

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ КЕРУВАННЯ НОТАТКАМИ

SOFTWARE FOR NOTE MANAGEMENT

Виконав(ла): студент(ка) 4 курсу, групи КНТ-113сп

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЯНСВ І.С.

(ПРИЗВИЩЕ та ініціали)

Керівник КОПЛАКОВА Т.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЯНСВА Івана Сергійовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для керування нотатками.
Software for Note Management

керівник проєкту (роботи) к.т.н., доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 7 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 03 червня 2026 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. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

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

7. Дата видачі завдання “ 7 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Іван ЯНСВ
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна КОЛПАКОВА
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
80 с., 3 табл., 29 рис., 3 дод., 17 джерел.

JAVASCRIPT, REACT, КЕРУВАННЯ НОТАТКАМИ, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНА БІБЛІОТЕКА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для керування нотатками.

Предмет дослідження – програмні засоби для керування особистими записами.

Мета роботи – розробка програмного забезпечення керування особистими записами для зменшення часу, що витрачається користувачем на пошук і систематизацію інформації.

Матеріали, методи та технічні засоби: мова програмування JavaScript, бібліотека React.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для керування особистими записами. Створено програмне забезпечення для керування особистими записами. Проведено тестування розробленої програмної системи для керування особистими записами.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для керування особистими записами за допомогою мови програмування JavaScript та бібліотеки React.

Галузь використання – планування робочого часу.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 80 pages, 3 tables, 29 figures, 3 appendixes, 17 sources.

JAVASCRIPT, REACT, NOTE MANAGEMENT, PROGRAMMING LANGUAGE, SOFTWARE LIBRARY, SOFTWARE.

The object of research is the process of developing software for managing notes.

The subject of the research is software for supporting the managing notes.

The purpose of this work is to develop software for managing notes to reduce the time spent by the user on searching and systematizing information.

Materials, methods and technical tools: JavaScript programming language, React library.

Results. Project solutions for the creation of software for managing notes has been designed. The software for managing notes has been developed. The developed software system for managing notes has been tested.

Conclusions. The goal of the work has been achieved. The software tools for managing notes using the JavaScript programming language and the React library have been developed.

Scope of use – working time planning.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Програмне забезпечення для керування особистими записами	11
1.2 Аналіз програмного забезпечення для керування особистими записами	16
1.3 Висновки за розділом 1	25
2 Матеріали і методи	26
2.1 Вибір мови програмування	26
2.2 Вибір програмної бібліотеки для створення програмного забезпечення для керування особистими записами	29
2.3 Висновки за розділом 2	32
3 Опис програми	34
3.1 Структура програмного забезпечення для керування особистими записами	34
3.2 Функціонування програмного забезпечення для керування особистими записами	37
3.3 Розробка бази даних програмного забезпечення для керування особистими записами	39
3.4 Проєктування інтерфейсу програмного забезпечення для керування особистими записами	42
3.5 Висновки за розділом 3	44
4 Експлуатація, тестування та експериментальне дослідження програми	46
4.1 Призначення й умови застосування програми	46
4.2 Характеристики програми для керування особистими записами	47
4.3 Інструкція по експлуатації програми	49
4.3.1 Звернення до програми	49
4.3.2 Вхідні й вихідні дані	49

4.3.3 Повідомлення.....	49
4.4 Виконання програмного забезпечення для керування особистими записами	50
4.5 Тестування програмного забезпечення для керування особистими записами	52
4.6 Висновки за розділом 4	53
Висновки.....	54
Перелік джерел посилання	57
Додаток А Технічне завдання.....	59
Додаток Б Фрагмент тексту програми	64
Додаток В Слайди презентації	73

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVC – Model-View-Controller;

UI – user interface;

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Актуальність програмного забезпечення для керування особистими записами зумовлюється загальною тенденцією до цифровізації освітньої, наукової та професійної діяльності, а також зростанням обсягів інформації, що опрацьовується користувачами щоденно. В умовах інформаційного перенасичення виникає об'єктивна потреба у впорядкованому збереженні, швидкому доступі та ефективному використанні текстових, графічних і мультимедійних матеріалів, що робить спеціалізовані програмні засоби для роботи з нотатками важливим інструментом організації інтелектуальної діяльності [1]-[3].

Актуальність програмного забезпечення для керування особистими записами підсилюється зростанням популярності віддаленої роботи та міждисциплінарної взаємодії. У таких умовах важливого значення набувають можливості синхронізації даних між різними пристроями, спільного доступу до інформації, швидкого пошуку та категоризації записів. Наявність таких функцій сприяє підвищенню продуктивності праці, зменшенню часу на пошук необхідних відомостей та мінімізації ризику втрати важливої інформації [4], [5].

Проте деякі програми для керування особистими записами можуть не забезпечувати достатньої гнучкості у налаштуванні структури збереження даних, що ускладнює адаптацію програмного продукту до індивідуальних потреб користувачів або специфіки конкретної предметної галузі. Суттєвим недоліком є також обмежені можливості інтеграції з іншими інформаційними системами та сервісами. У багатьох випадках програмне забезпечення для керування особистими записами функціонує ізольовано, що призводить до фрагментації інформації та потреби дублювання даних між різними програмними середовищами. Окрему проблему становлять питання продуктивності та масштабованості. При збільшенні обсягу збережених нотаток деякі програмні рішення демонструють зниження швидкості пошуку,

завантаження або синхронізації даних. Подібні обмеження є особливо критичними для користувачів, які працюють з великими масивами текстової інформації протягом тривалого часу [1]-[7].

Тому актуальною є розробка програмного забезпечення для керування особистими записами. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для керування особистими записами.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для керування особистими записами;
- здійснити проєктування програмного забезпечення для керування особистими записами;
- створити програмне забезпечення для керування особистими записами;
- виконати тестування розробленого програмного забезпечення для керування особистими записами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для керування особистими записами

Програмне забезпечення для керування особистими записами користувача призначено для створення, збереження, впорядкування та подальшого використання персональних записів у цифровому середовищі. Такі програмні засоби орієнтовані на підтримку роботи з інформацією, що виникає у процесі навчання, професійної діяльності або повсякденного життя, та забезпечують зручний перехід від фрагментарних записів до цілісної структури знань. Основною метою застосування таких програм є спрощення фіксації думок, ідей, завдань і довідкових відомостей із можливістю швидкого доступу до них у будь-який момент [1]-[6].

У межах подібних програмних систем зазвичай реалізується логіка централізованого зберігання даних, що дозволяє уникати втрати інформації та зменшувати залежність від паперових носіїв. Нотатки можуть об'єднуватися за змістом, контекстом або призначенням, що сприяє кращій організації інформаційного простору користувача. Вважається важливою властивістю підтримка різних форматів подання даних, оскільки це дає змогу працювати не лише з текстом, а й з іншими видами цифрового контенту, адаптуючи записи під конкретні потреби [1], [2].

Переваги програмного забезпечення для керування нотатками зазвичай пов'язуються з підвищенням ефективності роботи з інформацією та оптимізацією часу. Завдяки автоматизованому пошуку, фільтрації та впорядкуванню записів спрощується процес відновлення потрібних відомостей, що позитивно впливає на продуктивність діяльності. Крім того, такі системи сприяють формуванню звички до систематичного ведення записів, що в довгостроковій перспективі полегшує накопичення та переосмислення знань [3], [4].

Особливістю цього типу програмних продуктів вважається їхня орієнтація на персоналізацію та гнучке налаштування. Користувачеві

надається можливість адаптувати структуру зберігання даних, способів відображення інформації та механізми взаємодії з нотатками відповідно до власних уподобань. Важливу роль також відіграють питання доступності та захисту даних, оскільки сучасні рішення часто передбачають використання різних пристроїв і середовищ для роботи з одними й тими самими записами. У сукупності це дозволяє розглядати програмне забезпечення для керування нотатками як універсальний інструмент підтримки особистої інформаційної діяльності, що поєднує зручність, гнучкість і практичну цінність [2], [3].

Такі програмні системи спрямовані на підвищення контролю над власними даними, дозволяють структурувати думки, ідеї, плани, завдання та інші відомості, забезпечуючи швидкий та зручний доступ до них у будь-який момент. Сучасні програмні рішення для керування особистими записами користувачів підтримують різноманітні формати контенту, що включають текстові записи, мультимедійні елементи, посилання та документи, забезпечуючи інтеграцію різних типів інформації в єдину систему. Завдяки цьому користувач може ефективно організовувати свої ресурси, планувати дії та відстежувати прогрес у різних сферах діяльності [4], [5].

Переваги використання програм для керування особистими записами полягають у підвищенні продуктивності та економії часу. Завдяки наявності функцій пошуку, фільтрації, тегування та сортування інформації процес відновлення необхідних даних стає швидким та інтуїтивним. Також значення має можливість синхронізації між різними пристроями, що дозволяє підтримувати актуальність записів незалежно від того, де здійснюється робота. Крім того, такі системи допомагають формувати дисципліну щодо регулярного ведення записів, що сприяє накопиченню цінної інформації та її ефективному використанню у майбутньому [5], [6].

Особливості таких програмних систем пов'язані з можливістю персоналізації інтерфейсу та налаштування функціоналу відповідно до індивідуальних потреб користувача. Передбачаються гнучкі інструменти організації даних, можливість відстеження змін та збереження історії записів,

а також забезпечення безпеки та конфіденційності інформації. У цілому програмне забезпечення для керування особистими записами створює зручне, структуроване і надійне середовище для контролю та ефективного використання власної інформації, поєднуючи інтелектуальні функції з практичною користю для щоденної діяльності [2], [3].

Процес створення програмного забезпечення для керування особистими записами визначається як систематична діяльність зі створення цифрового інструменту, призначеного для організації, збереження та обробки персональної інформації користувачів. Метою цього процесу є розробка програмного продукту, який забезпечує ефективну роботу з даними, спрощує доступ до них, підвищує продуктивність користувача та гарантує безпеку і конфіденційність інформації. Процес розробки включає комплекс взаємопов'язаних етапів, що охоплюють як планування та проектування, так і тестування та впровадження системи [4], [5].

Першим етапом є визначення вимог до системи, що передбачає збирання та аналіз інформації про потреби користувачів, очікуваний функціонал та особливості інтерфейсу. На цьому етапі формулюються цілі програмного продукту, описуються основні сценарії його використання та уточнюються технічні та програмні обмеження, що будуть враховані при проектуванні [3], [4].

Наступним кроком є розробка архітектури та моделювання системи, що включає визначення структури програмного забезпечення, поділ його на модулі, а також опис взаємодії між компонентами. На цьому етапі вирішується, як дані будуть зберігатися та оброблятися, які технології й мови програмування будуть використані, а також як забезпечити масштабованість і надійність системи [3], [4].

Після цього здійснюється безпосередня реалізація програмного продукту, що включає написання коду, інтеграцію різних компонентів, розробку користувацького інтерфейсу та впровадження механізмів безпеки. Важливою складовою цього етапу є забезпечення сумісності між різними

платформами та пристроями, на яких планується використання програмного забезпечення, а також налаштування функцій синхронізації та резервного копіювання даних [3], [4].

Подальшим етапом є тестування системи, що включає перевірку функціональності, стабільності та продуктивності програмного забезпечення. На цьому етапі виявляються та усуваються помилки, перевіряється правильність роботи всіх модулів і взаємодія між ними, а також оцінюється зручність користувацького інтерфейсу. Тестування може охоплювати як автоматизовані перевірки, так і експериментальне використання програмного продукту потенційними користувачами [3], [4].

Завершальним етапом є впровадження та підтримка системи, що передбачає розгортання програмного забезпечення, надання доступу користувачам, а також організацію оновлень і технічної підтримки. Під час експлуатації системи здійснюється моніторинг її роботи, аналіз зворотного зв'язку від користувачів та внесення необхідних змін для підвищення ефективності та зручності використання. Таким чином, процес створення програмного забезпечення для керування особистими записами є комплексним циклом заходів, спрямованих на створення надійного, функціонального та зручного інструменту для організації персональної інформації [3], [4].

Особливості створення програмного забезпечення для керування особистими записами пов'язані насамперед із необхідністю забезпечити ефективну організацію та швидкий доступ до інформації користувача. Такі системи потребують продуманої архітектури, яка дозволяє інтегрувати різні формати даних, підтримувати багаторівневу класифікацію записів та забезпечувати можливість пошуку і сортування за різними критеріями. Важливим аспектом є забезпечення гнучкості у налаштуванні інтерфейсу та функціональних можливостей, щоб користувач міг адаптувати програму під власні потреби та робочі процеси. Особливістю цих систем є також необхідність синхронізації даних між різними пристроями та платформами, що дозволяє користувачу мати актуальний доступ до своїх записів у будь-

якому місці та у будь-який час. Крім того, значну увагу приділяється безпеці та конфіденційності інформації, оскільки персональні дані потребують захисту від несанкціонованого доступу та втрати [3], [4].

Створення подібного програмного забезпечення пов'язане з низкою проблем і складнощів. По-перше, виникає потреба у балансуванні між функціональністю та простотою використання: система повинна пропонувати достатньо інструментів для керування записами, але при цьому залишатися інтуїтивно зрозумілою. По-друге, значною проблемою є забезпечення стабільної синхронізації та збереження даних у реальному часі, особливо коли користувач працює з різними пристроями та платформами. Труднощі можуть виникати й у забезпеченні високого рівня безпеки, що включає шифрування інформації, захист від витоку даних та контроль доступу, а також дотримання вимог конфіденційності, що можуть відрізнятися у різних юрисдикціях. Ще одним викликом є масштабованість системи: програмне забезпечення повинно ефективно обробляти великі обсяги записів, підтримувати швидкий пошук і збереження інформації без зниження продуктивності. Також виникають труднощі, пов'язані з інтеграцією додаткових функцій, таких як робота з мультимедійними файлами, тегами, нагадуваннями або автоматизованими звітами, оскільки вони ускладнюють структуру системи та підвищують вимоги до ресурсів. У сукупності ці фактори роблять процес створення програмного забезпечення для керування особистими записами складним і багатогранним, вимагаючи ретельного планування, професійного підходу до архітектури та постійного тестування для забезпечення надійності, зручності та безпеки користувацьких даних [5], [6].

Визначено, що програмне забезпечення для керування особистими записами користувача призначено для створення, збереження, впорядкування та подальшого використання персональних записів у цифровому середовищі. Такі програмні засоби орієнтовані на підтримку роботи з інформацією, що виникає у процесі навчання, професійної діяльності або повсякденного життя.

1.2 Аналіз програмного забезпечення для керування особистими записами

Проаналізуємо програмне забезпечення для керування особистими записами [7]-[13].

Програмне забезпечення Fathom (рис. 1.1) є програмою інструмент для автоматизованого формування записів або нотаток щодо онлайн зустрічей і подальшої роботи з отриманою інформацією. Його функціонування спрямоване на те, щоб зняти з користувача потребу вести ручні записи під час розмов, забезпечуючи автоматичний запис, розпізнавання мовлення та формування узагальненого опису обговорень. У результаті увага зосереджується безпосередньо на комунікації, тоді як система самостійно створює структурований підсумок після завершення зустрічі [9], [11].

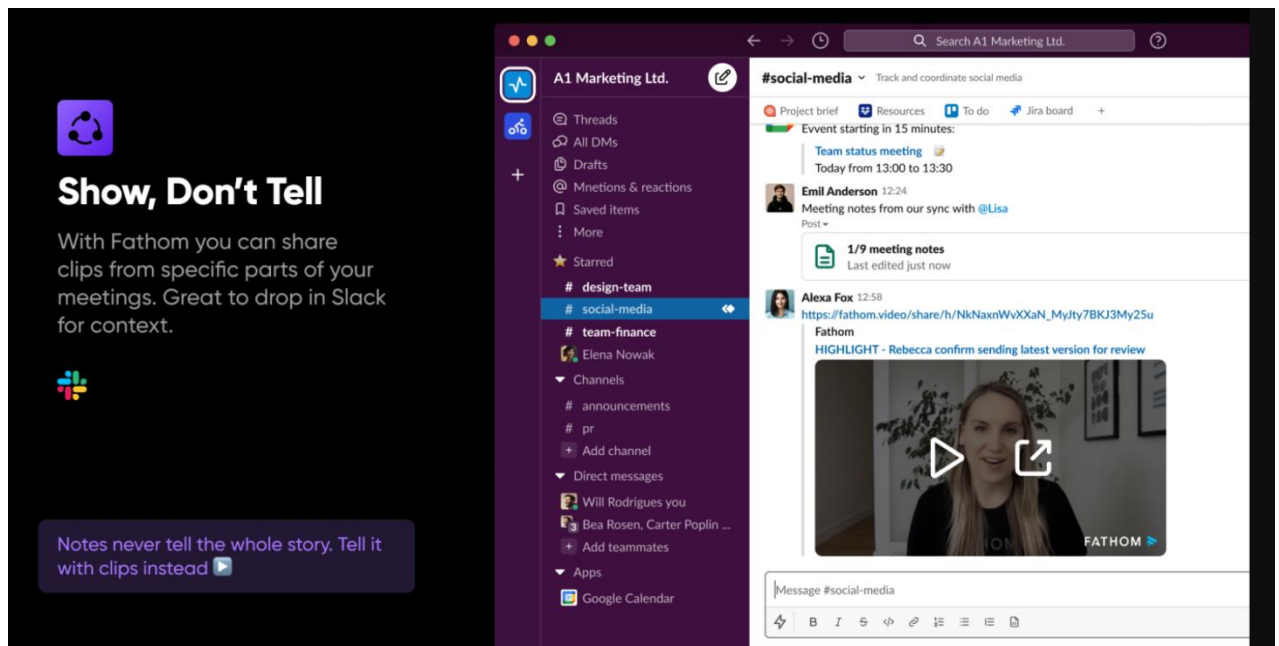


Рисунок 1.1 – Програмне забезпечення Fathom [11]

Отримані дані використовуються для формування змістовних звітів, у яких відображаються ключові моменти діалогу, важливі домовленості та завдання. Ця інформація може бути інтегрована з корпоративними системами управління взаємовідносинами з клієнтами або задіяна в автоматизованих

робочих процесах через зовнішні сервіси, що дозволяє скоротити час на обробку результатів зустрічей та підвищити узгодженість командної роботи [9], [11].

Використання застосунку Fathom не потребує складних налаштувань, що робить його доступним для широкого кола користувачів. Узагальнення, сформовані із застосуванням технологій штучного інтелекту, відзначаються актуальністю та зручністю для подальшого аналізу. Передбачається можливість ознайомлення з повною текстовою розшифровкою розмови, перегляду запитань, які виникали під час обговорення, а також швидкого переходу до відповідних фрагментів запису для уточнення деталей. Додатково у Fathom реалізовано функцію інтелектуального запиту до змісту зустрічі, що дозволяє отримувати відповіді на конкретні питання без необхідності самостійного пошуку [9], [11].

Окрему увагу приділено підтримці користувачів і питанням захисту даних. Оброблена інформація зберігається в межах облікового запису та не використовується для сторонніх цілей, що сприяє підвищенню довіри та відчуттю безпеки під час роботи. У відгуках користувачів зазвичай підкреслюється висока точність текстових розшифровок, зручність автоматизованого створення нотаток і практична цінність сформованих підсумків. Водночас іноді згадуються окремі технічні труднощі, обмежена гнучкість налаштувань узагальнених звітів та поодинокі неточності в розпізнаванні мовлення [9], [11].

Основними характеристиками програмного забезпечення Fathom є такі [9], [11]:

- автоматизоване ведення нотаток: програмне забезпечення Fathom забезпечує самостійний запис змісту онлайн зустрічей без участі користувача, що дозволяє повністю зосередитися на спілкуванні під час розмови [9], [11];

- транскрибування мовлення: реалізовано функцію перетворення усної мови учасників зустрічі у текстовий формат з подальшим збереженням результатів для аналізу та перегляду [9], [11];

- інтелектуальне узагальнення інформації: програмне забезпечення Fathom формує стислий підсумок обговорення із виділенням ключових моментів та важливих рішень із використанням алгоритмів штучного інтелекту [9], [11];

- інтеграція з зовнішніми сервісами: передбачено можливість синхронізації результатів зустрічей із системами управління клієнтськими даними та сервісами автоматизації робочих процесів [9], [11];

- зручна навігація за змістом: користувачеві програмного забезпечення Fathom надається доступ до повної стенограми з можливістю швидкого переходу до відповідних фрагментів запису зустрічі [9], [11];

- підтримка запитів до змісту зустрічі: у програмному забезпеченні Fathom реалізовано механізм отримання відповідей на запитання щодо обговорюваних тем без необхідності ручного перегляду матеріалів [9], [11];

- орієнтація на безпеку даних: оброблена інформація не використовується поза межами облікового запису користувача, що забезпечує конфіденційність та захист даних [9], [11];

- простота використання: інтерфейс та логіка роботи програмного забезпечення Fathom орієнтовані на швидке освоєння без складних налаштувань і додаткового навчання [9], [11].

Серед недоліків програмного забезпечення Fathom можна відзначити періодичні труднощі з автоматичним підключенням до онлайн зустрічей, обмежені можливості налаштування сформованих підсумків, а також можливу появу неточностей у результатах транскрибування під час обробки мовлення [9], [11].

Програмне забезпечення Amplenote (рис. 1.2) представляє собою комплексний інструмент, у якому процес фіксації інформації тісно пов'язується з організацією діяльності та управлінням особистою ефективністю. Його концепція ґрунтується на поєднанні нотаток із механізмами планування, що дозволяє не лише зберігати ідеї, а й одразу перетворювати їх на конкретні дії. Такий підхід особливо корисний у

ситуаціях, коли необхідно одночасно працювати з кількома напрямками діяльності та контролювати виконання поточних і перспективних завдань у межах єдиного середовища [10], [12].

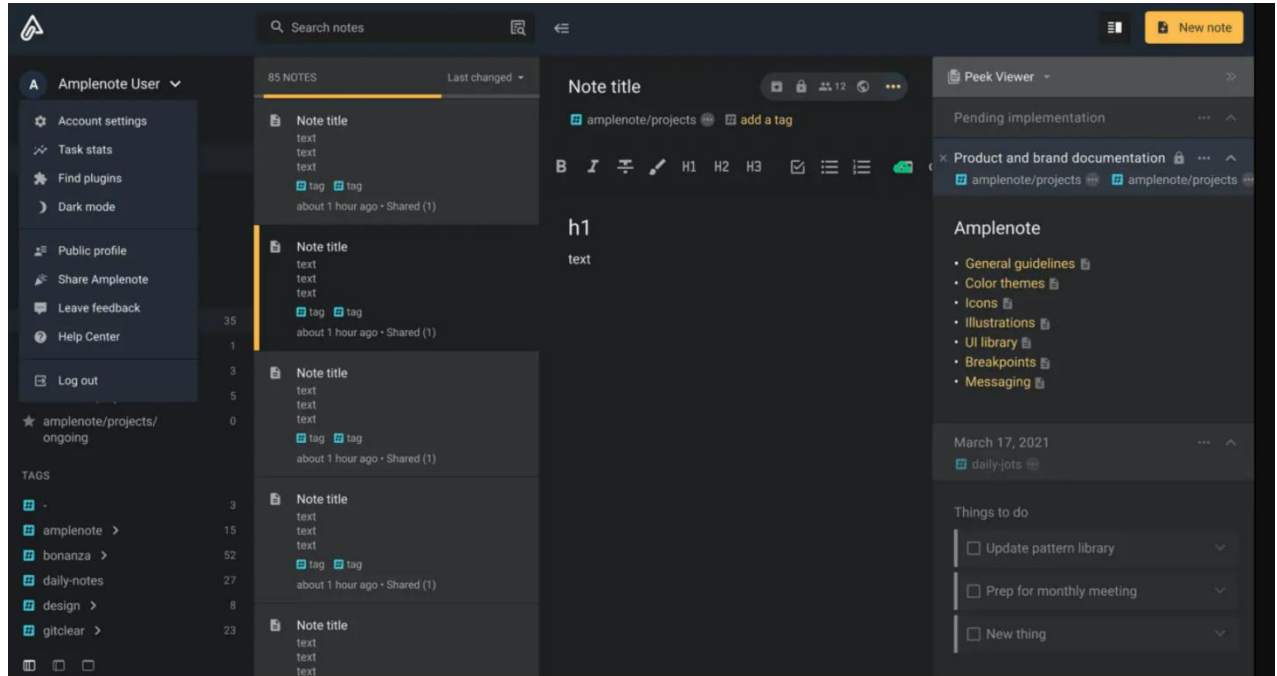


Рисунок 1.2 – Програмне забезпечення Amplenote [12]

Використання програмного забезпечення Amplenote орієнтоване на підвищення особистої продуктивності, оскільки кожен запис у системі може бути логічно пов'язаний із подальшими кроками користувача. На відміну від рішень, що зосереджуються виключно на зберіганні інформації або колективній роботі, дане програмне забезпечення спрямоване на підтримку індивідуального прогресу. Реалізовані механізми оцінювання важливості завдань допомагають впорядковувати робоче навантаження та концентрувати увагу на найбільш значущих цілях [10], [12].

Функціональні можливості програмного забезпечення Amplenote дозволяють перетворювати текстові записи на завдання з визначеними параметрами виконання, пов'язувати їх із календарним плануванням і контролювати виконання без необхідності використання сторонніх сервісів. Окрему роль відіграє підхід до безпеки даних, адже захист інформації

забезпечується сучасними методами шифрування, що підвищує рівень довіри до збережених матеріалів. Разом із тим варто враховувати, що екосистема розширень програмного забезпечення Amplenote є відносно обмеженою, а можливості спільної роботи реалізовані не так широко, як у деяких альтернативних рішеннях, при цьому доступ до окремих розширених функцій пов'язаний з умовами платного використання [10], [12].

Програмне забезпечення Amplenote має такі особливості [10], [12]:

- інтеграція нотаток і завдань: програмне забезпечення Amplenote поєднує збереження інформації з керуванням діями, що дозволяє перетворювати записи на повноцінні елементи робочого процесу [10], [12];

- орієнтація на продуктивність: кожна нотатка розглядається як потенційний крок до досягнення цілей, що сприяє більш усвідомленому плануванню особистої діяльності [10], [12];

- система пріоритетів: у програмному забезпеченні Amplenote реалізовано механізм оцінювання важливості завдань, який допомагає користувачеві зосереджувати увагу на найбільш значущих справах [10], [12];

- планування через календар: у програмному забезпеченні Amplenote забезпечується тісний зв'язок між нотатками та розкладом, що спрощує організацію часу й контроль виконання запланованих дій [10], [12];

- захист інформації: у програмному забезпеченні Amplenote застосовуються засоби наскрізного шифрування, які гарантують конфіденційність збережених даних і підвищують рівень безпеки [10], [12];

- універсальність використання: програмне забезпечення Amplenote адаптоване для роботи на різних платформах, що забезпечує доступ до нотаток у будь-якому середовищі [10], [12];

- індивідуальна спрямованість: основний акцент програмного забезпечення Amplenote зроблено на особистій роботі користувача, а не на розширених можливостях колективної взаємодії [10], [12].

Серед недоліків програмного забезпечення Amplenote можна відзначити відносно обмежену екосистему додаткових можливостей,

недостатньо розвинені інструменти спільної роботи, а також залежність доступу до окремих функцій від умов платного використання [10], [12].

Програмне забезпечення Joplin (рис. 1.3) є програмним інструментом для роботи з нотатками з відкритим вихідним кодом, орієнтований на користувачів, для яких важливими є конфіденційність, самостійний контроль над даними та можливість гнучкого налаштування. На відміну від багатьох комерційних рішень, у даному застосунку реалізовано свободу вибору способу зберігання інформації, що дозволяє розміщувати записи безпосередньо на локальному пристрої або синхронізувати їх із зовнішніми хмарними сервісами відповідно до власних уподобань [8], [10], [13].

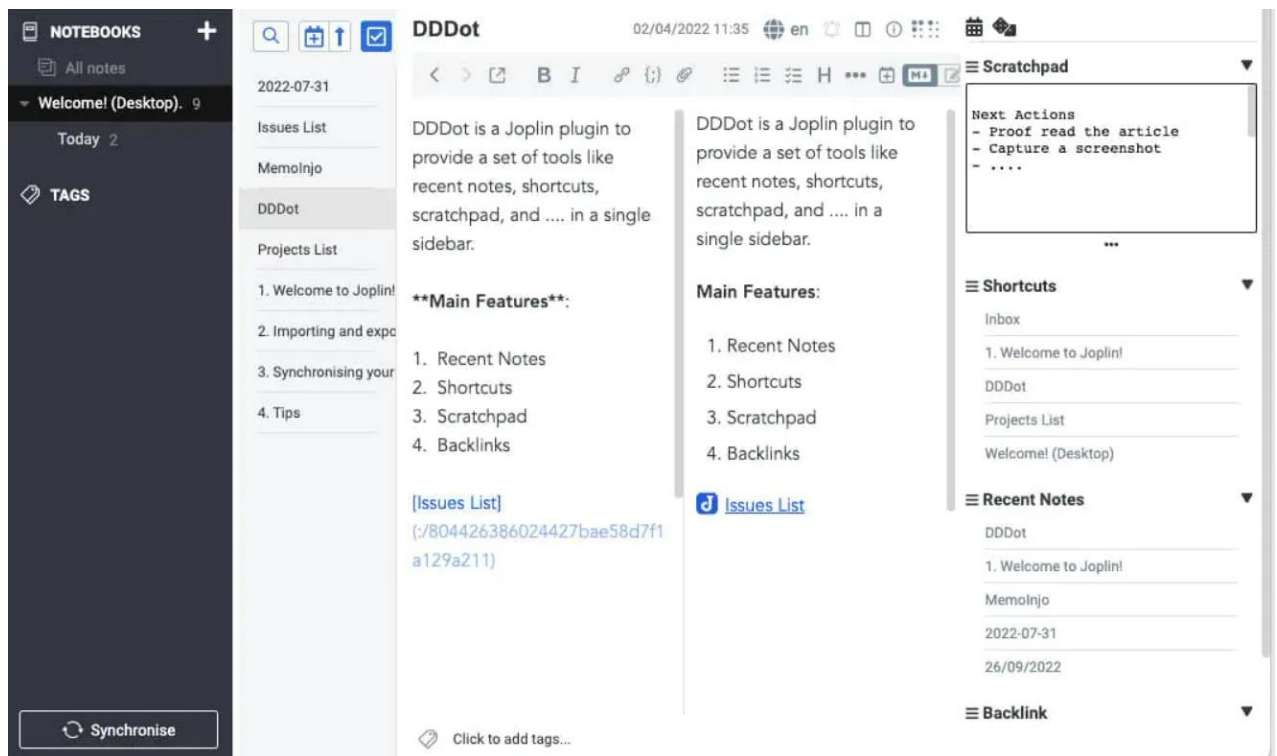


Рисунок 1.3 – Програмне забезпечення Joplin [13]

Особливістю Joplin є використання відкритих стандартів оформлення, зокрема підтримка розмітки Markdown, що робить його зручним для технічно підготовлених користувачів та тих, хто надає перевагу структурованому поданню текстової інформації. Такий підхід спрощує перенесення даних між різними середовищами та забезпечує незалежність від закритих форматів.

Додатково функціональність програмного забезпечення може бути розширена за рахунок підключення плагінів, які розробляються спільнотою, що сприяє адаптації інструмента до індивідуальних потреб [8], [10], [13].

Важливе місце у концепції Jorlin займає питання безпеки, оскільки реалізовано механізми захисту даних під час зберігання та синхронізації між пристроями. Окрім цього, застосунок дозволяє зберігати фрагменти вебсторінок і мережевий контент безпосередньо у власній базі нотаток, що розширює можливості збору інформації. Разом із тим інтерфейс програмного забезпечення може вимагати звикання, особливо для користувачів без досвіду роботи з текстовою розміткою, а вбудовані засоби колективної взаємодії залишаються обмеженими [8], [10], [13].

Програмне забезпечення Jorlin має такі особливості [8], [10], [13]:

- відкритий вихідний код: програмне забезпечення Jorlin розвивається як програмний проєкт з відкритою архітектурою, що забезпечує прозорість, можливість перевірки коду та гнучке доопрацювання спільнотою користувачів [8], [10], [13];

- орієнтація на конфіденційність: реалізовано підхід, за якого користувач зберігає повний контроль над власними даними, включно з можливістю локального збереження та захисту інформації [8], [10], [13];

- підтримка Markdown: створення та оформлення нотаток у Jorlin здійснюється з використанням текстової розмітки, що сприяє структурованості записів і сумісності з відкритими стандартами [8], [10], [13];

- захист даних: у системі Jorlin застосовуються механізми наскрізного шифрування, які забезпечують безпечну синхронізацію нотаток між різними середовищами використання [8], [10], [13];

- розширюваність функціоналу: у Jorlin передбачена підтримка плагінів, що дозволяє доповнювати базові можливості новими інструментами відповідно до індивідуальних потреб [8], [10], [13];

– збереження вебконтенту: у Joplin реалізовано засоби додавання інформації з вебресурсів безпосередньо до структури нотаток, що полегшує накопичення та систематизацію матеріалів [8], [10], [13];

– кросплатформність: програмне забезпечення Joplin доступне в різних програмних середовищах, що забезпечує зручний доступ до нотаток незалежно від пристрою [8], [10], [13].

Серед недоліків програмного забезпечення Joplin можна відзначити відносну складність інтерфейсу, обмежені можливості спільної роботи, а також необхідність адаптації для користувачів, які не мають досвіду роботи з текстовою розміткою [8], [10], [13].

Порівняльну характеристику програмного забезпечення для керування особистими записами наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для керування особистими записами

Критерій порівняння	Fathom [11]	Amplenote [12]	Joplin [13]
поєднання нотаток з управлінням завданнями користувачів	+—	+	+—
автоматизація нотаток з аудіо/відео	+	—	—
підтримка засобів розмітки тексту	—	+—	+
гнучкість зберігання даних	—	+	+
інструменти колективної роботи	+—	+—	—
простота інтерфейсу	+	+—	+—
розширення через плагіни	—	+—	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для керування особистими записами. Проте деякі програмні засоби для керування особистими записами можуть не забезпечувати достатньої гнучкості у налаштуванні структури збереження даних, що ускладнює адаптацію програмного продукту до індивідуальних потреб користувачів або специфіки конкретної предметної галузі. Суттєвим недоліком є також обмежені можливості інтеграції з іншими інформаційними системами та сервісами. У багатьох випадках програмне забезпечення для керування особистими записами функціонує ізольовано, що призводить до фрагментації інформації та потреби дублювання даних між різними програмними середовищами. Окрему проблему становлять питання продуктивності та масштабованості. При збільшенні обсягу збережених нотаток деякі програмні рішення демонструють зниження швидкості пошуку, завантаження або синхронізації даних. Подібні обмеження є особливо критичними для користувачів, які працюють з великими масивами текстової інформації протягом тривалого часу. Тому актуальною є розробка програмного забезпечення для керування особистими записами.

При розробці програмного забезпечення для керування особистими записами необхідно забезпечити такі функціональні вимоги:

- підтримка можливості відображення особистих записів користувача у вигляді нотаток;
- можливість персоналізації роботи програми (відображення нотаток) шляхом підтримки механізмів авторизації та реєстрації;
- можливість групування особистих записів користувача за категоріями;
- можливість створення нотаток та додавання їх до відповідних категорій;
- підтримка можливостей редагування та видалення вже створених нотаток.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для керування особистими записами користувача призначено для створення, збереження, впорядкування та подальшого використання персональних записів у цифровому середовищі. Такі програмні засоби орієнтовані на підтримку роботи з інформацією, що виникає у процесі навчання, професійної діяльності або повсякденного життя, та забезпечують зручний перехід від фрагментарних записів до цілісної структури знань.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для керування особистими записами. Проте деякі програмні засоби для керування особистими записами можуть не забезпечувати достатньої гнучкості у налаштуванні структури збереження даних, що ускладнює адаптацію програмного продукту до індивідуальних потреб користувачів або специфіки конкретної предметної галузі. Суттєвим недоліком є також обмежені можливості інтеграції з іншими інформаційними системами та сервісами. У багатьох випадках програмне забезпечення для керування особистими записами функціонує ізольовано, що призводить до фрагментації інформації та потреби дублювання даних між різними програмними середовищами. Окрему проблему становлять питання продуктивності та масштабованості. При збільшенні обсягу збережених нотаток деякі програмні рішення демонструють зниження швидкості пошуку, завантаження або синхронізації даних. Подібні обмеження є особливо критичними для користувачів, які працюють з великими масивами текстової інформації протягом тривалого часу. Тому актуальною є розробка програмного забезпечення для керування особистими записами.

Сформульовано функціональні вимоги до програмного забезпечення для керування особистими записами.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для керування особистими записами було використано мову програмування JavaScript [14], [15].

Мова програмування JavaScript належить до динамічно типізованих мов, що забезпечує гнучкість під час роботи з даними та спрощує процес прототипування і модифікації програмних рішень. Така властивість є особливо корисною на етапах швидкої розробки та експериментального впровадження функціоналу [14], [15].

Важливою рисою JavaScript є її подієво-орієнтована модель програмування, яка дозволяє ефективно обробляти дії користувачів та асинхронні процеси. Підтримка асинхронних механізмів, зокрема ключових слів `async/await`, сприяє підвищенню продуктивності додатків і забезпечує коректну роботу з мережевими запитами, файловими операціями та іншими ресурсомісткими завданнями. Це є суттєвою перевагою для програмних систем, що потребують швидкого реагування та обробки великої кількості подій [14], [15].

Окремої уваги заслуговує активна спільнота розробників та постійний розвиток стандарту мови. Регулярне оновлення специфікації ECMAScript сприяє появі нових мовних конструкцій, підвищенню безпеки та покращенню якості коду. Наявність великої кількості навчальних матеріалів, документації та відкритих проєктів спрощує процес навчання та впровадження JavaScript у практичну діяльність [14], [15].

Крім того, мова програмування JavaScript підтримує різні парадигми програмування, зокрема процедурну, об'єктно-орієнтовану та функціональну, що розширює можливості проєктування програмних систем і дозволяє обирати найбільш доцільний підхід залежно від складності та призначення програмного продукту. Це робить мову JavaScript універсальним

інструментом для реалізації як невеликих застосунків, так і складних багаторівневих інформаційних систем [14], [15].

Таким чином, JavaScript є не лише технологічною основою веброзробки, а й потужною, гнучкою та перспективною мовою програмування, яка відповідає сучасним вимогам до створення ефективного, масштабованого та зручного програмного забезпечення [14], [15].

Доцільність вибору мови програмування JavaScript для розробки програмного забезпечення для керування особистими записами зумовлюється її універсальністю, широкою екосистемою та здатністю забезпечувати створення сучасних, масштабованих і кросплатформних програмних рішень. У сучасних умовах мова програмування JavaScript є однією з базових мов розробки, що активно використовується як у клієнтській, так і в серверній частинах програмних систем, що відповідає вимогам до інтегрованих інформаційних продуктів [14], [15].

Використання JavaScript у поєднанні з вебтехнологіями дає змогу створювати програмне забезпечення для керування особистими записами, доступне через браузер без необхідності встановлення додаткового програмного забезпечення. Це забезпечує незалежність від операційної системи та спрощує доступ користувачів до власних даних з різних пристроїв. Крім того, застосування таких платформ, як Node.js, дозволяє реалізувати серверну логіку на тій самій мові, що і клієнтську частину, що знижує складність розробки та супроводу програмного продукту [14], [15].

Важливою перевагою JavaScript є наявність великої кількості бібліотек і фреймворків, які значно прискорюють процес розробки інтерфейсу користувача та обробки даних. Використання сучасних фреймворків сприяє створенню зручних, інтуїтивно зрозумілих і адаптивних інтерфейсів, що є критично важливим для програмного забезпечення, орієнтованого на щоденне використання. Розвинена екосистема також полегшує реалізацію функцій пошуку, фільтрації, синхронізації та збереження нотаток [14], [15].

Окремо слід відзначити можливості масштабування та інтеграції, які надає JavaScript. Завдяки підтримці REST-сервісів, роботи з базами даних і хмарними платформами, програмне забезпечення для керування особистими записами може бути легко розширене та адаптоване до зростання кількості користувачів і обсягів даних. Це робить JavaScript доцільним вибором для створення як простих персональних додатків, так і повнофункціональних корпоративних систем [14], [15].

Таким чином, вибір мови програмування JavaScript для розробки програмного забезпечення для керування особистими записами є обґрунтованим з огляду на її кросплатформність, універсальність, розвинену екосистему та відповідність сучасним вимогам до створення ефективних інформаційних систем [14], [15].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для керування особистими записами наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для керування особистими записами

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	Java	C#
Кросплатформність	+	+	+—
Ефективність веборієнтованої розробки	+	+	+—
Підтримка клієнтської та серверної частин однією мовою	+	—	—
Швидкість створення інтерфейсу користувача	+	+—	+
Підтримка асинхронної обробки даних	+	+—	+—

Отже, для реалізації програмного забезпечення для керування особистими записами обрано мову програмування JavaScript, яка активно використовується як у клієнтській, так і в серверній частинах програмних систем, що відповідає вимогам до інтегрованих інформаційних продуктів. Крім того, використання JavaScript дає змогу створювати програмне забезпечення для керування особистими записами, доступне через веббраузер без необхідності встановлення додаткового програмного забезпечення. Це забезпечує незалежність від операційної системи та спрощує доступ користувачів до власних даних з різних пристроїв.

2.2 Вибір програмної бібліотеки для створення програмного забезпечення для керування особистими записами

В якості програмної бібліотеки для створення інтерфейсу користувача програмного забезпечення для керування особистими записами було обрано бібліотеку React [16], [17].

React являє собою відкриту JavaScript-бібліотеку для побудови користувацьких інтерфейсів, орієнтовану насамперед на розробку односторінкових вебзастосунків. Вона була створена для ефективного відображення та оновлення інтерфейсу на основі стану даних і ґрунтується на компонентному підході, відповідно до якого інтерфейс користувача розглядається як сукупність незалежних, повторно використовуваних компонентів. Така архітектура спрощує проєктування складних інтерфейсів і підвищує керованість програмного коду [16], [17].

React є програмною бібліотекою, яка використовує декларативний підхід до опису інтерфейсу користувача, за якого розробник визначає, яким має бути стан інтерфейсу, а не як саме його необхідно оновлювати. Це зменшує кількість помилок, пов'язаних із ручним керуванням елементами інтерфейсу, та спрощує логіку застосунку [16], [17].

Важливою характеристикою React є односпрямований потік даних, який забезпечує передбачуваність поведінки інтерфейсу та полегшує налагодження програмного коду. Завдяки чітко визначеним правилам передачі стану між компонентами зменшується ризик виникнення прихованих залежностей, що є критичним для складних інтерактивних систем. Такий підхід сприяє підвищенню стабільності програмного забезпечення та спрощує його тестування [16], [17].

Бібліотека React підтримує повторне використання компонентів, що дозволяє створювати бібліотеки власних UI-елементів і забезпечувати єдність стилю інтерфейсу у межах усього програмного продукту. Це є особливо важливим для програмного забезпечення, орієнтованого на тривале використання та подальший розвиток, оскільки знижує витрати часу на реалізацію нових функцій і підтримку існуючого коду [16], [17].

Компонентна структура React полегшує модульне тестування окремих частин інтерфейсу, що підвищує надійність системи та сприяє своєчасному виявленню помилок. Наявність інструментів для аналізу стану застосунку та продуктивності додатково покращує процес розробки й оптимізації [16], [17].

Таким чином, React являє собою не лише як інструмент для створення інтерфейсів, а як комплексне технологічне рішення, що підтримує принципи сучасної програмної інженерії, сприяє підвищенню якості програмного коду та забезпечує ефективну реалізацію масштабованих і зручних у використанні користувацьких інтерфейсів [16], [17].

Доцільність використання React для розробки програмного забезпечення для керування особистими записами обумовлюється вимогами до високої інтерактивності, зручності та швидкої реакції інтерфейсу користувача. У процесі роботи з нотатками відбувається часте оновлення даних, зокрема створення, редагування, видалення та фільтрація записів, що потребує ефективного механізму оновлення інтерфейсу без повного перезавантаження сторінки. React реалізує цю можливість за рахунок

використання віртуального DOM, який оптимізує процес відображення змін і підвищує продуктивність застосунку [16], [17].

Бібліотека React забезпечує високу продуктивність, зручність розробки, масштабованість та відповідність сучасним вимогам до створення інтерактивних користувацьких інтерфейсів [16], [17].

Важливою перевагою React є чітке розділення логіки та представлення, що сприяє підвищенню читабельності коду та спрощує його супровід. Компонентна структура дозволяє легко масштабувати програмне забезпечення, поступово додаючи новий функціонал без суттєвого ускладнення загальної архітектури системи. Це є особливо важливим для програмного забезпечення для керування особистими записами, яке з часом може розширюватися за рахунок нових можливостей [16], [17].

Обґрунтування вибору програмної бібліотеки для створення програмного забезпечення наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору програмної бібліотеки для створення програмного забезпечення для керування особистими записами

Критерій порівняння програмних бібліотек	Програмні бібліотеки		
	React	Vue.js	Svelte
Масштабованість інтерфейсу	+	+—	+—
Простота супроводу великих проєктів	+	+—	+—
Гнучкість архітектурних рішень	+	+	+—
Розвиненість екосистеми та бібліотек	+	+—	—
Продуктивність при частих оновленнях даних	+	+	+—

Крім того, React має розвинену екосистему інструментів і бібліотек, що спрощують реалізацію типових функцій, зокрема маршрутизації, керування станом, взаємодії з серверною частиною та інтеграції з хмарними сервісами. Активна спільнота та широка підтримка забезпечують стабільність технології й доступність якісної документації, що зменшує ризики під час розробки та впровадження програмного продукту [16], [17].

Таким чином, для створення програмного забезпечення для керування особистими записами обрано бібліотеку React, оскільки ця бібліотека забезпечує високу продуктивність, зручність розробки, масштабованість та відповідність сучасним вимогам до створення інтерактивних користувацьких інтерфейсів. Крім того, React має розвинену екосистему інструментів і бібліотек, що спрощують реалізацію типових функцій, зокрема маршрутизації, керування станом, взаємодії з серверною частиною та інтеграції з хмарними сервісами. Активна спільнота та широка підтримка забезпечують стабільність технології й доступність якісної документації, що зменшує ризики під час розробки та впровадження програмного продукту.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для керування особистими записами обрано мову програмування JavaScript, яка активно використовується як у клієнтській, так і в серверній частинах програмних систем, що відповідає вимогам до інтегрованих інформаційних продуктів. Крім того, використання JavaScript дає змогу створювати програмне забезпечення для керування особистими записами, доступне через веббраузер без необхідності встановлення додаткового програмного забезпечення. Це забезпечує незалежність від операційної системи та спрощує доступ користувачів до власних даних з різних пристроїв.

Для створення програмного забезпечення для керування особистими записами обрано бібліотеку React, оскільки ця бібліотека забезпечує високу

продуктивність, зручність розробки, масштабованість та відповідність сучасним вимогам до створення інтерактивних користувацьких інтерфейсів. Крім того, React має розвинену екосистему інструментів і бібліотек, що спрощують реалізацію типових функцій, зокрема маршрутизації, керування станом, взаємодії з серверною частиною та інтеграції з хмарними сервісами.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для керування особистими записами

Структуру програмного забезпечення для керування особистими записами наведено на рис. 3.1.



Рисунок 3.1 – Структура програмного забезпечення
для керування особистими записами

Структура програмного забезпечення для керування особистими записами сформована на основі модульного підходу та принципу розмежування відповідальності між окремими частинами системи. Запропонована архітектура орієнтована на чітке відокремлення логіки обробки даних від представлення інтерфейсу користувача, що забезпечує гнучкість, масштабованість і зручність супроводу програмного продукту. Модульна організація програмного забезпечення водночас узгоджується з логікою побудови інтерфейсу, оскільки кожен функціональний блок системи відповідає певній групі екранів або елементів взаємодії з користувачем.

Ключову роль у структурі відіграє модуль ініціалізації та глобального керування застосунком, який виступає точкою входу до системи. Цей модуль відповідає за первинний запуск програмного забезпечення, виконання першого рендерингу інтерфейсу, завантаження базових ресурсів, конфігураційних параметрів і початкових даних. У його межах здійснюється підготовка середовища виконання, встановлення глобальних залежностей та ініціалізація контекстів, необхідних для подальшої роботи всіх інших модулів.

Модуль навігації та маршрутизації визначає логіку переходів між різними частинами програмного забезпечення. Він забезпечує динамічне завантаження сторінок відповідно до дій користувача та керує розподілом програмного коду на окремі пакети ресурсів. Завдяки цьому досягається оптимізація продуктивності, оскільки завантажуються лише ті компоненти, які є необхідними у конкретний момент часу. Даний модуль тісно пов'язаний з глобальним контекстом застосунку та координує взаємодію між функціональними частинами системи.

Модуль авторизації та керування доступом реалізує процеси реєстрації, входу та виходу користувачів із системи. У його межах обробляються облікові дані, контролюється стан автентифікації та визначаються права доступу до інших функціональних можливостей програмного забезпечення. Робота цього модуля є передумовою для активізації більшості функцій, пов'язаних із безпосереднім створенням і редагуванням нотаток.

Центральним з точки зору функціонального призначення є модуль керування особистими записами, який відповідає за створення, редагування, видалення та перегляд інформаційних записів. У його межах реалізуються операції над змістом нотаток, їх структуризація та підготовка до відображення в інтерфейсі користувача. Робота даного модуля активується лише після успішної авторизації та безпосередньо залежить від стану користувацьких даних.

Для забезпечення впорядкування інформації до структури програмного забезпечення включено модуль тегів і категоризації, який дозволяє здійснювати логічне групування нотаток. Він відповідає за створення, редагування та застосування тегів, а також за фільтрацію та пошук записів за заданими ознаками. Такий підхід значно підвищує зручність роботи з великими обсягами інформації та сприяє ефективному управлінню знаннями.

Важливу роль у взаємодії між модулями відіграє модуль керування станом даних, реалізований на основі механізмів Context API. У його межах створюються окремі контекстні менеджери, кожен з яких належить до конкретного функціонального модуля і зберігає відповідні дані разом із правилами їх обробки. Завдяки такій організації забезпечується швидкий обмін інформацією між компонентами, зменшується кількість зайвих перерендерингів і підвищується загальна продуктивність вебзастосунку. З огляду на відсутність власної серверної частини, у цьому модулі забезпечується взаємодія зі сторонніми сервісами. Цей модуль інкапсулює логіку ініціалізації зовнішніх ресурсів, обробки запитів до бази даних та виконання операцій збереження й отримання інформації.

Таким чином, запропонована структура програмного забезпечення для керування особистими записами, що складається з шести взаємопов'язаних модулів, забезпечує логічну цілісність системи, чітке розмежування функцій та ефективну організацію обробки даних. Такий підхід створює передумови для стабільної роботи застосунку, його подальшого розвитку та адаптації до зростаючих вимог користувачів.

3.2 Функціонування програмного забезпечення для керування особистими записами

Функціонування програмного забезпечення для керування особистими записами подамо за допомогою схеми, зображеної на рис. 3.2.

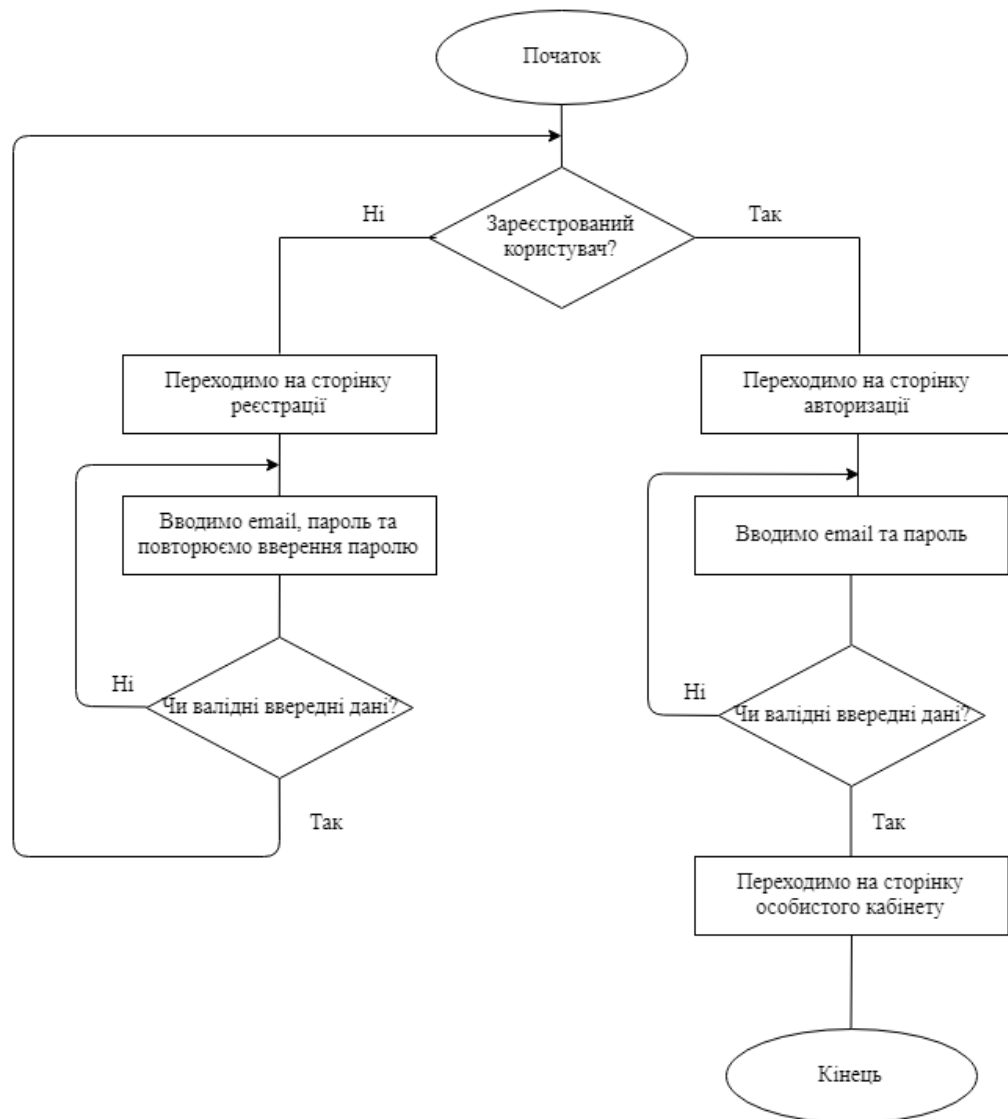


Рисунок 3.2 – Схема функціонування програмного забезпечення для керування особистими записами

Функціонування програмного забезпечення для керування особистими записами організовано за принципом послідовної взаємодії користувача з

вебзастосунком через стандартне браузерне середовище. Доступ до системи здійснюється шляхом переходу за відповідною вебадресою, яка є точкою входу до застосунку. На етапі розробки та тестування програмного продукту використовується окреме середовище з власною адресою, що дозволяє перевіряти працездатність функціоналу без впливу на кінцевих користувачів.

Після завантаження вебзастосунку користувач потрапляє на головну сторінку, яка виконує інформаційну та ознайомчу функцію. На цій сторінці представлено загальне уявлення про можливості сервісу, його призначення та основні переваги. Вона слугує своєрідною візитною карткою програмного забезпечення та доступна для перегляду без попередньої реєстрації, проте не надає доступу до основного функціоналу системи.

Для повноцінного використання сервісу користувачу необхідно пройти процедуру автентифікації або створити новий обліковий запис. Процеси реєстрації та авторизації передбачають введення персональних даних, які проходять перевірку на коректність і повноту. Механізми валідації реалізовані таким чином, щоб унеможливити надсилання некоректно заповнених форм, зокрема шляхом блокування кнопок підтвердження за відсутності обов'язкових даних. Такий підхід знижує ймовірність помилок і підвищує надійність взаємодії з системою. Доступ до внутрішніх розділів застосунку суворо обмежений, тому неавторизований користувач не може перейти до функцій створення або перегляду нотаток.

Після успішної авторизації відбувається автоматичне перенаправлення користувача до основної робочої області, де зосереджений повний функціонал програмного забезпечення. Водночас сторінки головного екрану, входу та реєстрації стають недоступними для взаємодії, що забезпечує логічну цілісність сценаріїв роботи та підвищує рівень безпеки. Подальша робота користувача відбувається в межах персонального простору, прив'язаного до його облікового запису.

Створення нової нотатки здійснюється через активацію відповідного елемента керування, після чого користувачу пропонується форма для введення

текстової інформації. За потреби нотатці може бути присвоєно тег і візуальне оформлення у вигляді кольору, що спрощує подальшу навігацію та візуальне розпізнавання записів. На етапі заповнення форми користувач має можливість скасувати дію без збереження даних.

Після підтвердження створення нотатка автоматично додається до загального списку записів користувача. У випадку, якщо жоден тег не було обрано, система застосовує стандартне позначення, що забезпечує коректне відображення запису в загальній структурі даних. Окремий функціональний блок дозволяє створювати власні теги та здійснювати сортування нотаток за тематичними групами, що особливо актуально при роботі з великим обсягом інформації.

Навігація між нотатками реалізована через перемикання між панелями тегів, що дає змогу швидко фільтрувати записи та зосереджуватися на потрібній категорії. Кожна окрема нотатка містить елементи керування, які забезпечують доступ до операцій редагування та видалення. Завдяки цьому користувач може в будь-який момент змінити вміст запису або повністю вилучити його з системи.

Таким чином, функціонування програмного забезпечення для керування особистими записами базується на чітко визначених сценаріях взаємодії, які поєднують захищений доступ, простоту користування та ефективну організацію інформації. Така логіка роботи забезпечує комфортне щоденне використання сервісу та відповідає сучасним вимогам до веборієнтованих програмних систем.

3.3 Розробка бази даних програмного забезпечення для керування особистими записами

База даних програмного забезпечення для керування особистими записами побудована з використанням нереляційного підходу до зберігання інформації, що відповідає сучасним вимогам до гнучкості та масштабованості

веборієнтованих застосунків. У якості сховища даних обрано NoSQL-рішення, структура якого ґрунтується на моделі типу «ключ–значення». Така модель дозволяє організувати швидкий доступ до інформації за унікальними ідентифікаторами без необхідності складних запитів і жорстких схем, характерних для реляційних баз даних.

Графічне подання структури даних програмного забезпечення для керування особистими записами наведено на рис. 3.3.

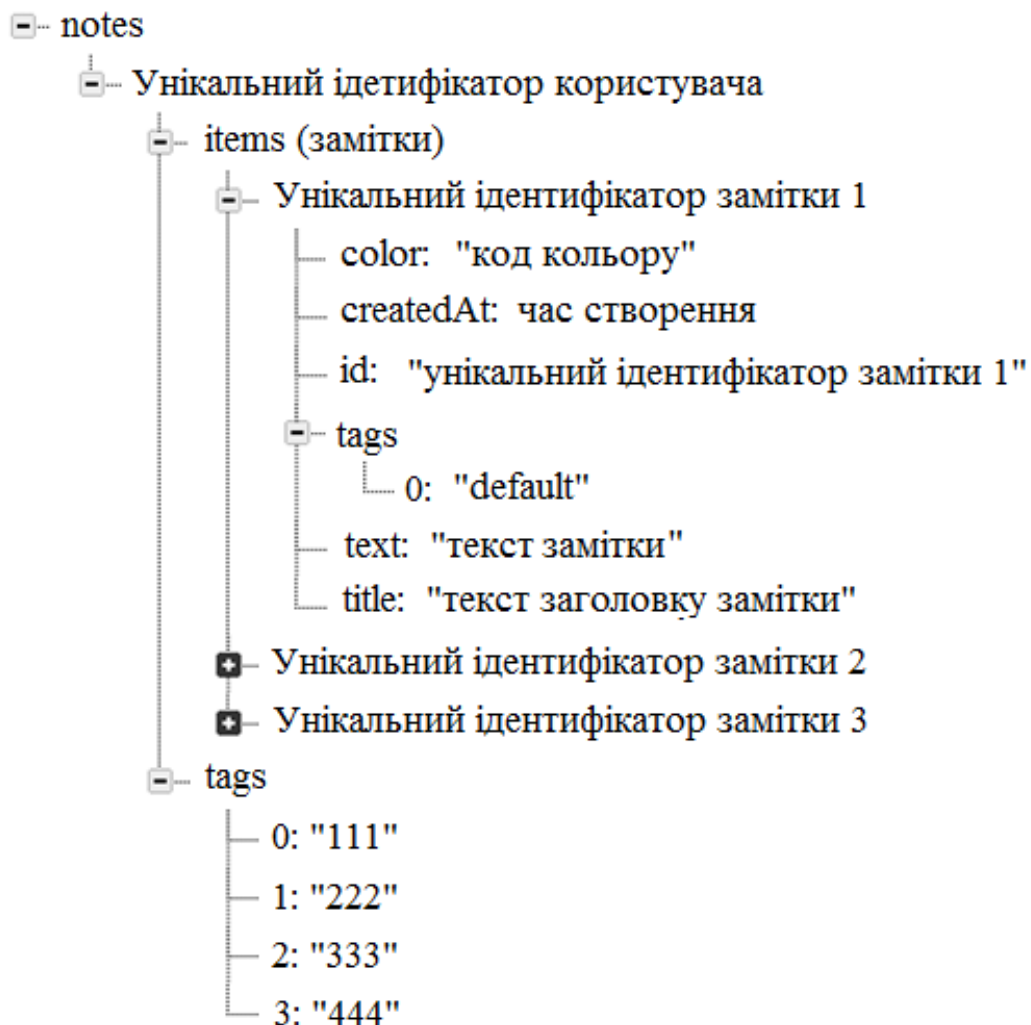


Рисунок 3.3 – Графічне подання структури даних програмного забезпечення для керування особистими записами

Центральним елементом структури виступає логічний розділ, що відповідає за збереження нотаток користувачів. У межах цього розділу всі

записи згруповані за унікальними ідентифікаторами облікових записів, що забезпечує ізоляцію даних різних користувачів та підвищує рівень безпеки. Для кожного користувача формується окремий набір записів, який містить колекцію нотаток, ідентифікованих власними унікальними ключами. Такий підхід дозволяє ефективно працювати з персональними даними та спрощує їх подальшу обробку.

Кожна окрема нотатка зберігається у вигляді об'єкта значень, що об'єднує всі необхідні атрибути для коректного відображення та редагування інформації. У структурі цього об'єкта передбачені поля для збереження заголовка запису, основного текстового вмісту, унікального ідентифікатора, а також часової мітки, яка фіксує момент створення нотатки. Додатково зберігається інформація про візуальне оформлення, зокрема обраний колір, що використовується для полегшення навігації та візуального групування записів у користувацькому інтерфейсі.

Окрему частину структури кожної нотатки становить вкладений об'єкт, відповідальний за зберігання тегів. У ньому міститься перелік значень, які відображають логічні позначки, присвоєні конкретному запису. Така організація дозволяє здійснювати гнучке сортування та фільтрацію нотаток без необхідності дублювання даних або створення додаткових таблиць, що є характерною перевагою NoSQL-підходу.

Паралельно з даними нотаток у базі даних передбачено окремий логічний розділ для збереження тегів, доступних у меню користувача. У цьому розділі кожен елемент представлений у вигляді поля, що містить назву відповідного тегу. Така структура забезпечує централізоване керування переліком тегів і спрощує їх використання під час створення або редагування нотаток.

Узагальнюючи, база даних програмного забезпечення для керування особистими записами характеризується простою, але водночас гнучкою структурою, що ґрунтується на моделі «ключ–значення». Обраний підхід дозволяє ефективно організувати зберігання персональних даних

користувачів, забезпечити швидкий доступ до нотаток і тегів, а також створити основу для подальшого розширення функціональних можливостей системи без суттєвих змін у структурі сховища.

3.4 Проєктування інтерфейсу програмного забезпечення для керування особистими записами

Інтерфейс основної частини застосунку спроектовано з урахуванням принципів зручності та інтуїтивності. Всі елементи керування розміщені таким чином, щоб виконання типових дій вимагало мінімальної кількості кроків.

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для керування особистими записами виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для керування особистими записами враховані функціональні вимоги до програми:

- підтримка можливості відображення особистих записів користувача у вигляді нотаток;
- можливість персоналізації роботи програми (відображення нотаток) шляхом підтримки механізмів авторизації та реєстрації;
- можливість групування особистих записів користувача за категоріями;
- можливість створення нотаток та додавання їх до відповідних категорій;
- підтримка можливостей редагування та видалення вже створених нотаток.

Інтерфейс головної сторінки програмного забезпечення для керування особистими записами наведено на рис. 3.4. Інтерфейс форми авторизації наведено на рис. 3.5.

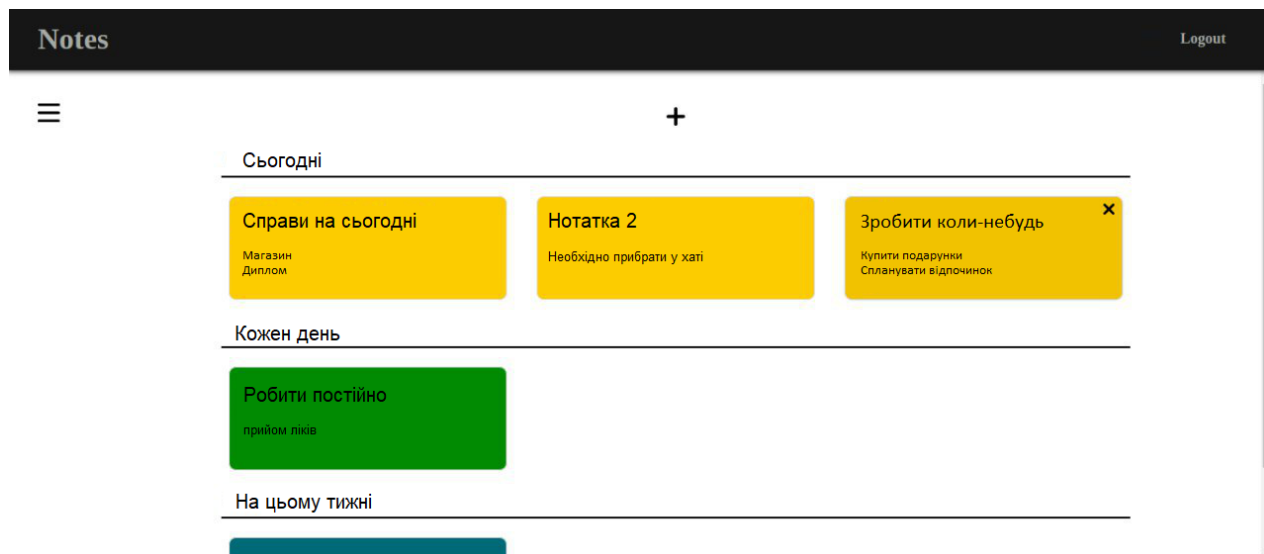


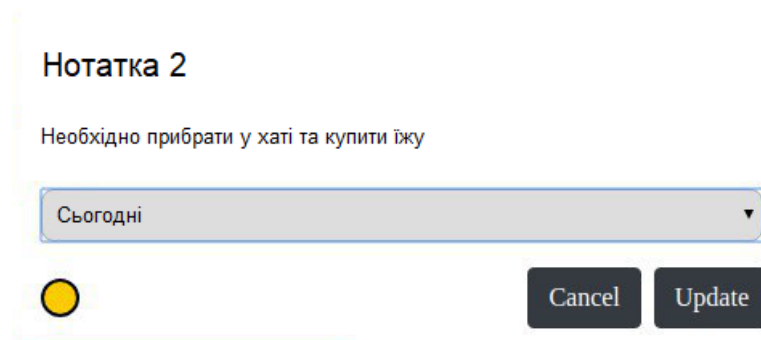
Рисунок 3.4 – Інтерфейс головної форми програмного забезпечення для керування особистими записами

Рисунок 3.5 – Інтерфейс форми авторизації

Форму створення нового запису наведено на рис. 3.6.

Рисунок 3.6 – Форма створення нового запису

Форму редагування особистого запису наведено на рис. 3.7.



Нотатка 2

Необхідно прибрати у хаті та купити їжу

Сьогодні

Cancel Update

Рисунок 3.7 – Форма редагування особистого запису

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для керування особистими записами.

Запропоновано структуру програмного забезпечення для керування особистими записами на основі модульного підходу та принципу розмежування відповідальності між окремими частинами системи. Визначено, що основними модулями програми є: модуль ініціалізації та глобального керування застосунком, модуль навігації та маршрутизації, модуль авторизації та керування доступом, модуль керування особистими записами, модуль тегів і категоризації, модуль керування станом даних. Запропонована архітектура орієнтована на чітке відокремлення логіки обробки даних від представлення інтерфейсу користувача, що забезпечує гнучкість, масштабованість і зручність супроводу програмного продукту. Модульна організація програмного забезпечення водночас узгоджується з логікою побудови інтерфейсу, оскільки кожен функціональний блок системи відповідає певній групі екранів або елементів взаємодії з користувачем.

Описано функціонування програмного забезпечення для керування особистими записами. Визначено, що функціонування програми базується на чітко визначених сценаріях взаємодії, які поєднують захищений доступ,

простоту користування та ефективну організацію інформації. Функціонування програми організовано за принципом послідовної взаємодії користувача з вебзастосунком через стандартне браузерне середовище. Доступ до системи здійснюється шляхом переходу за відповідною вебадресою, яка є точкою входу до застосунку. Така логіка роботи забезпечує комфортне щоденне використання сервісу та відповідає сучасним вимогам до веборієнтованих програмних систем.

Розроблено базу даних програмного забезпечення для керування особистими записами, яка використовує нереляційний підхід до зберігання інформації, що відповідає сучасним вимогам до гнучкості та масштабованості веборієнтованих застосунків. У якості сховища даних обрано NoSQL-рішення, структура якого ґрунтується на моделі типу «ключ–значення». Така модель дозволяє організувати швидкий доступ до інформації за унікальними ідентифікаторами без необхідності складних запитів і жорстких схем, характерних для реляційних баз даних.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для керування особистими записами.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для керування особистими записами. Програма написана на мові JavaScript, яка активно використовується як у клієнтській, так і в серверній частинах програмних систем, що відповідає вимогам до інтегрованих інформаційних продуктів. В якості програмної бібліотеки для створення інтерфейсу користувача програмного забезпечення для керування особистими записами обрано React, оскільки ця бібліотека забезпечує високу продуктивність, зручність розробки, масштабованість та відповідність сучасним вимогам до створення інтерактивних користувацьких інтерфейсів.

Програмне забезпечення для керування особистими записами забезпечує виконання таких функцій:

- підтримка можливості відображення особистих записів користувача у вигляді нотаток;
- можливість персоналізації роботи програми (відображення нотаток) шляхом підтримки механізмів авторизації та реєстрації;
- можливість групування особистих записів користувача за категоріями;
- можливість створення нотаток та додавання їх до відповідних категорій;
- підтримка можливостей редагування та видалення вже створених нотаток.

Для роботи програмного забезпечення для керування особистими записами потрібні такі технічні та програмні ресурси: центральний процесор з тактовою частотою не нижче 1,6 ГГц, що підтримує виконання багатозадачних операцій. Для коректної роботи інтерфейсу та обробки даних необхідно не менше 4 ГБ оперативної пам'яті, тоді як рекомендованим обсягом є 8 ГБ, що

забезпечує більш плавну роботу застосунку за одночасного використання інших програм.

Обсяг вільного місця на накопичувачі не є критичним, оскільки програмне забезпечення функціонує у вебсередовищі та не потребує локального встановлення. Водночас для тимчасового збереження кешованих даних браузера доцільно мати щонайменше 200 МБ вільного дискового простору. Графічна підсистема повинна підтримувати базові можливості апаратного прискорення відображення вебінтерфейсу, що є типовим для більшості сучасних інтегрованих або дискретних відеоадаптерів.

З програмної точки зору обов'язковою умовою є наявність актуальної операційної системи, здатної підтримувати сучасні веббраузери. Програмне забезпечення може функціонувати під управлінням операційних систем сімейств Windows, macOS або Linux без суттєвих обмежень. Для доступу до функціоналу необхідний сучасний веббраузер, зокрема Google Chrome, Mozilla Firefox, Microsoft Edge або Safari, з підтримкою стандартів HTML5, CSS3 та JavaScript (ECMAScript 6 і вище).

Окремою вимогою є стабільне підключення до мережі Інтернет, оскільки обмін даними, автентифікація користувачів та збереження нотаток здійснюються із залученням зовнішніх сервісів.

Функціональні характеристики розробленого програмного забезпечення для керування особистими записами наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для керування особистими записами наведено у додатку Б.

4.2 Характеристики програми для керування особистими записами

Програмне забезпечення для керування особистими записами характеризується орієнтацією на ефективну організацію інформації та підтримку повсякденної інтелектуальної діяльності користувачів. Його

функціонування спрямоване на спрощення процесів створення, збереження, упорядкування та подальшого використання текстових записів у цифровому середовищі. Особливістю такого програмного продукту є поєднання простоти використання з достатньо гнучкими можливостями налаштування, що дозволяє адаптувати роботу системи до різних сценаріїв застосування.

Характерною рисою програмного забезпечення є інтуїтивно зрозумілий користувацький інтерфейс, який забезпечує швидке освоєння основних функцій без потреби у спеціальній підготовці. Логіка взаємодії з системою побудована таким чином, щоб мінімізувати кількість дій, необхідних для виконання типових операцій, зокрема додавання або редагування нотаток. Це сприяє підвищенню продуктивності роботи та зменшенню когнітивного навантаження на користувача.

Важливою характеристикою є підтримка структуризації даних, що реалізується через механізми категоризації та логічного групування записів. Завдяки цьому забезпечується зручна навігація в межах інформаційного простору та спрощується пошук необхідних відомостей. Програмне забезпечення орієнтоване на роботу з динамічними даними, що дозволяє оперативно відображати зміни без порушення цілісності інформації.

Значну увагу приділено надійності збереження даних і контролю доступу. Робота з нотатками здійснюється в межах захищеного середовища, що гарантує розмежування інформації між різними користувачами та збереження конфіденційності персональних даних. Архітектура програмного забезпечення передбачає чітке відокремлення логіки обробки даних від представлення інтерфейсу, що позитивно впливає на стабільність і зручність супроводу системи.

Крім того, програмне забезпечення для керування особистими записами відзначається масштабованістю та готовністю до розширення функціональних можливостей. Закладені архітектурні рішення дозволяють без істотних змін інтегрувати додаткові сервіси або вдосконалювати наявний функціонал. Сукупність зазначених характеристик визначає програмне

забезпечення як сучасний, зручний та ефективний інструмент для роботи з інформацією в умовах цифрового середовища.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для керування особистими записами на клієнтській стороні необхідно перейти на відповідну вебсторінку.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для керування особистими записами є інформація, яку вводить користувач у процесі взаємодії з системою, зокрема текстовий вміст нотаток, їхні заголовки, позначки для логічного групування, вибрані параметри оформлення та дії, пов'язані з редагуванням або видаленням записів.

Вихідними даними програмного забезпечення для керування особистими записами є структуроване відображення збережених записів у користувацькому інтерфейсі, результати виконання пошукових запитів, оновлений стан нотаток після їх редагування та впорядковані набори даних відповідно до вибраних параметрів перегляду.

4.3.3 Повідомлення

Користувач програмного забезпечення для керування особистими записами може отримати такі повідомлення: повідомлення про успішне збереження або оновлення нотаток, попередження про некоректно введені дані чи спроби виконання недоступних дій, а також повідомлення про помилки, пов'язані з обробкою даних або втратою з'єднання з системою збереження. Окрім цього, система може інформувати про результати виконаних операцій,

підтверджувати критичні дії та повідомляти про обмеження, що виникають під час роботи з програмним забезпеченням.

4.4 Виконання програмного забезпечення для керування особистими записами

Після запуску програмного забезпечення для керування особистими записами користувачу відображається основна сторінка (рис. 4.1).

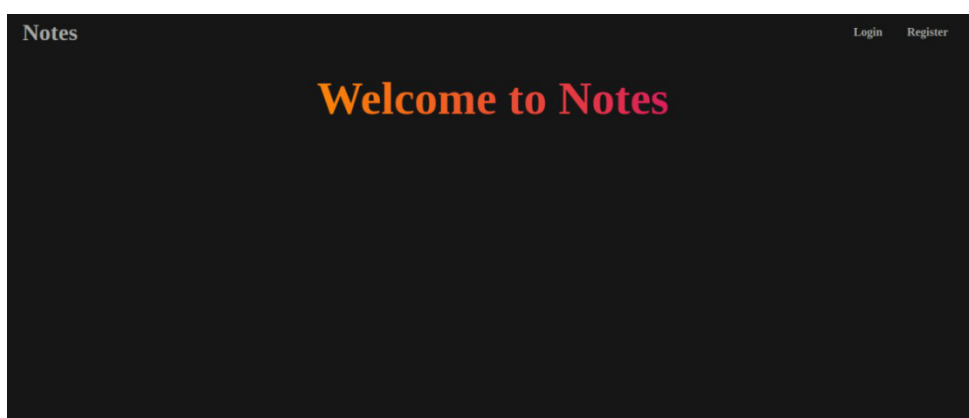


Рисунок 4.1 – Інтерфейс основної сторінки програмного забезпечення для керування особистими записами

Для авторизації потрібно перейти за посиланням Login та у формі (рис. 4.2) ввести свою електронну адресу та пароль.



Рисунок 4.2 – Форма авторизації у програмі для керування особистими записами

Для першого входу до системи необхідно зареєструватися (рис. 4.3).



The registration form consists of three input fields stacked vertically. The first field is labeled 'Email address:', the second 'Password:', and the third 'Confirm password:'. Each field is a simple white rectangle with a light gray border. Below the third field is a dark gray button with the word 'Register' in white text.

Рисунок 4.3 – Форма реєстрації у програмі для керування особистими записами

Після авторизації користувач переходить на головну сторінку (рис. 4.4), де відображаються особисті записи (нотатки) користувача. Нотатки згруповані за категоріями.

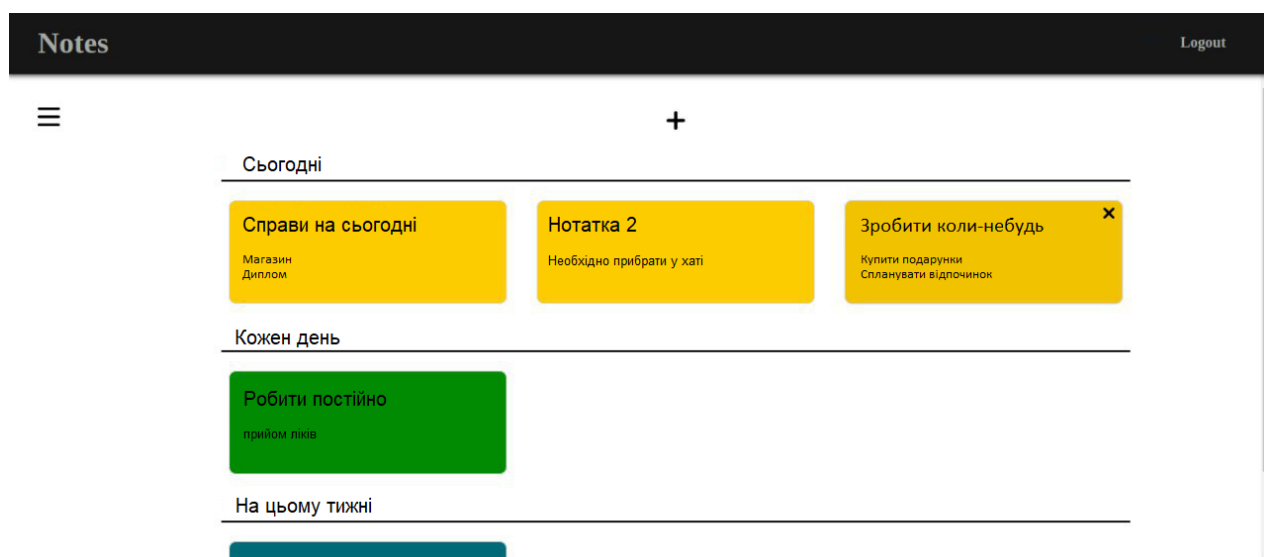


Рисунок 4.4 – Головна сторінка авторизованого користувача програми для керування особистими записами

Для створення нового особистого запису необхідно натиснути символ плюс та у відповідній формі (рис. 4.5) ввести назву нотатки, її зміст

(наприклад, що користувач хоче зробити), а також обрати з випадаючого списку категорію нотатки.

Рисунок 4.5 – Форма створення нового запису

Користувач має можливість видалення нотаток шляхом натискання символу x у правому верхньому куті відповідного запису. При подвійному натисканні на нотатці відображається форма редагування (рис. 4.6), де можливо змінити назву та зміст особистого запису, а також поміняти його категорію.

Рисунок 4.6 – Редагування особистого запису

4.5 Тестування програмного забезпечення для керування особистими записами

Виконано тестування програмного забезпечення для керування особистими записами.

Виявлено, що програма для керування особистими записами функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу керування особистими записами користувача.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою керування особистими записами користувача.

4.6 Висновки за розділом 4

Описано програмне забезпечення для керування особистими записами. Виконано тестування розробленого програмного забезпечення для керування особистими записами. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу керування особистими записами користувача.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для керування нотатками.

Визначено, що програмне забезпечення для керування особистими записами користувача призначено для створення, збереження, впорядкування та подальшого використання персональних записів у цифровому середовищі. Такі програмні засоби орієнтовані на підтримку роботи з інформацією, що виникає у процесі навчання, професійної діяльності або повсякденного життя, та забезпечують зручний перехід від фрагментарних записів до цілісної структури знань.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для керування особистими записами. Проте деякі програмні засоби для керування особистими записами можуть не забезпечувати достатньої гнучкості у налаштуванні структури збереження даних, що ускладнює адаптацію програмного продукту до індивідуальних потреб користувачів або специфіки конкретної предметної галузі. Суттєвим недоліком є також обмежені можливості інтеграції з іншими інформаційними системами та сервісами. У багатьох випадках програмне забезпечення для керування особистими записами функціонує ізольовано, що призводить до фрагментації інформації та потреби дублювання даних між різними програмними середовищами. Окрему проблему становлять питання продуктивності та масштабованості. При збільшенні обсягу збережених нотаток деякі програмні рішення демонструють зниження швидкості пошуку, завантаження або синхронізації даних. Подібні обмеження є особливо критичними для користувачів, які працюють з великими масивами текстової інформації протягом тривалого часу. Тому актуальною є розробка програмного забезпечення для керування особистими записами.

Сформульовано функціональні вимоги до програмного забезпечення для керування особистими записами.

Для реалізації програмного забезпечення для керування особистими записами обрано мову програмування JavaScript, яка активно використовується як у клієнтській, так і в серверній частинах програмних систем, що відповідає вимогам до інтегрованих інформаційних продуктів. Крім того, використання JavaScript дає змогу створювати програмне забезпечення для керування особистими записами, доступне через веббраузер без необхідності встановлення додаткового програмного забезпечення. Це забезпечує незалежність від операційної системи та спрощує доступ користувачів до власних даних з різних пристроїв.

Для створення програмного забезпечення для керування особистими записами обрано бібліотеку React, оскільки ця бібліотека забезпечує високу продуктивність, зручність розробки, масштабованість та відповідність сучасним вимогам до створення інтерактивних користувацьких інтерфейсів. Крім того, React має розвинену екосистему інструментів і бібліотек, що спрощують реалізацію типових функцій, зокрема маршрутизації, керування станом, взаємодії з серверною частиною та інтеграції з хмарними сервісами.

Розроблено програмне забезпечення для керування особистими записами.

Запропоновано структуру програмного забезпечення для керування особистими записами на основі модульного підходу та принципу розмежування відповідальності між окремими частинами системи. Визначено, що основними модулями програми є: модуль ініціалізації та глобального керування застосунком, модуль навігації та маршрутизації, модуль авторизації та керування доступом, модуль керування особистими записами, модуль тегів і категоризації, модуль керування станом даних. Запропонована архітектура орієнтована на чітке відокремлення логіки обробки даних від представлення інтерфейсу користувача, що забезпечує гнучкість, масштабованість і зручність супроводу програмного продукту. Модульна організація програмного

забезпечення водночас узгоджується з логікою побудови інтерфейсу, оскільки кожен функціональний блок системи відповідає певній групі екранів або елементів взаємодії з користувачем.

Описано функціонування програмного забезпечення для керування особистими записами. Визначено, що функціонування програми базується на чітко визначених сценаріях взаємодії, які поєднують захищений доступ, простоту користування та ефективну організацію інформації. Функціонування програми організовано за принципом послідовної взаємодії користувача з вебзастосунком через стандартне браузерне середовище. Доступ до системи здійснюється шляхом переходу за відповідною вебадресою, яка є точкою входу до застосунку. Така логіка роботи забезпечує комфортне щоденне використання сервісу та відповідає сучасним вимогам до веборієнтованих програмних систем.

Розроблено базу даних програмного забезпечення для керування особистими записами, яка використовує нереляційний підхід до зберігання інформації, що відповідає сучасним вимогам до гнучкості та масштабованості веборієнтованих застосунків. У якості сховища даних обрано NoSQL-рішення, структура якого ґрунтується на моделі типу «ключ–значення». Така модель дозволяє організувати швидкий доступ до інформації за унікальними ідентифікаторами без необхідності складних запитів і жорстких схем, характерних для реляційних баз даних.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для керування особистими записами.

Виконано тестування розробленого програмного забезпечення для керування особистими записами. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу керування особистими записами користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Personal Knowledge Management? A Complete Guide. URL: <https://betternotetaking.com/what-is-pkm/> (date of access: 23.04.2026).
2. Knowledge, Activity, and Content. A Perspective on the Note-Taking Wars. URL: https://www.reddit.com/r/NoteTaking/comments/syfche/knowledge_activity_and_content_a_perspective_on/ (date of access: 23.04.2026).
3. Features that you expect a note taking app to have. URL: https://www.reddit.com/r/ProductivityApps/comments/1qbggfh/features_that_you_expect_a_note_taking_app_to_have/ (date of access: 23.04.2026).
4. Boost Your Productivity and Organization with the Best Note Taking Apps. URL: <https://scholarly.so/blog/boost-your-productivity-and-organization-with-the-best-note-taking-apps> (date of access: 23.04.2026).
5. Best Note Taking Software. URL: <https://medium.com/@theo-james/best-note-taking-software-in-2024-bfa7a028c48c> (date of access: 23.04.2026).
6. Note-taking. URL: <https://infoepi.org/tools/notes> (date of access: 23.04.2026).
7. Best Note-Taking Software. URL: <https://www.capterra.com/note-taking-software/> (date of access: 23.04.2026).
8. The Best Note-Taking Apps. URL: <https://www.pcmag.com/picks/the-best-note-taking-apps> (date of access: 23.04.2026).
9. Best Note-Taking Software. URL: <https://www.g2.com/categories/note-taking-software> (date of access: 23.04.2026).
10. Top Note-Taking Apps to Consider. URL: <https://www.proofhub.com/articles/note-taking-apps> (date of access: 23.04.2026).
11. Fathom. URL: <https://www.fathom.ai/> (date of access: 23.04.2026).
12. Amplenote. URL: <https://www.amplenote.com/> (date of access: 23.04.2026).
13. Joplin. URL: <https://joplinapp.org/> (date of access: 23.04.2026).

14. JavaScript Guide. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide> (date of access: 23.04.2026).

15. JavaScript Introduction. URL: https://www.w3schools.com/js/js_intro.asp (date of access: 23.04.2026).

16. React resources. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/React_resources (date of access: 23.04.2026).

17. React The Library for Web and Native User Interfaces. URL: <https://react.dev/> (date of access: 23.04.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для керування особистими записами.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для керування нотатками», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 7 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для керування особистими записами користувача.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для керування особистими записами забезпечує виконання таких функцій:

- підтримка можливості відображення особистих записів користувача у вигляді нотаток;
- можливість персоналізації роботи програми (відображення нотаток) шляхом підтримки механізмів авторизації та реєстрації;
- можливість групування особистих записів користувача за категоріями;
- можливість створення нотаток та додавання їх до відповідних категорій;

– підтримка можливостей редагування та видалення вже створених нотаток.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для керування особистими записами повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

A.3.3 Вимоги до надійності

Програмне забезпечення для керування особистими записами повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для керування особистими записами необхідна наявність персонального комп'ютера.

A.3.5 Вимоги до складу та параметрів технічних засобів

Для роботи програмного забезпечення для керування особистими записами потрібні такі технічні та програмні ресурси: центральний процесор з тактовою частотою не нижче 1,6 ГГц, що підтримує виконання багатозадачних операцій. Для коректної роботи інтерфейсу та обробки даних необхідно не менше 4 ГБ оперативної пам'яті, тоді як рекомендованим обсягом є 8 ГБ, що забезпечує більш плавну роботу застосунку за одночасного використання інших програм.

Обсяг вільного місця на накопичувачі не є критичним, оскільки програмне забезпечення функціонує у вебсередовищі та не потребує локального встановлення. Водночас для тимчасового збереження кешованих даних браузера доцільно мати щонайменше 200 МБ вільного дискового простору. Графічна підсистема повинна підтримувати базові можливості апаратного прискорення відображення вебінтерфейсу, що є типовим для більшості сучасних інтегрованих або дискретних відеоадаптерів.

З програмної точки зору обов'язковою умовою є наявність актуальної операційної системи, здатної підтримувати сучасні веббраузери. Програмне забезпечення може функціонувати під управлінням операційних систем сімейств Windows, macOS або Linux без суттєвих обмежень. Для доступу до функціоналу необхідний сучасний веббраузер, зокрема Google Chrome, Mozilla Firefox, Microsoft Edge або Safari, з підтримкою стандартів HTML5, CSS3 та JavaScript (ECMAScript 6 і вище).

A.3.6 Вимоги до маркування і пакування

Програма для керування особистими записами може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для керування особистими записами».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для керування особистими записами є інформація, яку вводить користувач у процесі взаємодії з системою, зокрема текстовий вміст нотаток, їхні заголовки, позначки для логічного групування, вибрані параметри оформлення та дії, пов'язані з редагуванням або видаленням записів.

Вихідними даними програмного забезпечення для керування особистими записами є структуроване відображення збережених записів у користувацькому інтерфейсі, результати виконання пошукових запитів, оновлений стан нотаток після їх редагування та впорядковані набори даних відповідно до вибраних параметрів перегляду.

ДОДАТОК Б
Фрагмент тексту програми

```

export const strg = (() => {
  const stg = window.localStorage

  function lddt(k) {
    const d = stg.getItem(k)
    if (d === null) {
      return null
    }
    try {
      return JSON.parse(d)
    } catch (e) {
      return null
    }
  }

  function svdt(k, v) {
    const s = JSON.stringify(v)
    stg.setItem(k, s)
  }

  function rmdt(k) {
    stg.removeItem(k)
  }

  function chkstg() {
    try {
      const t = "__tst__"
      stg.setItem(t, t)
      stg.removeItem(t)
      return true
    } catch (e) {
      return false
    }
  }

  function gtll() {
    const rs = {}
    for (let i = 0; i < stg.length; i++) {
      const k = stg.key(i)
      rs[k] = lddt(k)
    }
    return rs
  }

  return {
    lddt,
    svdt,
  }
})

```

```

        rmdt,
        chkstg,
        gtll
    }
  })()

export const dtm = (() => {
  const ntk = "nts"
  const tgk = "tgs"

  function ldntrs() {
    const d = strg.lddt(ntk)
    if (!d) {
      strg.svdt(ntk, {})
      return {}
    }
    return d
  }

  function ldtgs() {
    const d = strg.lddt(tgk)
    if (!d) {
      strg.svdt(tgk, {})
      return {}
    }
    return d
  }

  function svnt(n) {
    const d = ldntrs()
    d[n.id] = n
    strg.svdt(ntk, d)
  }

  function rmnt(id) {
    const d = ldntrs()
    if (d[id]) {
      delete d[id]
      strg.svdt(ntk, d)
    }
  }

  function gtnl() {
    const d = ldntrs()
    return Object.values(d)
  }

  function svtg(nm) {
    const d = ldtgs()

```

```

        d[nm] = true
        strg.svdt(tgk, d)
    }

    function rmtg(nm) {
        const d = ldtgs()
        if (d[nm]) {
            delete d[nm]
            strg.svdt(tgk, d)
        }
    }

    function gttl() {
        const d = ldtgs()
        return Object.keys(d)
    }

    function ftrtg(tg) {
        const n = gtnl()
        return n.filter(x => x.tgs && x.tgs.includes(tg))
    }

    return {
        svnt,
        rmnt,
        gtnl,
        svtg,
        rmtg,
        gttl,
        ftrtg
    }
})()

import { dtm } from "./dtm.js"

export const ui = (() => {
    const rts = document.getElementById("rt")
    const lst = document.createElement("div")
    const frm = document.createElement("div")
    const tgl = document.createElement("div")

    function crtnpt(tp, plc) {
        const i = document.createElement("input")
        i.type = tp
        i.placeholder = plc
        return i
    }

    function crtbtn(tx) {

```

```

    const b = document.createElement("button")
    b.textContent = tx
    return b
  }

function clrnd(n) {
  while (n.firstChild) {
    n.removeChild(n.firstChild)
  }
}

function rndnt(n) {
  const c = document.createElement("div")
  const h = document.createElement("h3")
  const p = document.createElement("p")
  const b1 = crtbtn("EDIT")
  const b2 = crtbtn("DEL")

  h.textContent = n.ttl
  p.textContent = n.txt

  b2.onclick = () => {
    dtm.rmnt(n.id)
    rndll()
  }

  b1.onclick = () => {
    frmtl.value = n.ttl
    frmtx.value = n.txt
    frmtg.value = n.tgs.join(",")
    frmidx.value = n.id
  }

  c.appendChild(h)
  c.appendChild(p)
  c.appendChild(b1)
  c.appendChild(b2)
  lst.appendChild(c)
}

function rndll() {
  clrnd(lst)
  const n = dtm.gtnl()
  n.forEach(x => rndnt(x))
}

const frmtl = crtnpt("text", "TTL")
const frmtx = crtnpt("text", "TXT")
const frmtg = crtnpt("text", "TGS")

```

```

const frmId = crtnpt("hidden", "")
const sbt = crtbtn("SV")

sbt.onclick = () => {
  const id = frmId.value || Date.now().toString()
  const nt = {
    id,
    ttl: frmTtl.value,
    txt: frmTx.value,
    tgs: frmTg.value.split(",").map(x => x.trim())
  }
  dtm.svnt(nt)
  frmTtl.value = ""
  frmTx.value = ""
  frmTg.value = ""
  frmId.value = ""
  rndll()
  rndtg()
}

function rndtg() {
  clrnd(tgl)
  const t = dtm.gttl()
  t.forEach(x => {
    const b = crtbtn(x)
    b.onclick = () => {
      clrnd(lst)
      const f = dtm.ftrtg(x)
      f.forEach(n => rndnt(n))
    }
    tgl.appendChild(b)
  })
}

frm.appendChild(frmTtl)
frm.appendChild(frmTx)
frm.appendChild(frmTg)
frm.appendChild(frmId)
frm.appendChild(sbt)

rts.appendChild(frm)
rts.appendChild(tgl)
rts.appendChild(lst)

function ntui() {
  rndll()
  rndtg()
}

```

```

        return {
            ntui
        }
    })()

export function srchnt(qry) {
    const rs = []
    const nts = dtm.gtnl()
    for (let i = 0; i < nts.length; i++) {
        const n = nts[i]
        const ttl = n.ttl ? n.ttl.toLowerCase() : ""
        const txt = n.txt ? n.txt.toLowerCase() : ""
        const q = qry.toLowerCase()
        if (ttl.includes(q) || txt.includes(q)) {
            rs.push(n)
        }
    }
    return rs
}

export function srtnt(mth) {
    const nts = dtm.gtnl().slice()
    if (mth === "crt") {
        nts.sort((x, y) => (x.id > y.id ? -1 : 1))
    }
    if (mth === "ttl") {
        nts.sort((x, y) => {
            const a = x.ttl || ""
            const b = y.ttl || ""
            return a.localeCompare(b)
        })
    }
    return nts
}

export function pnnnt(id) {
    const nts = dtm.gtnl()
    for (let i = 0; i < nts.length; i++) {
        if (nts[i].id === id) {
            nts[i].pnn = true
            dtm.svnt(nts[i])
        }
    }
}

export function unpnnnt(id) {
    const nts = dtm.gtnl()
    for (let i = 0; i < nts.length; i++) {
        if (nts[i].id === id) {

```

```

        nts[i].pnn = false
        dtm.svnt(nts[i])
    }
}

export function gtpnn() {
    const rs = []
    const nts = dtm.gtnl()
    for (let i = 0; i < nts.length; i++) {
        if (nts[i].pnn) {
            rs.push(nts[i])
        }
    }
    return rs
}

export function rchvnt(id) {
    const nts = dtm.gtnl()
    for (let i = 0; i < nts.length; i++) {
        if (nts[i].id === id) {
            nts[i].rchv = true
            dtm.svnt(nts[i])
        }
    }
}

export function gtrchv() {
    const rs = []
    const nts = dtm.gtnl()
    for (let i = 0; i < nts.length; i++) {
        if (nts[i].rchv) {
            rs.push(nts[i])
        }
    }
    return rs
}

export function xprnt() {
    const nts = dtm.gtnl()
    const dt = JSON.stringify(nts)
    const blb = new Blob([dt], { type: "application/json" })
    const lnk = document.createElement("a")
    lnk.href = URL.createObjectURL(blb)
    lnk.download = "nts.json"
    lnk.click()
    URL.revokeObjectURL(lnk.href)
}

```

```

export function mprtnr(fls) {
  const rdr = new FileReader()
  rdr.onload = e => {
    try {
      const nts = JSON.parse(e.target.result)
      for (let i = 0; i < nts.length; i++) {
        if (nts[i] && nts[i].id) {
          dtm.svnt(nts[i])
        }
      }
    } catch (e) {}
  }
  rdr.readAsText(fls)
}

export function thmtgl() {
  const bd = document.body
  const cls = "drkmd"
  if (bd.classList.contains(cls)) {
    bd.classList.remove(cls)
    localStorage.setItem("thm", "lgt")
  } else {
    bd.classList.add(cls)
    localStorage.setItem("thm", "drk")
  }
}

export function ldthm() {
  const v = localStorage.getItem("thm")
  if (v === "drk") {
    document.body.classList.add("drkmd")
  }
}

export function sttsnt() {
  const nts = dtm.gtnl()
  let ttl = 0
  let pnn = 0
  let rchv = 0
  for (let i = 0; i < nts.length; i++) {
    ttl++
    if (nts[i].pnn) pnn++
    if (nts[i].rchv) rchv++
  }
  return { ttl, pnn, rchv }
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення для керування нотатками

Виконав: Іван ЯНЄВ

ст. групи КНТ-113сп

Керівник: Тетяна КОЛПАКОВА

к.т.н., доцент

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

- * Об'єкт дослідження – процес розробки програмного забезпечення для керування нотатками.
- * Предмет дослідження – програмні засоби для керування особистими записами.
- * Метою роботи є розробка програмного забезпечення керування особистими записами для зменшення часу, що витрачається користувачем на пошук і систематизацію інформації.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для керування особистими записами;
- здійснити проектування програмного забезпечення для керування особистими записами;
- створити програмне забезпечення для керування особистими записами;
- виконати тестування розробленого програмного забезпечення для керування особистими записами. 3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Fathom	Amplenote	Joplin
поєднання нотаток з управлінням завданнями користувачів	+–	+	+–
автоматизація нотаток з аудіо/відео	+	–	–
підтримка засобів розмітки тексту	–	+–	+
гнучкість зберігання даних	–	+	+
інструменти колективної роботи	+–	+–	–
простота інтерфейсу	+	+–	+–
розширення через плагіни	–	+–	+

4

Рисунок В.4 – Слайд 4

Постановка завдання

При розробці програмного забезпечення для керування особистими записами необхідно забезпечити такі функціональні вимоги:

- підтримка можливості відображення особистих записів користувача у вигляді нотаток;
- можливість персоналізації роботи програми (відображення нотаток) шляхом підтримки механізмів авторизації та реєстрації;
- можливість групування особистих записів користувача за категоріями;
- можливість створення нотаток та додавання їх до відповідних категорій;
- підтримка можливостей редагування та видалення вже створених нотаток.

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	Java	C#
Кросплатформність	+	+	+-
Ефективність веборієнтованої розробки	+	+	+-
Підтримка клієнтської та серверної частин однією мовою	+	-	-
Швидкість створення інтерфейсу користувача	+	+-	+
Підтримка асинхронної обробки даних	+	+-	+-

6

Рисунок В.6 – Слайд 6

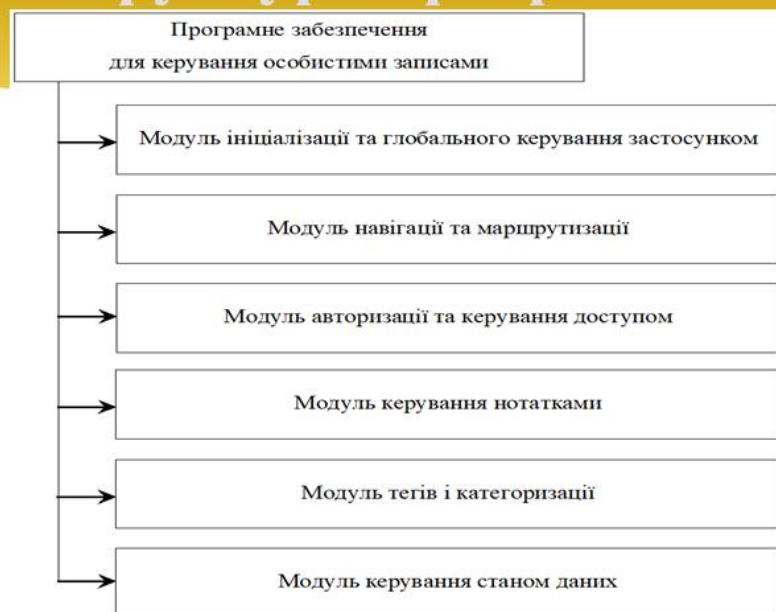
Порівняння середовищ розробки

Критерій порівняння програмних бібліотек	Програмні бібліотеки		
	React	Vue.js	Svelte
Масштабованість інтерфейсу	+	+—	+—
Простота супроводу великих проєктів	+	+—	+—
Гнучкість архітектурних рішень	+	+	+—
Розвиненість екосистеми та бібліотек	+	+—	—
Продуктивність при частих оновленнях даних	+	+	+—

7

Рисунок В.7 – Слайд 7

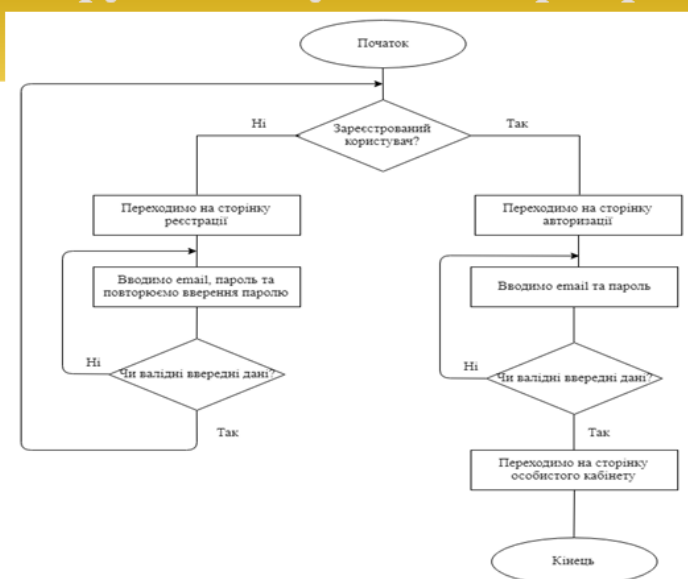
Структура програми



8

Рисунок В.8 – Слайд 8

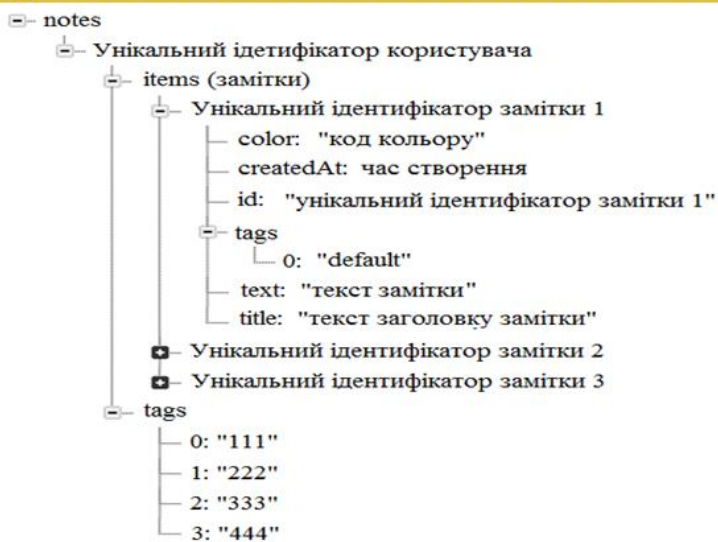
Схема функціонування програми



9

Рисунок В.9 – Слайд 9

Графічне подання структури даних ПЗ для керування особистими записами



10

Рисунок В.10 – Слайд 10

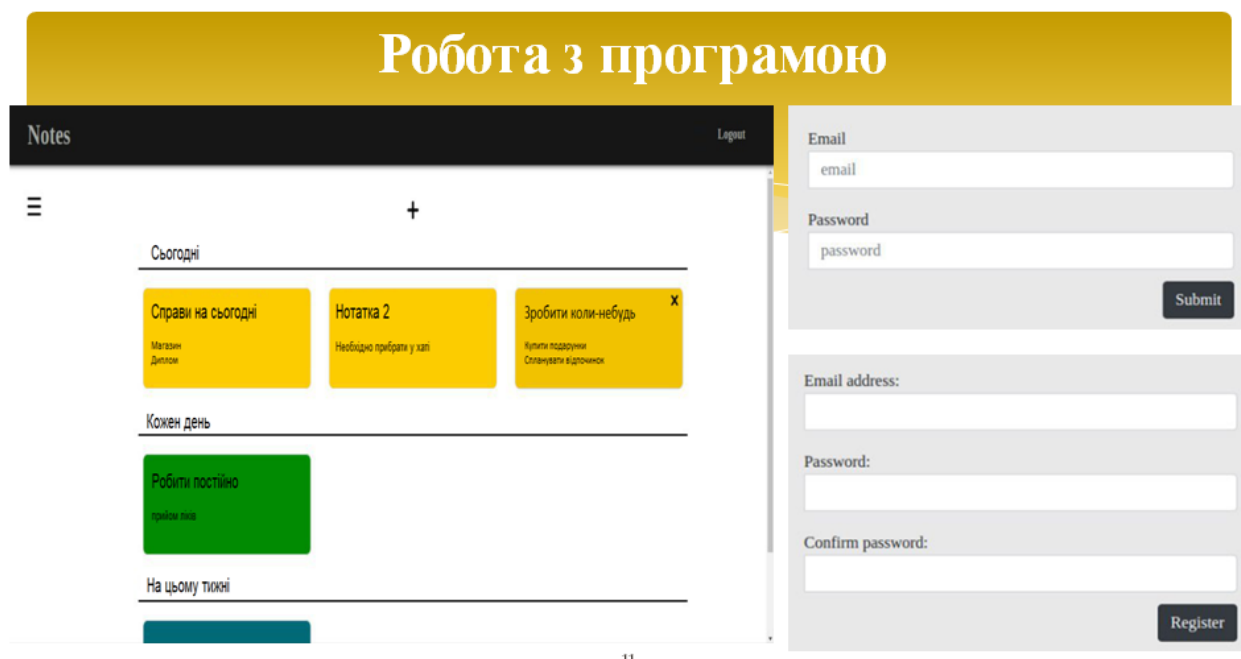


Рисунок В.11 – Слайд 11

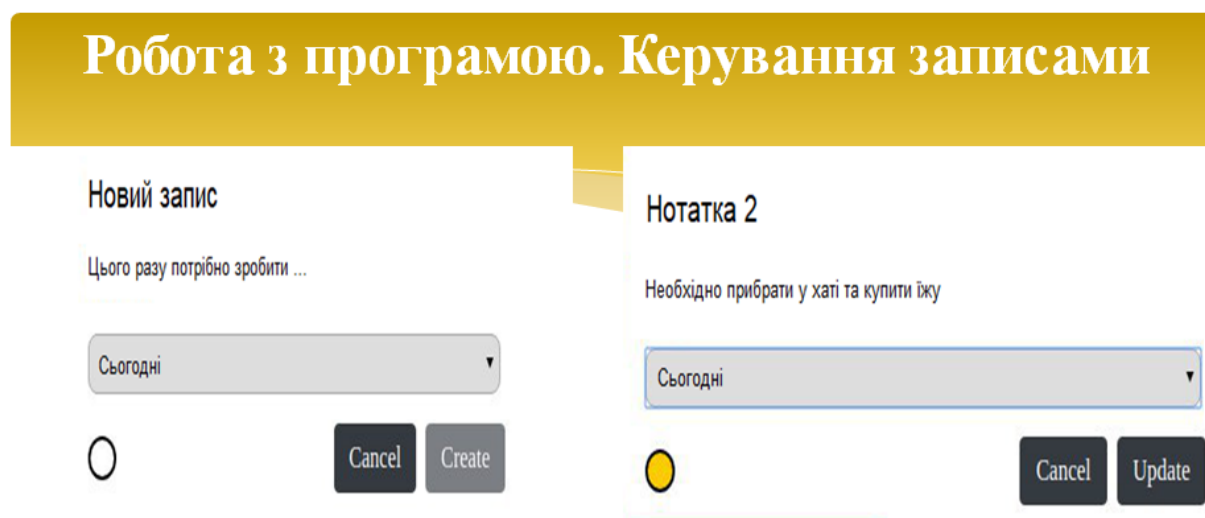


Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для керування нотатками.

Сформульовано функціональні вимоги до програмного забезпечення для керування особистими записами. Для реалізації програмного забезпечення для керування особистими записами обрано мову програмування JavaScript та бібліотеку React.

Запропоновано структуру програмного забезпечення для керування особистими записами на основі модульного підходу та принципу розмежування відповідальності між окремими частинами системи. Визначено, що основними модулями програми є: модуль ініціалізації та глобального керування застосунком, модуль навігації та маршрутизації, модуль авторизації та керування доступом, модуль керування особистими записами, модуль тегів і категоризації, модуль керування станом даних.

В результаті виконання завдання було отримано програмну систему, що відповідає всім вимогам визначеним на початку розробки та закріпленим в технічному завданні. Розроблена програмна система є зручною для використання та виконує всі необхідні функції.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

13

Рисунок В.13 – Слайд 13