Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра інтелектуальних програмних систем і технологій

Рекомендовано до захисту:
протокол засідання кафедри
№ ___ від ____.____.2025 р.
Завідувач кафедри _____
_____ _____
(підпис) (прізвище та ініціали)

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему «Розробка веб-застосунку для створення персоналізованих календарів користувачів»

Захищено на засіданні ЕК № _____
протокол № ____ від ____.____.2019 р.
Оцінка _____ / _____
Голова ЕК
_____ _____
(підпис) (прізвище та ініціали)

Виконала:
студентка 4 курсу, групи КС- 41
Спеціальності:
122 Комп’ютерні науки
(шифр і назва спеціальності підготовки)
Макарова Анастасія Іванівна
(прізвище, ім’я та по батькові, підпис)
Керівник ст. викл. Радоуцький К.Є.
(ступень, звання, прізвище та ініціали, підпис)
Рецензент к.т.н. Шеханін К.Ю.
(ступень, звання, прізвище та ініціали, підпис)

Харків – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра інтелектуальних програмних систем і технологій
Рівень вищої освіти (освітньо-кваліфікаційний рівень) Магістр (Бакалавр)
Галузь знань: Інформаційні технології
Спеціальність: Комп'ютерні науки
Освітня програма: Комп'ютені науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
_____
«____» _____20___року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

Макарової Анастасії Іванівни
(прізвище, ім'я, по батькові студента)

1. Тема роботи Веб-додаток для створення персоналізованих календарів

керівник роботи Радоуцький Костянтин Євгенович, старший викладач,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "16" квітня 2025року № 4101-5/96

2. Строк подання студентом роботи 2 червня 2025 року

3. Перелік питань, які потрібно розробити:
   1. Створення веб-додатку на основі React
   2. Реалізація функцій додавання, редагування та видалення подій
   3. Реалізація авторизації та реєстрації користувачів з використанням LocalStorage.
   4. Розробка можливості перегляду подій на обрану дату.
   5. Інтеграція календаря з можливістю вибору подій за місяць.
   6. Оцінка продуктивності та тестування веб-додатку.
   7. Порівняння з існуючими рішеннями в контексті безпеки і функціональності.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Розробка плану роботи та збору вимог до проекту	Перший тиждень
2	Вибір технологій для створення веб-додатку	Другий тиждень
3	Налаштування середовища	Третій тиждень
4	Реалізація базового функціоналу: реєстрація, авторизація, додавання подій	Четвертий тиждень
5	Розробка інтерфейсу користувача та інтеграція календаря	П’ятий тиждень
6	Тестування додатку, виправлення помилок, оптимізація	Шостий тиждень
7	Оформлення дипломної роботи, підготовка до захисту	Сьомий тиждень

5. Дата видачі завдання 10 жовтня 2024 року

Студент Макарова А.І. _____
ініціали, прізвище підпис

Керівник роботи Радоуцький К.Є. _____
ініціали, прізвище підпис

АНОТАЦІЯ

Загальний обсяг основного тексту становить 53 сторінок, включаючи 26 рисунків. Додатково в роботі наведено 3 додатки. Перелік використаних джерел налічує 15 найменувань. Робота оформлена відповідно до вимог до кваліфікаційних робіт.

Дипломна робота присвячена розробці веб-застосунку для створення персоналізованих календарів. У роботі розглянуто проблеми організації подій, аналіз існуючих календарних сервісів, їх функціональні можливості, обмеження та перспективи персоналізованого підходу до планування часу.

Особливу увагу приділено проектуванню архітектури системи, створенню зручного інтерфейсу користувача з використанням бібліотеки React, зберіганню даних у LocalStorage або через Firebase, реалізації авторизації та налаштуванню індивідуального вигляду календаря. У роботі описано розробку функцій додавання, редагування, видалення подій, встановлення нагадувань і синхронізації з іншими календарними сервісами.

Актуальність дослідження зумовлена потребою у гнучких інструментах планування часу, які відповідають індивідуальним вимогам користувачів. Запропонований веб-додаток дозволяє ефективно організовувати події, зберігати їх у хмарі або локально, забезпечувати мобільний доступ та підвищувати загальний рівень зручності та продуктивності.

Ключові слова: ВЕБ-ДОДАТОК, КАЛЕНДАР, РЕАКТ, FIREBASE, ПЕРСОНАЛІЗАЦІЯ, ПЛАНУВАННЯ ПОДІЙ, АВТОРИЗАЦІЯ, ІНТЕРФЕЙС, LOCALSTORAGE, МОДЕЛЮВАННЯ.

ABSTRACT

The total volume of the main text is 53 pages, including 26 figures. Additionally, the thesis includes 5 appendices. The list of references includes 15 sources. The work is formatted in accordance with the requirements for qualification papers.

The bachelor's thesis is dedicated to the development of a web application for creating personalized calendars. The paper examines the challenges of event organization, analyzes existing calendar services, their functional capabilities, limitations, and the prospects of a personalized approach to time planning.

Special attention is paid to designing the system architecture, developing a user-friendly interface using the React library, data storage via LocalStorage or Firebase, user authentication, and customization of the calendar appearance. The thesis describes the implementation of functionality for adding, editing, and deleting events, setting reminders, and synchronizing with other calendar services.

The relevance of the research is due to the growing need for flexible time management tools tailored to individual user requirements. The proposed web application allows users to efficiently organize events, store them in the cloud or locally, ensure mobile access, and enhance overall convenience and productivity.

Keywords: WEB APPLICATION, CALENDAR, REACT, FIREBASE, PERSONALIZATION, EVENT PLANNING, AUTHENTICATION, INTERFACE, LOCALSTORAGE, MODELING.

ЗМІСТ

ВСТУП

1) Актуальність теми

Зараз дедалі більше людей і компаній переходять на електронні інструменти для планування свого часу. Однак стандартні календарі, які встановлені в більшості пристроїв та додатків, не завжди підходять під індивідуальні вимоги користувачів. Тому часто виникає потреба створити власний, персоналізований календар, який можна гнучко налаштувати відповідно до конкретних завдань і особливостей користувача.

Персоналізовані календарі можуть бути дуже корисними як у звичайному житті, так і для бізнесу. Вони дозволяють зручно планувати різні події, синхронізувати їх з іншими сервісами, налаштовувати зручні нагадування, а також вести історію своїх активностей. У рамках цього дослідження ставиться завдання створити зручний веб-додаток, який допоможе користувачам легко створювати та налаштовувати власні календарі під свої потреби, роблячи процес планування часу простим і максимально комфортним.

Отже, створення веб-додатку для персоналізованих календарів є актуальним завданням, яке відповідає сучасним вимогам цифрових технологій. Такий додаток стане зручним і корисним рішенням для широкого кола користувачів — як для особистого використання, так і для ефективного управління часом у компаніях.

2) Мета та завдання дослідження

Метою даного дослідження є розробка веб-додатку для створення персоналізованих календарів, який дозволяє користувачам зручно планувати події, налаштовувати їх відповідно до власних вимог та забезпечити синхронізацію з іншими календарними системами.

Для досягнення цієї мети необхідно вирішити наступні завдання:

1) Проаналізувати існуючі рішення для створення та використання календарів.

2) Розробити загальну архітектуру веб-додатку для персоналізованого календаря.

3) Реалізувати функціонал додавання, редагування та видалення подій.

4) Розробити інтерфейс для налаштування зовнішнього вигляду календаря.

5) Реалізувати систему збереження даних про події з використанням локального хмарного хостингу або Firebase.

6) Створити механізм нагадувань і сповіщень про майбутні події.

7) Провести тестування додатку та оцінити його ефективність і зручність використання.

3) Об'єкт і предмет дослідження

Об'єктом цього дослідження є веб-додаток для створення персоналізованих календарів, який допомагає користувачам ефективно планувати й керувати своїми подіями. Ця система включає не тільки базові функції, як-от додавання чи видалення подій, але й дозволяє індивідуально налаштувати календар під свої конкретні потреби. Крім того, веб-додаток може синхронізувати інформацію з іншими сервісами, що забезпечує зручність у роботі, надійне збереження даних у хмарі й доступ до них із будь-якого пристрою.

Предметом дослідження є методи та інструменти для реалізації функцій персоналізованих календарів. До таких методів можна віднести:

1) Розробка функцій додавання, редагування та видалення подій: створення інтерфейсів для введення подій, їх редагування та видалення, а також створення логіки зберігання подій у базі даних або через хмарні сервіси.

2) Проектування та розробка користувацького інтерфейсу (UI/UX): створення зручного і інтуїтивно зрозумілого інтерфейсу для взаємодії з календарем, що дозволяє користувачеві швидко і без помилок виконувати необхідні дії, такі як додавання та редагування подій.

Зберігання та синхронізація даних:

Розробка зручних механізмів для зберігання інформації про події — як локально, так і в хмарних сервісах. Важливою є можливість легкої синхронізації даних з іншими популярними календарями, наприклад Google Calendar чи Microsoft Outlook.

Безпека персональних даних:

Використання методів шифрування, автентифікації та інших сучасних рішень, які гарантують, що персональні дані користувачів будуть надійно захищені.

Інтеграція з іншими календарями:

Забезпечення простого зв'язку з популярними календарними платформами. Це дозволяє користувачам легко імпортувати та експортувати свої події й синхронізувати їх з іншими сервісами для ефективнішого планування часу.

Отже, предмет дослідження включає не тільки технічну розробку веб-додатку для персоналізованих календарів, але й вирішення завдань, пов’язаних з інтеграцією додатку з іншими популярними сервісами, а також забезпечення максимальної зручності та надійного захисту даних користувачів.

3) Практична значущість роботи

Практична цінність цієї роботи полягає в тому, що персоналізований календар допомагає користувачам автоматизувати процес планування часу з урахуванням індивідуальних потреб. Створений веб-додаток дозволить легко організовувати події, налаштовувати зручні нагадування та отримувати своєчасні сповіщення. Це сприятиме ефективнішому плануванню та підвищенню продуктивності користувачів.

Розроблений веб-додаток може стати корисним інструментом для широкого кола людей: студентів, працівників, приватних осіб, а також невеликих команд, які цінують простий, інтуїтивно зрозумілий і гнучкий у налаштуваннях календар. Додаток дозволяє:

1) Індивідуалізація подій: користувач може додавати події з можливістю зазначення не тільки дати, а й часу, що дає можливість планувати свій день більш детально.

2) Редагування та видалення подій: система дозволяє змінювати події, що забезпечує високу гнучкість користування.

3) Вибір подій за місяцями: можливість перегляду подій на обраний місяць дозволяє краще організовувати свої плани, що актуально для довгострокового планування.

4) Робота з користувацькими акаунтами: можливість авторизації та реєстрації дозволяє зберігати персональні дані та календар подій в безпечному вигляді, що захищає приватність користувача.

5) Мобільність: доступ до календаря через браузер дозволяє користуватися додатком на будь-якому пристрої з доступом до інтернету, що підвищує мобільність і зручність для користувачів.

6) Безпека даних: завдяки зберіганню даних у хмарі через використання Firebase або інших платформ, дані користувачів будуть надійно захищені, що підвищує рівень безпеки.

Розроблений додаток має великий потенціал для використання в різних сферах діяльності, таких як навчання, особисте планування, управління робочими завданнями, а також у бізнесі для планування зустрічей, конференцій та інших заходів. Завдяки своїй простоті та ефективності, веб-календар стане корисним інструментом для широкой аудиторії користувачів.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ УПРАВЛІННЯ КАЛЕНДАРЯМИ

1.1 Огляд сучасних веб-додатків для роботи з календарями

У наш час безліч інструментів для організації часу, які активно використовуються як в особистому житті, так і в професійній діяльності. Календарі стали невід'ємною частиною сучасного управління часом і завданнями. Зокрема, завдяки інтеграції з іншими сервісами та простоті використання, вони стали необхідним інструментом для багатьох людей.

На сьогодні існує кілька популярних веб-додатків, які дозволяють організовувати час. Серед найбільш відомих можна виділити наступні:

1) Google Calendar (рисунок 1.1)

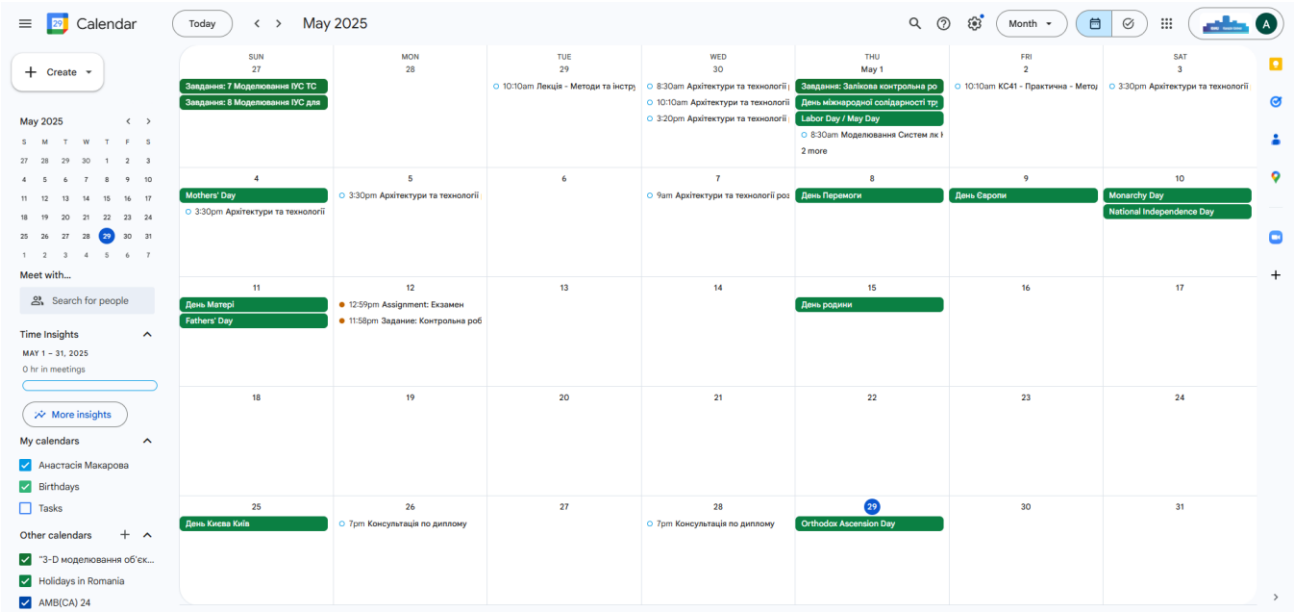

Рисунок 1.1 – Інтерфейс Google Calendar

Google Calendar є одним з найбільш використовуваних календарів на ринку. Він дозволяє створювати події, додавати нагадування і підключати інші календарі, що полегшує планування не тільки особистих подій, а й робочих зустрічей. Інтеграція з іншими продуктами Google, такими як Gmail, робить цей інструмент дуже зручним.

Особливості:

- Підтримка кількох календарів.
- Автоматичне створення подій з електронної пошти Gmail.
- Інтеграція з мобільними додатками на iOS та Android.

Недоліки:

- Відсутність великих можливостей для персоналізації інтерфейсу.
- Проблеми з використанням, якщо користувач не є частиною екосистеми Google.

2) Microsoft Outlook Calendar (рисунок 1.2)

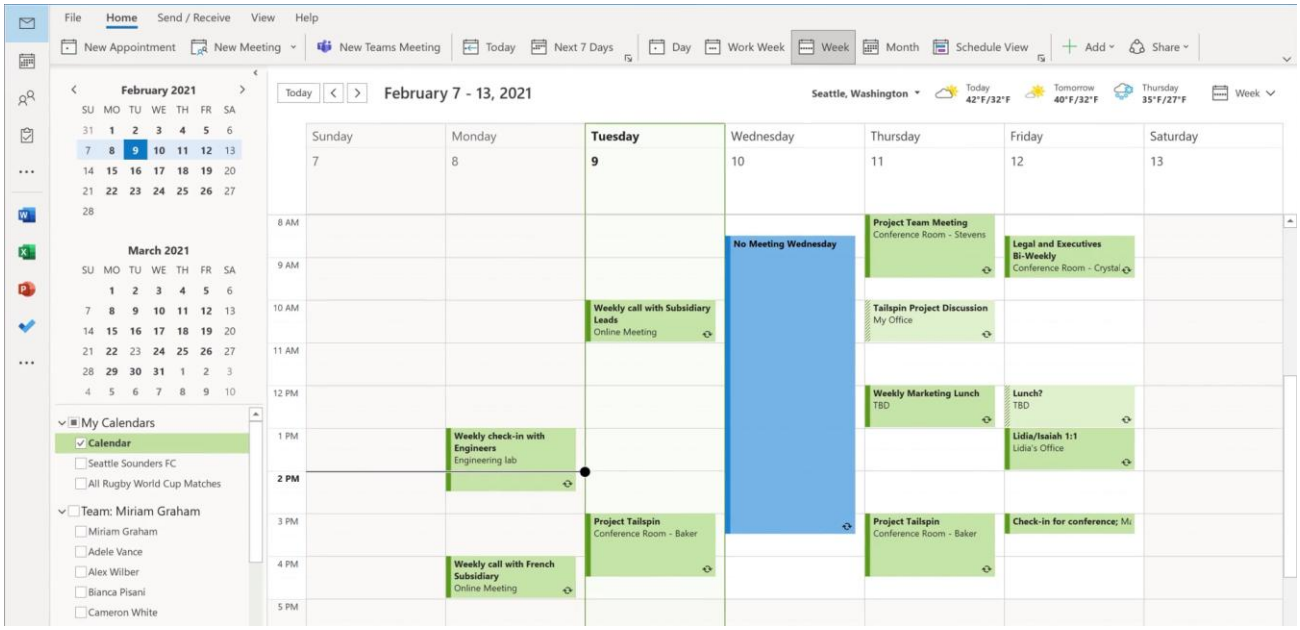

Рисунок 1.2 – Інтерфейс Microsoft Outlook Calendar

Відома платформа для роботи з електронною поштою, яка включає календар для планування подій. Outlook Calendar активно використовують як для особистих, так і для робочих потреб. Інтеграція з іншими продуктами Microsoft робить його зручним для підприємств.

Особливості:

- Глибока інтеграція з Microsoft Outlook.
- Легке планування зустрічей і синхронізація з іншими користувачами.
- Підтримка мобільних пристроїв.

Недоліки:

- Для повноцінного використання необхідно бути частиною екосистеми Microsoft.
- Інтерфейс не такий інтуїтивно зрозумілий, як у деяких інших продуктів.

3) Apple Calendar (iCloud Calendar) (рисунок 1.3)

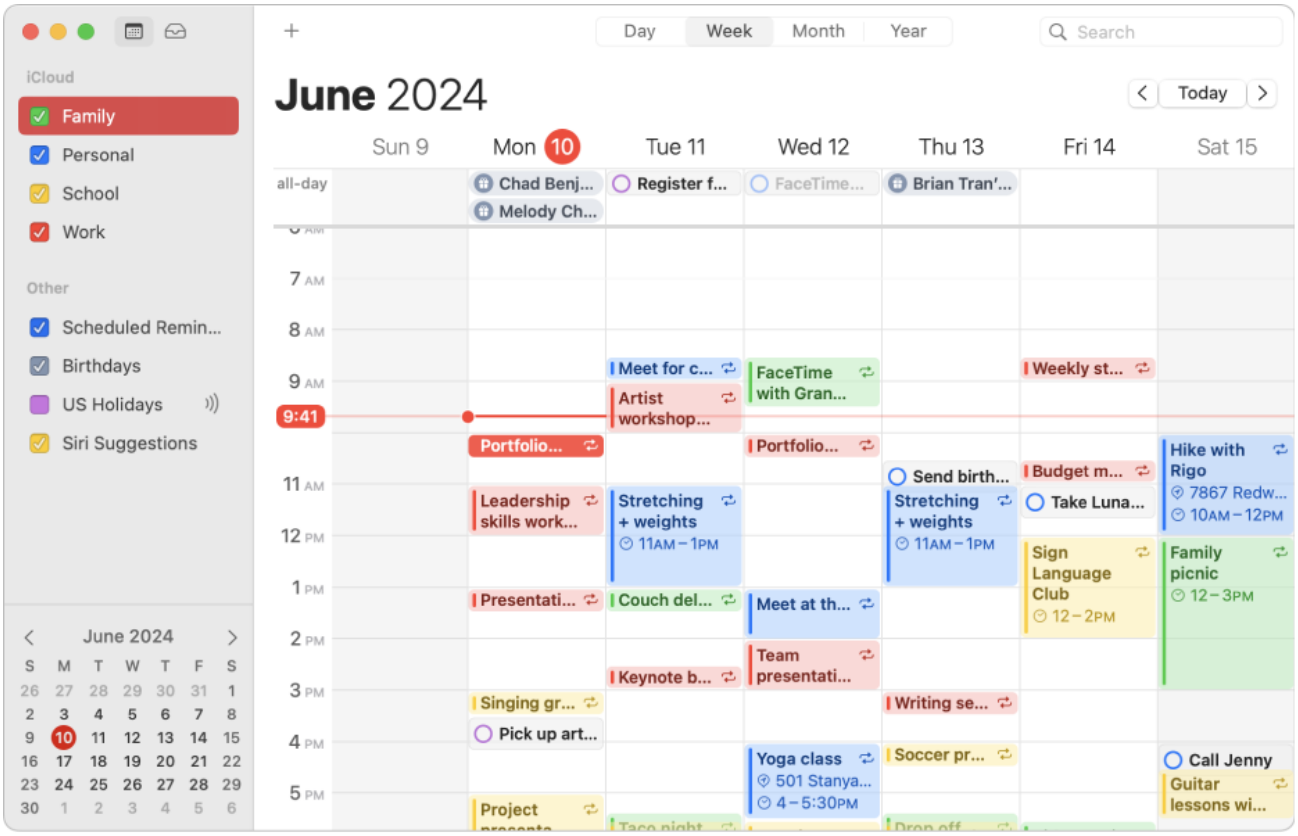

Рисунок 1.3 – Інтерфейс Apple Calendar

Календар від компанії Apple дозволяє синхронізувати події між всіма пристроями, що працюють на macOS та iOS. Він також має можливість інтеграції з іншими календарями, що дозволяє користувачам мати єдиний інтерфейс для управління подіями.

Особливості:

- Інтеграція з іншими продуктами Apple.
- Легка синхронізація подій між пристроями через iCloud.

Недоліки:

- Обмеження для користувачів, які не використовують продукцію Apple.
- Недостатньо гнучких можливостей для персоналізації.

4) Notion (рисунок 1.4)

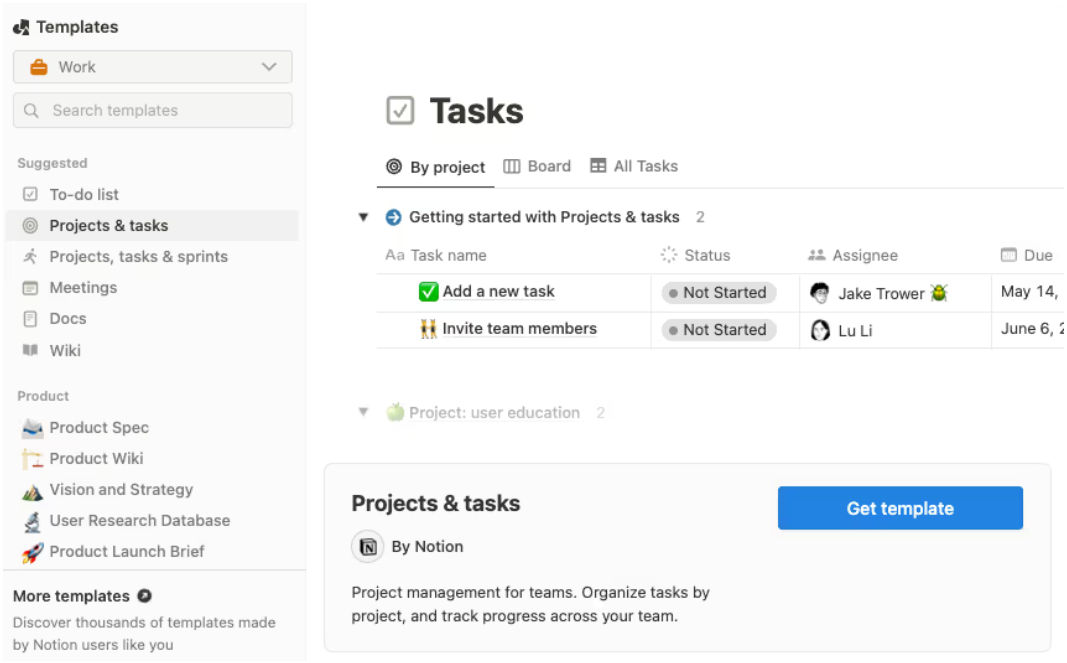

Рисунок 1.4 – Інтерфейс Notion

Notion — це багатофункціональний інструмент, який дозволяє не тільки організовувати події, а й створювати завдання, нотатки і таблиці. Хоча Notion більше відомий як інструмент для ведення нотаток, його функції календаря дозволяють зручно управляти подіями.

Особливості:

- Підтримка створення персоналізованих шаблонів для завдань і подій.
- Можливість візуалізувати завдання в календарному форматі.
- Зручний інтерфейс для організації роботи.

Недоліки:

- Довгий процес освоєння, оскільки це не лише календар, а й більш складний інструмент.

- Для деяких користувачів може бути занадто універсальним інструментом.

5) Trello (рисунок 1.5)

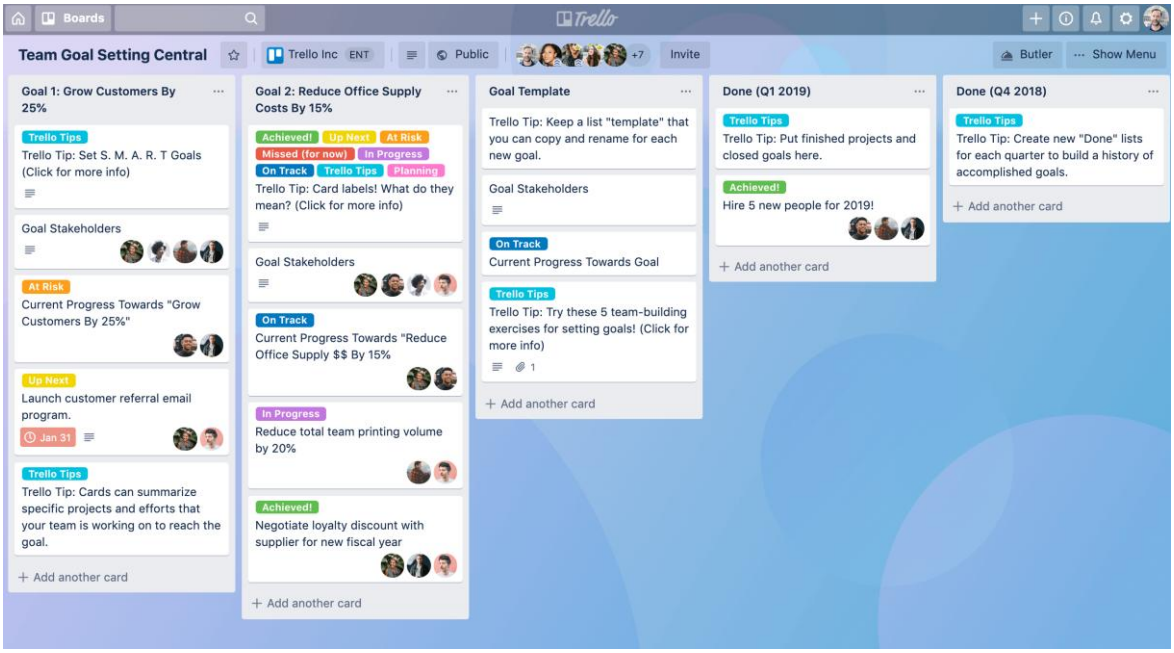

Рисунок 1. 5 – Інтерфейс Trello

Trello — це інструмент для управління проектами, який дозволяє організовувати завдання за допомогою карток. Одна з функцій — можливість додавати події до календаря, що дозволяє ефективно управляти дедлайнами та зустрічами.

Особливості:

- Візуальне управління завданнями і подіями.
- Легка інтеграція з іншими інструментами через Power-Ups.
- Підтримка групової роботи.

Недоліки:

- Відсутність спеціалізованого календаря для особистих подій.
- Не всі функції доступні безкоштовно.

Усі розглянуті календарні додатки мають свої сильні сторони та недоліки, які залежать від конкретних потреб користувачів. Так, наприклад,

Google Calendar і Microsoft Outlook надають хороші базові можливості для планування й управління часом, тоді як Notion і Trello краще інтегруються з іншими сервісами й робочими інструментами. Водночас жоден із цих сервісів не дозволяє повною мірою налаштувати календар під свої специфічні потреби, і саме тому виникає необхідність створити гнучкий, персоналізований додаток для більш зручної організації подій.

У наступних розділах роботи буде описано розробку саме такого персоналізованого календаря, який поєднає високу функціональність із простотою використання і зможе максимально точно задовольнити індивідуальні вимоги користувачів.

1.2 Основні функції календарних додатків

Календарні додатки, що широко використовуються в наш час, мають різноманітні функції, що дозволяють користувачам організовувати своєчасне виконання завдань і ефективно планувати своє робоче і особисте життя.

1.2.1 Створення подій і завдань

Найголовніша функція будь-якого календаря — це можливість створювати та керувати подіями чи завданнями. Події можуть бути як одноразовими (наприклад, важлива зустріч або співбесіда), так і такими, що повторюються регулярно (наприклад, щотижневі наради або щоденні завдання). Під час створення події зазвичай потрібно вказати її назву, дату й час, місце, а також короткий опис або коментар.

Деякі сучасні календарі також дозволяють прикріплювати до подій різноманітні документи або корисні посилання, які можуть знадобитися під час зустрічі або виконання завдання.

1.2.2 Нагадування та сповіщення

Більшість календарів дозволяють налаштовувати нагадування, які повідомляють користувачеві про подію заздалегідь — наприклад, за 10

хвилин, за годину чи навіть за день. Окрім цього, сучасні додатки часто підтримують налаштування кількох нагадувань для однієї події, а також пропонують різні типи сповіщень: пуш-сповіщення на телефон, повідомлення на електронну пошту або SMS.

1.2.3 Синхронізація з іншими сервісами та пристроями

Інтеграція календаря з іншими сервісами та пристроями суттєво спрощує та покращує його використання. Наприклад, завдяки синхронізації з поштою (як-от Gmail) можна автоматично створювати події прямо з електронних листів. Інтеграція з робочими інструментами, такими як Trello, Asana чи Slack, дозволяє миттєво оновлювати календар новими завданнями й планами. Крім того, синхронізація між різними пристроями гарантує, що користувач матиме зручний доступ до свого розкладу в будь-який момент і з будь-якого пристрою.

1.2.4 Спільний доступ до календаря

Сучасні календарні сервіси надають можливість користувачам ділитися власними подіями з окремими особами або цілими групами. Такий спільний доступ дозволяє надсилати запрошення через електронну пошту чи генеровані посилання, встановлювати різні рівні прав для перегляду чи редагування, а також переглядати розклади інших учасників для зручного узгодження спільних планів.

1.2.5 Повторювані події

Функція створення повторюваних подій є незамінною для користувачів, чий розклад включає регулярні зустрічі або завдання з певною періодичністю. Більшість онлайн-календарів дозволяє налаштовувати цикл повторення — щоденний, щотижневий, щомісячний (за датою або днем тижня), щорічний чи за індивідуальним інтервалом, наприклад, раз на два тижні або щомісяця.

Такий підхід спрощує організацію розкладу, зменшує потребу у ручному додаванні однотипних подій і забезпечує їх автоматичне оновлення.

1.2.6 Персоналізація інтерфейсу

Більшість сучасних календарних сервісів дозволяють адаптувати інтерфейс під індивідуальні вподобання користувача — змінювати кольорову палітру, формат відображення подій і спосіб подання часу. Така гнучкість не лише покращує зручність роботи з додатком, а й сприяє ефективнішій організації власного розкладу. Водночас дедалі актуальнішою стає потреба в персоналізованих рішеннях, що враховують специфічні запити користувачів. У наступних розділах буде детальніше розглянуто підходи до розробки таких інструментів із акцентом на їхню адаптивність та продуктивність.

1.3 Недоліки існуючих рішень та мотивація для створення персоналізованого календаря

Незважаючи на те, що існує безліч готових календарних додатків, які пропонують користувачам широкий набір функцій для планування та організації подій, ці рішення мають ряд суттєвих обмежень, які часто не дозволяють задовольнити потреби всіх користувачів. Серед основних недоліків існуючих рішень можна виділити:

1.3.1 Обмежена налаштовуваність

Сучасні календарі часто пропонують широкі можливості налаштування — користувач може змінити кольори інтерфейсу, обрати зручний вигляд подій або формат часу. Це робить роботу з додатком більш зручною та дозволяє краще впорядковувати щоденні справи. Водночас зростає інтерес до рішень, що можуть враховувати унікальні потреби кожного. Далі в роботі буде описано підхід до створення таких інструментів, зосереджений на їхній гнучкості та практичності.

1.3.2 Відсутність інтеграції з іншими системами

Багато популярних календарних додатків мають певні обмеження щодо інтеграції з іншими сервісами. Зокрема, не завжди зручно підключати календар до проєктних чи бізнес-платформ — таких як Trello, Asana чи Slack. Наприклад, Google Calendar хоч і підтримує базову інтеграцію, однак її функціональність часто виявляється недостатньою для ефективної роботи в командних середовищах. Крім того, синхронізація між пристроями іноді відбувається з затримками або з втратою частини можливостей на окремих платформах, що створює незручності для користувачів, які працюють із кількох пристроїв.

1.3.3 Безпека та конфіденційність даних

Питання безпеки та захисту приватної інформації залишається одним із найважливіших при використанні календарних сервісів. Збереження особистих даних у хмарних сховищах пов’язане з ризиком їхнього витоку чи несанкціонованого використання. У багатьох наявних рішеннях питання кіберзахисту не отримують належної уваги: інформація може зберігатися без шифрування або без чіткої системи контролю доступу, що викликає обґрунтоване занепокоєння серед користувачів.

1.3.4 Відсутність гнучкості для специфічних потреб

Чимало календарних сервісів не адаптовані до нестандартних сценаріїв використання. Наприклад, для фрілансерів або тих, хто працює за гнучким графіком, важливо мати можливість створювати події з нетиповими параметрами й групувати їх не за днями, а за проєктами. Однак більшість стандартних рішень обмежені типовим форматом ділових зустрічей і не дозволяють додавати події зі змінною структурою чи особливими налаштуваннями.

1.3.5 Висока залежність від інтернет-з’єднання

Деякі популярні календарні додатки, зокрема Google Calendar, потребують постійного підключення до Інтернету для перегляду або редагування подій. У випадках, коли мережеве з’єднання нестабільне чи повністю відсутнє, це створює труднощі для користувачів, які бажають мати безперебійний доступ до свого розкладу незалежно від наявності Інтернету.

1.3.6 Відсутність можливості створювати повністю персоналізовані календарі

Ураховуючи вказані обмеження, постає потреба у створенні персоналізованих календарних рішень, здатних забезпечити більшу гнучкість у плануванні подій, керуванні завданнями та інтеграції з іншими сервісами. Такі системи краще відповідатимуть індивідуальним запитам користувачів і водночас гарантуватимуть належний рівень безпеки й захисту особистих даних.

Використання сучасних технологій у розробці таких рішень відкриває можливість створення зручних і зрозумілих інтерфейсів, що працюють на різних платформах — від смартфонів до комп’ютерів. Це значно спростить процес планування, допоможе ефективніше керувати завданнями та позитивно вплине на загальний рівень продуктивності користувачів.

1.4 Вибір технологічного стеку для розробки (React, Firebase, Node.js тощо)

Під час створення веб-додатка для персоналізованих календарів було використано низку ключових технологій, що забезпечують зручну розробку, надійну масштабованість і високий рівень безпеки. Основою для побудови користувацького інтерфейсу стала бібліотека React, а для обробки та зберігання даних було застосовано хмарну платформу Firebase.

React — це популярна бібліотека для розробки динамічних інтерфейсів, яка надає можливість працювати з окремими компонентами, кожен з яких

може мати власний стан. Завдяки декларативному підходу до опису елементів інтерфейсу, розробка стає менш складною та більш зрозумілою. Велика екосистема і наявність численних готових рішень дозволяють швидко створювати адаптивні та масштабовані веб-додатки. У цьому проєкті React було використано для реалізації форми додавання подій, а також для побудови інтерфейсу календаря з відображенням запланованих подій.

Firebase було обрано як основну платформу для зберігання та обробки даних, оскільки цей сервіс надає широкий набір інструментів для реалізації функціоналу в реальному часі, автентифікації та безпеки. У рамках проєкту використовувався Firestore — масштабована NoSQL база даних, придатна для збереження подій, налаштувань користувача та іншої динамічної інформації. Додатково, можливість інтеграції авторизації через Google спрощує процес входу, дозволяючи користувачам швидко отримати доступ без необхідності створення окремих облікових записів.

Для забезпечення додаткової функціональності було використано Node.js для розробки серверної частини, що дозволяє обробляти запити, виконувати обчислення, працювати з базою даних та надавати API для клієнтської частини.

2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКА

2.1. Загальна архітектура системи

Загальна архітектура веб-додатка для створення персоналізованих календарів має базуватись на врахуванні низки важливих аспектів — від зручності користування до надійного захисту даних і стабільної взаємодії між усіма частинами системи. Такий підхід дозволяє забезпечити узгоджену роботу інтерфейсу, бекенду та бази даних, що є критично важливим для комфортного користувацького досвіду. Веб-додаток складається з кількох основних частин:

1) Фронтенд (інтерфейс користувача)

Інтерфейс користувача реалізовано за допомогою бібліотеки React, яка забезпечує швидку та динамічну взаємодію з елементами сторінки. У додатку використовується спеціальний компонент Календар, що дозволяє відображати події за обраною датою. Користувач може додавати нові події, редагувати наявні або видаляти їх за потреби, що робить керування розкладом простим і зручним.

Користувач може вводити назву події, дату та час, після чого подія зберігається в локальному сховищі браузера (LocalStorage) і автоматично відображається в календарі. Користувач також має можливість переглядати всі події за обраний місяць або конкретну дату.

Веб-додаток підтримує темний і світлий режими, що дозволяє змінювати стиль інтерфейсу в залежності від переваг користувача.

2) Бекенд (обробка даних)

У цьому проєкті не передбачено використання складного серверного бекенду, оскільки всі дані про події зберігаються локально на стороні клієнта за допомогою LocalStorage браузера. Такий підхід забезпечує автономність додатка та дозволяє користувачам працювати з календарем без підключення до зовнішніх серверів чи Інтернету.

Однак, якщо в майбутньому буде необхідно масштабувати додаток або додати нові функції, можна інтегрувати бекенд-сервер, наприклад, з використанням Firebase або Node.js. Це дозволить зберігати дані в хмарі та забезпечити доступ до календаря з будь-якого пристрою.

3) Безпека даних

Оскільки додаток оперує особистими подіями користувачів, питання захисту даних є надзвичайно важливим. Для цього впроваджено базові механізми безпеки, зокрема шифрування паролів і автентифікацію користувачів. У випадку використання серверної інфраструктури, обов’язковим є застосування протоколу HTTPS для забезпечення безпечної передачі інформації між клієнтською частиною та сервером.

4) Інтеграція з іншими сервісами

Додаток може бути інтегрований з іншими сервісами для синхронізації подій. Наприклад, використовуючи Google Calendar API, можна додавати події в Google Calendar, що дозволить користувачам синхронізувати події між кількома пристроями.

5) Архітектура клієнт-сервер

Архітектура веб-додатка побудована за принципом клієнт-серверної взаємодії, хоча наразі система функціонує лише на клієнтській стороні. Усі події зберігаються локально в браузері користувача, що дозволяє працювати з календарем без звернення до зовнішнього сервера. У майбутньому можлива реалізація серверної частини для централізованого зберігання даних у базі.

6) Масштабованість і додаткові можливості

Додаток розроблено з урахуванням можливості подальшого розширення функціоналу. У перспективі можна реалізувати функцію спільного доступу до календаря для кількох користувачів, а також передбачити інтеграцію з іншими календарними платформами, зокрема Outlook або iCalendar, що зробить систему більш універсальною та зручною для різних категорій користувачів.

2.2. Вимоги до функціоналу

Для реалізації персоналізованого календаря, що дає змогу користувачам легко додавати, редагувати та видаляти події, а також переглядати їх у зручному форматі, потрібно впровадити низку ключових функцій. Важливо, щоб усі елементи були інтуїтивно зрозумілими, доступними у використанні та відповідали вимогам безпеки й комфорту для користувача. Нижче наведено основні вимоги до функціоналу веб-додатка:

1) Додавання, редагування та видалення подій

Користувач повинен мати змогу додавати нові події до календаря, вказуючи такі параметри, як назва події, дата та час.

2) Усі події мають зберігатися в локальному сховищі браузера (LocalStorage) та відображатися у відповідні дні календаря. Для кожної події передбачена можливість редагування — змінювати її назву, дату або час — а також повне видалення. Усі зміни повинні миттєво оновлюватися в інтерфейсі, щоб користувач одразу бачив актуальний стан розкладу.

3) Налаштування кольорових схем та стилів календаря

Користувач повинен мати можливість змінювати стиль інтерфейсу календаря для зручності використання. Це може включати:

- Перемикання між темним і світлим режимами.
- Вибір кольорів для різних типів подій або днів
- Зміна шрифтів і відображення елементів інтерфейсу для покращення візуального сприйняття.

4) Інтеграція з іншими сервісами (наприклад, Google Calendar API)

Для розширення функціональності веб-додатка можна додати інтеграцію з іншими календарними сервісами, такими як Google Calendar API. Це дозволить користувачам:

- Синхронізувати події між веб-додатком і Google Calendar.
- Додавати, оновлювати та видаляти події безпосередньо через Google Calendar, що дозволить більш зручне управління подіями на різних пристроях.

Інтеграція з іншими сервісами може бути реалізована через API, що дозволяє взаємодіяти з хмарними платформами та зберігати дані у хмарі.

5) Додавання нагадувань та сповіщень

Додаток повинен мати можливість створювати нагадування для подій. Це можуть бути сповіщення за певний час до початку події, повідомлення через браузер або через електронну пошту.

6) Авторизація користувачів та збереження даних

Додаток має підтримувати авторизацію користувачів для забезпечення доступу до персоналізованих подій:

- Реєстрація нового користувача з ім'ям та паролем.
- Вхід через введення правильних облікових даних.
- Збереження даних користувача в локальному сховищі браузера або серверній базі даних при підключенні серверної частини.
- Безпека облікових даних через шифрування паролів (наприклад, за допомогою алгоритмів на базі base64 або більш складних методів, таких як bcrypt).

Після проходження авторизації користувач отримує доступ до своїх особистих подій з можливістю їх перегляду, редагування та створення нових записів. Усі дані зберігаються локально, що дозволяє працювати з календарем у режимі офлайн. У разі підключення до Інтернету інформацію можна синхронізувати для забезпечення актуальності та резервного копіювання.

7) Можливість перегляду подій на різних рівнях часу

Користувач повинен мати можливість переглядати події за конкретну дату та всі події за місяць; переглядати події на тиждень або годину, якщо додаткові опції будуть реалізовані в майбутньому.

Це дозволить користувачу гнучко налаштовувати відображення подій відповідно до його потреб та зручності.

8) Тестування та перевірка функціональності

Після розробки всіх функцій календаря необхідно провести детальне тестування:

- Перевірку коректності роботи всіх функцій.
- Перевірку правильності додавання, редагування та видалення подій.
- Перевірку налаштувань кольорових схем та режимів.
- Перевірку безпеки даних та авторизації користувачів.

2.3. Проектування бази даних

Для зручного зберігання та обробки подій у веб-додатку використовується локальне сховище браузера (LocalStorage), що дозволяє миттєво отримувати доступ до даних без звернення до сервера. Події кожного користувача зберігаються окремо у форматі JSON під унікальним ключем, наприклад, "user123_events". Кожен запис містить назву події, дату, час і ідентифікатор користувача, що забезпечує розмежування даних між різними обліковими записами.

Під час входу події завантажуються з localStorage, а після будь-яких змін — знову зберігаються шляхом оновлення відповідного ключа. Такий підхід забезпечує високу швидкість роботи та автономність системи.

Основні переваги LocalStorage — миттєвий доступ до даних, незалежність від серверів і простота реалізації. Водночас існують обмеження: обмежений обсяг пам’яті та відсутність синхронізації між пристроями. Тому в разі масштабування може виникнути потреба у використанні серверної бази даних.

2.4. UI/UX дизайн: розробка макетів та вибір стилю інтерфейсу

UI/UX дизайн відіграє вирішальну роль у створенні веб-додатка для персоналізованих календарів, оскільки саме від нього залежить зручність і комфорт роботи користувача. Інтерфейс має бути максимально простим, інтуїтивно зрозумілим і доступним навіть для людей із базовими навичками користування цифровими технологіями. Особлива увага приділяється мінімалістичному підходу: відсутність зайвих елементів, чітка структура та

логічне розміщення функціоналу дозволяють швидко додавати, редагувати й переглядати події.

Календар реалізовано як інтерактивний елемент з зручною навігацією між датами. Користувач має можливість переглядати події у денному або місячному форматі. Головний екран містить сам календар, форму для додавання подій, список запланованих подій і функціональні кнопки для взаємодії. Також створено окремі екрани для авторизації, реєстрації та повного перегляду подій у місячному режимі.

Дизайн додатка адаптовано для різних пристроїв — він коректно працює як на комп’ютерах, так і на мобільних телефонах завдяки медіа-запитам і гнучкій верстці. Візуальне оформлення базується на поєднанні м’якого фону з яскравими акцентами, що покращує сприйняття інформації. Шрифти підібрано таким чином, щоб забезпечити легке читання. Для візуального комфорту додано легкі анімації, а також передбачено темний режим, зручний для використання в умовах низького освітлення.

Для оформлення інтерфейсу використано Tailwind CSS разом із класичним CSS, що дозволяє оперативно стилізувати елементи без створення надлишкових класів. Адаптивність забезпечується завдяки використанню Flexbox і Grid, що дозволяє гнучко розташовувати елементи на різних екранах. Компонентний підхід у React дає змогу багаторазово використовувати ті самі елементи, що пришвидшує розробку та спрощує подальше обслуговування коду.

У результаті створено зручний, адаптивний і візуально привабливий інтерфейс, який враховує потреби користувачів, забезпечує швидкий доступ до функцій календаря і можливість налаштувань відповідно до індивідуальних уподобань. На рисунку 2.1 зображена діаграма компонентів інтерфейсу.

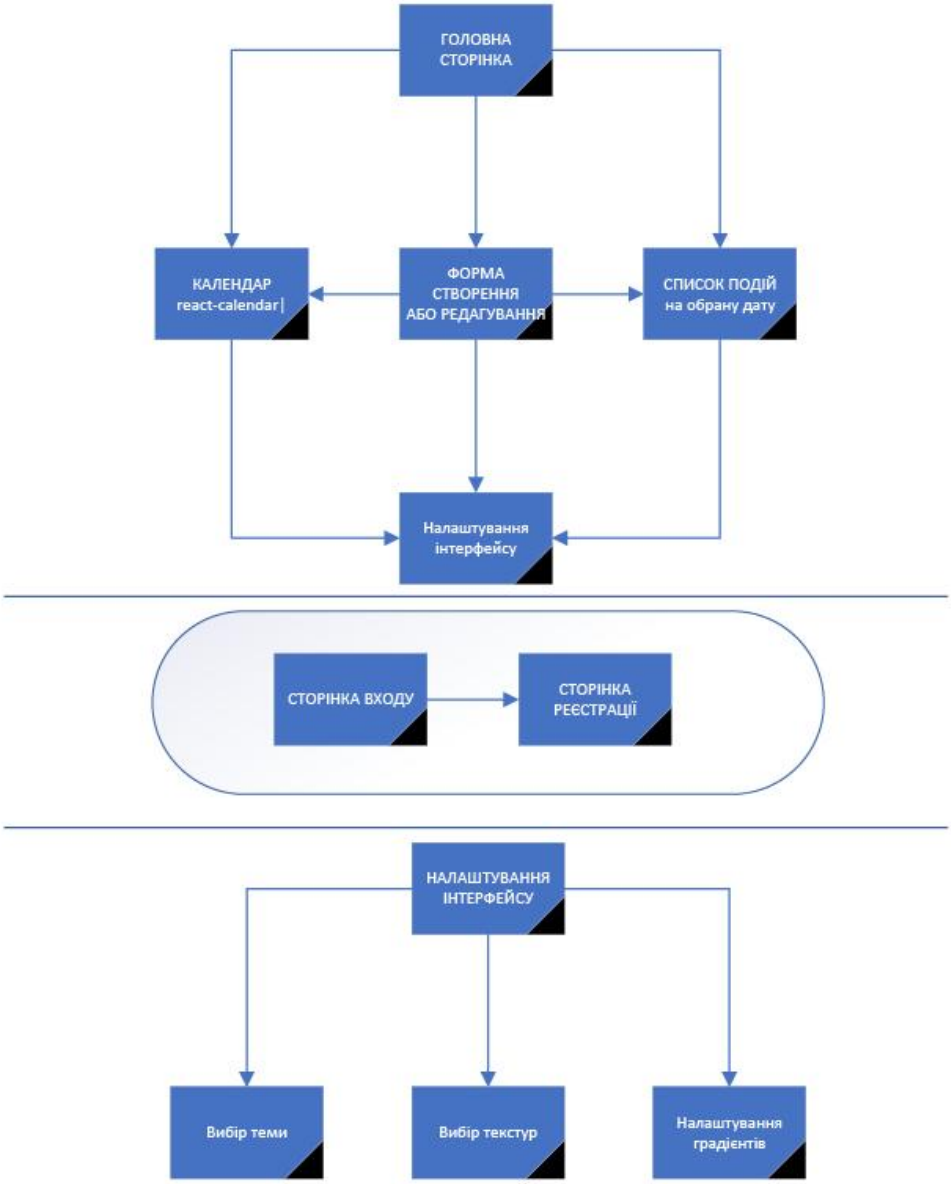

Рисунок 2.1 – Діаграма компонентів інтерфейсу

3 РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКА

3.1. Розробка фронтенду

Веб-додаток створено з використанням React — популярної бібліотеки для побудови користувацьких інтерфейсів. Вона дозволяє реалізовувати інтерактивні додатки за допомогою компонентів, що мають власний стан і можуть динамічно оновлюватися. У рамках проєкту реалізовано функціональний календар, можливість додавання та редагування подій, а також систему авторизації та реєстрації для користувачів. Розглянемо основні етапи та принципи реалізації.

1) Створення компонента App

На початку я імпортую необхідні залежності:

React – для роботи з компонентами.

useState, useEffect – для роботи зі станом компонентів та ефектами.

react-calendar – для використання компонента календаря.

App.css – для стилів додатка.

2) Функціональність додавання подій

Один з основних аспектів додатка – це можливість додавання подій на календар. Щоб це зробити, реалізуємо форму для введення подій, код на рисунку 3.1.

```
1  const [eventText, setEventText] = useState('');
2  const [eventDate, setEventDate] = useState('');
3  const [eventTime, setEventTime] = useState('');
4
```

Рисунок 3.1 – Код форми для введення подій

Ми створюємо три стани для збереження тексту події, дати та часу події. Вони оновлюються, коли користувач вводить дані у форму.

Реалізуємо кнопку додавання подій на рисунку 3.2

```
const addEvent = () => {
if (!eventText || !eventDate || !eventTime) return;
const formattedDateTime = formatDateTime(new Date(eventDate), eventTime);
setEvents([...events, { text: eventText, dateTime: formattedDateTime }]);
setEventText('');
setEventDate('');
setEventTime('');
};
```

Рисунок 3.2 – Фрагмент коду для додавання подій

У функції addEvent перевіряємо, чи заповнені всі поля.

3) Вхід та реєстрація

Важливий момент в додатку – це можливість реєстрації та входу користувачів. Ми реалізуємо це за допомогою простого механізму зберігання даних у localStorage (рисунок 3.3).

```
const encryptPassword = (password) => {
  return btoa(password); // шифруємо з допомогою base64
};
const decryptPassword = (password) => {
  return atob(password); // розшифровуємо з допомогою base64
};
```

Рисунок 3.3 - Фрагмент коду для шифрування та збереження пароля

Ми використовуємо Base64 для шифрування паролів користувачів перед їх збереженням у браузері. Це дозволяє захистити паролі від прямого доступу. Скріншот входу в застосунок можна побачити на рисунках 3.4 та 3.5.

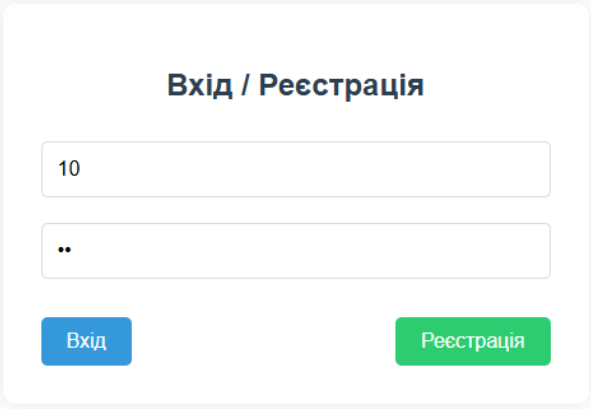

Рисунок 3.4 - Вхід / Реєстрація в додаток

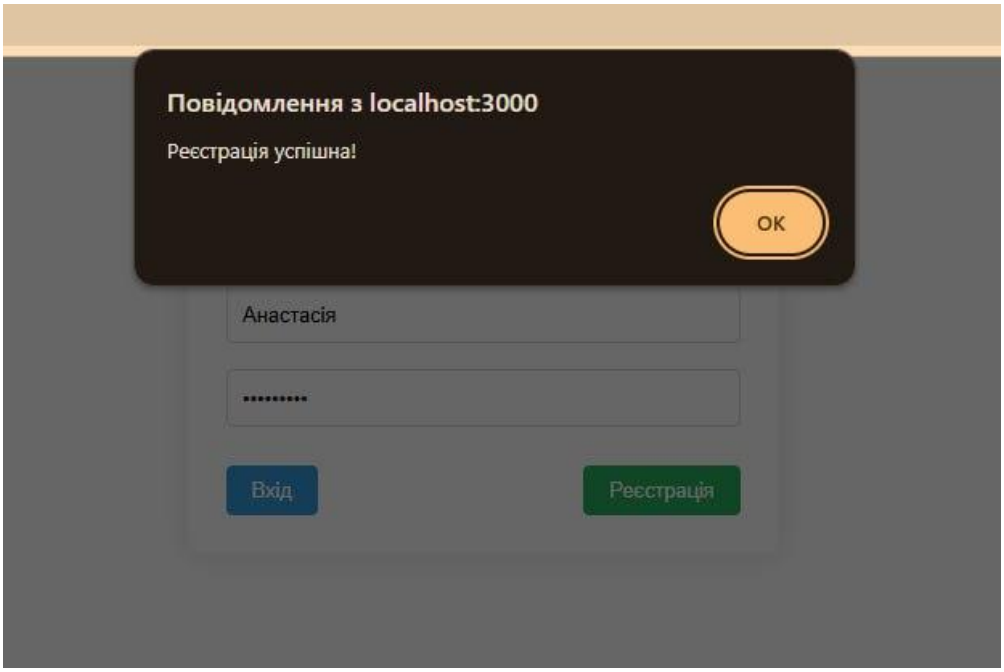

Рисунок 3.5 – Повідомлення про успішну реєстрацію

Коли користувач реєструється, ми зберігаємо ім’я користувача і зашифрований пароль у localStorage. Також створюємо для користувача пустий список подій.

Під час входу ми перевіряємо ім’я користувача та його пароль. Якщо пароль співпадає з тим, що збережено в localStorage, ми дозволяємо користувачу увійти до додатку, якщо ні – бачимо відповідне повідомлення (рисунок 3.6).

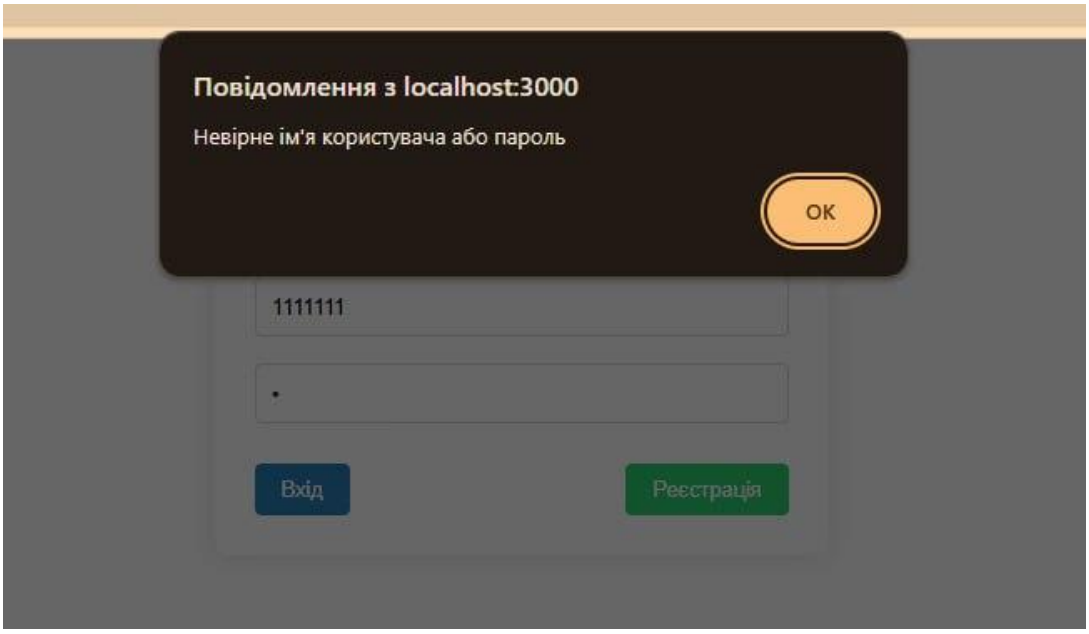

Рисунок 3.6 – Повідомлення про неправильні дані для входу

Функція handleLogout дозволяє користувачу вийти з додатку, скидаючи всі дані та скидаючи стан авторизації.

4) Календар та події

Для відображення календаря ми використовуємо бібліотеку react-calendar:

<Calendar onChange={onDateChange} value={selectedDate} />

При виборі дати на календарі ми оновлюємо вибрану дату через onDateChange. Події фільтруються за вибраною датою, і якщо події є для цієї дати, вони відображаються у списку.

Список подій фільтрується по вибраній даті, і відображається або повідомлення про те, що немає подій на цю дату, або самі події (рисунок 3.7).

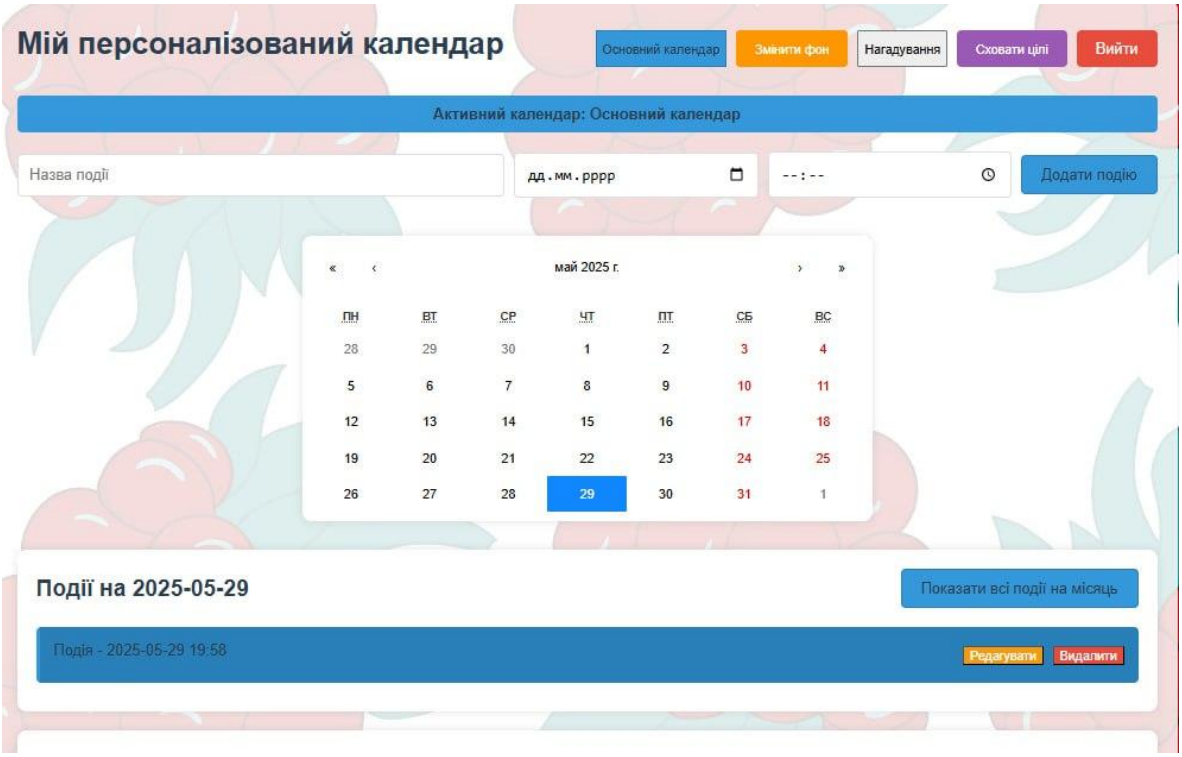

Рисунок 3.7 – Події на поточну дату

Темний режим додається за допомогою простого перемикача.

При натисканні на кнопку toggleDarkMode ми змінюємо стан darkMode і додаємо клас dark до основного контейнера додатка. Це дозволяє використовувати стилі для темної теми.

3.2 Реалізація бекенду (Node.js/Express або Firebase)

Розглянемо, як реалізувати бекенд для нашого веб-додатку для персоналізованого календаря. Для цього ми вибрали два варіанти — використання Node.js з фреймворком Express або використання Firebase для автоматизації більшості завдань.

1) Реалізація з використанням Node.js та Express:

Node.js — це середовище виконання JavaScript на сервері, яке дозволяє обробляти запити від клієнта і взаємодіяти з базою даних. Express — це мінімалістичний фреймворк для Node.js, який спрощує створення серверних додатків і обробку HTTP-запитів.

Кроки для налаштування бекенду з Node.js і Express:

1) Інсталяція необхідних пакетів:

Спочатку потрібно ініціалізувати проект і встановити необхідні бібліотеки:

- express — для створення сервера.
- mongoose — для роботи з MongoDB (якщо ви вибираєте MongoDB).
- cors — для підтримки крос-доменних запитів.
- body-parser — для парсингу тіла запитів.

2) Створення простого сервера на Express:

В базових налаштуваннях Express сервера ми створюємо два маршрути:

POST /addEvent — для додавання події.

GET /events — для отримання всіх подій.

Тепер, коли сервер налаштований, ми можемо підключити його до нашого фронтенду. Замість використання localStorage, ми будемо робити HTTP-запити до нашого бекенду.

Після додавання нової події за допомогою POST /addEvent, ми отримуємо всі події з серверу через GET /events.

Використання Firebase дозволяє нам уникнути налаштування сервера, оскільки Firebase надає хмарну платформу для зберігання даних, а також для аутентифікації користувачів.

Кроки для інтеграції Firebase у наш додаток:

1) Створила проект у Firebase Console.
2) Додала Firebase SDK у додаток.
3) Налаштувала Firebase у файлі мого додатку (рисунок 3.8)

```
import { initializeApp } from 'firebase/app';
import { getFirestore, collection, addDoc, getDocs } from 'firebase/firestore';
const firebaseConfig = {
  apiKey: 'YOUR_API_KEY',
  authDomain: 'YOUR_AUTH_DOMAIN',
  projectId: 'YOUR_PROJECT_ID',
  storageBucket: 'YOUR_STORAGE_BUCKET',
  messagingSenderId: 'YOUR_MESSAGING_SENDER_ID',
  appId: 'YOUR_APP_ID',
};
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);
```

Рисунок 3.8 – Фрагмент коду для налаштування Firebase

4) Для збереження подій у Firestore, використовувала методи addDoc та getDocs (рисунок 3.9)

```
const addEventToFirestore = async (eventText, eventDateTime) => {
  try {
    const docRef = await addDoc(collection(db, 'events'), {
      text: eventText,
      dateTime: eventDateTime
    });
    console.log('Подія додана: ', docRef.id);
  } catch (e) {
    console.error('Помилка при додаванні події: ', e);
  }
};
const getEventsFromFirestore = async () => {
  const querySnapshot = await getDocs(collection(db, 'events'));
  const eventsList = querySnapshot.docs.map(doc => doc.data());
  setEvents(eventsList);
};
```

Рисунок 3.9 – Фрагмент коду для збереження подій

5) Використовувала ці функції для зберігання та отримання подій. Замість локального зберігання, я використовувала Firestore для синхронізації даних між клієнтом і сервером у реальному часі.

3.3. Реалізація функціоналу календаря

Розглянемо реалізацію календарної функціональності у рамках веб-додатку. Календар виступає центральним елементом інтерфейсу, адже саме через нього користувач переглядає свої події за вибраний день, місяць або рік. Для спрощення інтеграції та налаштування використовується бібліотека react-calendar, яка дозволяє швидко впровадити календар у проєкт і забезпечити базову функціональність без значних витрат часу.

Крок 1: Встановлення бібліотеки react-calendar

Для того щоб додати календар, спочатку потрібно встановити бібліотеку react-calendar.

Ця бібліотека забезпечить нам можливість інтегрувати календар з мінімальними зусиллями.

Крок 2: Інтеграція календаря в компонент

Після встановлення бібліотеки, наступним кроком буде додавання календаря до нашого додатку. В компонентах, які ми вже створювали для подій, додаємо календар за допомогою react-calendar у файлі App.js (див. Додаток А)

Пояснення функціоналу:

1) Бібліотека react-calendar надає компонент <Calendar />, який відображає календар. Він інтегрується в наш додаток і дозволяє користувачу вибирати дату.

2) Фільтрація подій по даті: При виборі дати на календарі викликається функція onDateChange, яка змінює стан selectedDate на вибрану дату (рисунок 3.10). Події фільтруються за цією датою, і тільки ті події, що відповідають вибраній даті, відображаються у списку.

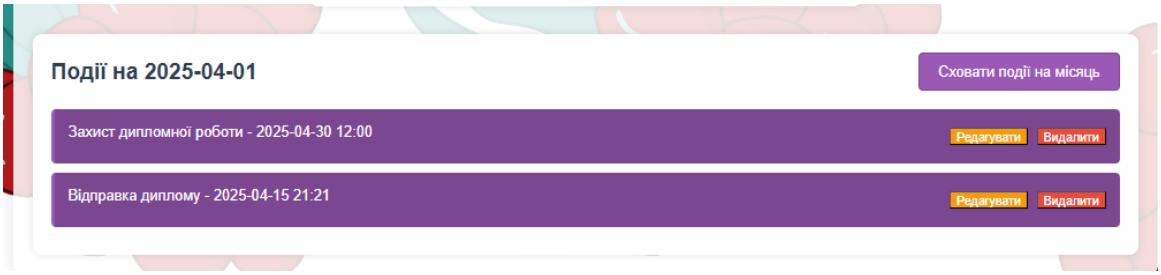

Рисунок 3.10 – Реалізація перегляду подій

3) Додавання події (рисунок 3.11): Після вибору дати та введення даних про подію (текст, дата, час), користувач може додавати події, які автоматично зберігаються у стані і згодом можуть бути збережені в базі даних (як було описано раніше).

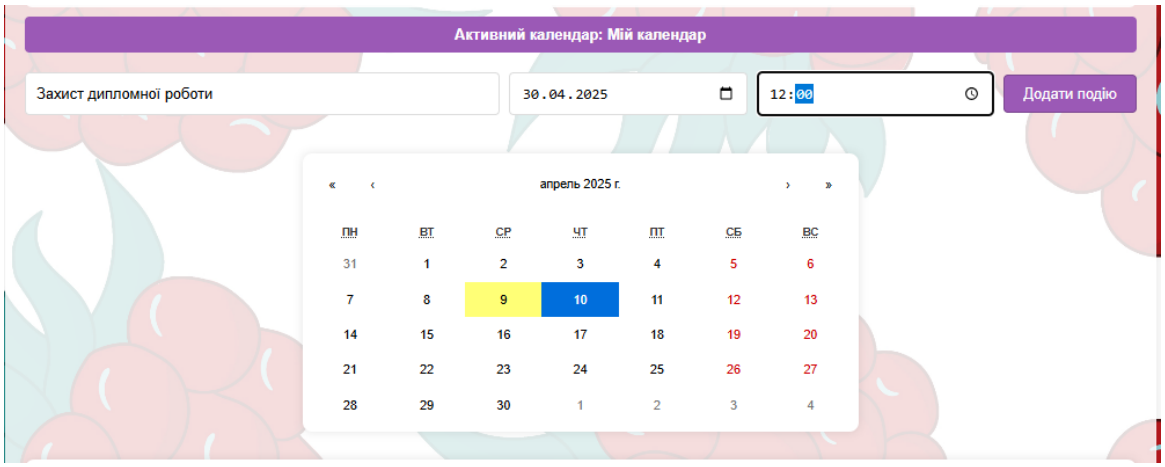

Рисунок 3.11 - Створення подій

4) UI компоненти: Для зручності користувача додано інтерфейс для введення тексту події, вибору дати та часу.

Крок 3: Налаштування вигляду календаря

Календар налаштували за допомогою стилів, використовуючи CSS.

Крок 4: Завантаження подій за місяць

Щоб реалізувати можливість переглядати події на весь місяць, ми використовуємо функцію getMonth() з JavaScript, щоб отримати всі події за поточний місяць. Це вже частина функціоналу, яку ми реалізували раніше з кнопкою "Показати всі події на місяць".

Компонент для управління цілями (рисунок 3.12) був реалізований у файлі Goals.js. Він дозволяє користувачам вводити текст цілі, зберігати її, а також видаляти непотрібні. Для збереження даних використовувалося локальне сховище браузера (localStorage), що забезпечує збереження цілей навіть після перезавантаження сторінки або закриття браузера (рисунок 3.13).

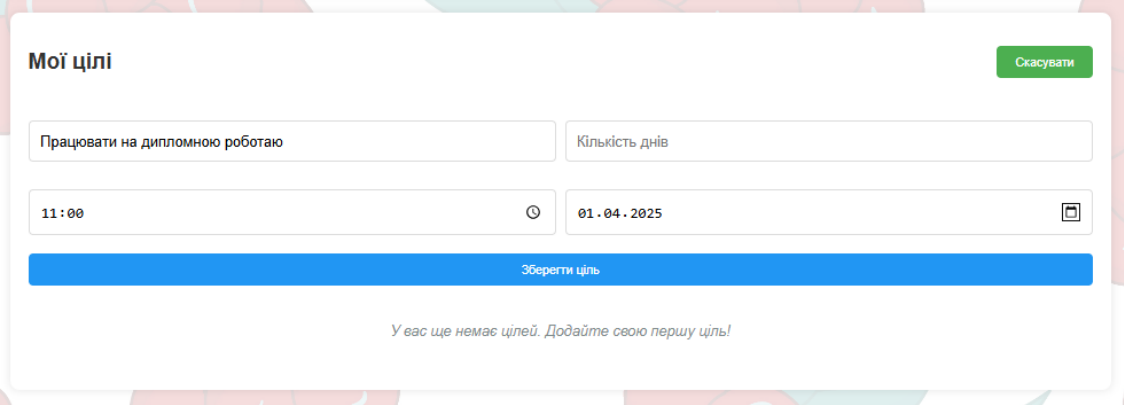

Рисунок 3.12 - Створення лічильника та відслідковування цілей

Рисунок 3.13 - Реалізація лічильника

Порівняння з іншими календарями

На відміну від традиційних сервісів на кшталт Google Calendar, даний додаток надає користувачам змогу не лише створювати події, а й зберігати особисті цілі. Це розширює функціональність і дозволяє більш структуровано планувати час — як на день, так і на тиждень чи місяць. Можливість встановлювати цілі без залучення сторонніх інструментів робить систему більш автономною та зручною для індивідуального використання.

Вибір дати:

При виборі нової дати в календарі через функцію onDateChange ми оновлюємо стан обраної дати:

```
const onDateChange = (date) => {
  setSelectedDate(date);  // Оновлюємо стан з новою вибраною датою
};
```

Відображення подій на вибрану дату:

Для кожної дати ми відображаємо список подій, пов'язаних з нею. Усі події зберігаються у локальному сховищі та завантажуються при вході користувача. У файлі App.js (додаток А) ми фільтруємо події для поточної вибраної дати. Це дозволяє показувати лише ті події, що прив'язані до обраної дати.

В App.css (додаток Б) додаємо стилі для календаря, щоб зробити його зручним для користувачів. Наприклад, можна змінити розміри контейнера:

Файл goals.css (додаток В) для налаштування фону та додаткових елементів

Для зручності ми також додали стилі в goals.css, які дозволяють змінювати фон календаря або додавати текстури (рисунок 3.14 та 3.15).

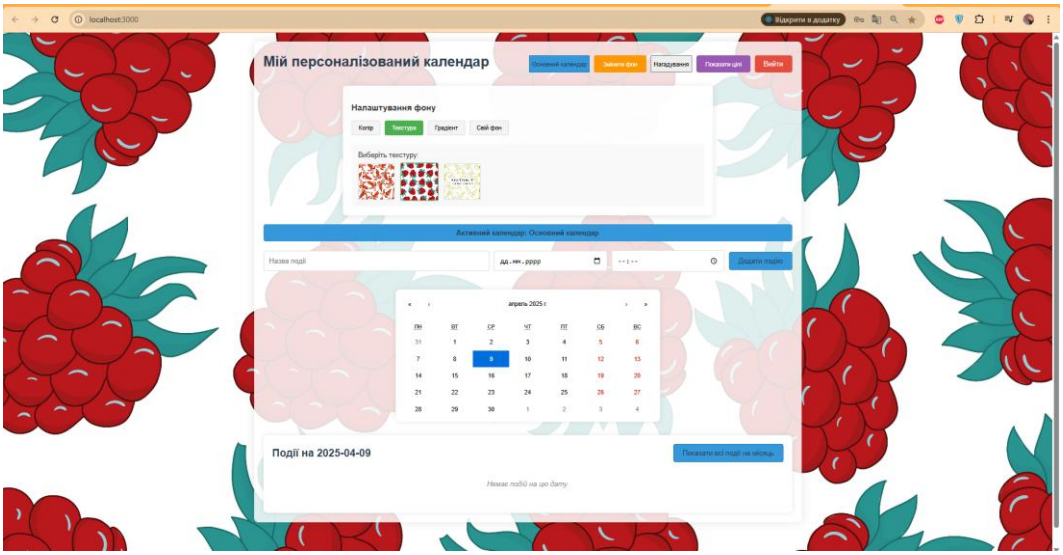

Рисунок 3.14 - Налаштування фону персоналізованого календаря

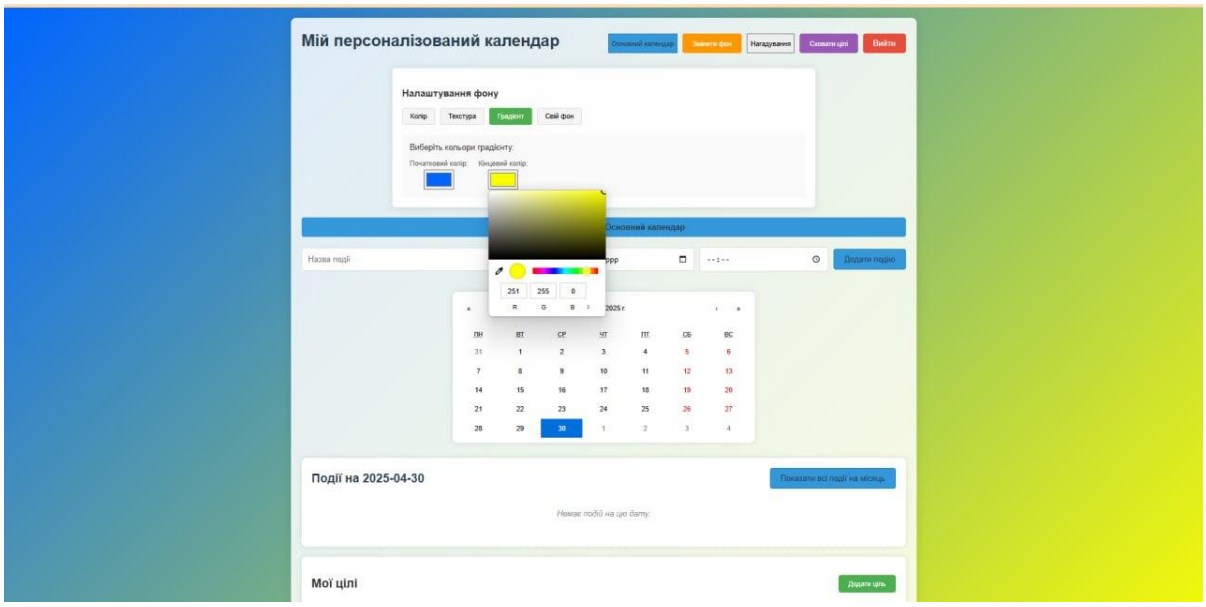

Рисунок 3.15 – Налаштування градієнту фону

Щоб додати більше можливостей для користувачів налаштовувати свій календар, ми створили можливість змінювати текстури фону та градієнти через стилі в goals.css. (рисунок 3.16).

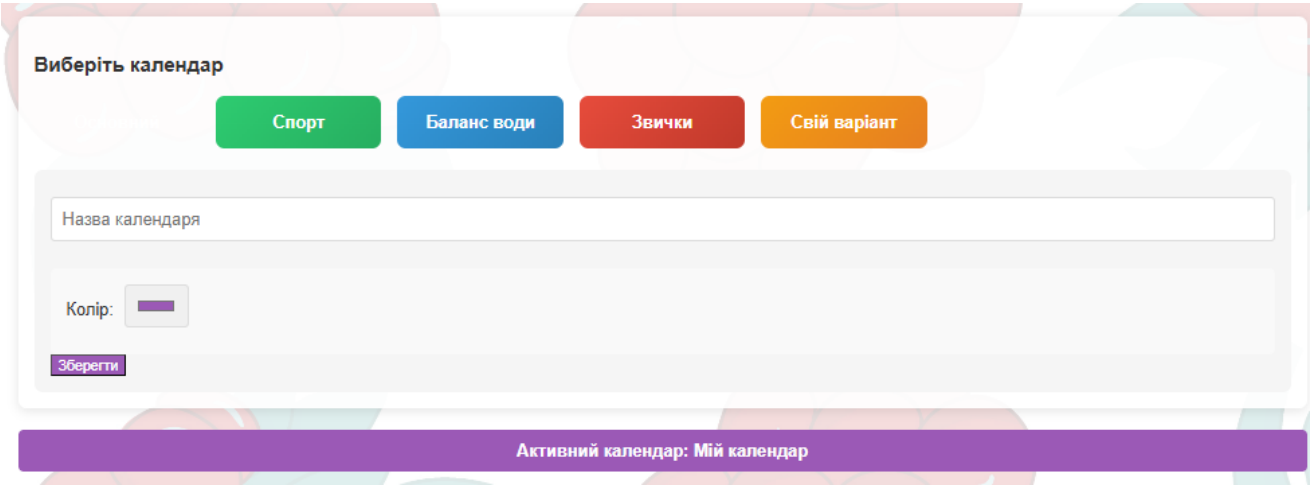

Рисунок 3.16 - Створення нового календаря або використання шаблону

Таким чином, використання react-calendar та налаштування його стилів допомагає зробити календар інтуїтивно зрозумілим і легким у використанні для користувачів. Також додавання можливості налаштовувати фон і градієнти допомагає зробити інтерфейс більш персоналізованим.

Функціональність додавання і управління цілями дозволяє значно розширити можливості календаря.

Завдяки використанню локального сховища всі цілі зберігаються навіть після перезавантаження додатку.

Користувачі можуть зберігати та переглядати не лише події, а й цілі, що робить календар більш багатофункціональним.

3.4. Тестування додатка

Тестування є ключовим етапом розробки, що дозволяє впевнитися в коректній роботі додатка, його стабільності та зручності для користувача. Під час перевірки веб-додатка для створення персоналізованих календарів було протестовано основні функції: додавання й видалення подій, навігацію по календарю, процес авторизації та реєстрації, а також взаємодію з локальним сховищем даних (localStorage).

Тестування функціоналу додавання та видалення подій

Правильний сценарій:

1) Вводимо коректні дані:

   Назва події: "Зустріч"

   Дата: 2025-04-10

   Час: 14:00

2) Натискаємо кнопку "Додати подію".

3) Перевіряємо, що подія додалася в список подій і відображається на вибрану дату.

Некоректний сценарій:

Залишаємо поле "Назва події" порожнім і намагаємося додати подію:

Вводимо дату 2025-04-10 та час 14:00, але не заповнюємо назву.

Після натискання кнопки "Додати подію" перевіряємо, що подія не додалася, і з’являється повідомлення про необхідність заповнити назву події.

1) Залишаємо поле "Дата" порожнім і намагаємося додати подію:

   Вводимо назву "Зустріч" та час 14:00, але не заповнюємо дату.

Після натискання кнопки "Додати подію" перевіряємо, що подія не додалася, і з’являється повідомлення про необхідність вибрати дату події.

2) Вводимо некоректний формат часу (наприклад, "25:00") і намагаємося додати подію.

Перевіряємо, що додаток не додає подію та виводить повідомлення про неправильний формат часу.

3) Після додавання подій натискаємо кнопку "Видалити" і перевіряємо, що подія зникає зі списку.

Тестування роботи календаря.

Правильний сценарій:

1) Вибираємо дату 2025-04-10 на календарі.

2) Перевіряємо, що відображаються події на вибрану дату, якщо вони були додані.

3) Після вибору іншої дати (наприклад, 2025-04-15) перевіряємо, що події для цієї дати відображаються коректно.

Некоректний сценарій:

1) Вибираємо дату, на яку немає подій, і перевіряємо, що з’являється повідомлення "Немає подій на цю дату" (рисунок 3.17).

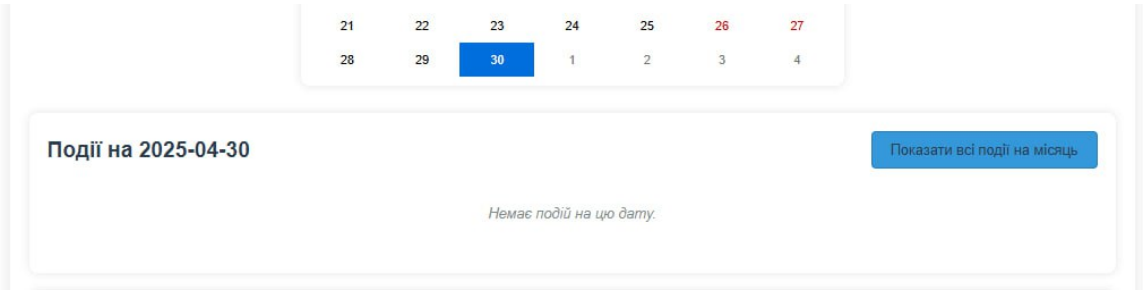

Рисунок 3.17 – Повідомлення про відсутність подій на поточну дату

2) Намагаємось вибрати дату в майбутньому (наприклад, 2025-12-31) і перевіряємо, що події на цю дату відображаються коректно.

3) Вибираємо дату в минулому (наприклад, 2024-04-01) і перевіряємо, що події відображаються коректно.

Тестування входу та реєстрації користувачів

Правильний сценарій:

1) Вводимо коректні дані для реєстрації (ім’я користувача: "user1", пароль: "password123").

2) Процес реєстрації проходить без помилок, дані зберігаються в localStorage.

3) Намагаємось увійти з правильним логіном і паролем. Перевіряємо, що додаток дозволяє увійти та відображає відповідні події.

Некоректний сценарій:

1) Задишаємо поле "Ім’я користувача" порожнім і намагаємося зареєструватися.

Перевіряємо, що додаток видає повідомлення про помилку: "Заповніть ім’я користувача".

2) Залишаємо поле "Пароль" порожнім і намагаємося зареєструватися.

Перевіряємо, що додаток видає повідомлення про помилку: "Заповніть пароль".

3) Вводимо неправильні дані для входу (наприклад, "user1" та "wrongpassword").

Перевіряємо, що додаток видає повідомлення: "Невірне ім’я користувача або пароль".

Тестування роботи з localStorage.

Правильний сценарій:

1) Додаємо події і перевіряємо, що дані зберігаються в localStorage під ключем "<ім’я користувача>_events".

2) Перезавантажуємо сторінку і перевіряємо, що всі події залишаються на місці.

3) Увійшовши в систему з іншим користувачем, перевіряємо, що його події завантажуються коректно з localStorage.

Некоректний сценарій:

1) Очищаємо localStorage і перевіряємо, що при повторному завантаженні сторінки не відображаються жодні події.

2) Перевіряємо, що якщо при спробі додати подію не заповнені необхідні поля, то дані не зберігаються в localStorage.

Результати тестування веб-додатка для персоналізованих календарів свідчать про його високу стабільність. Усі ключові функції — додавання, редагування та видалення подій, навігація по календарю, авторизація користувачів і робота з localStorage — працюють без збоїв. У процесі тестування було виявлено кілька недоліків, зокрема недостатню обробку помилок при некоректному введенні даних, однак загалом додаток показав надійність і готовність до подальшого використання.

4. АНАЛІЗ РЕЗУЛЬТАТІВ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

4.1. Оцінка продуктивності веб-додатка

Оцінювання продуктивності є важливою складовою тестування веб-додатка, адже дозволяє визначити, наскільки швидко та ефективно система виконує основні дії. У межах цього проєкту було проаналізовано швидкодію при додаванні та видаленні подій, а також час відгуку інтерфейсу під час відображення календаря. Отримані результати свідчать про стабільну роботу додатка навіть за активного використання основного функціоналу.

Оцінка часу завантаження додатка.

Час завантаження додатка виявився досить коротким і в середньому не перевищував 3 секунд. Навіть на пристроях із нижчими технічними характеристиками або при повільному інтернет-з’єднанні показники залишалися в межах прийнятного. Для невеликих проєктів подібного типу завантаження у проміжку 1–3 секунди вважається оптимальним і не впливає негативно на зручність користування.

Оцінка часу додавання події.

Час, необхідний для додавання події, вимірювався від моменту натискання кнопки «Додати подію» до її появи у списку подій. Операція виконується миттєво, без затримок, що свідчить про високу швидкодію та ефективність роботи додатка.

Оцінка часу видалення події.

Час видалення події також не перевищує 1 секунди. Після натискання кнопки "Видалити" подія моментально зникає зі списку подій і зміни зберігаються в localStorage, що також підтверджує високу швидкість виконання цієї операції.

Оцінка роботи календаря.

Перемикання між датами та відображення подій на календарі відбувається без затримок. Календар працює плавно, навіть при наявності великої кількості подій. Однак, на слабших пристроях із обмеженими

ресурсами може спостерігатися незначна затримка при завантаженні місяців з великою кількістю подій.

Оцінка швидкості збереження даних у localStorage.

Збереження даних у localStorage також відбувається швидко і не створює затримок для користувача. Після додавання або видалення події дані одразу зберігаються в localStorage, що дає можливість зберігати їх при перезавантаженні сторінки.

Продуктивність на мобільних пристроях.

Результати тестування на мобільних пристроях показали, що додаток працює стабільно та виконує всі основні функції — додавання, редагування, видалення подій і навігацію між місяцями — без помітних затримок. Водночас було виявлено, що на екранах з невеликою діагоналлю інтерфейс може бути менш зручним. Зокрема, деякі елементи потребують збільшення для поліпшення взаємодії, тому доцільно передбачити подальшу адаптацію дизайну для мобільних користувачів.

4.2 Аналіз зручності використання (відгуки тестових користувачів)

Для того, щоб оцінити наскільки зручним вийшов створений веб-додаток, я попросила протестувати його групу з 15 людей. Серед тестувальників були мої, знайомі, а також кілька працівників офісів та фрілансерів. Завданням було спробувати виконати основні дії в додатку (додати подію, змінити фон, створити новий календар, додати нагадування) і написати свої враження. Після тесту я попросила кожного відповісти на кілька запитань у невеликій анкеті.

Запитання, які були в анкеті:

1) Чи легко було розібратися з інтерфейсом?

2) Чи зручно користуватись календарем і перемикатися між режимами?

3) Чи подобається дизайн додатку?

4) Чи виникали помилки під час роботи?

5) Які є побажання щодо покращення додатку?

Ось які результати я отримала:

Простота роботи з додатком

Більшість користувачів (13 з 15) сказали, що інтерфейс зрозумілий і простий. Двоє людей сказали, що спочатку було незрозуміло, де знаходяться налаштування фону, але після декількох хвилин все стало зрозуміло (рисунок 4.1).

Рисунок 4.1 – Результат опитування щодо розуміння інтерфейсу

Зручність календаря

Усі тестувальники сказали, що перемикатися між днями та місяцями зручно. Також позитивно відзначили швидкість додавання і редагування подій. Особливо сподобалось, що додаток швидко реагує на кліки й зміни в календарі.

Дизайн додатку

Користувачі позитивно оцінили сучасний дизайн додатка, відзначивши його естетичність і зручність. Особливу увагу привернула можливість перемикання між світлою та темною темами, що зробило використання

додатка комфортнішим у різних умовах освітлення. Деякі користувачі також відзначили, що інтерфейс виглядає стильно й приємно для сприйняття.

Помилки та технічні проблеми

В більшості випадків додаток працював добре. Лише троє користувачів помітили невеликі затримки при завантаженні текстур для фону. Інших серйозних помилок чи збоїв виявлено не було (рисунок 4.2).

Рисунок 4.2 – Результати опитування щодо помилок

Загальні враження і побажання

Користувачі в цілому позитивно оцінили додаток. Вони сказали, що готові використовувати його в майбутньому для особистих чи робочих потреб. Також було кілька побажань:

- додати можливість експортувати події (наприклад, у форматі PDF або CSV);
- зробити інтеграцію з іншими календарями, особливо Google Calendar;
- реалізувати повідомлення або нагадування у вигляді сповіщень на телефоні.

Ці деталі я врахую для подальших покращень додатку.

Отже, після тестування я зрозуміла, що мій додаток дійсно вийшов зручним та цікавим для користувачів. Всі критичні зауваження було враховано та виправлено, а рекомендації будуть реалізовані в майбутньому.

4.3 Можливі напрямки подальшого розвитку

Після завершення розробки веб-додатка я дійшла висновку, що, попри його зручність і функціональність, існує низка напрямків для подальшого вдосконалення. У перспективі можна реалізувати додаткові можливості, які зроблять систему ще кориснішою для користувачів. На рисунку 4.3 представлено основні ідеї щодо покращення та розширення функціоналу в майбутньому.

Рисунок 4.3 – Перспективи розвитку

1) Мобільна версія додатку:

На поточному етапі додаток функціонує як веб-сайт з адаптивним дизайном для мобільних пристроїв. Однак розробка повноцінного мобільного застосунку для Android або iOS могла б суттєво підвищити зручність

користування. Це дало б змогу користувачам отримувати сповіщення безпосередньо на телефон і мати постійний доступ до календаря, що зробило б роботу з додатком ще більш комфортною та мобільною.

2) Інтеграція з іншими сервісами

Багато користувачів відзначали, що їм хотілося б інтегрувати свій календар з іншими популярними сервісами, наприклад, Google Calendar чи Outlook. Це дозволило б синхронізувати події між різними пристроями, платформами та додатками, а також легко експортувати або імпортувати свої заплановані події.

3) Розширення можливостей нагадувань та сповіщень.

Зараз додаток дозволяє налаштувати прості нагадування. У майбутньому можна додати можливість отримувати нагадування через push-сповіщення на мобільний телефон, електронну пошту чи навіть SMS-повідомлення. Це буде дуже зручно для людей, які активно користуються календарем і не хочуть пропустити важливі події.

4) Аналітика та статистика використання.

Цікавою ідеєю було б додати функції аналітики: наприклад, відстежувати кількість запланованих і виконаних завдань за певний період, показувати статистику досягнення цілей у вигляді простих діаграм або графіків. Це допоможе користувачам краще контролювати свій час та підвищить мотивацію використовувати додаток регулярно.

5) Робота з командами та спільний доступ.

Ще один важливий напрямок розвитку – додати можливість створювати спільні календарі для команд або груп людей. Це особливо корисно для невеликих компаній, студентських груп чи проектних команд. Користувачі могли б разом планувати події, синхронізувати завдання, призначати відповідальних та бачити загальну картину.

6) Вдосконалення безпеки.

У майбутньому варто більше уваги приділити безпеці та конфіденційності користувачів. Наприклад, впровадити більш надійні методи

авторизації, такі як двофакторна аутентифікація, та посилити шифрування даних, які зберігаються на сервері.

Отже, створений додаток має значні перспективи для подальшого вдосконалення. Запропоновані покращення здатні розширити його функціональність, підвищити зручність використання та зробити календар більш адаптованим до потреб різних категорій користувачів.

ВИСНОВКИ

У межах цієї дипломної роботи було створено веб-додаток для формування персоналізованих календарів. Головною метою стало розроблення зручного інструменту для організації особистого часу, який дозволяє користувачам додавати, редагувати та видаляти події, встановлювати нагадування й адаптувати інтерфейс відповідно до власних уподобань. Завдяки послідовному виконанню всіх етапів проєкту поставлену мету було повністю досягнуто.

На початковому етапі було проведено аналіз популярних інструментів, що використовуються для планування — таких як Google Calendar, Outlook, Notion, Trello тощо. Кожне з цих рішень має свої переваги та недоліки. Наприклад, Google Calendar відзначається простотою й широкою популярністю, однак обмежує можливості персоналізації інтерфейсу. Натомість Notion надає великий простір для налаштувань, але має складний інтерфейс, який вимагає часу для освоєння. В результаті аналізу було виявлено потребу у створенні веб-додатка, що поєднує простоту використання з гнучкими можливостями налаштування під індивідуальні потреби користувача.

На наступному етапі було виконано проєктування системи та сформовано перелік функціональних вимог. Основні з них включали можливість додавання й редагування подій, налаштування кольорових тем і фону, створення особистих цілей, реалізацію авторизації, збереження даних користувача та нагадування про заплановані події. Для втілення цих завдань було обрано сучасні технології: React — для побудови інтерфейсу, а для зберігання даних у початковій версії застосовано LocalStorage браузера, що дало змогу значно спростити розробку та забезпечити високу швидкодію й доступність додатка.

Реалізація веб-додатку стала найоб’ємнішою частиною проєкту. У процесі розробки використовувались React-компоненти, що дозволили

ефективно керувати станом додатка та створити динамічний, гнучкий інтерфейс. Було реалізовано ключові елементи, зокрема форму авторизації, календар з можливістю додавання й редагування подій, а також налаштування зовнішнього вигляду — темний режим, вибір кольорових схем, текстур і градієнтів фону. Крім того, впроваджено функції для створення особистих цілей та відстеження їх виконання. У підсумку додаток отримав широкий функціонал, який відповідає основним потребам користувачів.

Для оцінки якості та зручності використання розробленого додатка було проведено тестування за участю потенційних користувачів. До нього долучилися мої одногрупники, знайомі, а також кілька представників офісної сфери та фрілансу. Кожен учасник мав виконати завдання з перевірки основних функцій додатка та надати зворотний зв’язок. Результати виявилися загалом позитивними: більшість респондентів відзначили інтуїтивний інтерфейс, високу зручність у використанні, швидку роботу додатка, а також комфортну навігацію між датами та широкі можливості персоналізації. Серед зауважень згадувалася лише незначна затримка при завантаженні деяких фонових текстур, яка була оперативно усунута.

У процесі оцінювання зручності користування додатком учасники тестування надали низку цінних рекомендацій щодо його вдосконалення. Зокрема, було запропоновано реалізувати синхронізацію з популярними календарними сервісами, такими як Google Calendar, додати функцію експорту даних у форматах PDF або CSV, а також розробити мобільну версію з підтримкою сповіщень. Усі ці ідеї були враховані та включені до планів подальшого розвитку проєкту.

У заключному розділі роботи я окреслила перспективні напрямки подальшого розвитку додатку. На мою думку, пріоритетним є створення окремого мобільного застосунку, оскільки більшість сучасних користувачів планують свій час саме через мобільні пристрої. Важливо також посилити інтеграцію з популярними сервісами для зручної синхронізації подій між різними платформами та пристроями. Крім того, планується розширення

функціоналу нагадувань, зокрема впровадження push-сповіщень і email-інформування, а також розробка інструментів для командної роботи — спільних календарів і завдань для невеликих груп.

За результатами виконаної роботи було підготовлено тези доповіді на тему «Вебзастосунок для створення персоналізованих календарів» для участі в конференції «Modern Information Technologies and Artificial Intelligent Systems MIT@AIS-2025». Матеріали пройшли експертну перевірку та були офіційно прийняті до публікації.

Підсумовуючи виконану роботу, можна зазначити, що розроблений веб-додаток повністю відповідає поставленим цілям. Він відрізняється зручністю та інтуїтивністю, а також надає широкі можливості для персоналізації. Реалізовані функції сприяють ефективному управлінню часом користувачів і покращують організацію їхніх подій та завдань. Робота над дипломом надала цінний практичний досвід у застосуванні сучасних технологій, проектуванні та розробці програмного забезпечення, а також поглибила розуміння створення зручних і корисних продуктів для людей.

Отже, я переконана, що дипломна робота виявилася успішною, а створений додаток має вагомий потенціал для подальшого розвитку і широкого використання. Усі отримані рекомендації та ідеї щодо покращень будуть враховані в подальшій роботі, щоб зробити продукт ще більш якісним та зручним для користувачів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Banks A., Porcello E. Learning React: Functional Web Development with React and Redux. — O’Reilly Media, 2020. — 350 с.

2. Wieruch R. The Road to React: Your journey to master React.js in JavaScript. — Self-published, 2021. — 280 с.

3. Brown E. Web Development with Node and Express. — O’Reilly Media, 2019. — 330 с.

4. Pop F. Firebase Essentials: Streamline Web and Mobile App Development with Firebase. — Independently published, 2022. — 220 с.

5. Mozilla Developer Network (MDN). “Web APIs: localStorage.” URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (дата звернення: 04.04.2025).

6. Документація React. React.js Documentation. URL: https://reactjs.org/docs/getting-started.html (дата звернення: 04.04.2025).

7. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (дата звернення: 04.04.2025).

8. Google Firebase Documentation. URL: https://firebase.google.com/docs (дата звернення: 04.04.2025).

9. Node.js Documentation. URL: https://nodejs.org/en/docs/ (дата звернення: 04.04.2025).

10. Krug S. Don't Make Me Think: A Common Sense Approach to Web Usability. — New Riders, 2014. — 216 с.

11. Duckett J. JavaScript and JQuery: Interactive Front-End Web Development. — Wiley, 2014. — 640 с.

12. Cherny B. Programming TypeScript. — O’Reilly Media, 2019. — 324 с.

13. Lukša M. Cloud Native Applications with Firebase. — Packt Publishing, 2021. — 352 с.

14. Sheiko D. “A Comparative Study of Web Calendar APIs: Google Calendar, Microsoft Outlook, and iCalendar.” — International Journal of Web Engineering, Vol. 12, No. 3, 2021. — 17 с.

15. Atencio L. Functional Web Development with React and Redux. — Manning Publications, 2018. — 275 с.

ДОДАТОК А. ПРОГРАМНИЙ КОД ФАЙЛУ APP.JX

```
// Шифрування/розшифрування паролю
const encryptPassword = (password) =>
btoa(encodeURIComponent(password));
const decryptPassword = (password) =>
decodeURIComponent(atob(password));

// Теми календарів
const CALENDAR_THEMES = {
  default: { primaryColor: '#3498db', secondaryColor: '#2980b9',
textColor: '#2c3e50' },
  sport:   { primaryColor: '#2ecc71', secondaryColor: '#27ae60',
textColor: '#ffffff' },
  habits:  { primaryColor: '#e74c3c', secondaryColor: '#c0392b',
textColor: '#ffffff' }
};

// Завантаження даних користувача після входу
useEffect(() => {
  if (isLoggedIn) {

setEvents(JSON.parse(localStorage.getItem(`events_${userName}`))
|| []);

setReminders(JSON.parse(localStorage.getItem(`reminders_${userNa
me}`)) || []);
    const bg =
JSON.parse(localStorage.getItem(`bgSettings_${userName}`)) || {};
    setBackgroundType(bg.backgroundType || 'color');
    setSelectedColor(bg.selectedColor || '#ffffff');
  }
}, [isLoggedIn, userName]);

// Перевірка нагадувань
useEffect(() => {
  const checkReminders = setInterval(() => {
    const now = new Date();
    const hh = String(now.getHours()).padStart(2, '0');
    const mm = String(now.getMinutes()).padStart(2, '0');
    reminders.forEach(rem => {
      if (rem.enabled && rem.time === `${hh}:${mm}`) {
        new Notification("Нагадування", { body: rem.text });
      }
    });
  }, 30000);
  return () => clearInterval(checkReminders);
}, [reminders]);

// Отримання стилю фону
const getBackgroundStyle = () => ({
```

```
  backgroundColor: backgroundType === 'color' ? selectedColor :
'#ffffff'
});

// Додавання події
const addEvent = () => {
  if (eventText && eventDate && eventTime) {
    setEvents([...events, {
      id: Date.now(),
      text: eventText,
      dateTime: `${eventDate} ${eventTime}`,
      calendar: activeCalendar
    }]);
  }
};

// Реєстрація користувача
const handleRegister = () => {
  if (userName && userPassword) {
    const encrypted = encryptPassword(userPassword);
    localStorage.setItem(`user_${userName}`,     JSON.stringify({
password: encrypted }));
    localStorage.setItem(`events_${userName}`,
JSON.stringify([]));
    localStorage.setItem(`reminders_${userName}`,
JSON.stringify([]));
  }
};

// Вхід користувача
const handleLogin = () => {
  const                          user                          =
JSON.parse(localStorage.getItem(`user_${userName}`));
  if  (user  &&  decryptPassword(user.password)  ===  userPassword)
setIsLoggedIn(true);
};

// Компонент цілей
{showGoals         &&         <Goals         userName={userName}
activeCalendar={activeCalendar} />}
```

ДОДАТОК Б. ПРОГРАМНИЙ КОД ФАЙЛУ APP.CSS

```
/* Загальні стилі */
.App {
  font-family: 'Arial', sans-serif;
  min-height: 100vh;
  padding: 20px;
  color: #333;
  transition: background 0.3s ease;
}

/* Вхід і реєстрація */
.login-container {
  max-width: 400px;
  margin: 50px auto;
  padding: 30px;
  background-color: #fff;
  box-shadow: 0 0 20px rgba(0,0,0,0.1);
  border-radius: 10px;
}

/* Базовий вигляд календаря */
.calendar-app {
  background-color: rgba(255,255,255,0.85);
  border-radius: 10px;
  padding: 20px;
  max-width: 1200px;
  margin: 0 auto;
}

/* Форма додавання подій */
.event-form {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr auto;
  gap: 10px;
  margin-bottom: 30px;
}

/* Календар */
.react-calendar {
  width: 100%;
  border: none;
  border-radius: 10px;
  background-color: white;
  padding: 10px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

/* Список подій */
.events-section {
  background-color: white;
  padding: 20px;
```

```
      border-radius: 10px;
    }
    .event-item {
      background-color: #f8f9fa;
      padding: 15px;
      border-radius: 5px;
      margin-bottom: 10px;
      display: flex;
      justify-content: space-between;
      flex-wrap: wrap;
    }

    /* Секція цілей */
    .goals-section {
      background-color: white;
      padding: 20px;
      border-radius: 10px;
      margin-top: 20px;
    }

    /* Теми календарів */
    .theme-option.sport    { background: linear-gradient(135deg,
#2ecc71, #27ae60); }
    .theme-option.water    { background: linear-gradient(135deg,
#3498db, #2980b9); }
    .theme-option.habits   { background: linear-gradient(135deg,
#e74c3c, #c0392b); }
    .theme-option.custom   { background: linear-gradient(135deg,
#f39c12, #e67e22); }

    /* Налаштування фону */
    .background-settings {
      background-color: rgba(255,255,255,0.95);
      padding: 20px;
      border-radius: 8px;
      margin: 20px auto;
      max-width: 800px;
    }

    /* Градієнти */
    .gradient-color input[type="color"] {
      width: 60px;
      height: 40px;
      cursor: pointer;
    }

    /* Адаптивність */
    @media (max-width: 768px) {
      .event-form {
        grid-template-columns: 1fr;
      }
      .app-header {
```

```
    flex-direction: column;
  }
}

/* Секція нагадувань */
.reminder-settings {
  padding: 15px;
  border-radius: 8px;
  background-color: rgba(255,255,255,0.9);
}

/* Активний календар */
.current-calendar {
  text-align: center;
  padding: 10px;
  border-radius: 5px;
  font-weight: bold;
}

/* Кнопки */
.login-button     { background: #3498db; color: white; }
.register-button  { background: #2ecc71; color: white; }
.logout-button    { background: #e74c3c; color: white; }
.add-button       { background: #9b59b6; color: white; }
.view-month-button { background: #3498db; color: white; }
.background-button { background: #ff9800; color: white; }
.goals-button      { background: #9b59b6; color: white; }
```

ДОДАТОК В. ПРОГРАМНИЙ КОД ФАЙЛУ GOALS.JX

```
// Завантаження цілей із localStorage при вході
useEffect(() => {
  if (userName) {
    const savedGoals = JSON.parse(localStorage.getItem(`goals_${userName}`)) || [];
    setGoals(savedGoals);
  }
}, [userName]);

// Збереження цілей у localStorage
const saveGoals = () => {
  if (userName) {
    localStorage.setItem(`goals_${userName}`, JSON.stringify(goals));
  }
};

// Додавання нової цілі
const addGoal = () => {
  if (!newGoal.title || !newGoal.days || !newGoal.reminderTime) {
    alert('Заповніть усі поля');
    return;
  }
  const goal = {
    id: Date.now(),
    title: newGoal.title,
    totalDays: parseInt(newGoal.days),
    completedDays: 0,
    reminderTime: newGoal.reminderTime,
    startDate: newGoal.startDate,
    calendar: activeCalendar,
    lastCheckDate: null
  };
  setGoals([...goals, goal]);
  setNewGoal({ title: '', days: '', reminderTime: '', startDate: new Date().toISOString().split('T')[0] });
  setShowGoalForm(false);
  setTimeout(saveGoals, 0);
};

// Позначення виконаної цілі на сьогодні
const markGoalCompleted = (id, completed) => {
  const updatedGoals = goals.map(goal => {
    const today = new Date().toISOString().split('T')[0];
    if (goal.id === id && goal.lastCheckDate !== today) {
      return { ...goal, completedDays: goal.completedDays + 1, lastCheckDate: today };
    }
    return goal;
```

```
  });
  setGoals(updatedGoals);
  setTimeout(saveGoals, 0);
};

// 🗑️ Видалення цілі
const removeGoal = (id) => {
  setGoals(goals.filter(goal => goal.id !== id));
  setTimeout(saveGoals, 0);
};
```