити час завантаження вебсторінки.

Webpack — це пакувальник, з відкритим кодом, статичних модулів для сучасних додатків JavaScript. Він створений переважно для JavaScript, але може трансформувати зовнішні ресурси, такі як HTML, CSS та зображення, якщо підключити відповідні завантажувачі. Webpack приймає модулі із залежностями та генерує статичні файли, що представляють ці модулі [9].

Gulp — це інструмент збірки JavaScript, який забезпечує потокову обробку файлів та є простим у використанні. Він дозволяє використовувати різні плагіни, такі як Babel для транспірації коду або UglifyJS для зменшення розміру файлів.

Parcel — це пакувальник для веб-додатків для розробників з різним досвідом. Він пропонує чудову швидку роботу з використанням багатоядерної обробки і не вимагає налаштувань. Parcel має вбудований сервер розробки, який буде автоматично збирати заново ваш додаток, так як підтримує гарячу заміну модуля для збільшення швидкості розробки [10].

Фреймворки — це набір бібліотек, інструментів та правил, які допомагають програмістам швидше розробляти складні вебдодатки на JavaScript. Ось декілька популярних фреймворків JavaScript:

Angular — це повноцінний MVC-фреймворк від Google з підтримкою Typescript та Web Workers з коробки, а також з прекрасним CLI, за допомогою якого ви можете згенерувати шаблони для компонентів, маршрутів, сервісів за допомогою простих команд командного рядка.

Vue.js — це фреймворк, який знайшов баланс між обмеженнями та гнучкістю. Його ядро вирішує передусім задачі представлення даних, тому він легко інтегрується в наявні проекти поступово. Проте підходить для створення повноцінних SPA (Single-Page-Application) застосунків, адже має повний набір функціоналу [11].

React — це фреймворк від Facebook, який дозволяє розробляти динамічні інтерфейси за допомогою компонентного підходу. React широко використовується в розробці односторінкових додатків та мобільних додатків.

Система керування версіями (англ. Version Control System, VCS) — це програмний інструмент, який дозволяє розробникам зберігати, відстежувати та керувати змінами в програмному коді та інших документах. VCS зберігає всі версії документів та коду, що дозволяє розробникам з легкістю повертатися до попередніх версій документів, відновлювати випадково видалені файли, відстежувати, хто та коли вніс зміни в код і багато іншого.

GIT — це кросплатформна система контролю і управління версіями коду, розроблена у 2005 році. GIT поширюється за безкоштовній основі за GNU-ліцензією і являє собою набір утиліт командного рядка. Дозволяє працювати над одним проектом цілим командам користувачів, де кожна зміна додається у вигляді «комітів» (commit — знімок змін). Це дозволяє відстежувати різні версії коду документа й при необхідності скасовувати зміни та повертатися до попередніх версій. Також є можливість створювати окремі гілки — «форки» (відхилення, модифікована копія проекту).

Почати застосовувати GIT на практиці можна з допомогою спеціальних Git-хостингів і репозиторіїв [12].

Текстовий редактор — це програма для редагування текстових файлів. Текстові редактори використовуються розробниками для написання програмного коду, написання документації, редагування конфігураційних файлів та багато іншого.

Компоненти. У веброботці компонентами називають частини вебдодатка, які можна використовувати багаторазово, додавати, змінювати та використовувати їх в різних місцях додатка. Компоненти допомагають зменшити час розробки, покращити структуру коду та забезпечити більшу повторюваність [13].

Далі наведені основні типи компонентів у веброботці.

HTML-компоненти — це частини HTML-коду, які можна використовувати багаторазово на сторінці або в інших документах. Наприклад, заголовки, списки, кнопки, форми та інші.

CSS-компоненти — це набір стилів, які можна використовувати багаторазово на сторінці або в інших документах. Наприклад, стилі для кольорів, шрифтів, фонів, рамок, позиціювання та інші.

JavaScript-компоненти — це функції або класи JavaScript, які можна використовувати багаторазово на сторінці або в інших документах. Наприклад, скрипти для валідації форм, анімації, обробки подій та інші.

Бібліотеки — це набір компонентів, які можна використовувати для певної задачі або функціоналу. Наприклад, бібліотеки для роботи з DOM, AJAX, датами та інші.

Фреймворки — це комплексні набори бібліотек та інших компонентів, які надають структуру та функціональність для розробки вебдодатків. Наприклад, Angular, React, Vue та інші [14].

1.3 Огляд інформаційних дошок для нотаток

Інформаційна онлайн дошка для нотаток — це вебпрограма, яка надає можливість створювати, зберігати та організовувати нотатки в цифровому форматі. Вона дозволяє зручно записувати і зберігати різні види інформації, такі як текстові записи, списки завдань, ідеї, нагадування та інші помітки.

Далі детально розглянемо деякі існуючі програмні продукти цієї області.

1.3.1 «Блокнот Онлайн»

«Блокнот Онлайн» - вебдодаток для створення та управління нотатками. Він надає різноманітні функції для зручного ведення та організації інформації (рис. 1.1).

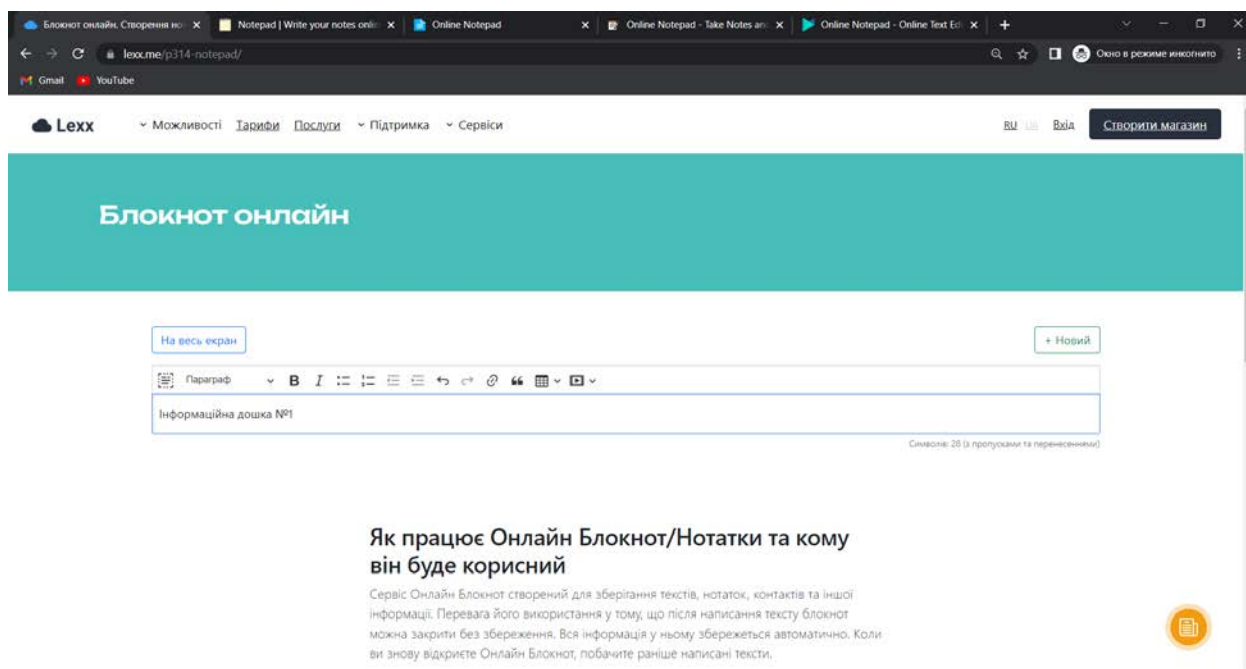


Рисунок 1.1 — Інтерфейс програми «Блокнот Онлайн» [15]

Основні функції даної програми наведені нижче.

Створення нотаток. Можна швидко створити нові записи, вводячи потрібний текст. Програма дозволяє створювати необмежену кількість нотаток.

Автоматичне збереження. Перевагою цієї програми є можливість закрити вкладку з записами без збереження, оскільки вся інформація зберігається автоматично. При наступному відкритті програми будуть відображатись раніше написані тексти.

Простий та зрозумілий інтерфейс. Програма має простий та інтуїтивно зрозумілий інтерфейс, що дозволяє легко орієнтуватися та використовувати всі функції.

Відсутність реєстрації. Для використання «Онлайн Блокнота» не потрібно реєструватися або створювати обліковий запис. Можна відразу розпочати створення записів.

Безкоштовна та без реклами. Програма є доступною та повністю безкоштовною для всіх користувачів. Крім того, вона не містить рекламних повідомлень, що полегшує використання.

Захист приватності. Всі дані зберігаються лише у браузері користувача і не передаються через Інтернет. Інформація не зберігається на серверах розробника і не доступна стороннім особам [15].

1.3.2 «Rapidtables»

«Rapidtables» — це вебпрограма для створення та редагування нотаток з різними функціями (рис. 1.2).

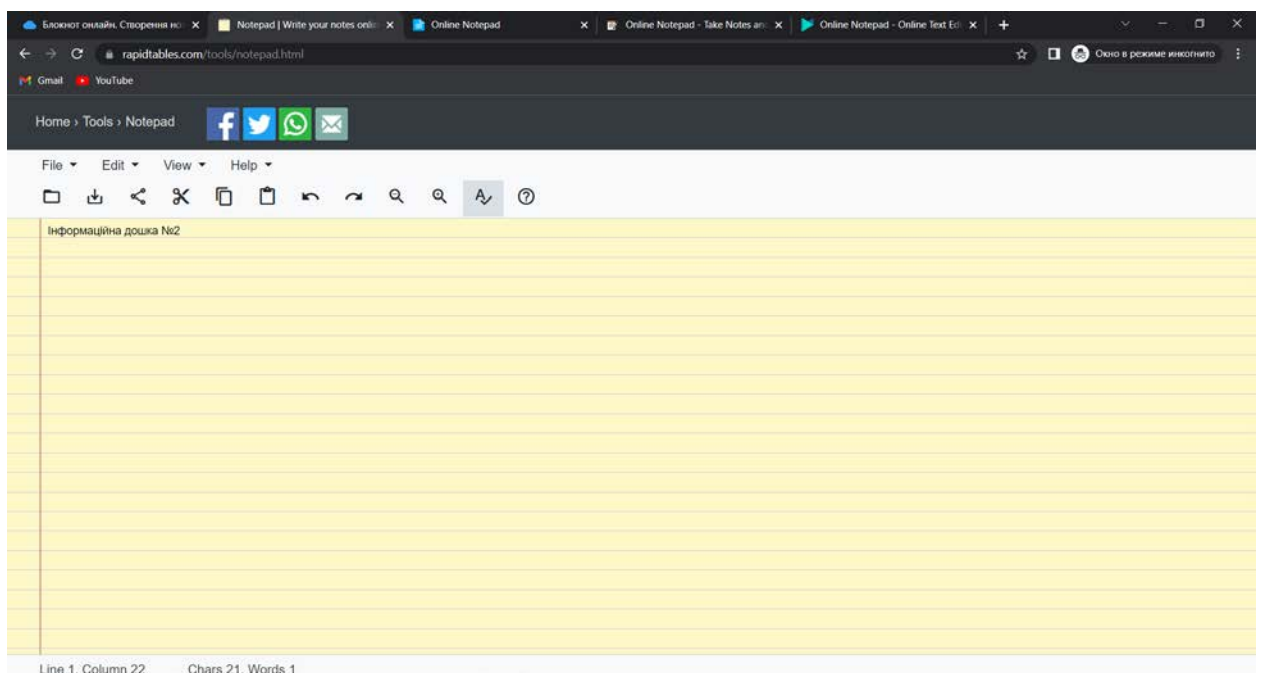


Рисунок 1.2 — Інтерфейс програми «Rapidtables» [16]

Основні функції програми з роботи [16] перераховані далі.

Створення та редагування нотаток: Можна створювати нові записи та редагувати існуючі.

Збереження та завантаження. Можна зберегти нотатки на локальний комп'ютер у вигляді текстових файлів. Також є можливість завантажувати збережені раніше записи для подальшого редагування.

Інструменти форматування. Програма надає ряд інструментів форматування тексту, таких як вирівнювання, списки, вставка посилань та

зображень, вибір шрифтів та інші опції форматування для зручності роботи з записами.

Пошук та заміна. Інструмент пошуку дозволяє знаходити конкретний текст у записах, а функція заміни дозволяє замінювати певний текст іншим.

Доступність та зручність. Програма доступна онлайн і не вимагає установки додаткового програмного забезпечення. Можна легко працювати з нотатками з будь-якого пристрою, який має доступ до Інтернету.

Безкоштовна та без реклами.

1.3.3 «Online Notepad»

«Online Notepad» інформаційна дошка для створення, редагування та збереження нотаток (рис. 1.3).

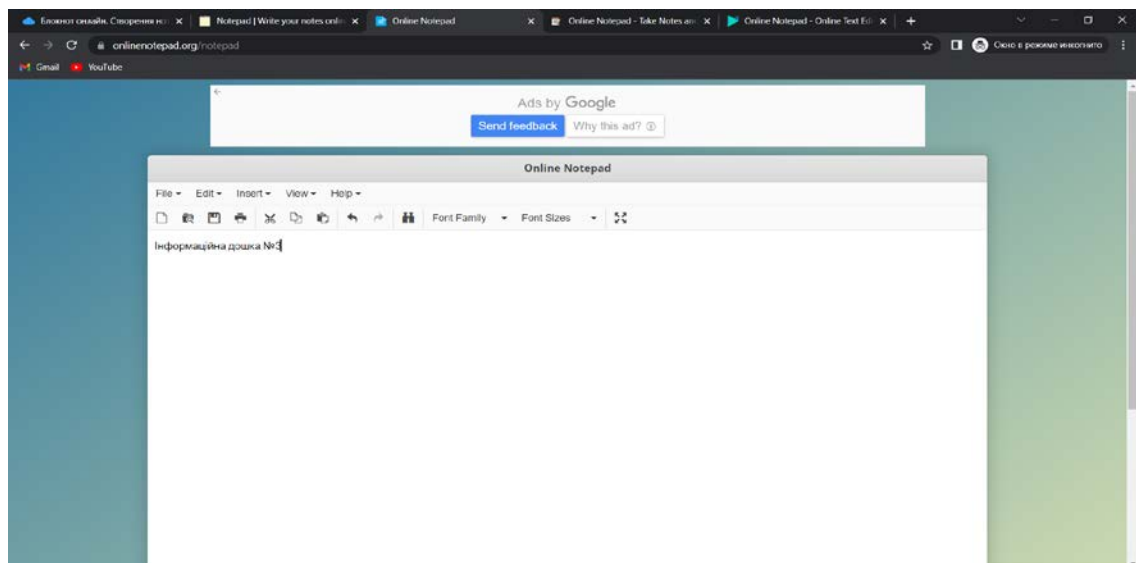


Рисунок 1.3 — Інтерфейс програми «Online Notepad» [17]

Далі наведені основні функції цієї програми.

Простий інтерфейс. «Online Notepad» має простий і легкий у використанні інтерфейс.

Безкоштовне використання. Програма доступна повністю безкоштовно, не потрібно реєструватися або створювати обліковий запис, щоб користуватися нею.

Автоматичне збереження. Нотатки автоматично зберігаються після внесення змін. Можна безперервно працювати над своїми записами без страху втрати даних.

Відкриття і збереження файлів. Програма дозволяє відкривати тексти з раніше збережених файлів і зберігати тексти як окремі файли на комп'ютері.

Підтримка форматування тексту. Можна формувати текст у своїх нотатках, використовуючи різні стилі та сімейства шрифтів, розміри, жирність, курсив та підкреслення.

Можливість друку. Можна надрукувати готові нотатки безпосередньо з програми, що зручно для збереження або обміну текстом в паперовій формі [17].

1.3.4 «aNotepad»

«aNotepad» — це онлайн-інструментом для створення, редагування та збереження нотаток (рис. 1.4).

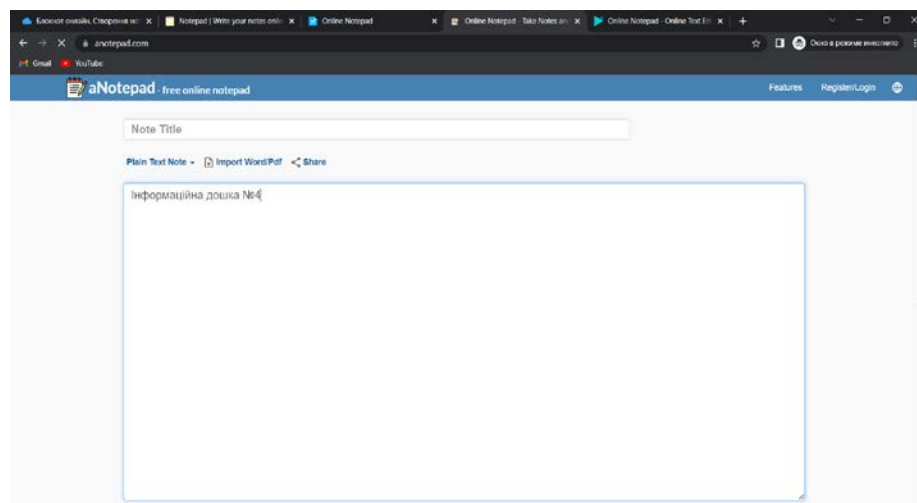


Рисунок 1.4 — Інтерфейс програми «aNotepad» [18]

Веб-програма пропонує ряд корисних функцій.

Легкий доступ. «aNotepad» є вебпрограмою, тому можна отримати доступ до нотаток з будь-якого пристрою з підключенням до Інтернету. Не потрібно завантажувати або встановлювати програму.

Створення та редагування нотаток. Можна створювати нові записи та редагувати існуючі. В програму можна завантажувати готовий текст в різних форматах.

Збереження нотаток. «aNotepad» автоматично зберігає записи після внесення змін. Крім того, можна зберегти нотатки вручну, натиснувши спеціальну кнопку.

Підтримка списків. Можна створювати нумеровані та марковані списки, що допомагає структурувати ваші нотатки та зробити їх більш організованими.

Сортування. Можна сортувати готові нотатки за назвою заголовків (алфвітом) або датою створення.

Доступність і приватність. Програма дозволяє працювати з записами без необхідності реєстрації або облікового запису. Нотатки залишаються приватними і не доступними для інших користувачів [18].

Для детальнішого порівняння функціоналу даних програм розробимо наступну таблицю (таб. 1.1).

З порівняльної таблиці можна зробити наступні висновки. Усі чотири програми, «Rapidtables», «Online Notepad», «aNotepad» та «Блокнот Онлайн», надають можливості створення та редагування нотаток, а також збереження і завантаження файлів.

Програма «Rapidtables» пропонує ширший набір інструментів форматування тексту та функцію пошуку та заміни, яка відсутня у решти програм. «Online Notepad» має додаткові функції підтримки списків та друку, «aNotepad» пропонує автоматичне збереження нотаток, а «Блокнот Онлайн» має простий інтерфейс. Всі програми є доступними, безкоштовними та не містять реклами.

Таблиця 1.1 – Порівняння оглянутих програмних продуктів (аналогів)

Функції	Rapidthables	Online Notepad	aNotepad	Блокнот Онлайн
Створення та редагування нотаток	+	+	+	+
Збереження та завантаження файлів	+	+	+	+
Інструменти форматування	+	Частково	Частково	Частково
Пошук та заміна	+	-	-	-
Доступність та зручність	Онлайн	Онлайн	Онлайн	Онлайн
Безкоштовна та без реклами	+	+	+	+
Простий інтерфейс	-	+	+	+
Автоматичне збереження	-	+	+	+
Відкриття і збереження файлів	-	+	+	+
Підтримка форматування тексту	-	+	+	+
Можливість друку	-	+	+	+
Сортування	-	+	+	-
Доступність і приватність	+	+	+	+

1.4 Постановка завдання та вимоги до проекту

1.4.1 Мета

Метою даного ПП є проєктування і створення інформаційної дошки для створення нотаток. Веб продукт дозволить зручно записувати і зберігати

різні види інформації, такі як текстові нотатки, списки завдань, ідеї, нагадування та інші записи.

1.4.2 Функціональність

Користувачі повинні мати можливість створювати нові нотатки та зберігати їх у системі. Це може включати створення текстових записів, списків завдань, ідей, нагадувань та інших типів записів.

Перегляд усіх створених записів. ПЗ має відображати всі записи, які вносив раніше користувач. Дата створення запису. Програма повинна зберігати та відображати точну дату коли було створено запис. Видалення записів. Система повинна дозволяти користувачам видаляти необхідні записи та нотатки. Забезпечення безпеки та конфіденційності. Важливо, щоб ПЗ забезпечувало безпеку та конфіденційність нотаток користувачів. Це може включати шифрування даних, контроль доступу тощо. Зручний інтерфейс, мінімалістичний, лаконічний і інтуїтивно зрозумілий інтерфейс та вигляд.

На основі цих вимог була створена сценарій та діаграма прецедентів користувача (рис. 1.5).

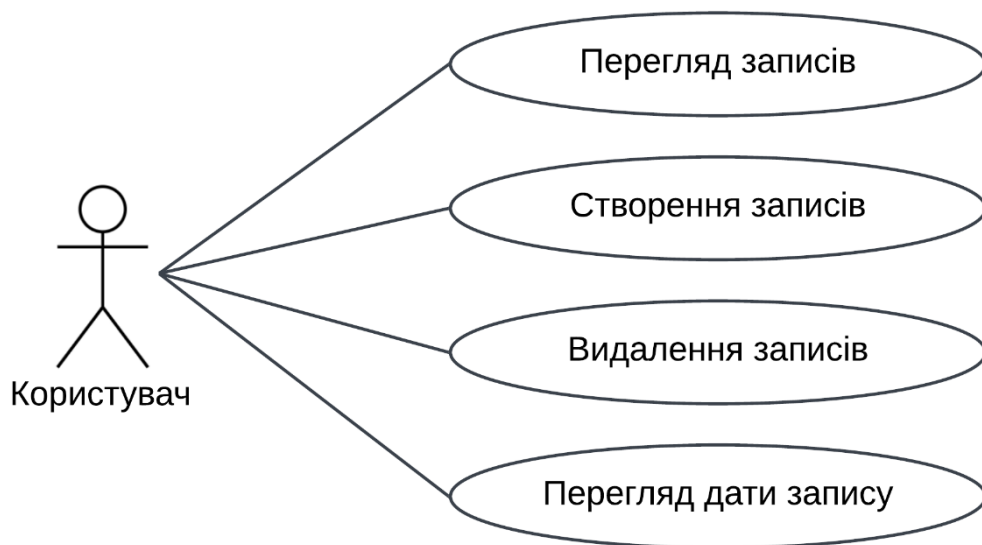


Рисунок 1.5 — Діаграма прецедентів користувача

1.4.3 Інтерфейс

Вимоги до інтерфейсу користувача.

Простота та зрозумілість. Інтерфейс інформаційної дошки повинен бути простим у використанні і зрозумілим для будь-якого користувача. Елементи управління та опції повинні бути логічно розташовані та інтуїтивно зрозумілі.

Естетика та зручність. Важливо, щоб візуальні елементи вебдодатку були привабливими та збалансованими, сприяли комфортній взаємодії користувача з системою.

Консистентність. Різні частини інтерфейсу повинні мати однорідний стиль та зовнішній вигляд. Це допомагає користувачеві швидше орієнтуватися та відчувати зручність в роботі з системою.

Адаптивність. Інтерфейс повинен адаптуватись під будь-який пристрій, з якого його буде відкрито.

1.5 Висновки до розділу 1

В даному розділі було розглянуто різні аспекти, пов'язані з дипломним проектом, а саме — створення інформаційної дошки для нотаток. Було визначено та пояснено поняття веброзробки, описано етапи створення вебпродуктів та їх оптимізації. Також був проведений огляд існуючих ІД для нотаток і досліджено їх функціональність.

В результаті аналізу було сформовано постановку завдання та вимоги до проекту. Метою ПП є створення зручного та функціонального вебпродукту, який дозволить користувачам зручно записувати, зберігати та організовувати різні типи нотаток. Вимоги до проекту включають створення та редагування записів, їх видалення, а також забезпечення безпеки даних.

2 ВИБІР МЕТОДІВ І ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Мови програмування

Існує багато мов програмування для розробки вебдодатків, серед яких кожна має свої особливості та використовується для різних цілей. Далі розглянемо детальніше основні з них.

2.1.1 JavaScript

JavaScript є високорівневою, інтерпретованою мовою програмування, яка використовується для розробки динамічних та інтерактивних веб-сторінок. Основними особливостями JavaScript наводяться далі.

JavaScript виконується безпосередньо у веб-браузері, що дозволяє створювати веб-сторінки зі змінною поведінкою та реагуванням на дії користувача. Він може змінювати вміст та стилізацію елементів сторінки, оброблювати події, взаємодіяти з користувачем і здійснювати запити до сервера.

JS має синтаксис, подібний до інших мов програмування, зокрема C і Java, що робить його легким для вивчення та розуміння. Він використовується для створення змінних, функцій, об'єктів та керування потоком виконання програми.

JavaScript є об'єктно-орієнтованою мовою, що дозволяє створювати та маніпулювати об'єктами. Об'єкти можуть мати властивості (змінні) та методи (функції), і їх можна унаслідувати та використовувати для структурування коду та повторного використання.

JS використовує динамічну типізацію, що означає, що типи змінних визначаються автоматично під час виконання програми. Змінні можуть змінювати свій тип під час виконання, що надає більшу гнучкість, але вимагає уваги при роботі з даними.

Дана мова програмування має велику кількість фреймворків та бібліотек, таких як React, Angular, Vue.js, що полегшують розробку складних веб-додатків та дозволяють прискорити процес розробки.

Крім веброзробки, JavaScript також може використовуватись для розробки мобільних додатків (за допомогою фреймворків, таких як React Native або Ionic), серверного програмування (за допомогою Node.js) та інших додатків, що використовують JavaScript як мову розширення [19].

2.1.2 PHP

PHP (Hypertext Preprocessor) є серверною мовою програмування, спеціально розробленою для веб-розробки. Основні особливості PHP.

PHP призначений для розробки динамічних веб-сторінок і взаємодії з веб-серверами. Він може генерувати HTML, взаємодіяти з базами даних, обробляти форми, створювати, читати, записувати та видаляти файли на сервері.

Синтаксис PHP подібний до синтаксису мови C, що робить його легким для вивчення та використання. PHP-код вбудовується безпосередньо в HTML-сторінки, використовуючи теги `<?php ?>`.

PHP використовує слабку, динамічну типізацію, що дозволяє змінювати типи змінних під час виконання програми. Він автоматично визначає тип змінної залежно від присвоєння значення.

Дана мова програмування має велику та активну спільноту розробників, що розробляють фреймворки, бібліотеки та допоміжні інструменти. Це дозволяє легко знайти документацію, підтримку та готові рішення для багатьох завдань.

PHP має велику кількість вбудованих функцій та можливостей, а також підтримує розширення за допомогою розширень PHP, які додають додаткові функціональні можливості.

Ця мова програмування має вбудовану підтримку для багатьох типів баз даних, включаючи MySQL, PostgreSQL, Oracle, SQLite та інші [20].

2.1.3 Python

Python є високорівневою, інтерпретованою мовою програмування, яка зосереджується на простоті та зрозумілості коду. Основні риси та особливості Python.

Синтаксис Python дуже читабельний і лаконічний, що робить його легким для вивчення і розуміння. Він заснований на використанні пробілів для відділення блоків коду, замість фігурних дужок або інших роздільників.

Python має велику кількість вбудованих функцій і бібліотек, що робить його універсальним і потужним інструментом для різних задач. Від веб-розробки та наукових обчислень до штучного інтелекту та аналізу даних, Python має багато готових рішень.

Дана мова програмування підтримує об'єктно-орієнтоване програмування, що дозволяє розбити код на окремі об'єкти з властивостями і методами. Це сприяє структуруванню та повторному використанню коду, а також полегшує розвиток складних додатків.

Python має активну та велику спільноту розробників, що сприяє швидкому поширенню знань, підтримці та обміну досвідом. Існує багато ресурсів, форумів та бібліотек, які допомагають вирішувати проблеми та розширювати можливості мови.

Код, написаний на одній платформі (наприклад, Windows), може бути легко виконаний на інших платформах (наприклад, Linux або macOS) [21].

Для детальнішого порівняння цих мов програмування розробимо таблицю (таб. 2.1).

Таблиця 2.1 — Порівняння мов програмування для веброзробки

	JavaScript	PHP	Python
Тип мови	Клієнтська та серверна мова	Серверна мова	Універсальна мова
Використання	Веб-розробка, фронтенд	Веб-розробка, серверний сценарій	Веб-розробка, наукові обчислення, штучний інтелект, аналіз даних
Об'єктно-орієнтований	Так	Так	Так
Розширення	Фреймворки та бібліотеки	Розширення PHP	Багато вбудованих функцій та бібліотек, встановлення додаткових пакетів
Динамічна типізація	Так	Так	Так
Переносимість	Залежить від веб-браузера	Багато платформовий	Багато платформовий
Використання в інших областях	Мобільна розробка (React Native, Ionic), серверне програмування (Node.js)	-	Наукові обчислення, штучний інтелект, аналіз даних, автоматизація та скриптовання, робототехніка,
Швидкодія	Висока	Висока	Висока

Для розробки програми була обрана саме JavaScript з кількох причин. По-перше, JavaScript є однією з основних мов програмування для веб-розробки, що дозволяє створювати динамічні інтерактивні вебдодатки та

сайти. По-друге, вона має багатий стек технологій, який включає фреймворки і бібліотеки, такі як React, що спрощують розробку веб-інтерфейсу.

Крім того, JavaScript має широку підтримку та активну спільноту розробників, що забезпечує доступ до багато ресурсів, документації і готових рішень. Вона також є мовою з великою популярністю і використовується багатьма веб-розробниками по всьому світу.

2.2 Фреймворки та база даних

Фреймворки — це набір інструментів, бібліотек та загальних структур, які допомагають розробникам вебдодатків прискорити процес розробки та забезпечити консистентність, ефективність і розширюваність проєктів. Вони надають різноманітні функціональність та шаблони, які можна використовувати для реалізації різних аспектів веб-додатків.

Залежно від специфіки завдань, які вони можуть вирішувати, можна виділити три категорії.

Backend. Працюють виключно на стороні сервера. Вони орієнтовані на виконання окремих функцій, пов'язаних із зберіганням, захистом даних, забезпеченням життєво важливих процесів. З їх допомогою можна створювати прості сайти з досить обмеженим інтерфейсом. Відомі бекенд фреймворки: Django, Express.js, Symfony.

Frontend. Не пов'язані з логікою роботи програми, так як функціонують безпосередньо в браузері. Призначені для створення користувацьких інтерфейсів з різноманітною графікою, анімацією. Приклади: Svelte, Angular.

Fullstack. Універсальні інструменти, які ефективно працюють на обох сторонах. Їх особливість в тому, вони використовують одну мову програмування для серверної і клієнтської частин. Тому коли backend і frontend фреймворки передають між сторонами HTML-код, fullstack обмінюються даними. Приклад: Meteor [22].

2.2.1 MERN-стек

MERN-стек — це набір технологій, що використовуються для розробки веб-додатків, і включає:

- MongoDB — це документова база даних;
- Express.js — це веб-фреймворк для Node.js;
- React.js — це клієнтський JavaScript-фреймворк;
- Node.js — це платформа для розробки серверних додатків на JavaScript [23].

Express та Node утворюють середній (додатковий) рівень. Express.js є серверним веб-фреймворком, а Node.js — популярною та потужною платформою для розробки серверів на JavaScript. MERN є ідеальним підходом для роботи з JavaScript та JSON на всіх етапах розробки.

На верхньому рівні стеку MERN знаходиться React.js, декларативний фреймворк JavaScript для створення динамічних клієнтських додатків в HTML. React дозволяє будувати складні інтерфейси за допомогою простих компонентів, з'єднувати їх з даними на сервері і відтворювати їх у вигляді HTML.

Сильна сторона React полягає в обробці станових, з даними пов'язаних інтерфейсів з мінімальним кодом та мінімальними зусиллями, і він має всі необхідні функції для розробки сучасного веб-фреймворка: відмінна підтримка форм, обробка помилок, події, списки та багато іншого.

Наступний рівень — це серверний фреймворк Express.js, який працює в середовищі сервера Node.js. Express.js описує себе як «швидкий, нейтральний, мінімалістичний веб-фреймворк для Node.js». У Express.js є потужні моделі для маршрутизації URL (відповідність вхідного URL серверній функції) та обробки HTTP-запитів і відповідей.

За допомогою XML HTTP Requests (XHRs), GET або POST запитів з React.js фронтенду, можна підключатися до функцій Express.js, які виконують функції додатка, що розроблюється. Ці функції, в свою чергу, використовують

Node.js драйвери MongoDB, які працюють через зворотні виклики або за допомогою промісів, для доступу до даних у базі даних MongoDB.

Express.js (що працює на Node.js) і React.js роблять стек MERN повністю JavaScript/JSON. Express.js є фреймворком для створення серверних додатків, який обробляє HTTP-запити і відповіді, і спрощує відображення URL-адрес на серверні функції. React.js є фреймворком для побудови інтерактивних користувацьких інтерфейсів у HTML та здійснення зв'язку з віддаленим сервером.

Nodemon — «гаряче» перезавантаження вузла-сервера у разі змін. Завдяки Nodemon можна не зупиняти та не перезапускати сервер після внесення будь-яких змін [23].

Нижче наведено список найпоширеніших пакетів, які використовуються у проектах з Express.js [24]:

- Mongoose — це об'єктно-документний менеджер (ODM), який забезпечує взаємодію між додатком Express.js та базою даних MongoDB;
- bodyParser — бібліотека, яка дозволяє додатку Express.js читати тіло (зміст) вхідних запитів;
- DotENV — дозволяє використовувати файли з розширенням .env для роботи з конфіденційними даними;
- Passport.js — забезпечує аутентифікацію в нашому додатку і надає кілька різних методів аутентифікації;
- Validator — проста валідація багатьох типів даних;
- bcrypt — шифрування конфіденційних даних, наприклад, паролів.

2.2.2 MongoDB

MongoDB — це документоорієнтована система управління базами даних (СУБД) типу NoSQL, яка не вимагає опису схеми таблиць і використовує документи у форматі JSON для зберігання та організації даних. Вона надає гнучкість і масштабованість для зберігання великого обсягу даних,

а також підтримує розподілену архітектуру для високошвидкісного доступу до даних.

MongoDB була розроблена для зберігання даних у форматі JSON (технічно вона використовує бінарну версію JSON, відому як BSON), і все, починаючи від її інтерфейсу командного рядка до мови запитів (MQL, або мова запитів MongoDB), побудовано на JSON та JavaScript.

MongoDB працює дуже добре з Node.js і дозволяє легко зберігати, маніпулювати та представляти дані у форматі JSON на кожному рівні вашого додатка. Для хмарних додатків MongoDB Atlas робить це ще простіше, надаючи автоматично масштабований кластер MongoDB на області хмарних обчислень за вашим вибором всього кількома клацаннями кнопок [25].

2.2.3 TailwindCss

Tailwind CSS — це сучасний CSS-фреймворк, який надає набір готових класів для стилізації вебінтерфейсу. Замість написання окремих CSS-правил, можна використовувати класи з Tailwind CSS, які визначають різноманітні властивості стилів, такі як розміри, кольори, відступи, шрифти та інші. Це дозволяє швидко і зручно стилізувати елементи і складати їх у компоненти, що спрощує розробку веб-інтерфейсу. Tailwind CSS також пропонує можливість налаштування і розширення, що дозволяє адаптувати його під потреби конкретного проекту [26].

2.3 Середовище розробки та редактор коду

Редактор коду — це програма, яка використовується для написання, редагування та форматування програмного коду. Він надає зручне середовище для розробників, де вони можуть створювати та змінювати свій код.

Редактори коду зазвичай надають ряд корисних функцій.

Підсвічування синтаксису. Редактори підсвічують різні елементи коду (ключові слова, змінні, функції) різними кольорами, що полегшує читання та розуміння коду.

Автодоповнення. Редактори можуть автоматично доповнювати код, запропонувавши варіанти, що відповідають тому, що ви пишете. Це допомагає зекономити час та зменшити кількість помилок.

Відступи та форматування. Редактори можуть автоматично вирівнювати код та застосовувати правила форматування (наприклад, відступи), щоб зробити код більш зрозумілим та зручним для читання.

Налагодження. Деякі редактори надають можливість налагодження коду, дозволяючи вставляти точки зупинки, відстежувати значення змінних та аналізувати виконання програми.

Управління версіями. Редактори мають інтегровану підтримку систем керування версіями, що дозволяє вам легко працювати з репозиторіями та виконувати операції коміту, співпрацювати з іншими розробниками та керувати версіями вашого коду [27].

Існує багато редакторів коду, які можна використовувати для розробки програмного коду вебзастосунків. Далі розглянемо найпопулярніші з них.

2.3.1 Visual Studio Code

Visual Studio Code (VS Code) — це безкоштовний редактор коду, розроблений компанією Microsoft. Він є дуже популярним серед розробників завдяки своїй потужності, розширюваності та зручному інтерфейсу. Далі наведені деякі ключові особливості Visual Studio Code.

VS Code має велику кількість розширень, які дозволяють розширити його функціональність. Є можливість встановлювати розширення для підтримки різних мов програмування, інтеграції з інструментами розробки, автоматичного доповнення, тем оформлення та багато іншого.

Редактор коду надає кольорове підсвічування синтаксису для багатьох мов програмування, що полегшує читання та редагування коду. Крім того, він підтримує функцію IntelliSense, яка автоматично доповнює код та надає контекстні рекомендації на основі типів, функцій та змінних у проекті.

Програма надає можливість налагодження коду безпосередньо в редакторі. Можна встановлювати точки зупинки, відстежувати значення змінних, крокувати по коду та аналізувати виконання програми.

VS Code має вбудовану підтримку системи керування версіями Git. Можна виконувати операції коміту, розглядати внесені зміни, гілки, розв'язувати конфлікти злиття та багато іншого, прямо в редакторі.

Редактор коду має вбудований термінал, що дозволяє запускати команди та виконувати різні операції безпосередньо з редактора, що полегшує розробку та налагодження [28].

2.3.2 Sublime Text

Sublime Text — це швидкий і легкий редактор коду, призначений для програмістів і веброзробників. Ось деякі особливості Sublime Text.

Sublime Text відомий своєю швидкодією та ефективністю. Він дозволяє швидко відкривати та редагувати великі файли без затримок.

Текстовий редактор дозволяє створювати багато курсорів одночасно, що дозволяє редагувати багато рядків коду одночасно. Це дуже корисна функція для редагування і форматування великих блоків коду.

Sublime Text автоматично підсвічує синтаксис коду для багатьох мов програмування, роблячи його більш зрозумілим та читабельним. Крім того, можна налаштувати кольорову схему відповідно до своїх вподобань.

Програма має вбудовану систему розширень, що дозволяє розширити його функціональність. Існує велика кількість сторонніх плагінів, які додають нові можливості та функції до редактора.

Sublime Text має потужну командну палітру, яка дозволяє виконувати різні операції швидко і зручно. Можна виконувати команди, переходити до файлів, шукати й замінювати текст і багато іншого, не відходячи від клавіатури [29].

2.3.3 Atom

Atom — це відкритий та розширюваний редактор коду, розроблений командою GitHub. Він пропонує ряд цікавих можливостей для розробників. Далі наведені деякі особливості цієї програми.

Atom має потужну систему розширень, яка дозволяє розширити функціональність редактора. Користувач може встановлювати розширення з багатьох доступних пакетів, що додають нові можливості, підтримку мов програмування, теми оформлення та інше.

Редактор коду надає зручні інструменти для керування проектами. Можна легко відкривати та перемикатися між різними файлами та папками, шукати файлів, створювати закладки та виконувати інші операції, що полегшують навігацію по проекту.

Atom має інтеграцію з GitHub, що дозволяє здійснювати операції з GitHub безпосередньо з редактора. Можна виконувати коміти, створювати гілки, відкривати запити на злиття та інші операції без необхідності переходити до веб-інтерфейсу GitHub.

Середовище Atom підсвічує синтаксис коду для багатьох мов програмування, роблячи його більш зрозумілим та легким для читання. Він також має функцію автодоповнення, яка допомагає швидко вводити код, надаючи пропозиції на основі контексту.

Редактор коду дозволяє налаштовувати його вигляд та поведінку відповідно до вподобань користувача. Можна вибрати тему оформлення, налаштувати шрифти, включити або виключити певні функції та змінити інші параметри, щоб редактор відповідав особистим потребам [30].

2.3.4 IntelliJ IDEA

IntelliJ IDEA — це інтегроване середовище розробки (IDE), розроблене компанією JetBrains. Воно спеціалізується на розробці програм на мовах Java, Kotlin, Groovy та інших мовах, що працюють на віртуальній машині Java. Ось деякі особливості даної програми.

IntelliJ IDEA надає потужне автодоповнення коду, яке аналізує контекст і надає пропозиції для методів, змінних, класів та інших елементів вашого проекту. Це полегшує і прискорює процес написання коду.

Редактор має вбудовані інструменти для відлагодження коду, включаючи можливість встановлювати точки зупинки, крокувати по коду, переглядати значення змінних та багато іншого. Він також підтримує відлагодження коду в режимі живого редактора.

IntelliJ IDEA не обмежується лише розробкою на мові Java. Воно підтримує такі мови, як Kotlin, Groovy, Scala, JavaScript, TypeScript, HTML, CSS та інші. Це робить його універсальним інструментом для розробки веб-додатків, мобільних додатків, десктопних додатків та інших типів програм.

Програма надає широкі можливості для рефакторингу коду. Можна легко перейменовувати змінні, методи, класи, витягувати фрагменти коду в окремі методи, виносити спільний код в суперкласи або інтерфейси та багато іншого. Це спрощує підтримку та модифікацію кодової бази.

IntelliJ IDEA має підтримку популярних систем контролю версій, таких як Git, SVN, Mercurial і CVS. Ви можете виконувати операції коміту, гілкування, злиття гілок та інші операції безпосередньо з редактора [31].

Для детальнішого порівняння складемо наступну таблицю (таб. 2.2).

Для розробки проекту було обрано інтегроване середовище IntelliJ IDEA. Перевага надалась йому з наступних причин: високої продуктивності, зручного інтерфейсу та широкої функціональності, такої як автодоповнення коду, перевірка синтаксису, рефакторинг і налагодження коду. Також IntelliJ IDEA має підтримку інших інструментів розробки, інтеграцію з системами

керування версіями і можливості налаштування через плагіни, що робить його зручним та потужним інструментом для розробки проекту на JavaScript.

Таблиця 2.2. — Порівняння функцій чотирьох редакторів коду

	VS Code	Sublime Text	Atom	IntelliJ IDEA
Багатомовність	+	+	+	+
Автодоповнення	+	+	+	+
Відлагодження	+	+	-	+
Універсальність	+	+	+	+
Рефакторинг коду	-	+	-	+
Контроль версій	+	+	+	+
Розширюваність	+	+	+	+
Легкість використання	+	+	+	+
Система інструментів	+	+	+	+
Налаштування та розширення	+	+	+	+

2.4 Висновок до розділу 2

У даному розділі було розглянуто різні аспекти вибору методів і технологій розробки для проекту. Зокрема, було проведено порівняльний аналіз мов програмування та середовищ розробки програмного коду. Зрештою, були зроблені наступні висновки.

JavaScript є однією з найпоширеніших мов програмування, яка надає можливості для створення динамічних та інтерактивних веб-сторінок. Обрання JavaScript для проекту було обумовлено його широкою підтримкою, великою кількістю ресурсів та бібліотек, а також використанням у MERN-стеку. IntelliJ IDEA є потужним інтегрованим середовищем розробки, яке надає розширені можливості для розробки на різних мовах програмування.

Обрання IntelliJ IDEA було обумовлено його продуктивністю, зручним інтерфейсом та широким набором функціональності.

Також було розглянуто MERN-стек, який включає MongoDB, Express, React та Node.js. Цей стек є повноцінним рішенням для розробки веб-додатків, оскільки об'єднує в собі клієнтську та серверну частини, а також базу даних. Обрання MERN-стеку було обумовлено його зручністю використання, швидкістю розробки та ефективним використанням JavaScript та JSON.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Структурна схема розробки

Загальний процес роботи даного ПП включає наступні кроки.

Крок 1. Користувач взаємодіє з клієнтською частиною додатку, яка забезпечує користувачу доступ до інтерфейсу та функціональності системи. Клієнтська частина може бути розроблена з використанням React, Tailwind CSS та інших технологій. Користувач може виконувати дії, такі як створення, перегляд або видалення записів тощо.

Крок 2. Клієнтська частина взаємодіє з серверною частиною за допомогою HTTP-запитів, зазвичай використовуючи методи POST та GET. Запити містять необхідну інформацію, таку як дані для створення або оновлення записів, або запити на отримання певних даних з бази.

Крок 3. Серверна частина додатку, яка побудована з використанням фреймворка Express та Node.js, отримує запити від клієнтської частини. Вона обробляє запити, виконує необхідні операції, такі як збереження даних в базу даних або витягування даних з бази. Для цього сервер взаємодіє з базою даних MongoDB, яка зберігає дані в документоорієнтованому форматі.

Крок 4. База даних MongoDB зберігає дані, які були передані від сервера. MongoDB дозволяє зберігати дані у вигляді JSON-подібних документів, без необхідності визначення схеми таблиць. Це робить її гнучкою та легко масштабовуваною системою управління базами даних.

Далі розглянемо структурну схему роботи даного програмного забезпечення (рис. 3.1).

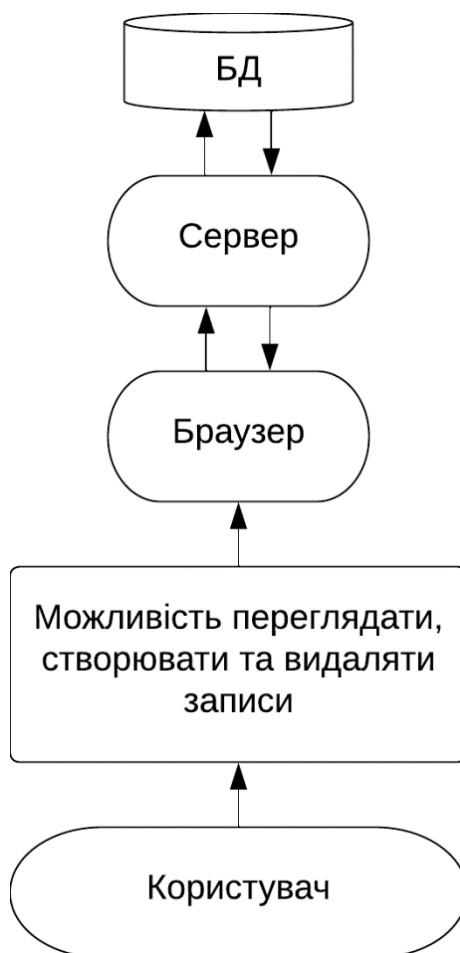


Рисунок 3.1 — Загальна схема роботи ПЗ

3.2 Клієнтська частина

Клієнтська частина додатку відповідає за взаємодію з користувачем та відображення інтерфейсу. Вона використовується користувачем для взаємодії з функціональністю системи, такою як створення, перегляд або редагування записів.

Клієнтська частина даної інформаційної дошки складається з наступних елементів:

- користувацький інтерфейс: реалізований за допомогою React, який дозволяє створювати динамічні веб-сторінки;
- стилізація: використовується фреймворк Tailwind CSS для швидкого та зручного створення стилів;

– бізнес-логіка: заснована на бібліотеці Redux разом з reduxjs/toolkit для ефективного керування станом додатку.

3.2.1 Головна сторінка додатку

Розглянемо файл «Home.jsx». Компонент Home є домашньою сторінкою додатку. Він використовує useDispatch та useSelector для отримання доступу до функції dispatch та стану posts з Redux store відповідно. Компонент також використовує ефект useEffect, який викликає dispatch(fetchPosts()) для отримання нотаток з сервера та збереження їх у Redux store. У розмітці відображається заголовок «Заметки» з іконкою, форма для створення нової нотатки та список окремих нотаток з Redux store (рис. 3.2).

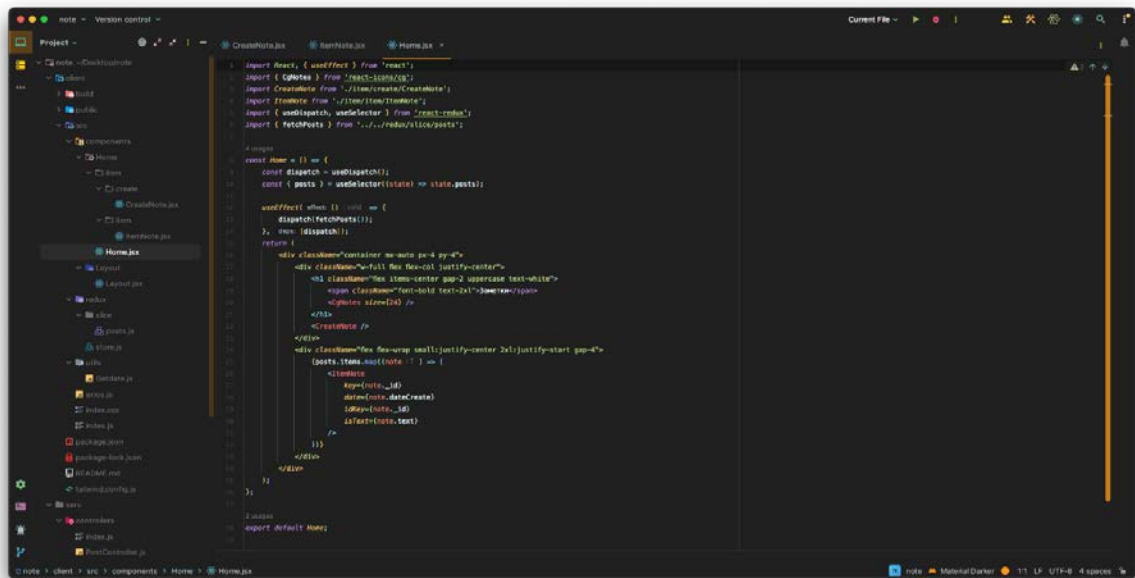


Рисунок 3.2 — Код головної сторінки додатку

3.2.2 Елемент «нотатка»

Файл «ItemNote.jsx» визначає компонент, який представляє окрему нотатку або запис у списку заміток. Компонент містить інформацію про текст нотатки, дату створення та кнопку видалення. В компоненті

використовуються іконки з бібліотеки `react-icons` для відображення кнопки видалення.

Компонент також взаємодіє з `Redux`-стором, використовуючи функцію `useDispatch` для відправки асинхронного запиту на видалення запису. У розмітці компоненту використовуються класи `CSS` для задання стилів, таких як фоновий колір, шрифт, округлені кути тощо. Загалом код відповідає за відображення окремого запису і обробку видалення запису (рис. 3.3).

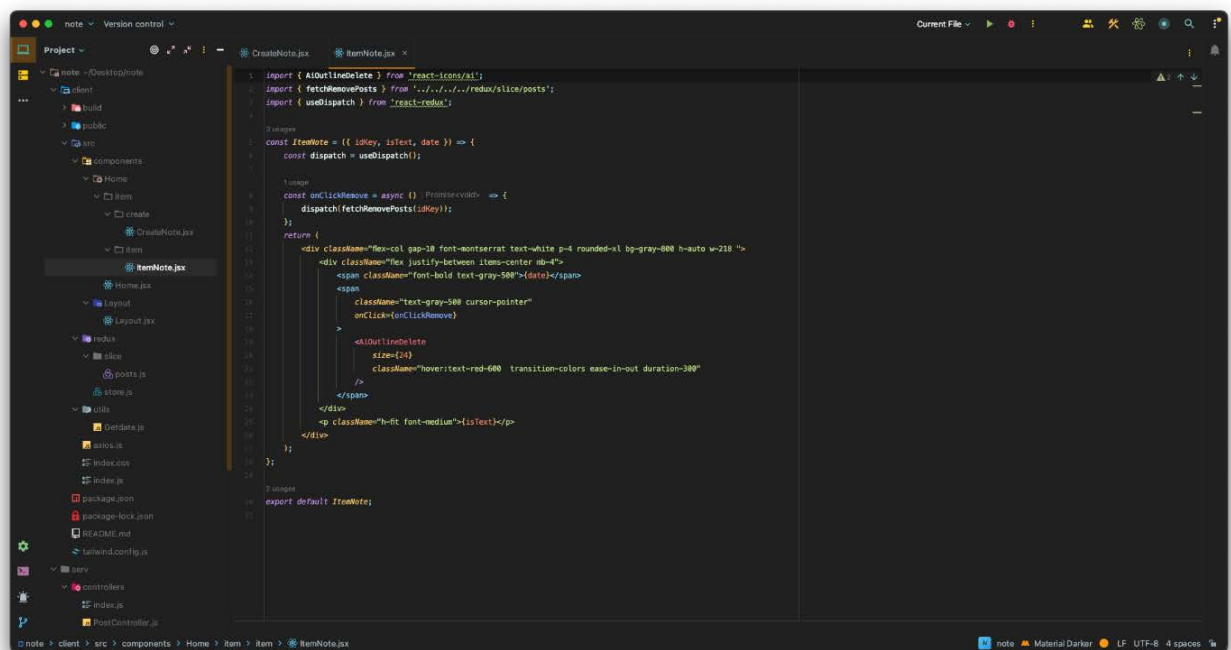


Рисунок 3.3 — Код елемента «нотатка»

3.2.3 Створення «нотатки»

Файл `CreateNote.jsx` визначає компонент, який відповідає за створення нової нотатки або запису. Цей код відповідає за створення нової нотатки та взаємодію з `Redux`-стором і сервером (рис. 3.4).

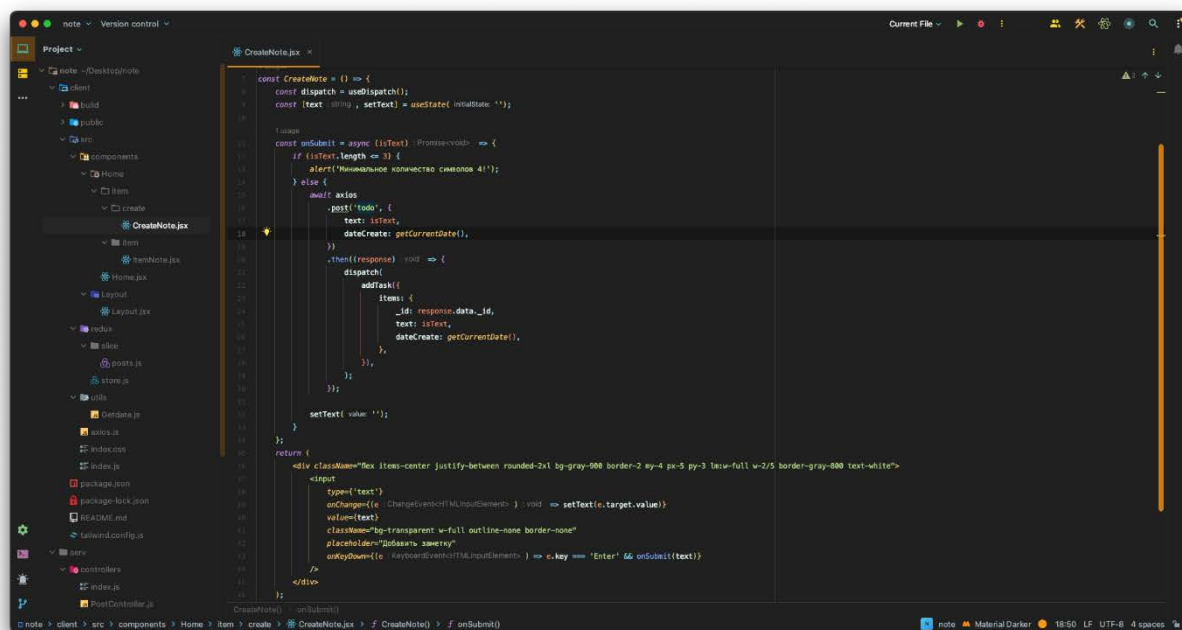


Рисунок 3.4 — Код створення «нотатки»

Компонент містить текстове поле для введення тексту нотатки та кнопку додавання. При введенні тексту і натисканні клавіші Enter або кнопки додавання, відбувається перевірка на мінімальну кількість символів. Якщо введений текст відповідає вимогам, виконується асинхронний запит до сервера за допомогою Axios для створення нового запису.

Після успішного створення запису, відбувається оновлення Redux-стору за допомогою dispatch, додаючи новий запис до списку нотаток. Компонент також використовує хук useState для зберігання та оновлення значення введеного тексту. У розмітці компоненту також використовуються класи CSS для стилізації вигляду (рис. 3.5).

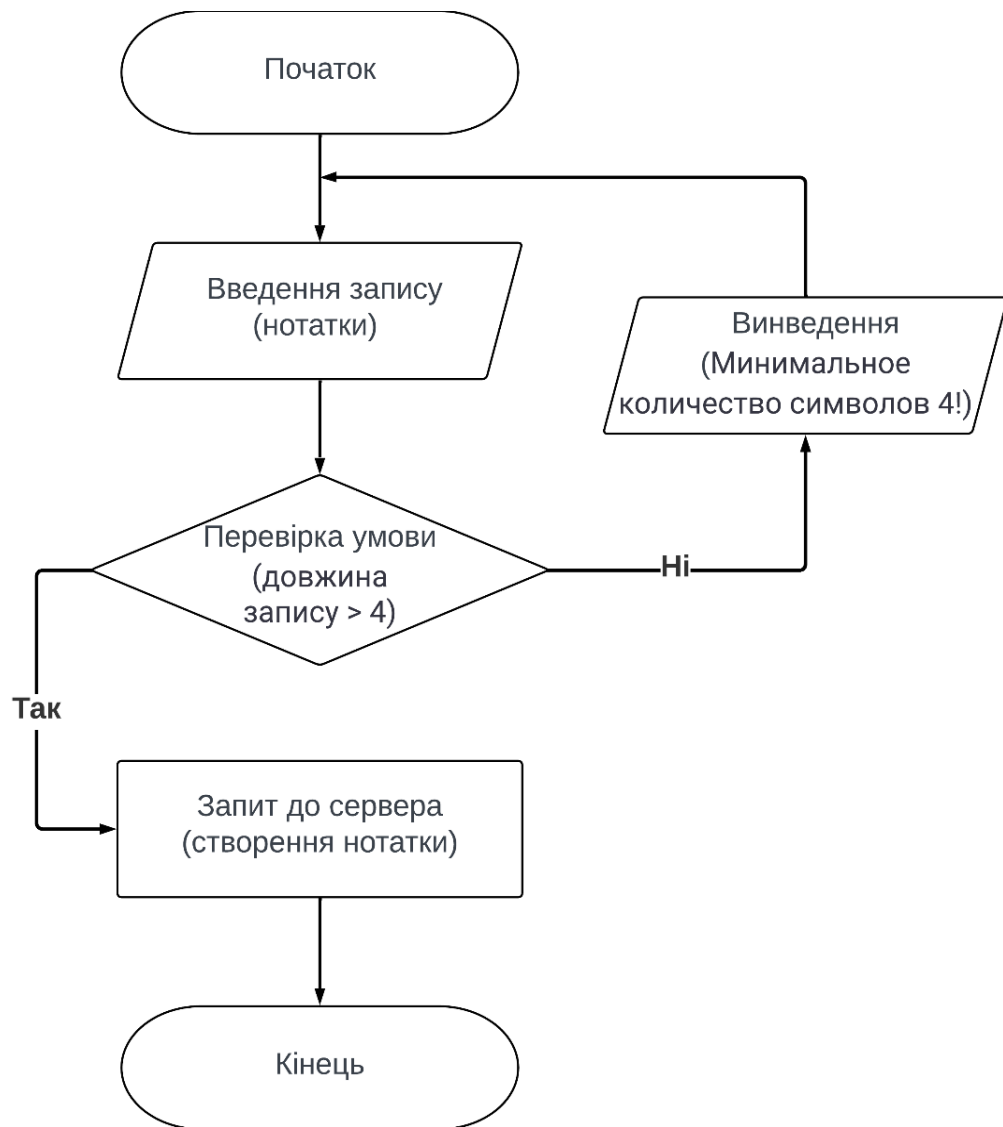


Рисунок 3.5 — Блок-схема алгоритму введення «нотатки»

Поточну дату визначає функція `getCurrentDate`, яка повертає її у форматі «день.місяць.рік». Функція отримує роздільник (параметр `separator`), за замовчуванням це крапка. В середині функції створюється новий об'єкт `Date`, який представляє поточну дату і час. З цього об'єкту витягуються день, місяць і рік. Значення місяця збільшують на 1, оскільки місяці в JavaScript нумеруються з 0 (січень - 0, лютого - 1 і так далі.). Далі формується рядок, в якому день, місяць і рік об'єднуються з допомогою роздільника.

Функція експортується за допомогою `export default`, що дозволяє її використовувати в інших модулях.

3.2.4. Бізнес-логіка

Файл «posts.js» описує бізнес-логіку даного програмного продукту (рис. 3.6). Бізнес-логіка в даному контексті описує функціональність та логіку взаємодії з сервером та обробки даних в додатку.

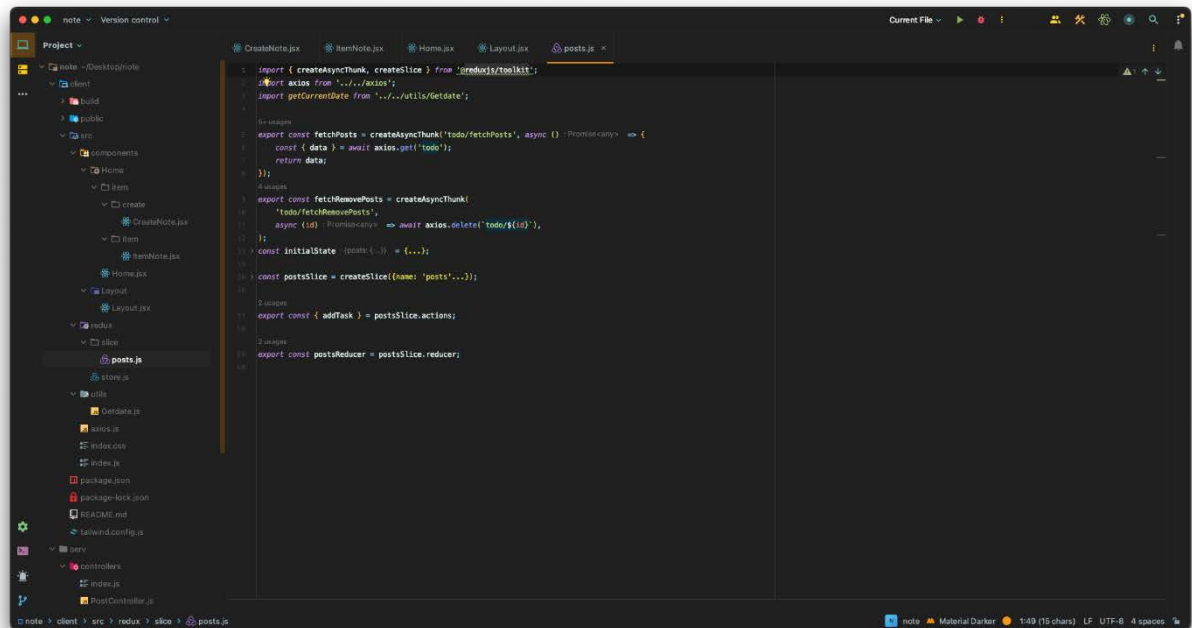


Рисунок 3.6 — Бізнес-логіка інформаційної системи

Далі розглянемо основні її складові і функції.

`fetchPosts` — отримання списку заміток з сервера за допомогою GET-запиту і збереження їх у стані.

Дана функція виконує GET-запит до шляху `'/todo'` за допомогою бібліотеки `axios` і отримує дані з сервера. Отримані дані (список заміток) зберігаються у стані `Redux`. Це робиться за допомогою виклику `createAsyncThunk` з бібліотеки `@reduxjs/toolkit`, який автоматично генерує диспетчери для обробки станів `"pending"`, `"fulfilled"` та `"rejected"`.

`fetchRemovePosts` — видалення замітки з сервера за допомогою DELETE-запиту.

Вона отримує ідентифікатор замітки (id) як параметр і виконує DELETE-запит до шляху '/todo/{id}' за допомогою бібліотеки axios. Це також використовує createAsyncThunk, щоб автоматично генерувати диспетчери для обробки станів "pending", "fulfilled" та "rejected".

addTask — додавання нової замітки до списку заміток у стані.

Функція отримує об'єкт замітки як параметр, який містить поля, такі як ідентифікатор (_id), текст (text) та дата створення (dateCreate). Функція створює новий об'єкт замітки з цими значеннями і додає його до списку заміток у стані Redux. Це виконується через редуктор, який визначений у createSlice, а саме в reducers частині.

Ці дії виконуються за допомогою Redux та допоміжних функцій з бібліотеки @reduxjs/toolkit. Всі зміни стану зберігаються у відповідних полях items і status об'єкта posts у Redux store.

3.3 Серверна частина

Серверна частина даної інформаційної системи побудована на стеку MERN (MongoDB, Express, React, Node.js). MongoDB використовується в якості бази даних, яка працює на віртуальному приватному сервері (VPS). MongoDB — це система керування базами даних NoSQL, яка використовує документоорієнтований підхід і не потребує опису схеми таблиць, використовуючи JSON-подібні документи і схему бази даних.

Для перегляду бази даних можна використовувати MongoDB Compass. Для підключення використовується URI `mongodb://45.94.158.55:27017/dashboardProject`. Express використовується як фреймворк для побудови серверної частини на Node.js. У цьому випадку він використовується для створення серверного API.

Клієнтська частина додатку розроблена за допомогою React, відкритої JavaScript-бібліотеки для розробки користувацького інтерфейсу. Для

стилізації використовується фреймворк Tailwind CSS, який надає широкий набір готових компонентів та стилів.

Для взаємодії з сервером використовуються методи POST та GET за допомогою протоколу HTTP.

3.3.1 Контролер записів

Файл «PostController.js» містить код, що описує функції для взаємодії з базою даних за допомогою моделі PostModel (рис. 3.7).

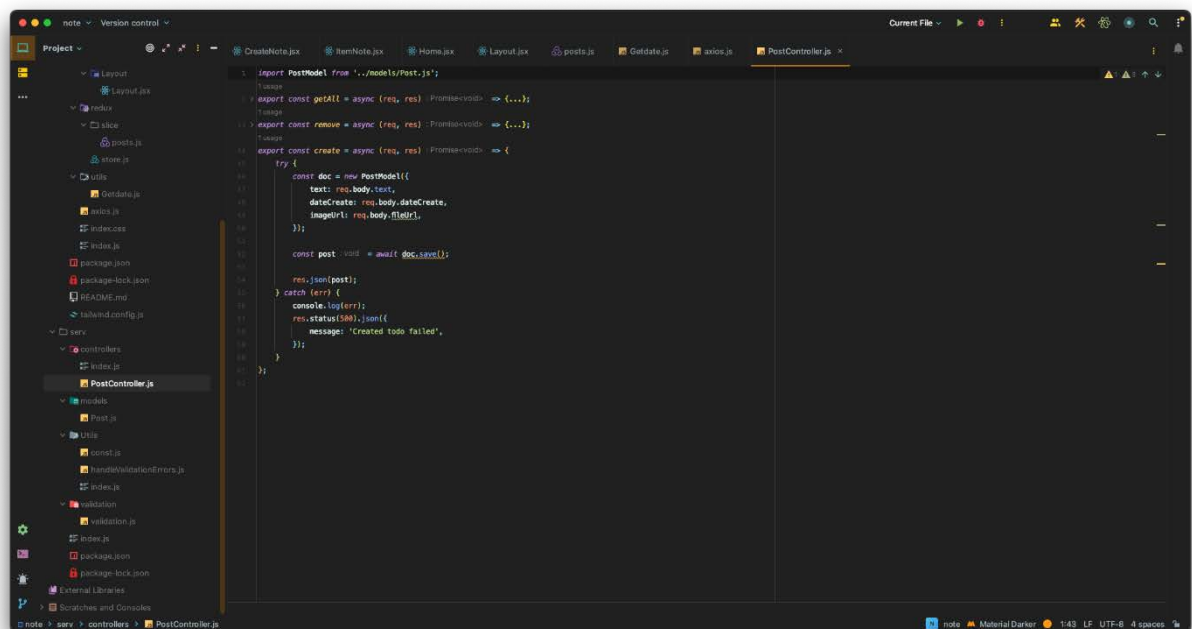


Рисунок 3.7 — Контролер записів

Функція `getAll` отримує всі записи з бази даних, використовуючи метод `find()` моделі `PostModel`. Вона повертає знайдені записи у форматі JSON у відповідь серверу.

Функція `remove` видаляє запис з бази даних за його ідентифікатором, який отримує з параметрів запиту (`req.params.id`). Вона використовує метод `findOneAndDelete()` моделі `PostModel`, щоб знайти і видалити запис за вказаним ідентифікатором. Якщо виникає помилка під час виконання операції

або якщо запис не знайдений, сервер повертає відповідну помилку у форматі JSON.

Create створює новий запис в базі даних на основі отриманих даних з тіла запиту (`req.body`). Вона створює новий об'єкт моделі `PostModel` з цими даними і зберігає його в базі даних за допомогою методу `save()`. Після збереження, створений запис повертається у форматі JSON у відповідь серверу. Якщо виникає помилка під час створення запису, сервер повертає відповідну помилку у форматі JSON.

3.3.2 API сервер

Основний файл API сервера «`index.js`» містить код, що створює сервер, який здатний обробляти HTTP запити для отримання, створення та видалення постів. Він також забезпечує з'єднання з базою даних MongoDB та налаштовує HTTPS сервер для безпечної комунікації (рис. 3.8).

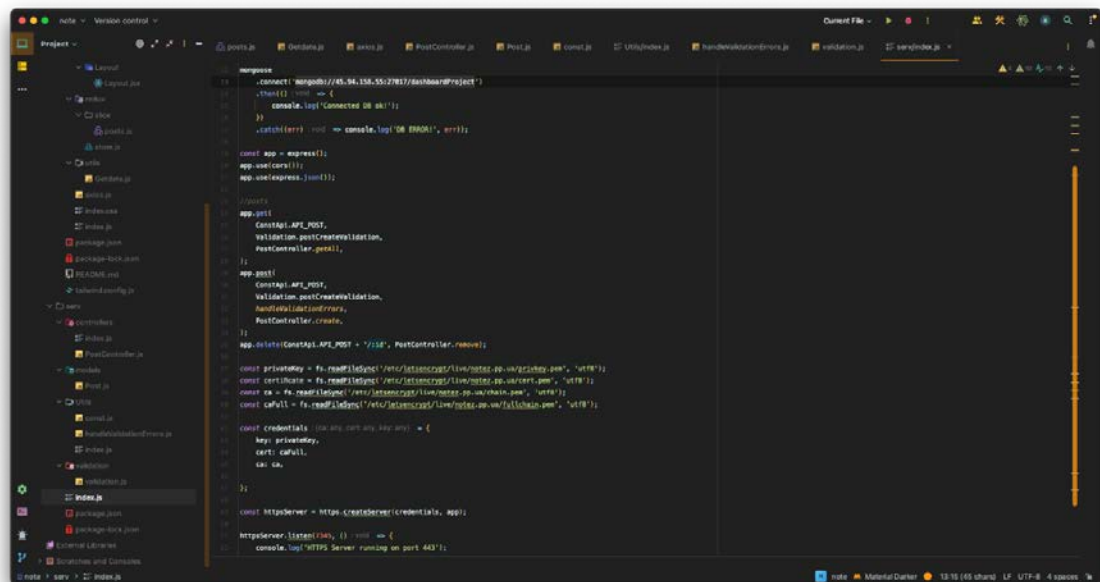


Рисунок 3.8 — Файл API сервера

Далі описані основні дії, які виконуються.

Імпортується необхідні модулі та бібліотеки, такі як `express`, `mongoose`, `cors`, `https`, `fs`.

Встановлюється з'єднання з базою даних MongoDB за допомогою модуля `mongoose`. URL підключення до бази даних передається в метод `connect()`. Якщо підключення успішне, виводиться повідомлення "Connected DB ok!". У разі помилки підключення, виводиться повідомлення про помилку.

Створюється новий екземпляр додатку Express за допомогою `express()`. Застосовується `middleware cors`, який дозволяє перетинати межі доменів для взаємодії з сервером. Застосовується `middleware express.json()`, який розпізнає та парсить вхідні дані у форматі JSON.

Налаштовуються маршрути для роботи з пости. Використовується HTTP метод GET для отримання всіх постів за допомогою `PostController.getAll`. Використовується HTTP метод POST для створення нового поста за допомогою `PostController.create`. Використовується HTTP метод DELETE для видалення поста за допомогою `PostController.remove`.

Налаштовується HTTPS сервер за допомогою сертифікатів, які завантажуються з файлів. Використовується порт 7345 для прослуховування HTTPS запитів. При успішному запуску серверу виводиться повідомлення "HTTPS Server running on port 443".

3.3.3 Взаємодія з базою даних

Цей код описує модель поста для бази даних MongoDB за допомогою бібліотеки `mongoose` (рис. 3.9). Алгоритм дозволяє визначити структуру та валідацію даних для моделі поста, що зберігатиметься в базі даних MongoDB.

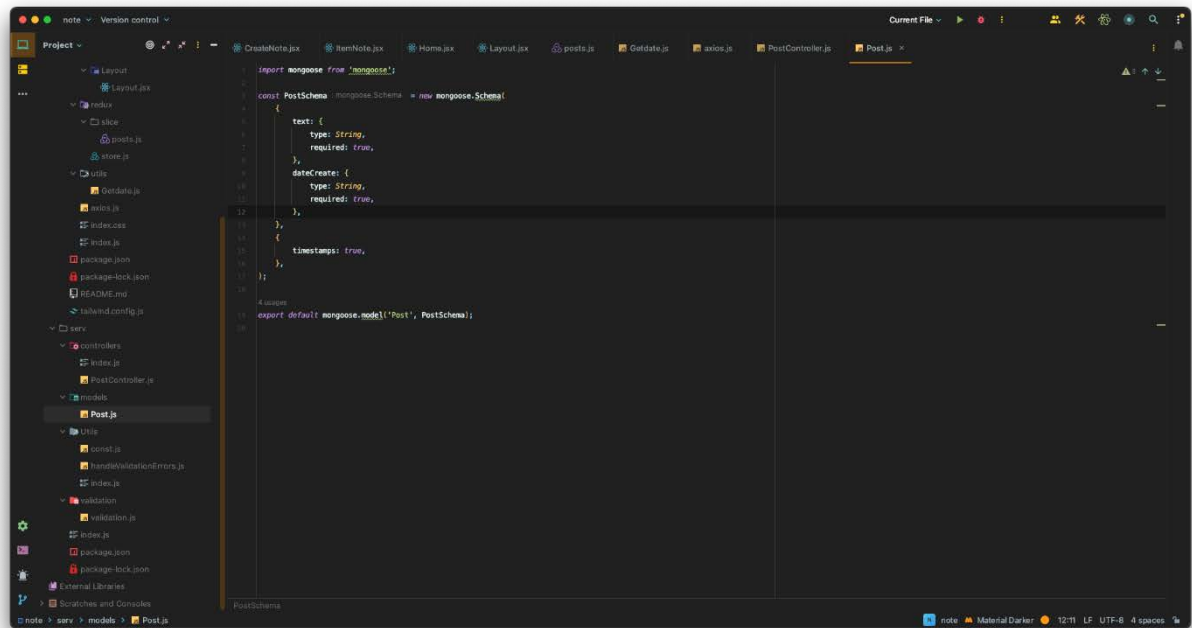


Рисунок 3.9 — Запис нотаток в базу даних

Далі описані основні дії, які виконує даний код.

Імпортується модуль `mongoose`, який дозволяє працювати з MongoDB.

Створюється схема `PostSchema` для моделі поста. Схема містить два поля:

- `text`: тип даних - рядок (`String`), обов'язкове поле (`required: true`), яке відповідає за текст поста;
- `dateCreate`: тип даних - рядок (`String`), обов'язкове поле (`required: true`), яке відповідає за дату створення поста.

До схеми додається параметр `timestamps: true`, який автоматично додає два поля до документа: `createdAt` (дата і час створення) та `updatedAt` (дата і час останнього оновлення). Це дозволяє відстежувати час створення і оновлення постів.

Експортується модель `Post`, створена за допомогою `mongoose.model()`. Модель отримує назву "Post" і використовує створену раніше схему `PostSchema`.

3.4 Висновки за розділом 3

У даному розділі була представлена структурна схема розробки програмного продукту. Використовувався стек технологій MERN (MongoDB, Express, React, Node.js) для розробки клієнтської та серверної частини. Було застосовано фреймворки і бібліотеки для зручної та швидкої розробки програми.

У клієнтській частині програми була використана JavaScript-бібліотека React для розробки інтерфейсу користувача. Для стилізації інтерфейсу був використаний фреймворк Tailwind CSS. Бізнес-логіка програми базується на використанні бібліотеки Redux та reduxjs/toolkit для управління станом даних.

У серверній частині програми використовувався фреймворк Express для розробки API. Для зберігання даних використовувалась база даних MongoDB, яка забезпечує документоорієнтовану систему управління базами даних. Зв'язок з базою даних та обробка запитів від клієнта здійснюється за допомогою HTTP-методів POST і GET.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення й умови використання програми

Розроблена інформаційна дошка використовується для зручного зберігання та організації коротких записів, нагадувань, списків справ і важливих відомостей. Основна мета програми нотаток полягає у наданні користувачеві простого і ефективного інструменту для створення, редагування, зберігання та сортування нотаток.

Деякі основні мотиви для використання цієї програми приводяться нижче.

Запис ідей і творчих думок: Користувачі можуть швидко зафіксувати свої ідеї, натхнення і творчі думки в нотатках, щоб не втратити їх і мати можливість повернутися до них пізніше.

Організація завдань і списків справ. ПП дозволяє створювати списки завдань, контрольні пункти, нагадування і терміни виконання, допомагаючи вам керувати своїми завданнями та планувати ділові справи.

Зберігання важливих відомостей. Можна використовувати ПЗ для зберігання інформації, як-от адреси, номери телефонів, паролі, списки покупок, рецепти та інші корисні відомості, до яких вам зручно мати швидкий доступ.

Організація досліджень та навчання. ІД дозволяє структурувати і зберігати знання, записи, посилання та інші важливі деталі, щоб користувачі могли легко організувати свою роботу та повертатися до матеріалів в майбутньому.

Умови використання інформаційної дошки для нотаток можуть включати деякі аспекти.

Веб-браузер. Для використання інформаційної дошки для нотаток користувачу потрібен сучасний веб-браузер, такий як Google Chrome, Mozilla Firefox, Safari або Microsoft Edge. Важливо мати оновлену версію браузера, щоб забезпечити сумісність та оптимальну роботу з програмою.

Інтернет-з'єднання. Для доступу до інформаційної дошки для нотаток потрібне активне підключення до Інтернету. Без стабільного та швидкого інтернет-з'єднання користувач не зможе взаємодіяти з сервером і отримувати оновлення даних.

Захист даних. Варто переконатись, що користуєтеся надійним і безпечним з'єднанням для доступу до ІД. Забезпечуйте захист свого з'єднання та уникайте передачі конфіденційної інформації через незахищені мережі або ненадійні пристрої.

4.2 Опис інтерфейсу та функціоналу програмного продукту

Основні компоненти програми включають інтерфейс користувача, функціональність створення, редагування та видалення нотаток.

Інтерфейс розробленої ПП для створення нотаток має зручну та інтуїтивно зрозумілу структуру, яка дозволяє користувачам легко знаходити потрібні нотатки (рис. 4.1).

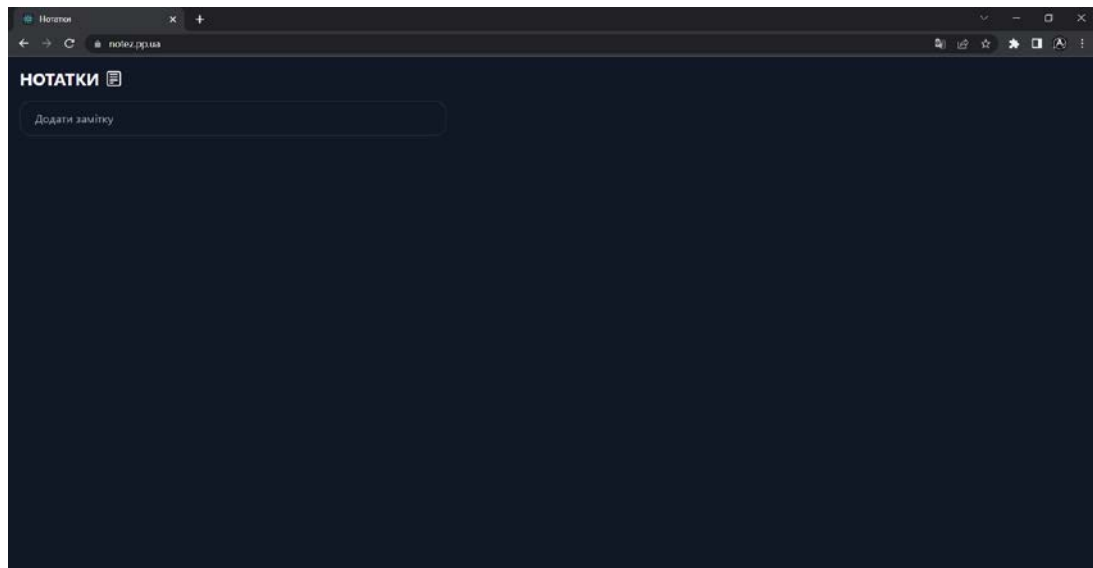


Рисунок 4.1 — Інтерфейс програми (браузер, ноутбук)

Інтерфейс програми є адаптивним, що означає, що він автоматично пристосовується до різних розмірів екранів і пристроїв, забезпечуючи

оптимальне відображення та взаємодію з користувачем незалежно від того, чи використовується комп'ютер, планшет або смартфон (рис. 4.2).

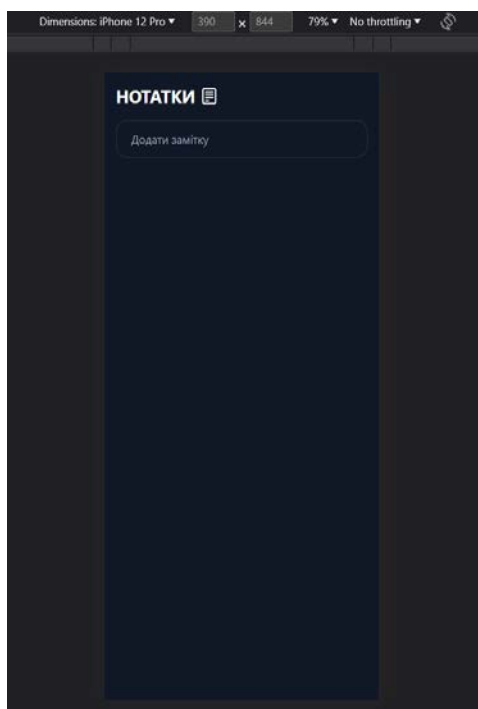


Рисунок 4.2 — Інтерфейс програми (смартфон)

Користувач може створювати нові нотатки, вводячи текстовий контент та іншу інформацію. Потім відображаються всі створені замітки (рис. 4.3 та рис. 4.4).

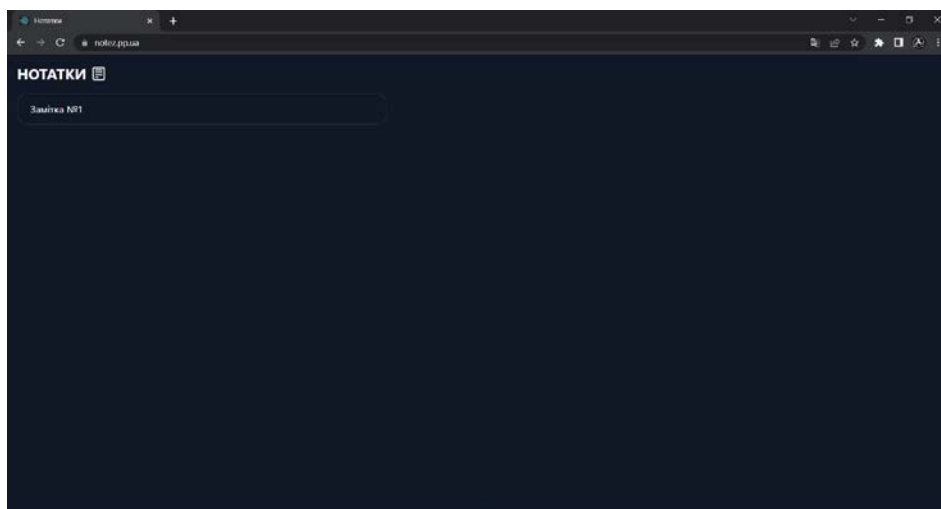


Рисунок 4.3 — Створення заміток

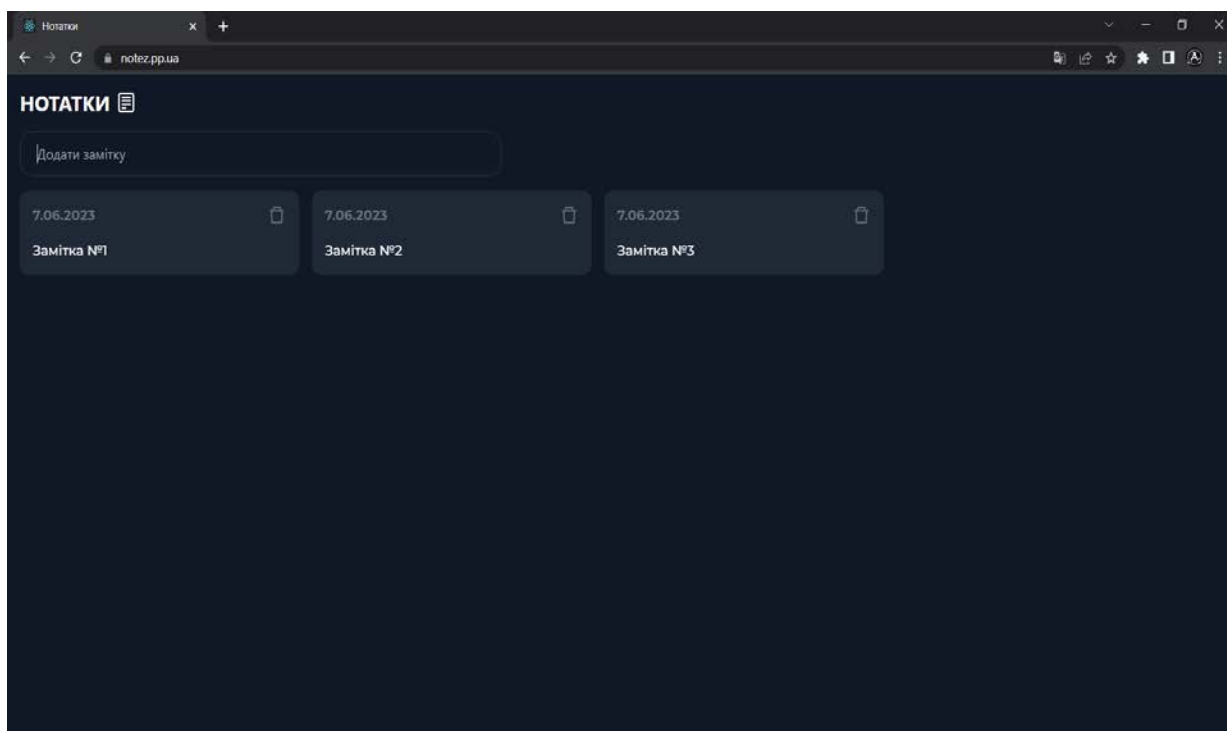


Рисунок 4.4 — Відображення створених заміток

Також є функція видалення створених нотаток (рис. 4.5 та рис. 4.6).

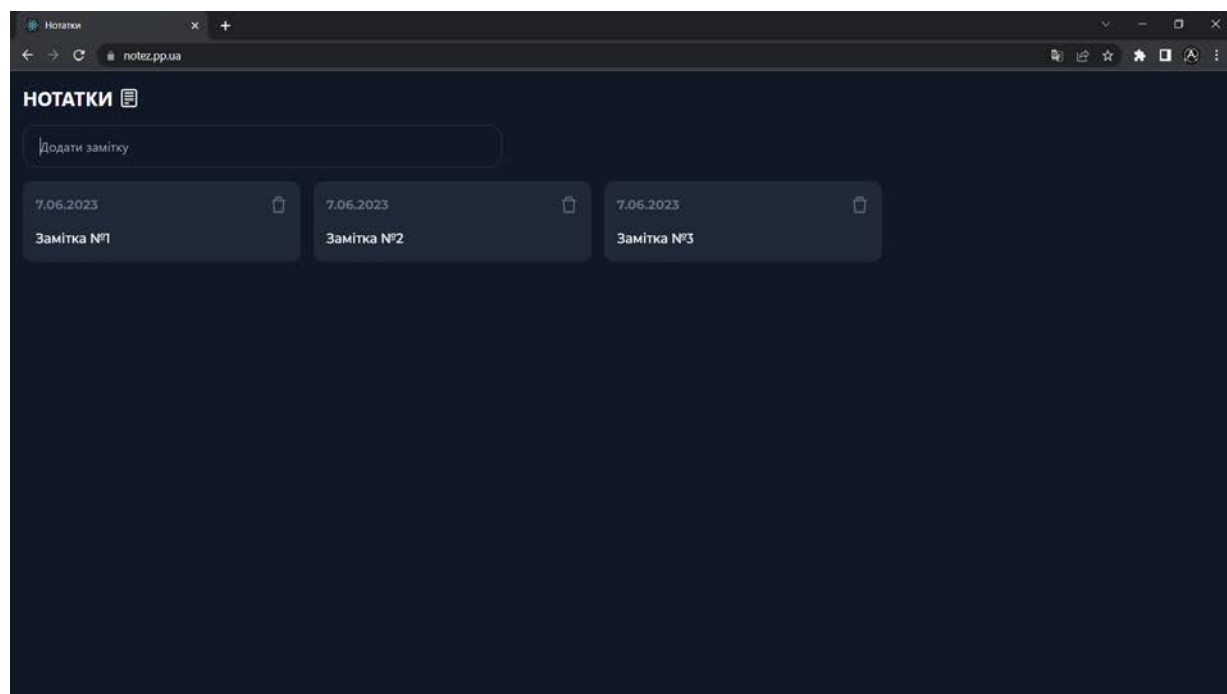


Рисунок 4.5 — Процес видалення замітки №2

При наведенні на кнопку видалення (у вигляді іконки) вона змінює колір на червоний.

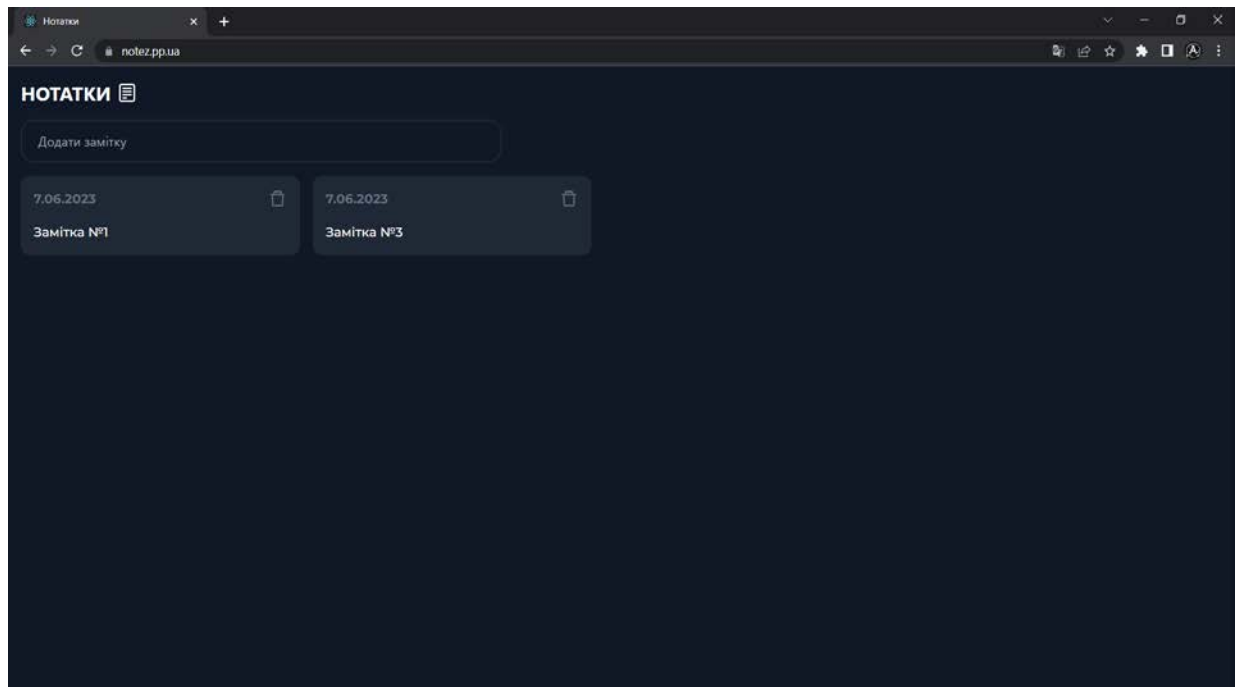


Рисунок 4.6 — Новий перелік заміток (без видаленої замітки №2)

Безпека даних в програмі забезпечується за допомогою різних заходів. Використовується база даних MongoDB, яка пропонує документоорієнтовану модель зберігання даних. Крім того, для забезпечення безпеки використовуються HTTPS-з'єднання для комунікації з сервером, а також встановлені заходи безпеки, такі як автентифікація та авторизація, для захисту від несанкціонованого доступу до даних.

4.3 Тестування програми

Тестування програми проводиться з метою перевірки її функціональності, якості і надійності перед її випуском. Це допомагає виявити та виправити помилки, а також забезпечити відповідність програми поставленим вимогам і очікуванням. Тестування дозволяє переконатися, що ПП працює належним чином, і допомагає забезпечити її якість і надійність.

Тестування розробленої інформаційної дошки для нотаток включає наступні кроки.

Функціональне тестування. Варто переконатись, що всі основні функції програми працюють належним чином. Наприклад, створення нових нотаток, перегляд і видалення існуючих нотаток тощо.

Тестування інтерфейсу користувача. Варто перевірити, що інтерфейс програми нотаток є зрозумілим і зручним для користувача. Потрібно перевірити навігацію, розміщення елементів, реакцію на дії користувача, наявність необхідних кнопок і функцій.

Тестування збереження і відновлення даних. Варто переконатись, що нотатки зберігаються правильно і можуть бути відновлені після закриття і перезапуску програми. Треба перевірити, що зміни в нотатках зберігаються автоматично.

Тестування безпеки. Необхідно перевірити, що нотатки і дані захищені від несанкціонованого доступу. Варто протестувати як програма нотаток обробляє та зберігає конфіденційну інформацію, чи використовується шифрування, чи є налаштування доступу або використовуються інші механізми безпеки.

У відповідності з технічним завданням був створений набір функцій, для кожної з яких було в кінці виконано виклик і перевірку отриманого результату, реалізуючи ці дії щонайменше раз на кожну функцію. При перевірці співставлявся очікуваний результат з отриманим. Усі результати в кінцевому варіанті співпали. Тому тестування в підсумку можна вважати успішним, а значить і реалізація ПП в підсумку виявилась успішною.

За результатами була розроблена таблиця (чеклист) тестування програми (табл. 4.1). ОК — результат задовільний (проблеми не виявлено). НЕЗ — результат не задовільний (виявлено проблеми).

Таблиця 4.1 — Чеклист тестування програми

Тест	Опис тесту	Результат
Функціональність		
Створення	Перевірка можливості створення нової нотатки	ОК
Видалення	Перевірка можливості видалення нотатки	ОК
Інтерфейс		
7. Навігація	Перевірка роботи кнопок, посилань і меню	ОК
8. Відображення	Перевірка коректного відображення нотаток	ОК
9. Реакція	Перевірка реакції програми на дії користувача	ОК
Збереження даних		
Автозбереження	Перевірка автоматичного збереження нотаток	ОК
Відновлення	Перевірка відновлення нотаток після перезапуску	ОК
Безпека		
Захист доступу	Перевірка наявності захисту доступу до нотаток	ОК
Обробка даних	Перевірка безпечної обробки та зберігання конфіденційної інформації	ОК

4.4 Висновки за розділом 4

У розділі було розглянуто призначення програми і визначені умови її використання. Програма призначена для створення нотаток (записів) і надає можливість користувачам зберігати і керувати своїми текстовими даними. Вона доступна через браузер з використанням Інтернет-з'єднання.

Було надано опис інтерфейсу та функціоналу програмного продукту. Інтерфейс програми є адаптивним, що дозволяє користувачам зручно використовувати програму на різних пристроях і розмірах екранів. Функціонал програми включає можливість створення, перегляду і видалення нотаток, а також збереження даних і взаємодію з серверною частиною для отримання і збереження даних.

Також було проведено тестування програми з метою перевірки його функціональності, надійності і відповідності вимогам. Були складені тестові сценарії і проведено різні види тестування. Результати тестування дозволили виявити і виправити помилки та недоліки програми, забезпечуючи його якісну роботу та відповідність поставленим вимогам.

ВИСНОВКИ

У ході даної дипломної кваліфікаційної роботи було розроблено програмний продукт, який надає користувачам зручний інструмент для зберігання, організації та керування нотатками. Проект був розроблений з використанням методів та технологій, таких як IntelliJ IDEA, JavaScript, React, а також бази даних MongoDB.

У першому розділі було проведено аналіз предметної області, визначено поняття веброзробки, етапи створення вебпродуктів та проведено огляд існуючих інформаційних дошок для нотаток. Також були поставлені завдання та вимоги до проєкту.

У другому розділі було вибрано методи та технології розробки, включаючи мови програмування, фреймворки та базу даних. Було обрано JavaScript як мову програмування, React як JavaScript-бібліотеку для розробки інтерфейсу користувача, а також базу даних MongoDB для зберігання даних.

У третьому розділі було проведено розробку програмного продукту, включаючи структурну схему розробки, реалізацію клієнтської та серверної частини програми. Клієнтська частина була реалізована з використанням React, а серверна частина — з використанням Node.js та Express. Застосовувались принципи розробки MERN (MongoDB, Express, React, Node.js) стеку.

У четвертому розділі була описана експлуатація та тестування програми. Було визначено призначення й умови використання програми, описаний інтерфейс та функціонал програмного продукту. Також було проведено тестування програми з метою перевірки його функціональності і відповідності вимогам.

В результаті дипломного проєкту був розроблений програмний засіб для створення інформаційної дошки, який має зручний та зрозумілий інтерфейс та функціонал. Проект відповідає поставленим завданням і вимогам, а його реалізація включає в себе використання сучасних технологій

розробки програмного забезпечення. Розроблений програмний засіб має великий потенціал для полегшення повсякденних завдань користувачів і покращення їх продуктивності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Веброзробка [Електрон. ресурс]. — Режим доступу: <https://uk.wikipedia.org/wiki/Веброзробка> (дата звернення: 31.05.2023).
2. IT-словник української мови [Електрон. ресурс]. — Режим доступу: http://xn--r1a3b.xn--b1amgblet.xn--j1amh/index.php/Словник_з_інформатики (дата звернення: 31.05.2023).
3. Front-end розробка [Електрон. ресурс]. — Режим доступу: <https://wezom.com.ua/ua/blog/chto-takoe-front-end-razrabotka> (дата звернення: 31.05.2023).
4. Що таке backend-розробка і чим вона відрізняється від frontend [Електрон. ресурс]. — Режим доступу: <https://www.buh24.com.ua/shho-take-backend-rozrobka-i-chym-vona-vidriznyayetsya-vid-frontend/> (дата звернення: 31.05.2023).
5. Веб-дизайнер - розповідаємо про професію та як їм стати [Електрон. ресурс]. — Режим доступу: https://cloud.itstep.org/blog_3/web-designer-we-talk-about-the-profession-and-how-to-become-one (дата звернення: 31.05.2023).
6. Етапи створення веб сайтів [Електрон. ресурс]. — Режим доступу: <https://webtune.com.ua/statti/web-rozrobka/etapy-stvorenniya-veb-sajtiv/> (дата звернення: 31.05.2023).
7. Оптимізація [Електрон. ресурс]. — Режим доступу: <http://xn--r1a3b.xn--b1amgblet.xn--j1amh/index.php/Оптимізація> (дата звернення: 31.05.2023).
8. Методологія БЕМ в дії [Електрон. ресурс]. — Режим доступу: <https://avivi.pro/ua/blog/metodologiya-bem-v-deystvii/> (дата звернення: 31.05.2023).
9. Webpack [Електрон. ресурс]. — Режим доступу: <https://uk.wikipedia.org/wiki/Webpack> (дата звернення: 31.05.2023).

10. Parcel - початок роботи [Електрон. ресурс]. — Режим доступу: https://uk.parceljs.org/getting_started.html (дата звернення: 31.05.2023).
11. Огляд фреймворків JavaScript. Що, для чого і коли використовувати [Електрон. ресурс]. — Режим доступу: <https://dou.ua/forums/topic/34739/> (дата звернення: 31.05.2023).
12. Що таке GIT та як почати працювати з ним? [Електрон. ресурс]. — Режим доступу: <https://blog.kr-labs.com.ua/що-таке-git-та-як-почати-працювати-з-ним-90c7f460f564> (дата звернення: 31.05.2023).
13. Web Components [Electronic resource]. — Access mode: https://developer.mozilla.org/en-US/docs/Web/Web_Components (дата звернення: 31.05.2023).
14. Фреймворки [Електрон. ресурс]. — Режим доступу: <https://vyspiansky.gitbook.io/introduction-to-web-development/getting-started/frameworks> (дата звернення: 31.05.2023).
15. Блокнот онлайн [Електрон. ресурс]. — Режим доступу: <https://lexx.me/p314-notepad/> (дата звернення: 31.05.2023).
16. Notepad [Electronic resource]. — Access mode: <https://www.rapidtables.com/tools/notepad.html> (дата звернення: 31.05.2023).
17. Online Notepad [Electronic resource]. — Access mode: <https://onlinenotepad.org/notepad> (дата звернення: 31.05.2023).
18. aNotepad – free online notepad [Electronic resource]. — Access mode: <https://anotepad.com/> (дата звернення: 31.05.2023).
19. Сучасний підручник з JavaScript [Електрон. ресурс]. — Режим доступу: <https://uk.javascript.info/> (дата звернення: 02.06.2023).
20. PHP Tutorial [Electronic resource]. — Access mode: <https://www.w3schools.com/php/> (дата звернення: 02.06.2023).
21. Що таке мова програмування python? [Електрон. ресурс]. — Режим доступу: <https://freehost.com.ua/ukr/faq/wiki/chto-takoe-jazik-programmirovanija-python/> (дата звернення: 02.06.2023).

22. Фреймворки для веб-розробки [Електрон. ресурс]. — Режим доступу: <https://voll.com.ua/uk/blog/frejmworki-dlya-veb-rozrobki> (дата звернення: 02.06.2023).
23. MERN Stack Explained [Electronic resource]. — Access mode: <https://www.mongodb.com/mern-stack> (дата звернення: 02.06.2023).
24. Что такое стек MERN, и как с ним работать? [Электрон. ресурс]. — Режим доступа: <https://habr.com/ru/companies/piter/articles/458096/> (дата звернення: 02.06.2023).
25. MongoDB [Electronic resource]. — Access mode: <https://www.mongodb.com/> (дата звернення: 02.06.2023).
26. Beautiful UI components, crafted with Tailwind CSS [Electronic resource]. — Access mode: <https://tailwindui.com/components?ref=sidebar> (дата звернення: 02.06.2023).
27. Редактори коду [Електрон. ресурс]. — Режим доступу: <https://vyspiansky.gitbook.io/introduction-to-web-development/getting-started/code-editors> (дата звернення: 02.06.2023).
28. Visual Studio Code. Code Editing. Redefined [Electronic resource]. — Access mode: <https://code.visualstudio.com/> (дата звернення: 02.06.2023).
29. Sublime Text [Electronic resource]. — Access mode: <https://www.sublimetext.com/> (дата звернення: 02.06.2023).
30. Atom [Electronic resource]. — Access mode: <https://atom.ru.uptodown.com/windows> (дата звернення: 02.06.2023).
31. IntelliJ IDEA – the Leading Java and Kotlin IDE [Electronic resource]. — Access mode: <https://www.jetbrains.com/idea/> (дата звернення: 02.06.2023).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення інформаційної дошки для створення нотаток забезпечить зручний спосіб зберігання записів, списків завдань, нагадувань та інших важливих даних. Розробка даного програмного продукту має великий потенціал для полегшення повсякденних завдань користувачів і покращення їх продуктивності.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Розробка вебзастосунку інформаційної дошки», затверджене наказом Національного університету «Запорізька політехніка» № 87 від 04 квітня 2023 р.

A.2 Призначення роботи

Програмний продукт створення нотаток призначений для фіксування ідей, організації завдань, зберігання важливих відомостей та організації досліджень і навчання.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмний продукт повинен надавати наступні можливості:

- можливість створювати записи (нотатки);
- можливість переглядати записи;
- можливість видаляти нотатки;
- автоматичне збереження записів.

A.3.2 Вимоги до інтерфейсу

Інтерфейс має бути інтуїтивно зрозумілий і легкий у використанні. Необхідна зручна навігація та швидкий доступ до функцій. Також інтерфейс повинен мати адаптивний дизайн, що підлаштовується під різні розміри екранів.

A.3.3 Вимоги до надійності.

Вимоги до надійності програми включають стабільну та безперебійну роботу, що означає, що програма повинна функціонувати без сильних збоїв або непередбачених витоків пам'яті. Крім того, важливим аспектом є захист від втрати даних та несанкціонованого доступу.

A.3.4 Умови експлуатації.

Для використання інформаційної дошки для нотаток потрібен сучасний веббраузер, такий як Google Chrome, Mozilla Firefox, Safari або Microsoft Edge. Для доступу до програмного забезпечення потрібне активне підключення до Інтернету. Без стабільного та швидкого інтернет-з'єднання користувач не зможе взаємодіяти з сервером і отримувати оновлення даних.

A.4 Вхідні дані

Вхідні дані до роботи програми включають текстові дані, які вводяться користувачем. Це може бути текст нотаток, завдання, список справ або будь-які інші відомості, які користувач бажає зберегти.

ДОДАТОК Б
Текст програми

Б.1 Home.jsx

```

import React, { useEffect } from 'react';
import { CgNotes } from 'react-icons/cg';
import CreateNote from './item/create/CreateNote';
import ItemNote from './item/item/ItemNote';
import { useDispatch, useSelector } from 'react-redux';
import { fetchPosts } from '../redux/slice/posts';

const Home = () => {
  const dispatch = useDispatch();
  const { posts } = useSelector((state) => state.posts);

  useEffect(() => {
    dispatch(fetchPosts());
  }, [dispatch]);

  return (
    <div className=»container mx-auto px-4 py-4»>
      <div className=»w-full flex flex-col justify-center»>
        <h1 className=»flex items-center gap-2 uppercase text-white»>
          <span className=»font-bold text-2xl»>Заметки</span>
          <CgNotes size={24} />
        </h1>
        <CreateNote />
      </div>
      <div className=»flex flex-wrap small:justify-center 2xl:justify-start gap-4»>
        {posts.items.map((note) => (
          <ItemNote
            key={note._id}
            date={note.dateCreate}

```

```

        idKey={note._id}
        isText={note.text}
      />
    )))
  </div>
</div>
);
};
export default Home;

```

B.2 ItemNote.jsx

```

import { AiOutlineDelete } from 'react-icons/ai';
import { fetchRemovePosts } from '../redux/slice/posts';
import { useDispatch } from 'react-redux';

const ItemNote = ({ idKey, isText, date }) => {
  const dispatch = useDispatch();
  const onClickRemove = async () => {
    dispatch(fetchRemovePosts(idKey));
  };
  return (
    <div className="flex-col gap-10 font-montserrat text-white p-4 rounded-xl
bg-gray-800 h-auto w-218 ">
      <div className="flex justify-between items-center mb-4">
        <span className="font-bold text-gray-500">{date}</span>
        <span
          className="text-gray-500 cursor-pointer"
          onClick={onClickRemove}
        >
          <AiOutlineDelete

```

```

        size={24}
        className="hover:text-red-600 transition-colors ease-in-out
duration-300"
      />
    </span>
  </div>
  <p className="h-fit font-medium">{isText}</p>
</div>
);
};
export default ItemNote;

```

Б.3 CreateNote.jsx

```

import { useState } from 'react';
import { useDispatch } from 'react-redux';
import axios from '../../axios';
import { addTask } from '../../redux/slice/posts';
import getCurrentDate from '../../utils/Getdate';

const CreateNote = () => {
  const dispatch = useDispatch();
  const [text, setText] = useState("");

  const onSubmit = async (isText) => {
    if (isText.length <= 3) {
      alert('Минимальное количество символов 4!');
    } else {
      await axios
        .post('todo', {
          text: isText,

```

```

        dateCreate: getCurrentDate(),
    })
    .then((response) => {
        dispatch(
            addTask({
                items: {
                    _id: response.data._id,
                    text: isText,
                    dateCreate: getCurrentDate(),
                },
            }),
        );
    });
    setText("");
}
};
return (
    <div className="flex items-center justify-between rounded-2xl bg-gray-900
border-2 my-4 px-5 py-3 lm:w-full w-2/5 border-gray-800 text-white">
        <input
            type={'text'}
            onChange={(e) => setText(e.target.value)}
            value={text}
            className="bg-transparent w-full outline-none border-none"
            placeholder="Добавить заметку"
            onKeyDown={(e) => e.key === 'Enter' && onSubmit(text)}
        />
    </div>
);
};

```

```
export default CreateNote;
```

B.4 Getdate.js

```
function getCurrentDate(separator = '.') {
  let newDate = new Date();
  let date = newDate.getDate();
  let month = newDate.getMonth() + 1;
  let year = newDate.getFullYear();

  return `${date}${separator}${
    month < 10 ? `0${month}` : `${month}`
  }${separator}${year}`;
}
export default getCurrentDate;
```

B.5 posts.js

```
import { createAsyncThunk, createSlice } from '@reduxjs/toolkit';
import axios from '../././axios';
import getCurrentDate from '.././utils/Getdate';

export const fetchPosts = createAsyncThunk('todo/fetchPosts', async () => {
  const { data } = await axios.get('todo');
  return data;
});

export const fetchRemovePosts = createAsyncThunk(
  'todo/fetchRemovePosts',
  async (id) => await axios.delete(`todo/${id}`),
);

const initialState = {
  posts: {
```

```

    items: [],
    status: 'loading',
  },
};

const postsSlice = createSlice({
  name: 'posts',
  initialState,
  reducers: {
    addTask: (state, action) => {
      const itemsTodoAdd = {
        _id: action.payload.items._id,
        text: action.payload.items.text,
        dateCreate: getCurrentDate(),
      };
      state.posts.items.push(itemsTodoAdd);
    },
  },
  extraReducers: {
    [fetchPosts.pending]: (state) => {
      state.posts.items = [];
      state.posts.status = 'loading';
    },
    [fetchPosts.fulfilled]: (state, action) => {
      state.posts.items = action.payload;
      state.posts.status = 'loaded';
    },
    [fetchPosts.rejected]: (state) => {
      state.posts.items = [];
      state.posts.status = 'error';
    },
  },
});

```

```

    },
    [fetchRemovePosts.pending]: (state, action) => {
      state.posts.items = state.posts.items.filter(
        (obj) => obj._id !== action.meta.arg,
      );
    },
    [fetchRemovePosts.rejected]: (state) => {
      state.posts.status = 'error';
    },
  },
});
export const { addTask } = postsSlice.actions;
export const postsReducer = postsSlice.reducer;

```

B.6 PostController.js

```

import PostModel from '../models/Post.js';
export const getAll = async (req, res) => {
  try {
    const posts = await PostModel.find();
    res.json(posts);
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: 'Get all todo failed',
    });
  }
};
export const remove = async (req, res) => {
  try {
    const postId = req.params.id;

```

```

PostModel.findOneAndDelete({
  _id: postId,
}).then((doc, err) => {
  if (err) {
    console.log(err);
    return res.status(500).json({
      message: 'Delete todo failed',
    });
  }
  if (!doc) {
    console.log(err);
    return res.status(500).json({
      message: 'Get one todo failed',
    });
  }

  res.json({
    success: true,
  });
});
} catch (err) {
  console.log(err);
  res.status(500).json({
    message: 'Remove todo failed',
  });
}
};

export const create = async (req, res) => {
  try {

```

```

const doc = new PostModel({
  text: req.body.text,
  dateCreate: req.body.dateCreate,
  imageUrl: req.body.imageUrl,
});

const post = await doc.save();

res.json(post);
} catch (err) {
  console.log(err);
  res.status(500).json({
    message: 'Created todo failed',
  });
}
};

```

B.7 index.js

```

import express from 'express';
import mongoose from 'mongoose';
import {ConstApi, handleValidationErrors} from './Utils/index.js';
import {PostController} from './controllers/index.js';
import * as Validation from './validation/validation.js';
import cors from 'cors';
import * as https from "https";
import * as fs from "fs";

const directory = 'uploads';

mongoose

```

```

    .connect('mongodb://45.94.158.55:27017/dashboardProject')
    .then(() => {
        console.log('Connected DB ok!');
    })
    .catch((err) => console.log('DB ERROR!', err));

const app = express();
app.use(cors());
app.use(express.json());

//posts
app.get(
    ConstApi.API_POST,
    Validation.postCreateValidation,
    PostController.getAll,
);
app.post(
    ConstApi.API_POST,
    Validation.postCreateValidation,
    handleValidationErrors,
    PostController.create,
);
app.delete(ConstApi.API_POST +('/:id', PostController.remove);

const privateKey = fs.readFileSync('/etc/letsencrypt/live/notez.pp.ua/privkey.pem',
'utf8');
const certificate = fs.readFileSync('/etc/letsencrypt/live/notez.pp.ua/cert.pem',
'utf8');
const ca = fs.readFileSync('/etc/letsencrypt/live/notez.pp.ua/chain.pem', 'utf8');

```

```
const caFull = fs.readFileSync('/etc/letsencrypt/live/notez.pp.ua/fullchain.pem',  
'utf8');
```

```
const credentials = {  
  key: privateKey,  
  cert: caFull,  
  ca: ca,  
  
};
```

```
const httpsServer = https.createServer(credentials, app);
```

```
httpsServer.listen(7345, () => {  
  console.log('HTTPS Server running on port 443');  
});
```

ДОДАТОК В
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
кафедра програмних засобів

дипломна робота на тему:

Розробка вебзастосунку інформаційної дошки Development of the Information Board Web Application

Виконав: Євген ДЕМ'ЯНЕНКО
Керівник: Ольга ГЛАДКОВА

Рисунок В.1 — Слайд 1

Об'єкт дослідження

— програмне забезпечення ведення інформаційних дошок.

Предмет дослідження

— методи та технології розробки програмного забезпечення для створення інформаційних дошок.

Мета дослідження

— розробка вебзастосунку для створення інформаційної дошки, що надає користувачам зручний інструмент для зберігання, організації та керування нотатками.

Рисунок В.2 — Слайд 2

Порівняння існуючих аналогів

Функції	Rapidtables	Online Notepad	aNotepad	Блокнот Онлайн
Створення та редагування нотаток	+	+	+	+
Збереження та завантаження файлів	+	+	+	+
Інструменти форматування	+	Частково	Частково	Частково
Пошук та заміна	+	-	-	-
Доступність та зручність	Онлайн	Онлайн	Онлайн	Онлайн
Безкоштовна та без реклами	+	+	+	+
Простий інтерфейс	-	+	+	+
Автоматичне збереження	-	+	+	+
Відкриття і збереження файлів	-	+	+	+
Підтримка форматування тексту	-	+	+	+
Можливість друку	-	+	+	+
Сортування	-	+	+	-
Доступність і приватність	+	+	+	+

Рисунок В.3 — Слайд 3

Діаграма варіантів використання програмного забезпечення

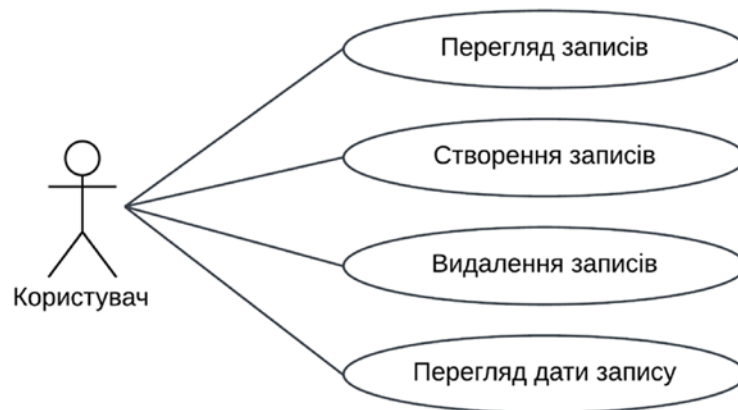


Рисунок В.4 — Слайд 4

Вибір мови програмування

	JavaScript	PHP	Python
Тип мови	Клієнтська та серверна мова	Серверна мова	Універсальна мова
Використання	Веб-розробка, фронтенд	Веб-розробка, серверний сценарій	Веб-розробка, наукові обчислення, штучний інтелект, аналіз даних
Об'єктно-орієнтований	Так	Так	Так
Розширення	Фреймворки та бібліотеки	Розширення PHP	Багато вбудованих функцій та бібліотек, встановлення додаткових пакетів
Динамічна типізація	Так	Так	Так
Переносимість	Залежить від веб-браузера	Багатоплатформовий	Багатоплатформовий
Використання в інших областях	Мобільна розробка (React Native, Ionic), серверне програмування (Node.js)		Наукові обчислення, штучний інтелект, аналіз даних, автоматизація таскриптовання, робототехніка,
Швидкодія	Висока	Висока	Висока

Рисунок В.5 — Слайд 5

Порівняння середовищ розробки

	VS Code	Sublime Text	Atom	IntelliJ IDEA
Багатомовність	+	+	+	+
Автодоповнення	+	+	+	+
Відлагодження	+	+	-	+
Універсальність	+	+	+	+
Рефакторинг коду	-	+	-	+
Контроль версій	+	+	+	+
Розширюваність	+	+	+	+
Легкість використання	+	+	+	+
Система інструментів	+	+	+	+
Налаштування та розширення	+	+	+	+

Рисунок В.6 — Слайд 6

Загальна схема системи

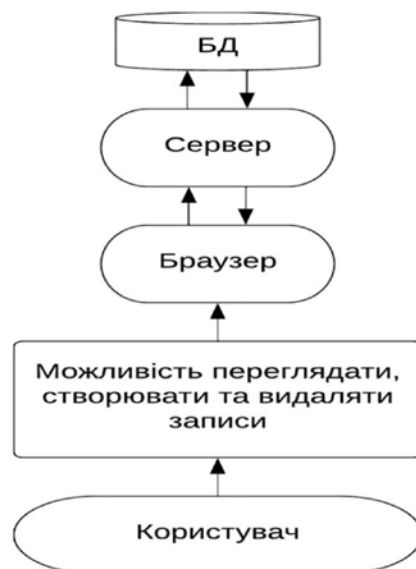


Рисунок В.7 — Слайд 7

Блок- схема алгоритму введення «нотатки»

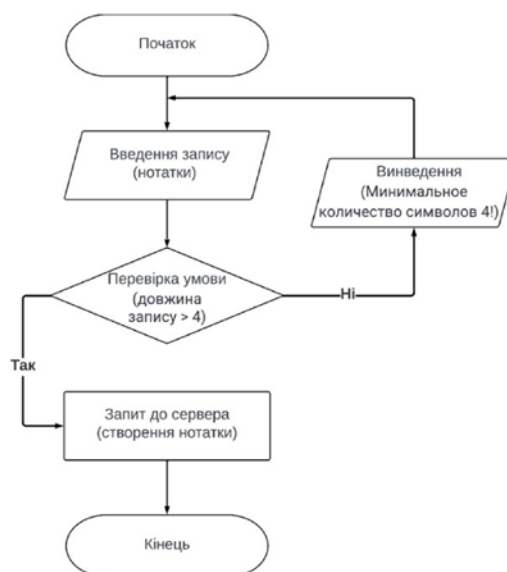


Рисунок В.8 — Слайд 8

Інтерфейс програмного продукту (екран ноутбука)

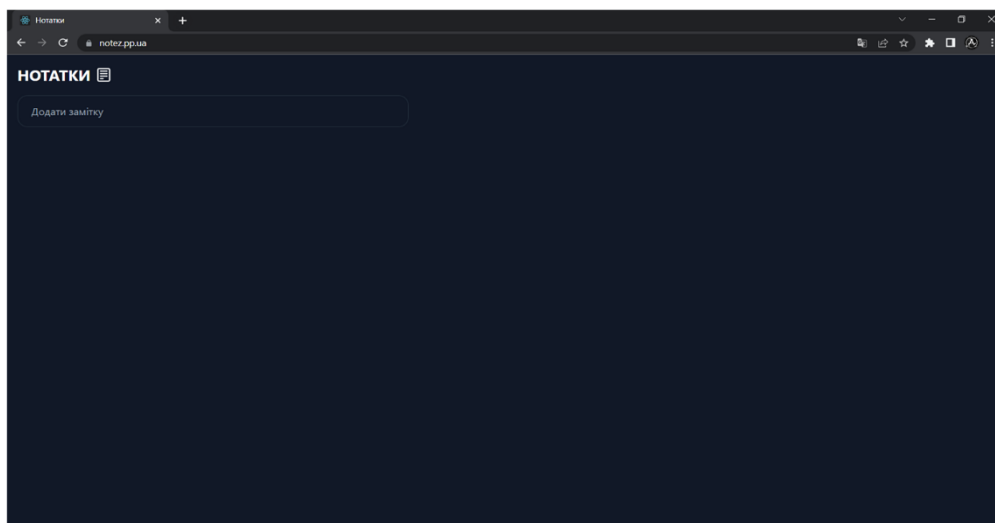


Рисунок В.9 — Слайд 9

Інтерфейс програмного продукту (екран смартфона)

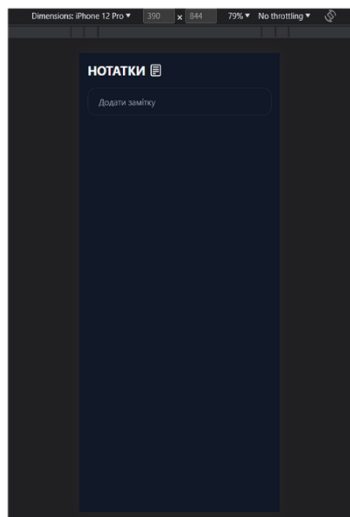


Рисунок В.10 — Слайд 10

Створення записів

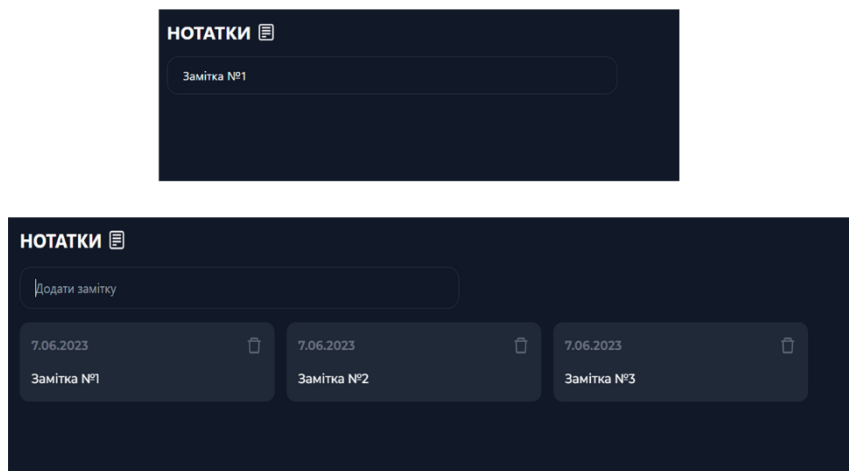


Рисунок В.11 — Слайд 11

Видалення записів

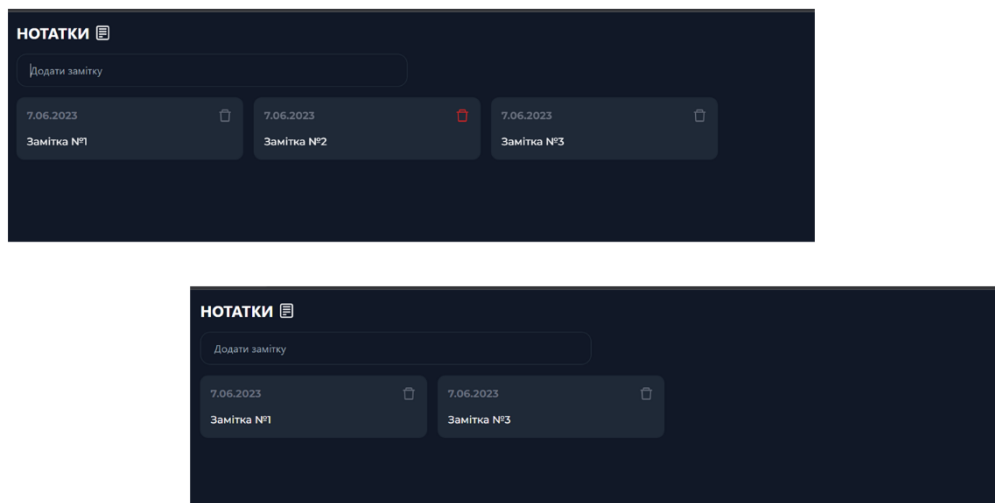


Рисунок В.12 — Слайд 12

Чек-лист тестування програмного продукту

Тест	Опис тесту	Результат
Функціональність		
Створення	Перевірка можливості створення нової нотатки	ОК
Видалення	Перевірка можливості видалення нотатки	ОК
Інтерфейс		
7. Навігація	Перевірка роботи кнопок, посилань і меню	ОК
8. Відображення	Перевірка коректного відображення нотаток	ОК
9. Реакція	Перевірка реакції програми на дії користувача	ОК
Збереження даних		
Автозбереження	Перевірка автоматичного збереження нотаток	ОК
Відновлення	Перевірка відновлення нотаток після перезапуску	ОК
Безпека		
Захист доступу	Перевірка наявності захисту доступу до нотаток	ОК
Обробка даних	Перевірка безпечної обробки та зберігання конфіденційної інформації	ОК

Рисунок В.13 — Слайд 13

Висновки

У ході даної дипломної кваліфікаційної роботи було розроблено програмний продукт, який надає користувачам зручний інструмент для зберігання, організації та керування нотатками. Проект був розроблений з використанням методів та технологій, таких як IntelliJ IDEA, JavaScript, React, а також бази даних MongoDB.

Програмний засіб для створення інформаційної дошки має зручний та зрозумілий інтерфейс та функціонал. Проект відповідає поставленим завданням і вимогам, а його реалізація включає в себе використання сучасних технологій розробки програмного забезпечення. Розроблений програмний засіб має великий потенціал для полегшення повсякденних завдань користувачів і покращення їх продуктивності.

Рисунок В.14 — Слайд 14