Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного засобу обліку фінансових витрат під мобільну платформу Android

Виконав(ла): студент(ка) 4 курсу, групи СПс-41
спеціальності 121 Програмна інженерія
(шифр і назва спеціальності)

_____ Голояд І.В.
(підпис) (прізвище та ініціали)

Керівник _____ Яворська Є.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
« » 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Голояду Івану Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного засобу обліку фінансових витрат під мобільну платформу Android

Керівник роботи Яворська Євгенія Богданівна, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_15__» *квітня* 2024 року № *4/7-343*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Вимоги замовника, технічні умови, технічне завдання

4. Зміст роботи (перелік питань, які потрібно розробити)

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	Лазарюк В.В., доцент кафедри МТ		
основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання		
2	Аналіз завдання		
3	Виконання розділу 1		
4	Виконання розділу 2		
5	Виконання розділу 3		
6	Оформлення пояснювальної записки		
7	Оформлення графічного та презентаційного матеріалу		
8	Попередній захист		
9	Захист		

Студент _____ Голояд І.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яворська Є.Б.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Темою кваліфікаційної роботи на здобуття освітнього ступеню «бакалавр» є: «Розробка програмного засобу обліку фінансових витрат під мобільну платформу Android» написана студентом групи СПс-41 Голоядом Іван Васильовичем.

Кваліфікаційна робота бакалавра містить: 63 с., 38 рис., 5 табл., 38 джер.

У представленій до захисту кваліфікаційній роботі представлено розробку програмного засобу, у вигляді веб-застосунку, для надання послуг користувачам з контролю витрат і дає можливість користувачеві ефективно відстежувати і здійснювати контроль над соїм фінансовим станом.

При розробці веб-додатку для створення інтерфейсу користувача використовувалися сучасні веб-технології, JS, HTML, CSS і React framework. За допомогою них користувач зможе створити новий бюджет, вводити з можливістю подальшого перегляду витрати та надходження.

Програмний інструмент має інтуїтивно зрозумілий інтерфейс,що дозволяє легко орієнтуватися в його функціях. Користувачі можуть створити обліковий запис і увійти в нього, щоб отримати доступ до своїх фінансових даних.

В подальшому розроблений веб-застосунок може бути розширеним та удосконаленим.

Ключовими словами проекту є: розробка, програмний засіб, облік витрат, JavaSkript, HTML, CSS, React.

ABSTRACT

The topic of the qualifying work for obtaining the bachelor's degree is: "Development of a financial expense accounting software tool for the Android mobile platform" written by a student of the SPs-41 group, Ivan Vasyliovych Holoyad.

This qualification work presents the development of a software tool, in the form of a web application, for providing services to users for cost control. This enables users to competently and effectively monitor and control their own spending.

During the development of the web application, modern web technologies, JS, HTML, CSS and the React framework were used to create the user interface. With their help, the user will be able to create a new budget, enter with the possibility of further revision of expenses and income.

The software tool has an intuitive interface that allows you to easily navigate its functions. Users can create an account and log in to access their financial data.

The developed web application can be expanded and improved in the future.

The keywords of the project are: development, software, cost accounting, JavaScript, HTML, CSS, React.

ЗМІСТ

ВСТУП……8
1 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ……10
1.1 Аналізування вимог до програмного засобу……...10
1.2 Опис програмного засобу……10
1.3 Аналіз відомих рішень……..…11
1.3.1 Goodbudget……...……12
1.3.2. Mint……..13
1.3.3. Money Manager……...13
1.3.4. Expensify……14
1.3.5. Підсумок……15
1.4 Технології проєктування та створення програмного засобу……...15
1.4.1 Авторизація користувача……...15
1.4.2 Auth0……16
1.4.3 Clerk……16
1.4.4 Firebase Authentication……17
1.4.5 Прийняття рішення щодо вибору платформи автентифікації…....17
1.5 Технології для розроблення клієнтської частини……..…18
1.5.1 Вибір технології для розроблення клієнтської частини…….…...20
1.6 Обґрунтування вибору системи керування базами даних……..……23
1.7 Використані інструменти при розробленні програмного засобу……26
1.7.1 Налаштування клієнтської частини……..……26
1.7.2 Інструментарій налаштування БД…….……28
1.7.3 Допоміжний інструментарій при налаштуванні клієнської частини…….29
1.8 Висновок до розділу 1……31
2 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ……..…32
2.1 Опис моделі даних…….……..….32

2.1.1. Модель користувача User……..…..34
2.1.2. Модель Budget (бюджет)……36
2.1.3. Модель Expense (витрати)……..36
2.2 Огляд операцій з даними……..…37
2.2.1. Створення бюджету……...…38
2.2.2. Створення витрати…….38
2.2.3. Видалення бюджету……..…39
2.3 Огляд інтерфейсу веб-додатку……39
2.4 Тестування та перевірка функціоналу веб-додатку……..…44
2.5. Висновок до розділу 2……48
3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ……49
3.1.Значення адаптації в трудовому процесі……..49
3.2 Оцінка технологічного процесу, обладнання, щодо умов електробезпеки, безпеки……...51
ВИСНОВКИ……..…54
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ……55
ДОДАТКИ…….59

ВСТУП

Контролювання фінансового стану є важливим у сучасному світі, оскільки його стабільність та правильне управління ресурсом є основою для суспільства. З огояду на швидкий розвиток ІТ та використання Інтернету розроблення програмних засобів фінансового менеджменту є актуальним насьогодні завданням.

Метою цієї кваліфікаційної роботи є розробка та впровадження програмного засобу, який забезпечить ефективне управління власними фінансами та надасть їм практичні та легкодоступні інструменти для реєстрації своїх доходів і витрат.

Дана робота має на меті розв’язати комплекс завдань, конкретно пов’язаних з фінансовим менеджментом:

– Відслідковування доходів і витрат – ретельний контроль надходжень та витрат за категоріями, датами та сумами з метою отримання чіткого уявлення про свої фінансові надходження та прийняття рішення щодо витрат.

– Бюджетування – функціональні можливості для створення та управління бюджетом, а саме використання коштів для різних життєвих цілей, відстеження прогресу та отримання персоніфікеованих порад щодо їх заощадження.

– Аналіз фінансового стану з використанням графічного представлення (графіки, діаграми тощо) та формуванням звітної документації. Це допоможе визначити області, яким потрібно надати більше уваги та відповідно контролю.

– Нагадування та сповіщення – повідомлення щодо регулярних платежів, наближення терміну оплати тощо.

Даний застосунок допоможе мені дотримуватись фінансової дисципліни. Розробивши його, в мене буде потужний інструмент за допомогою якого я зможу керувати своїми фінансами та отримаю більше свободи та незалежності у своїх фінансових справах.

При виконанні поставленого завдання я використав сучасні web-технології та фреймворки для забезпечення надійності, безпеки та простоти використання засобу. Крім того, для забезпечення оптимальної взаємодії з додатком враховано сучасні підходи до дизайну та користувацьких інтерфейсів.

Сподіваюсь, що розроблений програмний засіб для управління фінансами сприятиме зростанню фінансової грамотності користувача та свідомому управлінню фінансами, що призведе до поліпшення його фінансового становища та життєвої стабільності.

1 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналізування вимог до програмного засобу

Успішна реалізація програмного засобу обліку фінансових витрат вимагає чіткого формулювання функціональних і властивих вимог до нього, зокрема відповідність вимог потребам користувача та забезпечення ефективногот керування фінансами.

Основними аспектами, які слід враховувати при розробці вимог є:

Реєстрація та автентифікація користувача, що забезпечить конфіденційність інформації.

Керування бюджетом – встановлення цілей та планування розподілу коштів за категоріями витрат.

Відслідковування надходжень і витрат – введення та відслідковування інформації про надходження та витрати, класифікація транзакцій, перегляд витрат та доступного залишку.

Зрозумілий інтерфейс користувача – простота та зрозумілість інтерфейсу для легкої орієнтації та виконання необхідних операцій.

Вищевказані вимоги дають можливість створення функціонального та зручного засобу з врахуванням потреб користувача для ефективного обліку фінансовим ресурсом. Майбутні версії програми також дозволять вам додавати різні функції до програми.

1.2 Опис програмного засобу

Засіб, що розробляється, є веб-програмою для управління фінансами, яка надає користувачу практичні та ефективні інструменти для управління своїми фінансами. Основна мета його — надати інструменти для відслідковування

надходжень і витрат, планування бюджету, фінансового аналізу та надання корисної статистики.

Програмний засіб з інтуїтивно зрозумілим інтерфейсом, надає легку орієтацію в його функціях. Користувач може створити обліковий запис і увійти в систему, для доступу до своїх фінансових даних [1].

Основними особливостями розроблюваного засобу є:

Додавання доходів і витрат за категоріями – користувач може додавати свої надходження та витарати у створену категорію (для прикладу, харчування, проживання, розваги тощо).

Планування бюджету – створення категорії бюджету та лімітування витрат за нею. Користувач зможе відстежувати свої витрати керуючись запланованим бюджетом.

Аналітика та звіти – створення звітів та графічних реалізацій із статистичними даними про надходження, витрати та інші фінансові показники. Це уможливить аналіз фінансового стану та прийняття обгрунтованих рішень щодо фінансів.

Нагадування та сповіщення – користувачу надходить інформація про необхідність здійснити регулярний платіж, заплановані витрати та інші фінансові зобов’язання.

Засіб розробляється для забезпечення управління фінансами користувача.

Вищевказані функції є базовими для удосконалення програмного засобу з метою задовільнити потреби користувача у плануванні та управлінні його фінансовим станом.

1.3 Аналіз відомих рішень

Як говорить стара приказка, гроші люблять, коли їх рахують. За допомогою сучасних гаджетів облік фінансів здійснюється автоматично за допомогою

відповідних програм, які надають можливість реєстрації, авторизації, введення та отримання фінансових даних. Слід сказати, що додатків українського походження для Android не знайдено.

1.3.1 Goodbudget (https://goodbudget.com/)

Goodbudget розпочався в 2009 році як експеримент Dayspring Technologies, фірми, яка спеціалізується на розробленні Інтернету та мобільних пристроїв у Сан-Франциско. Його метою було створення віртуальних конвертів. Пізніше було створено функції для звірки чекових книжок, рахунків ітповторних транзакцій [2].

Рисунок 1.1 – Вигляд інтерфейсу

Це поширений web-додаток для здійнення фінансового контролю за надходженнями та витратами оснований на envelope system. Він дозволяє користувачу контролювати свої фінанси та розподіляти кошти по віртуальних конвертах за категоріями. У додатку передбачена синхронізація між пристроями, формування звітної інформації для подальшого аналізування стану бюджету. Goodbudget практичний інструмент планування та керування фінансами [3].

1.3.2 Mint

Популярним додатком для управління та контролювання фінансами є Mint (Розробник Laks Inc).

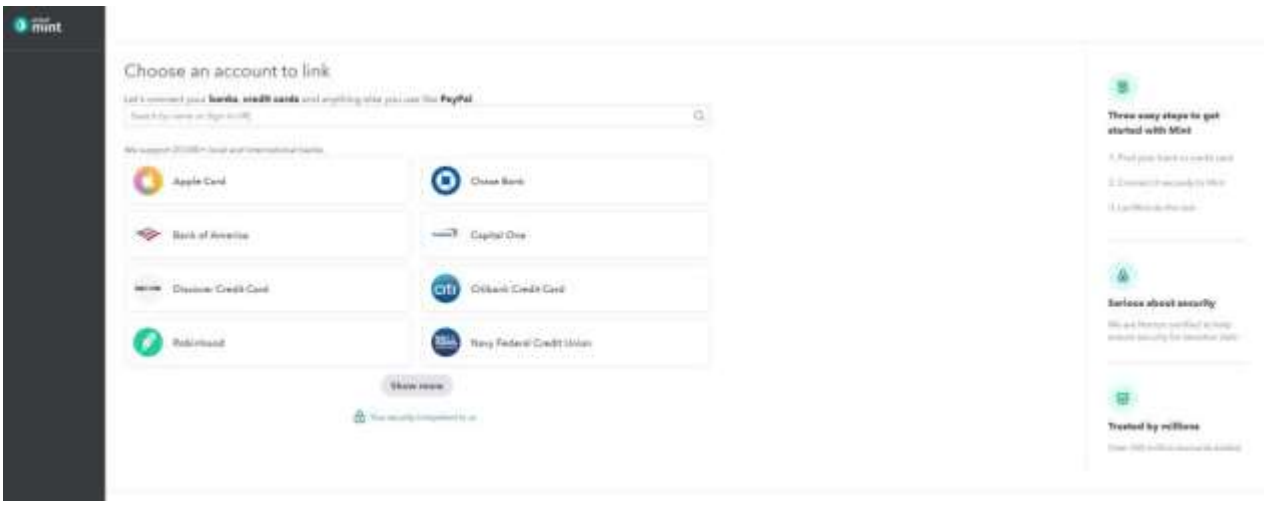


Рисунок 1.2 – Інтерфейс mint

За його допомогою можна не тільки відстежувати свої надходження та витрати, а й отримувати сповіщення про необхідність проведення різного роду фінансових операцій. Mint, базуючись на звичках користувача, формує корисні поради щодо заощадження фінансів.

1.3.3 Money Manager (Корея)

Додаток із доволі зручним і простим інтерфесом, має pro версію, опцію налаштування рахунків, дозволяє експорт даних.

Рисунок 1.3 – Скрін додатку

1.3.4 Expensify

Основна мета Expensify — полегшити контроль витрат за чеком. За фотознімком рахунку програма розпізнає суму та категорію витрат. Інтерфейс програми (англомовний) дозволить поділ фінансів особистих і для ведення бізнесу та формує звітну документацію.

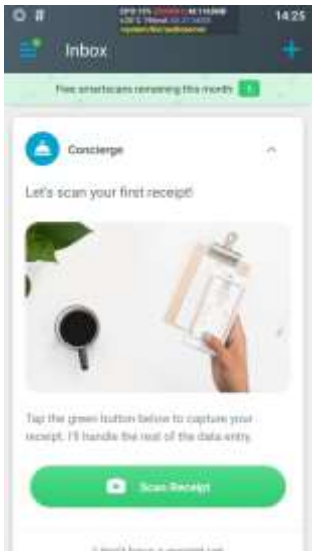


Рисунок 1.4 – Загальний вигляд Expensify

Ця програма автоматизує процес управління витратами, заощаджуючи час і спрощуючи фінансовий облік і є корисним інструментом для керування витратами та ведення фінансування.

1.3.5 Підсумок

Кожному із додатків (наведених у підрозділі 1.3) притаманні свої «плюси» і «мінуси». Так, Goodbudget прекрасно підходить користувачам для планування свого бюджету. Expensify є корисним для пошуку зручного способу відстеження витрат та керування надходженнями. Mint автоматично імпортує фінансові дані та додаткові інструменти керування інвестиціями. Money Manager має простий і зручний інтерфейс, доступний для звичайного користувача.

1.4 Технології проєктування та створення програмного засобу

1.4.1 Авторизація користувача

Важливою складовою web-застосунків є авторизація користувача з метою збереження конфіденційності наданої інформації та особистих даних. Авторизований користувач отримує доступ до визначених функцій та ресурсів. Основними функціями є: процедура реєстрації та створення облікового запису . Для цього користувач (для підтвердження даних) вводить свою електронну пошту та пароль.

Після перевірки відповідності введеної інформації із базою даних користувач отримує доступ до функціональних можливостей програми.

Керування правом доступу. Авторизований користувач отримує доступ до інформації в залежності від його ролі. Так, повний функціональний доступ має адміністратор, а звичайний користувач має обмежений доступ до відповідних ресурсів чи функцій.

Конфіденційність: При авторизації використовуються різного роду методи кодування для запобігання несанкціонованого доступу. До них належать кешування паролів і застосування маркерів автентифікації.

Відновлення пароля. У випадку втраченого паролю, його можна відновити за вказаною, при реєстрації, електронною поштою чи іншим способом підтвердження.

Автентифікація користувача забезпечує безпеку, конфіденційність та контролювння доступу до функціональних можливостей та ресурсу web-додатку.

Використання вищевказаного є необхідним для забезпечення захисту наданої інформації та рівня довіри користувача до допрограмного засобудатку.

Поширеними службами автентифікації користувачів є: Auth0 [14], Clerk [7 , 8], Firebase Authentication [15].

1.4.2 Auth0

Платформа, яка надає готові рішення для розробки сервісів будь-якого рівня складності. Auth0 підтримується командою JWT (JSON Web Token/веб-токен у форматі JSON). Це вселяє певну впевненість у безпеці Auth0-сервісів. Auth0 істотно полегшує виконання нетривіальної задачі розробки сервісу аутентифікації/авторизації веб-додатка.

1.4.3 Clerk

Платформа автентифікації та керування користувачами для веб-застосунків. Вона дозволяє розробникам швидко і безпечно реалізовувати системи реєстрації, входу та управління профілями користувачів, не потребуючи часу на створення власних рішень.

Це проста та безпечна автентифікація користувачів із готовим усім необхідним для забезпечення безпечної взаємодії з користувачами.

1.4.4 Firebase Authentication

Має серверні служби, прості, при використанні, SDK та створені бібліотеки інтерфейсу користувача для його автентифікації у веб-застосунках. Автентифікація підтримується за використанням паролів, номерів телефонів, популярних постачальників Google, Facebook, Twitter та ін.

Автентифікація Firebase фактично інтегрована з іншими службами Firebase і використовує інші стандарти, такі як Auth0 2.0 і Open ID Conect, тому її можна легко інтегрувати з бекендом користувача.

При оновленні Firebase Authentication з Identity Platform відбувається розблокування додаткових функцій, таких як: багатофакторна автентифікація, функції блокування, ведення журналу активності користувачів і аудиту, підтримку SAML і універсального OpenID Connect, мультитенантність таі підтримку на рівні підприємства.

1.4.5 Прийняття рішення щодо вибору платформи автентифікації

Для вибору платформи автентифікації при виконанні кваліфікаційної роботи мною проаналізовано наступні Auth0, FirebaseAuthentication та CIerk.

Враховуючи вимоги та потреби я зупинився на CIerk, оскільки дана платформа уможливлює швидку інтеграцію, має готові компоненти та інтерфейси, що спростить розробку та впровадження функціоналу авторизації. Завдяки цьому нівелюється необхідність створення нових складних кодів і зекономить затрати часу.

Наперед створені елементи інтерфейсу користувача (вхід, реєстрація) дозволять інтеграцію

Попередньо створені елементи інтерфейсу користувача, такі як форми входу та реєстрації, дозволяють швидко та легко інтегрувати функціонал авторизації у розроблюваному програмному засобі.

Застосування Clerk повністю налаштує зовнішній вигляд інтерфесу автентифікації, а це в свою чергу задовільняє мої вимоги щодо дизайну та брендингу та надає сучасний вигляд засобу.

За допомогою цієї платформи гарантована високорівнева безпека та захист даних користувача. Захищені з’єднання, кодування і кешування паролів зберігають конфіденційність даних. Також я можу використати інші способи авторизації: вхід через соціальну мережу чи за номером телефону.

Clerk легко інтегрується із іншими сервісами та інструментами, які я використовую у програмному засобі. А це розширює функціональність засобу та уможливлює використання додаткових функцій.

Для Clerk властива структурована документація та команду розробників, завдяки яким можна вирішити потенційні проблеми та отримати відповідь на поточні запитання.

Готові рішення та налаштування дозволяють швидко впроваджувати функції автентифікації у моєму web-застосунку, що забезпечит зручність та безпечність взаємодії з користувачами.

1.5 Технології для розроблення клієнтської частини

Розроблення клієнтської частини передбачає використання функціональних, ефективних та зовнішньопривабливих технологій. Серед них набули популярності наступні:

– html [10] – основна мова розмітки для web-сторінок, її використовують для структурування та організації контенту, а також для виначення заголовків, абзаців тощо.

– Каскадні таблиці стилів (cascading style sheeds) [11] — мова ієрархічних правил (табюлиць стилів, яка використовується для представлення зовнішнього вигляду документу, що написаний на html чи xml. Вона описує, як елемент має відображуватись.

CSS є одною із базових мов вільної веб розробки, що є стандартизованою специфікацією W3C.

– javascript [4] – безпечна мова прогрмування. Вона не надає низькорівневий доступ до пам’яті чи процесора, оскільки була створена для браузерів, які цього не потребують.

В браузері javascript є доступним все, що пов’язане із маніпулюванням веб-сторінками, взаємодією користувача та веб-сервера.

– React. js - це JavaScript-бібліотека для створення інтерфейсів сайтів та програм. З його допомогою створюються інтерактивні елементи – кнопки, віджети, чати, картки товарів тощо. React це не фреймворк як такий. Фреймворк включає набір бібліотек, правил та інструментів для створення сайтів або програм, тобто фреймворк – це комплексне рішення. React. js – це лише бібліотека, тобто частина фреймворку.

Головна особливість React в тому, що він дозволяє працювати з окремими компонентами інтерфейсу, за рахунок чого сайти та додатки стають стабільнішими. Кожен створений на Реакті елемент залежить від свого стану, і якщо одна опція перестане працювати коректно, на інші це не вплине. Це

перевага роботи з React перед написанням сайту на чистому JavaScript. Сторінка, написана на JS, монолітна, тобто її компоненти залежать один від одного, і якщо одна функція зламається, це вплине на інші.

Ще одна особливість елементів, створених на Реакті, їх не обов'язково прописувати заново щоразу, коли потрібно додати опцію. Кожен створений на Реакті компонент можна дублювати та додавати до інших розділів та інших сторінок. Це стає можливим, тому що кожен написаний за допомогою React компонент містить усі необхідні дані та методи для реалізації конкретної функції.

Компоненти React. js пишуться на JSX. Це розширення мови JavaScript, а виглядає він як суміш JS та HTML. Він потрібний саме для того, щоб працювати з різними об'єктами на сторінці та робити їх менш залежними від стану інших об'єктів. При цьому об'єкти, які виглядають у коді як HTML, залишаються синтаксисом JSX. Тому розробники користуються платформами веб-розробки за допомогою JavaScript-коду, наприклад Node.js для роботи з React.

– Angular (також відомий як «Angular 2+») [18] — це безкоштовний фреймворк односторінкових веб-додатків із відкритим кодом на основі TypeScript, який працює на Node.js. Він розроблений Google і спільнотою окремих осіб і корпорацій. Angular — це повністю переписана та сама команда, яка створила AngularJS. Екосистема Angular складається з різноманітної групи з понад 1,7 мільйонів розробників, авторів бібліотек і творців контенту. Згідно з опитуванням розробників Stack Overflow, Angular є одним із найбільш часто використовуваних веб-фреймворків, поступаючись React.

Google розробив Angular як фундаментальний перепис AngularJS. На відміну від AngularJS, Angular не має поняття «області» або контролерів; замість цього він використовує ієрархію компонентів як свою основну архітектурну характеристику. Angular має інший синтаксис виразу, зосереджений на «[ ]» для зв’язування властивостей і «( )» для зв’язування подій. Angular рекомендує використовувати мову Microsoft TypeScript, яка вводить такі функції, як статичний тип, генерики та анотації типів.

– Vue.js – це платформа JavaScript для створення інтерфейсів користувача. Він створений на основі стандартних HTML, CSS і JavaScript і забезпечує декларативну модель програмування на основі компонентів, яка допомагає вам ефективно розробляти інтерфейси користувача будь-якої складності.

Декларативне відображення: Vue розширює стандартний HTML синтаксисом шаблону, який дозволяє нам декларативно описувати вивід HTML на основі стану JavaScript.

Реактивність: Vue автоматично відстежує зміни стану JavaScript і ефективно оновлює DOM, коли відбуваються зміни.

Vue — це фреймворк та екосистема, яка охоплює більшість загальних функцій, необхідних для розробки інтерфейсу. Але Інтернет надзвичайно різноманітний – речі, які ми створюємо в Інтернеті, можуть кардинально відрізнятися за формою та масштабом. Зважаючи на це, Vue розроблено таким чином, щоб бути гнучким і поступово адаптуватися. Залежно від вашого випадку використання Vue можна використовувати різними способами:

- Покращення статичного HTML без етапу створення./
- Вбудовування як веб-компонентів на будь-якій сторінц.
- Односторінкова програма (SPA).
- Fullstack / Server-Side Rendering (SSR).
- Jamstack / Генерація статичного сайту (SSG).
- Орієнтація на комп’ютер, мобільний пристрій, WebGL і навіть термінал.

Перераховані технології дають можливість розробнику створити потужну, привабливу та ефективну клієнтську частину web-застосунку. В залежності від потреб проєкту та зручності його розробки, функціональних вимог і продуктивності обирається відповідна технологія.

1.5.1 Вибір технології для розроблення клієентської частини

Перелічені далі у підрозділі 1.5 технології є поширеними бібліотеками чи фреймворками.

Для проєктів в яких передбачено компонентний підхід, зручний синтаксис використовують Vue.js.

Для розроблення web-застосунків повноцінним фреймворком є Angular, який має достатній набір інструментів та функціоналів – директив, сервісів, модулів. Базований Angular на TypeScript [3] та є строгоструктурованим і типізованим. Застосовується для складних проєктів і підтримується Google.

Для розроблення інтерфейсу користувача використовують бібліотеку JavaScript React, який забезпечує поділ інтерфейсу на невеликі компоненти, які повторно використовуються. Реакт забезпечує продуктивність, віртуальний домен та легку інтеграція із іншими бібліотеками.

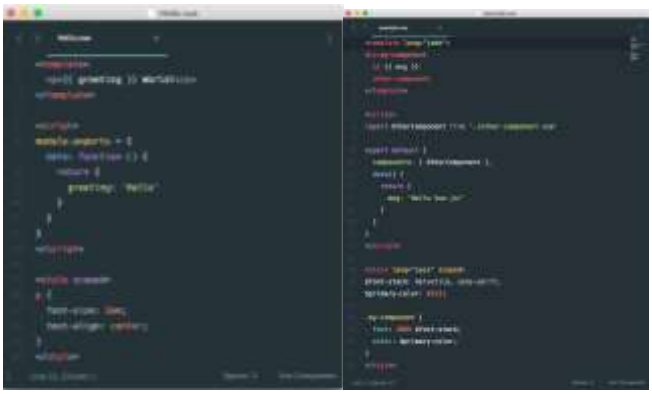


Рисунок 1.5 – Компоненти Vue.js

```
import { Component, OnInit,Input } from '@angular/core';
@Component({
  selector: "lib-testlibrary",
  template: `<div class="header">
  <ng-content select=".header" ></ng-content>
        </div>`,
  styles: [
    `.header { color: blue}`
  ]
})
export class TestlibraryComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {}
}
```

Рисунок 1.6 – Вид компоненту Angular

Рисунок 1.7 – Зразок коду Angular

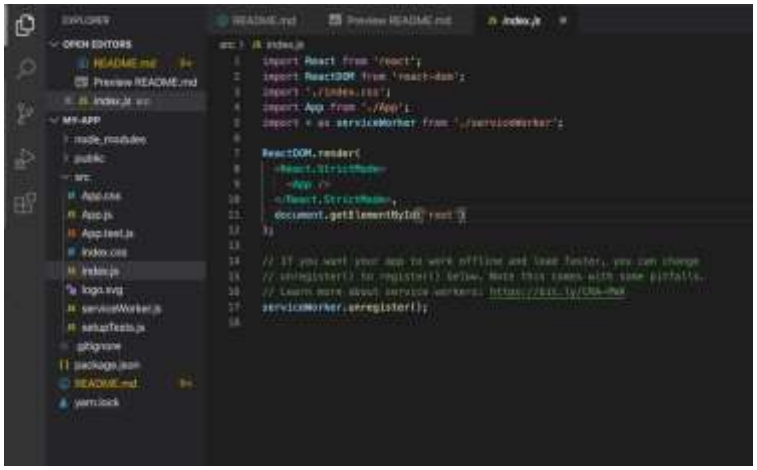


Рисунок 1.8 – Вигляд компоненти React.js

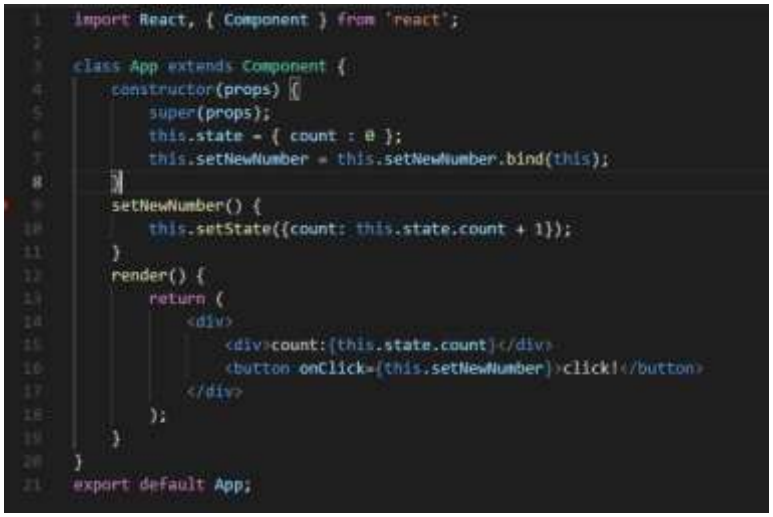


Рисунок 1.9 – Зразок коду React.js

Для виконання кваліфікаційної роботи я вибрав React.js. Оскільки він є популярним та має підтримку від спільноти розробників, забезпечений документацією та доступом до різних ресурсів.

За рахунок віртуального домену Реакт забезпечує ефективність керування станом програмного засобу та односторонній потік даних. Також він уможливлює створення швидких інтерфейсів реагуючих на зміну стану без потреби перерендерингу дерева компонент.

Завдяки можливості розроблення компонент, з подальшим повторним їх використанням, React.js значно економить затрати часу підчас розробки та підтримки засостосунку.

Така технологія має потужний інструментарій для створення інтерфейсу користувача мого web-застосунку.

1.6 Обґрунтування вибору системи керування базами даних

Для виконання кваліфікаційної роботи як базу даних я використовую Firebase. Підставою для мого вибору є те, що платформа легка у використанні та встановленні. Також вона є базою даних NoSQL і є зручною для користувача. Такого роду бази даних, на відміну від SQL, для якої нормою є таблиці і рядки, тут використовуються колекції із переліком документі без таблиць.

Firebase часто оновлюється і нові функції є доступними. Її можна використовувати для обробки авторизації, сповіщень, зберігання файлів та інших функцій, які можуть знадобитися вашим застосункам. А оскільки Google часто оновлює Firebase, нові функції завжди доступні. Взаємодія між Firebase Console та web-застосунком є проста завдяки консолі.

Зручність API уможливлює швидке підключення до БД і виконання операцій зберігання, отримання, оновлення та видалення.

У Firebase є набір сервісів і рішень, таких як CloudFirestore, Realtime Database, FirebaseAuthentication, CloudStorage, CloudMessaging, та Firebase MachineLearning тощо.

Основними відмінностями між CloudFirestore і Realtime DataBase є моделі даних і масштабованість.

Рисунок 1.10 – Відмінності між Cloud Firestore і Realtime Database

У більшості випадків працює CloudFirestore, оскільки її використання є частішим.

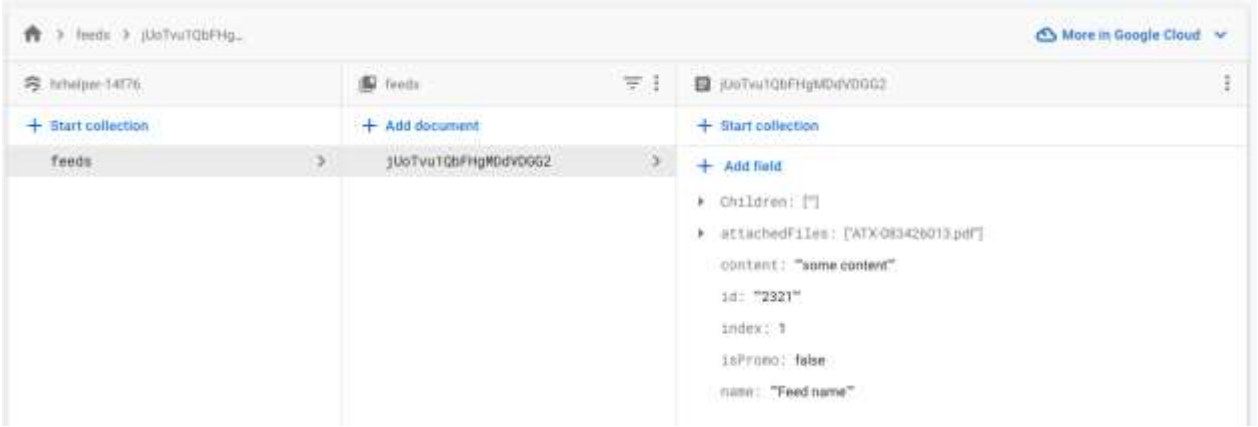


Рисунок 1.11 – БД на платформі Firestore

Однією із цікавих функцій Firestore є конструктор запитів, за допомогою якого здійснюється пошук за певними критеріями.

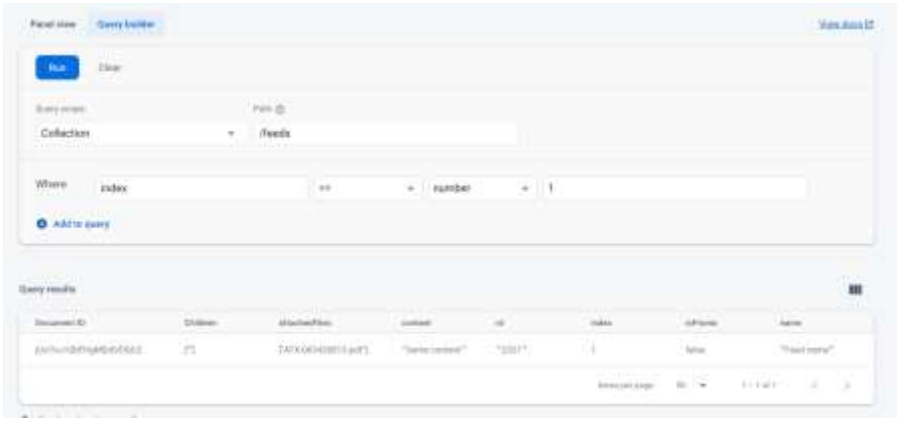


Рисунок 1.12 – Процедура пошуку

Для збереження файлів та керування ними Firebase має функцію сховища.

Рисунок 1.13 – Функція сховища

При необхідності запуску коду мобільного бекенду без керування серверами використовується CloudFunction. Ця технологія використовує JS-код, який буде виконаний в середовищі Node.js. Наприклад, при реєстрації нового користувача, за допомогою даної функції, відбувається посилання push-повідомлення.

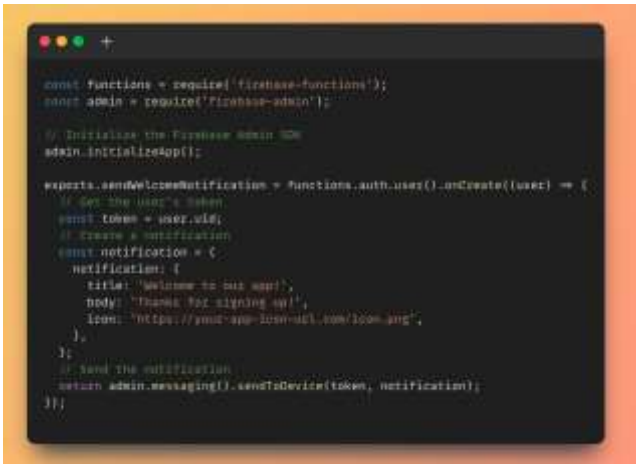
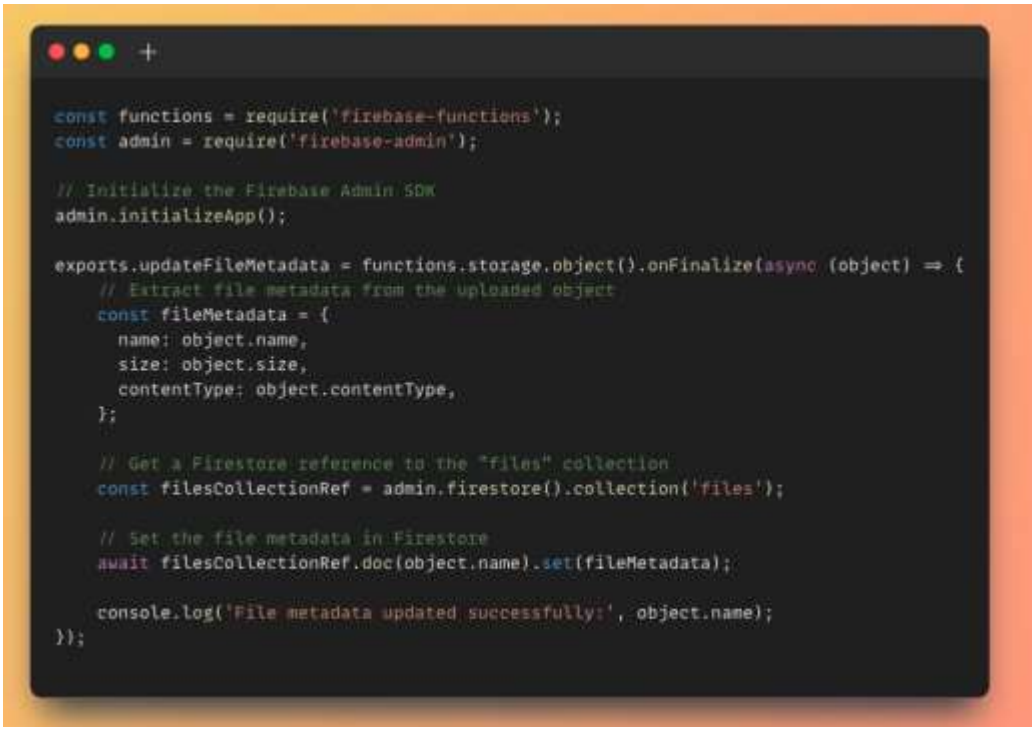


Рисунок 1.14 – Приклад реєстрації нового користувача



```
const functions = require('firebase-functions');
const admin = require('firebase-admin');

// Initialize the Firebase Admin SDK
admin.initializeApp();

exports.updateFileMetadata = functions.storage.object().onFinalize(async (object) => {
    // Extract file metadata from the uploaded object
    const fileMetadata = {
      name: object.name,
      size: object.size,
      contentType: object.contentType,
    };

    // Get a Firestore reference to the "files" collection
    const filesCollectionRef = admin.firestore().collection('files');

    // Set the file metadata in Firestore
    await filesCollectionRef.doc(object.name).set(fileMetadata);

    console.log('File metadata updated successfully:', object.name);
});
```

Рисунок 1.15 – Приклад оновлення даних

Для виконання авторизації використовується FirebaseAuth. Да ний сервіс автнентифікації є безпечним і простим у використанні. Він підтримує багато платформ і може інтегруватися з іншими службами FireBase.

Також, у ньому є рішення і сервіси, які є допоміжними при створенні програмного засобу.

1.7 Використані інструменти при розробленні програмного засобу

1.7.1 Налаштування клієнтської частини

Одними із популярних інструментів за допомогою яких можна швидко налаштувати та розробити клієнтську частину web-застосунку за використанням Реакт є CRA (Create React App) [13] і Vite [9].

CreateReac tApp – це інтерфейс командного рдка (ComandLineInterface, CLI), який дозволяє створювати попередньо налаштовані Реакт-проекти [1].

Vite – це інтструмент, який дає можливість проєкту швидко стартувати з основного шаблону для популярних фреймворків, налаштовує середовище розробки та зборки проекту. Крім того, - це сучасний збирач модулів (bundler) для JavaScript-додатків, менш кастомізований, ніж Webpack, але набагато швидший.

Створення шаблону.

```
yarn create vite [project-name] --template [template-name]
```

- [project-name] - назва проекта
- [template-name] - використовуваний шаблон
-

Використання Vite вбудованих можливостей браузера при завантаженні модулів значно скорочує час зборки иа перезавантаження сторинок при розробці. Тому, я його обрав. Мій вибір зумовлений:

– швидким часом ініціалізації проекту та швидким перезавантаженням сторінки при розробці, що забезпечує ефективність моєї роботи та швидкі результати;

– підтримка нативних ES модулів, що зумовлює простоту роботи модульної системи та використання сучасного синтаксису імортування та експортування;

– додаткові функції пристутні у Vite , а саме HMR (HotModulReplacement) І швидкість перезавантаження компонентів уможливлюють спрощення розробки і покращення продуктивності в ході роботи;

– постійна підтримка спільноти за потреби.

В загальному, мій вибір оснований на його характеристиках, що дає можливість розробити ефективну та зручну клієнтську частину.

1.7.2 Інструментарій налаштування БД

При виконанні кваліфікаційної роботи за основу для БД мною обрано FireStore – рішення Firebase для роботи із базами даних NOSQL [6]. Ця БД пропонує синхронізацію даних в реальному часі та клієнтські SDK для web- і мобільних платформ.

Це дасть мені можливість зберігати, отримувати та оновлювати дані з любого місця у Реакт-застосунку.

1.7.3 Допоміжний інструментарій при налаштуванні клієнської частини

У кваліфікаційній роботі мною використано Type Script [3] і ReactRouter. Розглянемо ці інструменти докладніше.

1) TypeScript це «вид» або «варіант» JavaScript. Зв’язок між TypeScript (TS) і JavaScript (JS) досить унікальний серед сучасних мов програмування.

Деякі мови взагалі не дозволяють запускати програми з помилками. Статична перевірка дозволяє виявити помилки в коді без його запуску. TypeScript здійснює перевірку програми на присутність помилок перед її виконанням. Це відбувається на основі типів значень, так звана статисна перевірка типів.

TypeScript — це мова, що є надмножиною JavaScript: тому синтаксис JS є правомірним TS. Синтаксис означає спосіб написання тексту для створення програми.

TypeScript не вважає жодний код JavaScript помилкою через його синтаксис. Він добавляє правила щодо використання різних типів значень. Засіб перевірки типів TypeScript розроблено, щоб дозволити правильні програми, виявляючи якомога більше поширених помилок.

При переміщенні коду з файлу JS у файл TS, то в залежності від написаного коду видно помилки типу. Це можуть бути правомірні проблеми з кодом або занадто консервативний TypeScript.

TypeScript також зберігає, під час виконання, поведінку JavaScript, а саме, при переміщенні коду із JavaScript у TypeScript, він гарантовано працюватиме так само, навіть якщо TypeScript вважає, що код містить помилки типу.

Збереження при виконанні поведінки, як у JavaScript, є основню задачею TypeScript, оскільки є можливість легкого переходу між 2 мовами, не зважаючи на незначні відмінності, які можуть вплинути на роботу програми.

TypeScript — це середовище виконання JavaScript із засобом перевірки типів на мамент компіляції. Сортування списку у TypeScript таке саме, як і в JavaScript.

Як тільки компілятор TypeScript завершить перевірку коду, він стирає типи, щоб створити «скомпільований» код – після компіляції коду у отриманому звичайному коді JS буде відсутня інформація щодо типу. Це вказує на те, що TS не змінить поведінку програми на базі встановлених типів. Суть в тому, що хоча можна бачити помилки типу при компіляції, сама система типів не впливає на роботу програми під час її виконання.

TypeScript не надає жодних додаткових бібліотек виконання. Розробленими програмами використовуютьс стандартні ббліотеки, так само к і програми JS.

2) На сьогоднішній день react-router – це, можна сказати, монополіст у світі React. Якщо в проекті потрібен роутинг, то, швидше за все, вибір паде саме на цю бібліотеку.

В React Router існує три категорії компонентів:

– роутер (routers) – <Browser Router >, <Hash Router > тощо;
– маршрут (Route Matchers) –< Route >, < Switch >;
– навігація (Navigation) – < Link >, <Nav Link >, <Redirect>.

Всі компоненти, які використовуються у web-застосунку, повинні імпортуватись з react-router-dom.

Будь-яка маршрутизація починається з роутера. Для web -проектів react-router-dom надає <BrowserRouter> та <HashRouter>. Основна відмінність між ними полягає у способі зберігання URL та взаємодії з сервером.

<BrowserRouter> використовує звичайні URL-адреси. У цьому випадку URL виглядають як звичайно, але потрібне певне налаштування сервера. Зокрема сервер повинен обслуговувати всі сторінки, що використовуються клієнтом.

CreateReactApp підтримує це в режимі розробки та містить інструкції для правильного налаштування сервера.

<HashRouter> зберігає поточну локацію в хеш-частині URL (після символу "#"), тому URL виглядає приблизно так: http://example.com/#/your/page. Оскільки хеш не надсилається серверу, його спеціальне налаштування не потрібно.

Існує два види компонентів для пошуку збігів із URL: Switch та Route. Під час рендерингу <Switch> визначає <Route>, що відповідає поточному URL. При виявленні такого маршруту він рендериться, інші маршрути ігноруються. А це значить, що потрібно більш специфічні маршрути розміщувати перед менш специфічними.

React Router надає компонент <Link> для створення посилань у програмі. При використанні <Link> у HTML рендерується <a>.

<NavLink> - це спеціальний тип <Link>, що дозволяє визначати стилі активного стану посилання.

Для примусової навігації використовується <Redirect>. Під час рендерингу <Redirect> виконується перенаправлення.

Використання Type Script і ReactRouter покращить якість та функціональність застосунку, спростить процес розробки. Це зручні інструменти при роботі із маршрутизацією і типізацією.

Я вибрав простий та доступний для ряду ОС редактор коду VisualStudio (VSCode). Він має вбудовану підстримку JavaScript, TypeScript, Node.js та має розширення для C++б C#, Python, php та ін. VSCode дає можливість простого написання, рудагувня та форматування коду. Він сприяэ швидкому створенню проекту та його файлової структури. Також висвітлює синтаксис коду та автоматично коригує помилки.

1.8 Висновок до розділу 1

У розділі визначено мету кваліфікаційної роботи, визначено основні вимоги та проведено огляд відомих додатків обліку фінансів та визначено переваги для кожного.

Проаналізовано та вибрано технології та інструменти для розроблення програмного засобу обліку фінансових витрат. Та роведено аналіз та вибір кож враховано вимоги, користувацькі потреби та переваги.

Для розроблення фронтенда мною вибрано React.js – для побудови швидкого, ефективного і масштабованого інтерфейсу користувача, разом із Type Script – для покращення організації коду, виявлення помилок на початкоих етапах розробки та надання переваги статичної типізації.

Декларативний підхід ReactRouter уможливив використати його для керування маршрутизацією, що дозволить зручне налаштування навігації між сторінками та гнучкості відображення вмісту.

Контроль авторизації здійснено за допомогою сервісу Clerk.

Для забезпечення швидкого та масштабованого збереження та синхронізації даних обрано БД Firebase Firestore.

Також для швидкого старту проекту з основного шаблону вибрано Vite.

Описані у розділі технології дадуть можливість процес розробки зробити надійним та зручним. Прийняті рішення забезпечать створення сучасного, функціонального і зручного при використанні web-застосунку, що задовільнить потреби користувачів при обліку їх фінансових витрат.

2 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

2.1 Опис моделі даних

Модель даних відноситься до абстрактного представлення структур даних, які використовуються для організації та керування даними в базі даних або інформаційній системі. Вона визначає зв’язок між елементами даних (що представляють об’єкти реального світу) і те, як вони організовані, збережені та отримані [20-26].

Моделі даних використовуються для створення баз даних і сховищ даних, керування даними для аналітичної обробки та впровадження програм, які дозволяють користувачам отримувати доступ до інформації значущими способами.

Моделі даних є основою системи та бази даних; вони не тільки зберігають дані користувача, але й допомагають забезпечити точність і послідовність цих даних, визначаючи, як різні дані пов’язані у візуальних формах. Це дозволяє користувачам легко зрозуміти, як об’єкти моделі даних взаємопов’язані.

Під час створення моделі даних визначаються структури даних і стандарти. Використання одного терміна для кожного об’єкта полегшує відображення даних і їх узгодженість у всій організації. Це покращує співпрацю між діловими та технічними командами, сприяючи кращому спілкуванню та розумінню.

Модель даних гарантує високу якість і ефективність програм для кінцевих користувачів, знижуючи ризик збоїв і полегшуючи технічне обслуговування. Завдяки частковому тестуванню моделей даних під час розробки можна виявити помилки та проблеми на ранній стадії, що призведе до зниження витрат і мінімізації ризику простоїв або збоїв у кінцевих користувачів.

Модель даних містить деякі ключові компоненти:

Сутності — це об’єкти, які ми хочемо представити в нашій моделі даних і зазвичай представлені таблицею. Приклади сутностей включають клієнтів, замовлення, продукти або категорії.

Атрибути – відображаються у вигляді стовпців у певних таблицях. Приклади включають ідентифікатор продукту та назву продукту (для Продукту), ідентифікатор категорії та назву категорії (для Категорії).

Записи – відображаються в рядках кожної таблиці.

Відносини – визначають асоціації між сутностями.

Кардинальність – означає числове співвідношення між двома сутностями в БД.

Первинний ключ – унікальний ідентифікатор кожного запису в поточній таблиці. Він відноситься до стовпця або комбінації стовпців, які мають унікальне значення для кожного рядка.

Зовнішній ключ – стовпець або група стовпців, які вказують на первинний ключ в іншій таблиці.

Після визначення окремих компонентів моделі даних наступним кроком є збірка їх у комплексну модель даних. Різні типи моделей даних зазвичай створюються перед кінцевою моделлю.

У моделюванні даних розрізнять три основні типи моделей даних, які створюються послідовно, і кожна будується на основі попередньої.

Концептуальна модель, зазвичай, будується як план майбутньої системи БД. Розроблена на основі збору інформації від зацікавлених сторін, вона забезпечує огляд усієї системи на високому рівні та визначає її компоненти та правила в термінах бізнесу. Для її побудови потрібно визначити сутності, які мають бути включені в модель та встановити їх зв’язки.

Логічна – надає більше деталей про сутності, атрибути, зв’язки систнми. Вона використовується для розробки схеми БД та допомогає встановити і зібрати всі вимоги до системи, та зрозуміти, які дані проходять у бізнес-процесах

Фізична модель даних – детальне представлення системи, має всі фізичні структури для технічної реалізації БД і може реалізуватись у системі керування БД готовій для збереження даних та підтримки запитів. Вона створена розробниками БД і несе інформацію щодо типів даних, індексування та методів доступу.

Процес побудови моделей даних з метою покращення розуміння, керування і використання називають моделюванням даних. Його використовують для розроблення схеми БД, інформаційної системи чи програмного забезпечення [20, 21, 22, 23, 24, 25, 26].

2.1.1. Модель користувача User.

Опис отримано з використанням сервісу для авторизації користувачів Clerk:

Таблиця 2.1 – Опис моделі User

Назва поля	Тип	Опис
id	string	**Унікальний ідентифікатор користувача.**
firstName	string | null	**Ім'я користувача.**
lastName	string	**Прізвище користувача.**
fullName	string | null	**Повне ім'я користувача.**
username	string | null	**Ім'я користувача (логін).**
profileImageUrl	string | null	**URL-адреса зображення профілю користувача.**
primaryEmailAddress	EmailAddress | null	**Інформація про основну електроннуадресу користувача.**
primaryEmailAddressId	string | null	**Унікальний ідентифікатор EmailAddress, який встановлено якосновний.**
emailAddresses	EmailAddress[]	**Масив об'єктів EmailAddress, пов'язаних з користувачем. Включаєосновну.**
primaryPhoneNumber	PhoneNumber[] | null	**Інформація про основний номертелефону користувача.**
primaryPhoneNumberId	string | null	**Унікальний ідентифікатор PhoneNumber, який встановлено якосновний.**
phoneNumbers	PhoneNumber[]	**Масив об'єктів PhoneNumber, пов'язаних з користувачем. Включаєосновний.**
primaryWeb3WalletId	string | null	**Унікальний ідентифікатор Web3Wallet, з яким користувач зареєструвався.**

web3Wallets	web3Wallets[]	**Масив об'єктів Web3Wallet, пов'язаних з користувачем. Включає основний.**
externalAccounts	ExternalAccount[]	**Масив об'єктів ExternalAccount, пов'язаних з користувачем через OAuth.**
passwordEnabled	boolean	**Булеве значення, що вказує, чи має користувач пароль на своєму обліковомузаписі.**
publicMetadata	{[string]: any} | null	**Метадані, які можуть бути зчитані з FrontendAPI та Backend API та можуть бути встановлені лише з Backend API.**
privateMetadata	{[string]: any} | null	**Метадані, які можуть бути зчитані тавстановлені лише з Backend API.**
unsafeMetadata	{[string]: any} | null	**Метадані, якіможуть бути зчитаніта встановлені з фронтенду. Один зі способів використання цього атрибута - реалізація власних полів, які будуть приєднані до об'єкта користувача. Будь ласка, зверніть увагу, що також існує атрибут unsafeMetadata у об'єкта SignUp. Значення цього поля буде автоматично скопійовано внебезпечні метадані користувача після завершення процесу реєстрації.**
lastSignInAt	Date	**Дата останнього входу користувача. Можебути порожнім, якщо користувач ніколи невходив.**
createdAt	Date	**Дата створення користувача.**
updatedAt	**Date**	**Дата останнього оновлення даних користувача.**

web3Wallets	web3Wallets[]	**Масив об'єктів Web3Wallet, пов'язаних з користувачем. Включає основний.**
externalAccounts	ExternalAccount[]	**Масив об'єктів ExternalAccount, пов'язаних з користувачем через OAuth.**
passwordEnabled	boolean	**Булеве значення, що вказує, чи має користувач пароль на своєму обліковомузаписі.**
publicMetadata	{[string]: any} | null	**Метадані, які можуть бути зчитані з FrontendAPI та Backend API та можуть бути встановлені лише з Backend API.**
privateMetadata	{[string]: any} | null	**Метадані, які можуть бути зчитані тавстановлені лише з Backend API.**
unsafeMetadata	{[string]: any} | null	**Метадані, якіможуть бути зчитаніта встановлені з фронтенду. Один зі способів використання цього атрибута - реалізація власних полів, які будуть приєднані до об'єкта користувача. Будь ласка, зверніть увагу, що також існує атрибут unsafeMetadata у об'єкта SignUp. Значення цього поля буде автоматично скопійовано внебезпечні метадані користувача після завершення процесу реєстрації.**
lastSignInAt	Date	**Дата останнього входу користувача. Можебути порожнім, якщо користувач ніколи невходив.**
createdAt	Date	**Дата створення користувача.**
updatedAt	**Date**	**Дата останнього оновлення даних користувача.**

2.1.2. Модель Budget (бюджет).

Таблиця 2.2 – Опис моделі Budget

Назва поля	Тип	Опис
id	string	Генерується за допомогою функції crypto.randomUUID() і використовується для ідентифікації конкретного бюджету у системі.
name	string | null	Використовується для ідентифікації і відображення назви бюджету в інтерфейсі користувача.
currency	string | null	Валюта бюджету. Використовується для визначення валюти, в якій зберігаються та відображаються суми грошей в бюджеті. Наприклад, USD, EUR, UAH тощо.
amount	Date	Сума грошей в бюджеті. Використовується для відстеження загальної суми грошей, які доступні в рамках даного бюджету.
color:	string | null	Генерується за допомогою функції generateRandomColor() і використовується для візуального відрізнення різних бюджетів у користувацькому інтерфейсі. Наприклад, кожен бюджет може мати власний унікальний колір для легкого розпізнавання.
createdAt	Date	Дата і час створення бюджету. Генерується за допомогою функції Date.now() і використовується для фіксації моменту створення бюджету.
updatedAt	Date	Дата останнього оновлення даних користувача.

Дану модель я використаю для вибраного бюджету. Також передбачено можливість створення нового.

2.1.3. Модель Expense (витрати).

Таблиця 2.3 – Опис моделі «Expense»:

Назва поля	Тип	Опис
id	string	Унікальний ідентифікатор витрат. Згенерований за допомогою цієї самої вбудованої функції crypto.randomUUID() модуля crypto і використовується для ідентифікації конкретної витрати у системі.
name	string | null	Використовується для ідентифікації і опису витрати. Наприклад, «Покупка продуктів», «Оплата комунальних послуг» тощо
amount	Date	Використовується для відстеження суми грошей, витрачених на дану позицію
budgetId	Date	Ідентифікатор бюджету, до якого належить витрата. Використовується для встановлення зв'язку між витратою і відповідним бюджетом, до якого вона відноситься.
createdAt	Date	Дата і час створення витрати. Генерується за допомогою вбудованої функції Date.now() модуля Date і використовується для збереження моменту створення витрати
updatedAt	Date	Дата останнього оновлення даних користувача.

Модель використано для відображегння конкретних витрат з метою відстеження їх та подальшого аналізу із відповідними висновками.

2.2 Опрацювання даних

Створення бюджету є важливою складовою web-застосунку для обліку витрат. Під час розробки web-застосунку для обліку витрат, створення бюджету передбачає наступні кроки:

1. Фінансова ціль	Перед створенням бюджету користувачі повинні визначити фінансові цілі, які допоможуть визначити необхідні суму і встановити реалістичний бюджет.
2. Фінансовий стан	Для ефективного управління фінансами користувачам необхідно чітке розуміння свого фінансового становища. Веб-додаток має надавати можливість введення фінансових даних ьа імпортувати їх з різних джерел фінансування
3. Бюджетна категорія	Щоб керувати фінансами, користувач має виділити категорії бюджету, з врахуванням відповідності фінансового забезпечення їх. До них відносяться харчування, проживання, транспорт і розваги тощо. Передбачено можливість розподіляти конкретні суми для цих категорій з подальшим відстеженням.
4. Моніторинг та аналіз	Функціонал, який надає можливість користувачу здійснювати перегляд фінансового звіту для розуміння ситуації щодо свого фінансового стану.

```
export const createBudget = async ({
  name,
  amount,
  currency,
}: {
  name: string;
  amount: string;
  currency: string;
}) => {
  const color = await generateRandomColor();
  const newItem = {
    id: crypto.randomUUID(),
    name,
    createdAt: Date.now(),
    currency,
    amount: +amount,
    color,
  };
  await addDoc(collection(db, 'budgets'), newItem);
  await dashboardLoader();
};
```

Рисунок 2.1 – Створення бюджету

Далі розглянемо можливість створювати дані користувача у вашій веб-програмі.

Створення витрат також є важливою функцією веб-додатків для управління фінансами. Ця функція дозволяє користувачам вводити та відстежувати фінансові

витрати, керувати та аналізувати бюджети. Під час розробки веб-додатку для управління фінансами створення витрат передбачає виконання таких кроків:

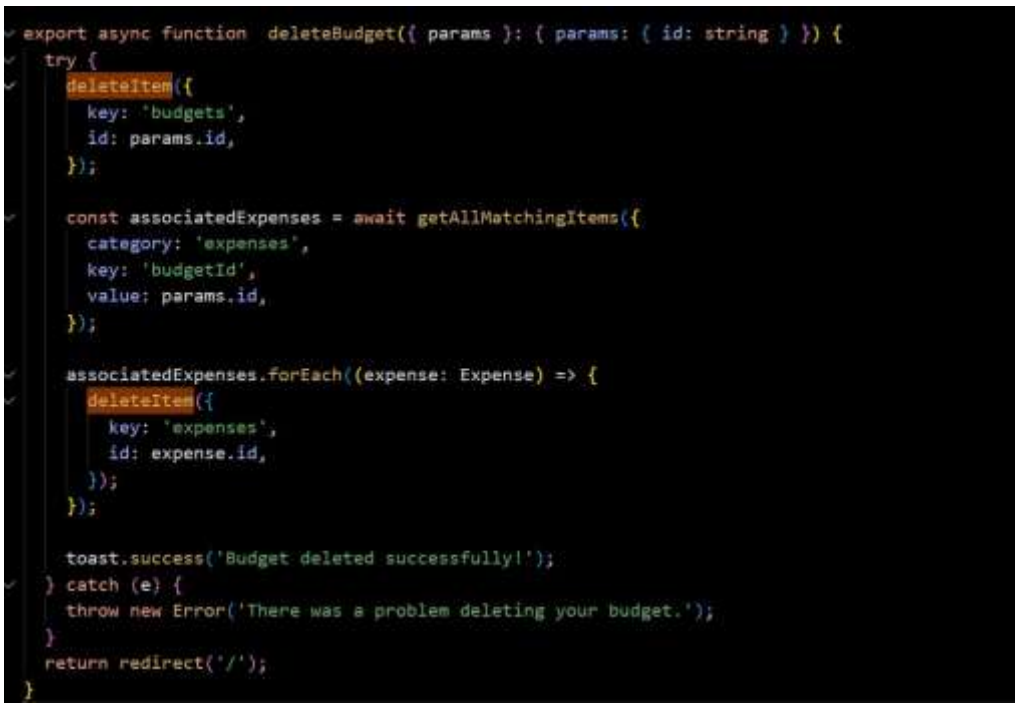
- Деталювання витрат.
- Визначення категорії витрат.
- Зберегти витрати.

Знімок екрана відповідної функції, що відповідає за цю операцію, показано на рисунку нижче.

```
export const createExpense = async ({
  name,
  amount,
  budgetId,
}: {
  name: string;
  amount: string;
  budgetId: string;
}) => {
  const newItem = {
    name,
    createdAt: Date.now(),
    amount: +amount,
    budgetId,
  };
  await addDoc(collection(db, 'expenses'), newItem);
  await dashboardLoader();
};
```

Рисунок 2.2 – Створення витрати

У веб-застосунку присутня можливість видалити бюджет. Програмний код наведено на рисунку.



```
export async function  deleteBudget({ params }: { params: { id: string } }) {
  try {
    deleteItem({
      key: 'budgets',
      id: params.id,
    });

    const associatedExpenses = await getAllMatchingItems({
      category: 'expenses',
      key: 'budgetId',
      value: params.id,
    });

    associatedExpenses.forEach((expense: Expense) => {
      deleteItem({
        key: 'expenses',
        id: expense.id,
      });
    });

    toast.success('Budget deleted successfully!');
  } catch (e) {
    throw new Error('There was a problem deleting your budget.');
  }
  return redirect('/');
}
```

Рисунок 2.3 – Видалення бюджету

2.3 Огляд інтерфейсу веб-додатку

У цьому підрозділі розглянуто інтерфейс web-застосунку для контролю фінансів для користувача як зручний та ефективний інструмент для керування його фінансовими ресурсами. Відповідний інтерфейс є одним з головних компонентів успішного функціонування застосунку. оскільки він визначає як буде взаємодіяти користувач із системою.

Перейду до опису основних розділів і сторінок, доступних у web-застосунку. Також, надано пояснення до кожного елемента та його взаємодії з користувачем і розглянуто різні додаткові екрану web-застосунку, такі як перегляд витрат, перегляд бюджету, видалення бюджету, видалення витрат, створення витрати та ін.

Отож, ось такий вигляд має початкова сторінка, на котру потрапляє користувач:

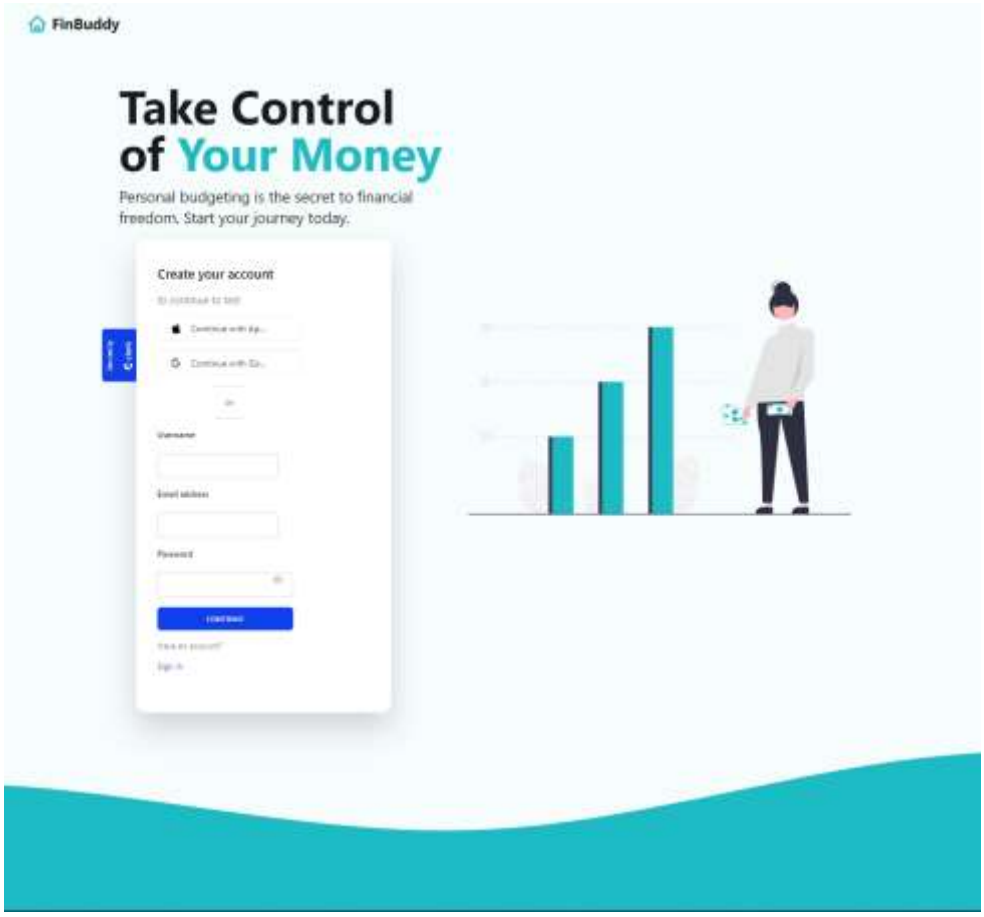


Рисунок 2.4 – Стартова сторінка

Тут відбувається реєстрація у веб- додатку з використанням облікових записів Android чи Google або за e-mail.

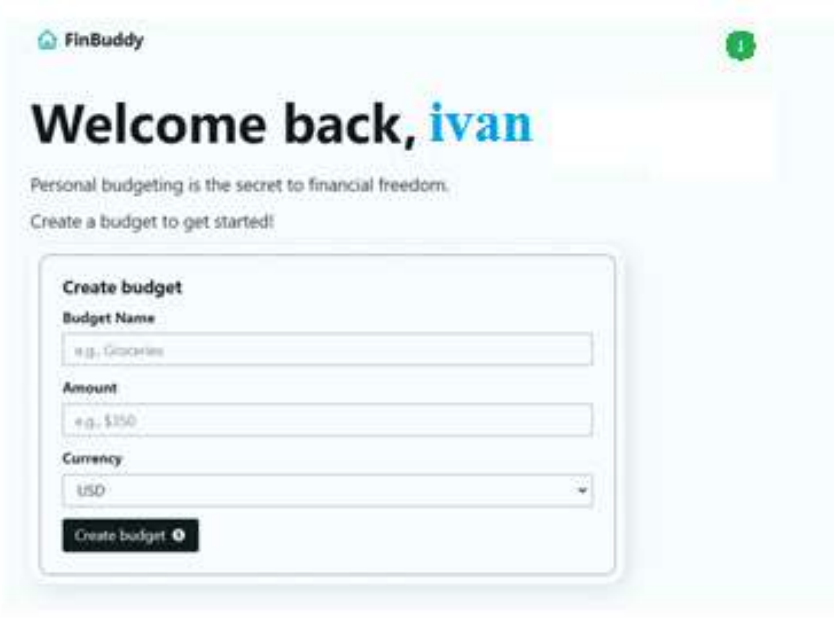


Рисунок 2.5 – Вигляд головної сторінки

Після успішної реєстрації відкривається форма, яку потрібно заповнити.

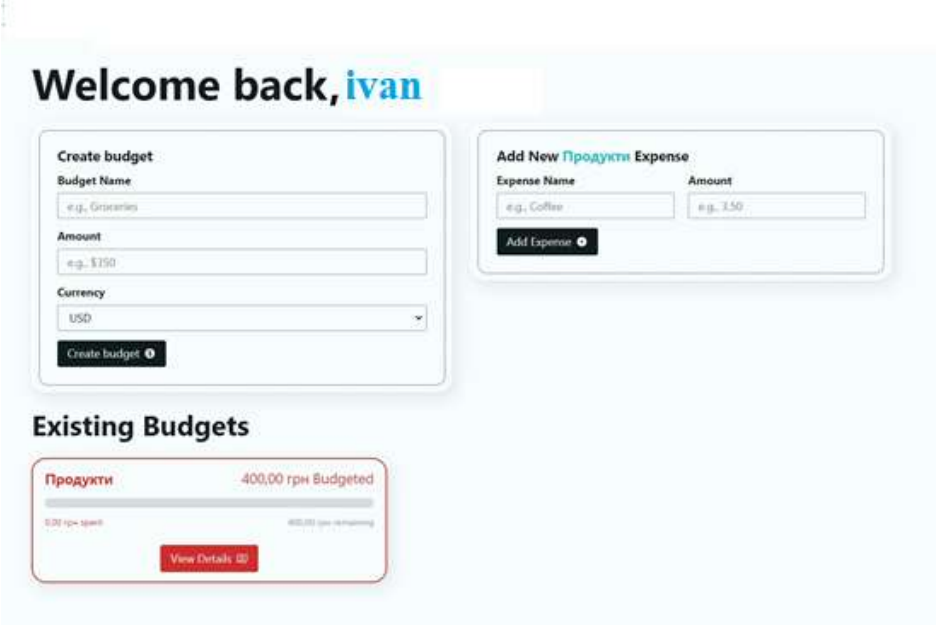


Рисунок 2.6 – Головна сторінка після створення бюджету

Після створення бюджету відкриється форма для введення витрат.

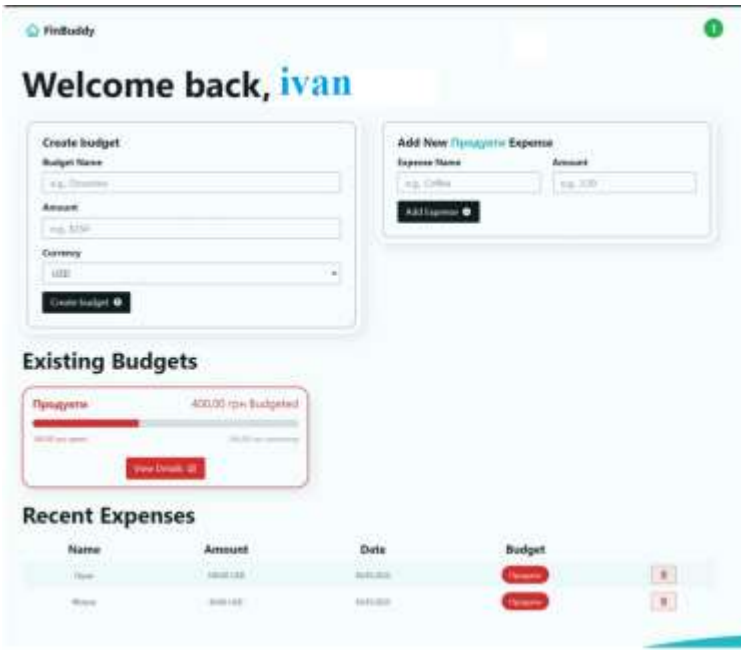


Рисунок 2.7 – Вигляд сторінки

Оновлення сторінки відбувається в процесі створення витрат користувачем. Передбачена можливістб перегляду фінансовго стану.

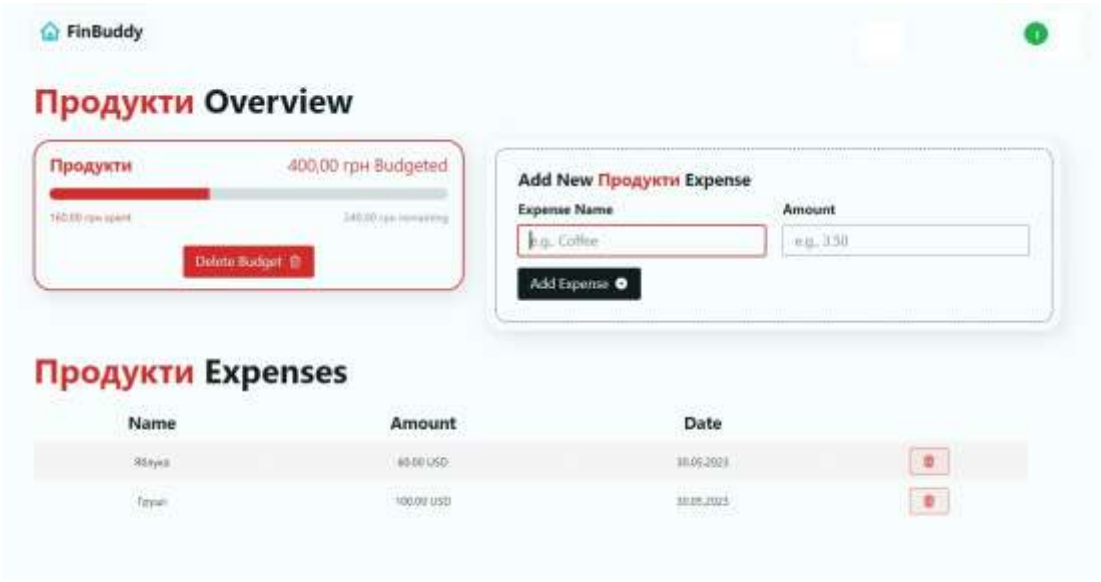


Рисунок 2.8 – Поточний стан бюджету

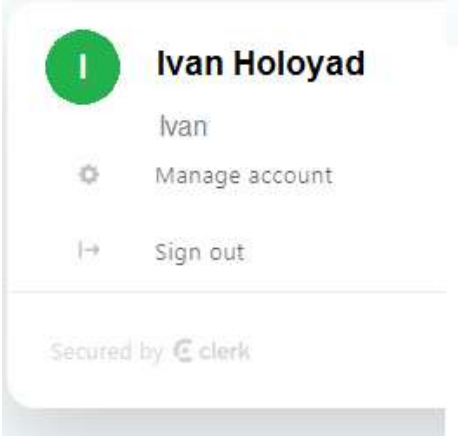


Рисунок 2.9 – Профіль користувача

Користувач має можливість змінити свій акаунт чи внести необхідні уточнення.

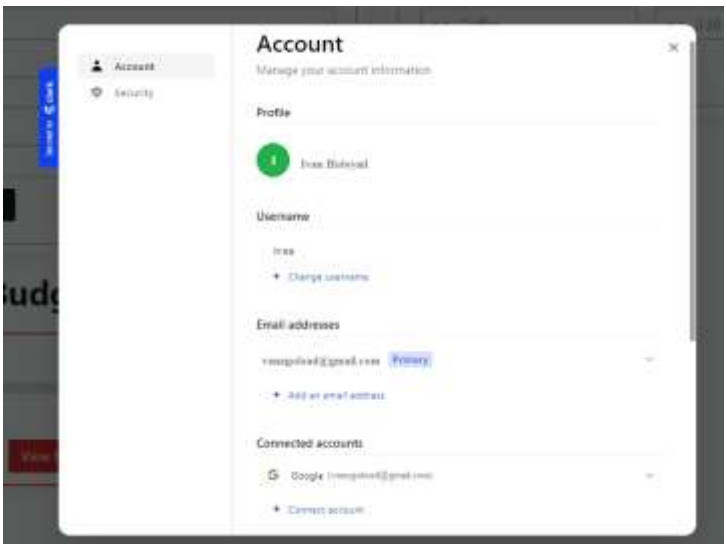


Рисунок 2.10 – Налаштування акаунту

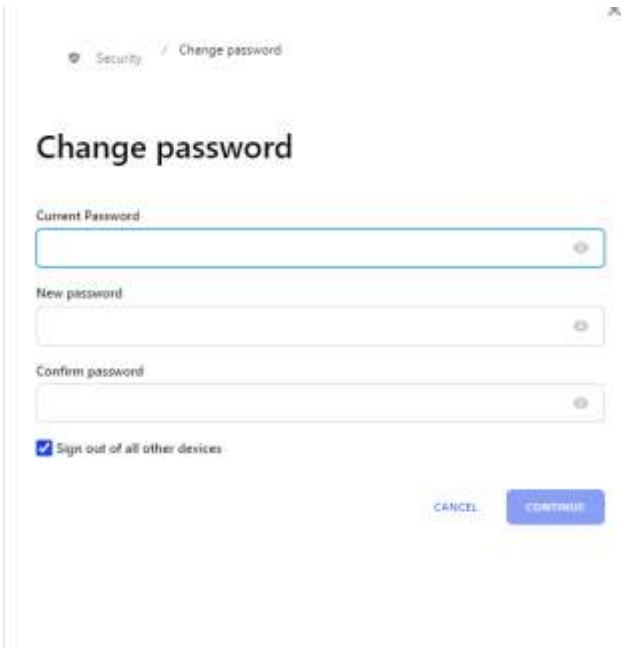


Рисунок 2.11 – Змінення пароля

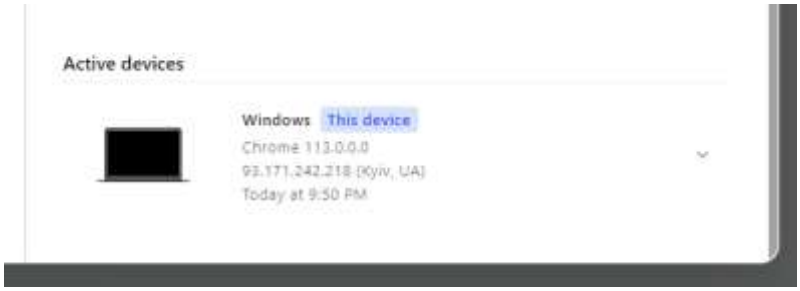


Рисунок 2.12 – Перелік активних пристроїв

2.4 Тестування та перевірка функціоналу веб-додатку

На завершення протестуємо функціонал веб-застосунку з метою перевірки та встановлення неточностей [20, 21, 22, 23, 24, 25, 26]:

- Реєстрація користувача.

Нижче показано процес реєстрації нового користувача та результат її.

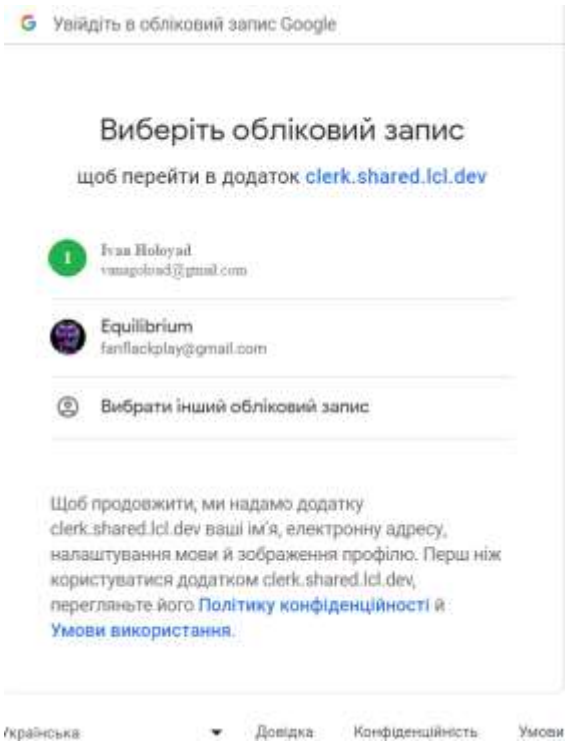


Рисунок 2.13 – Реєстрація Google

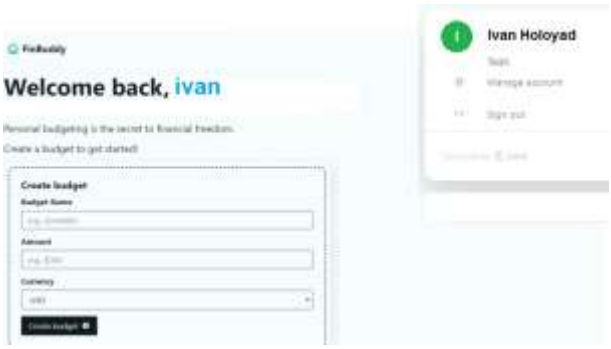


Рисунок 2.14 – Зареєстрований користувач

- Створення бюджету.

Рисунок 2.15 – Тестовий бюджет

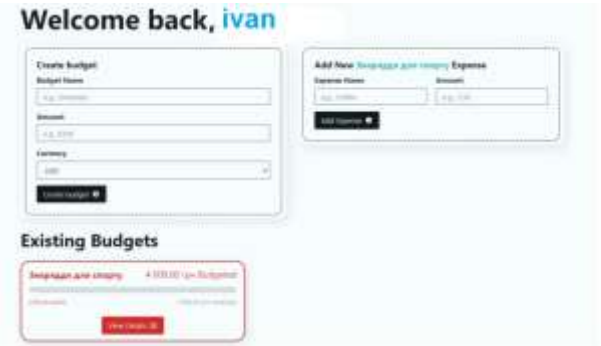


Рисунок 2.16 – Успішне створення бюджету

- Створення витрат.

Рисунок 2.17 – Витрати

- Видалення витрати.

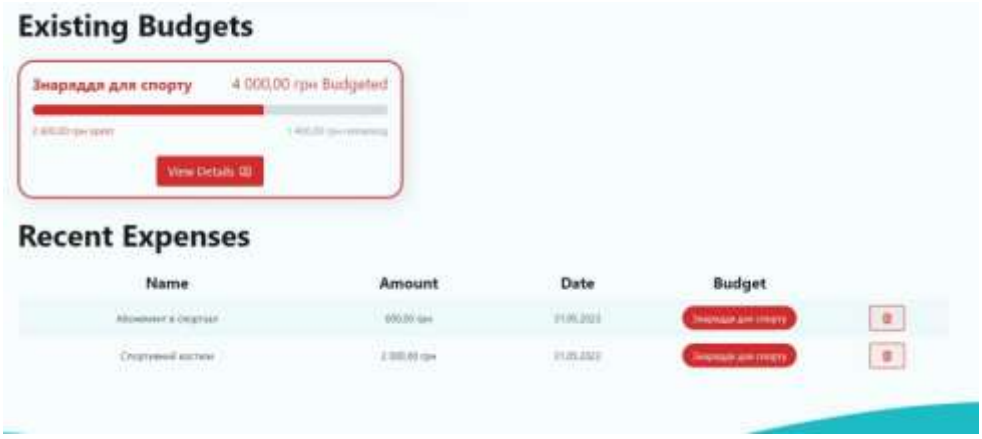


Рисунок 2.18 – Видалення обраних витрат

- Видалення бюджету.

Рисунок 2.19 – Результат процесу

Рисунок 2.20 – Повідомлення про видалення

- Вихід з облікового запису.

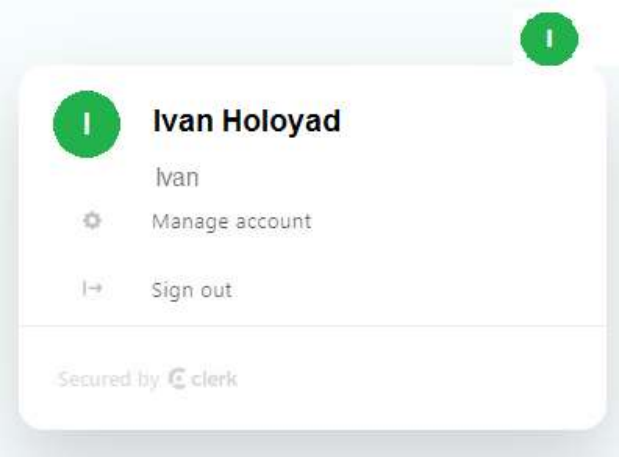


Рисунок 2.21 – Вихід з акаунту

- Можливість авторизації.

Рисунок 2.21 – Вхід до акаунту

Таким чином встановлено, що додаток працює нормально.

2.5. Висновок до розділу 2

У цьому розділі ми розглянули та протестували завершену веб-програму управління фінансами. Під час роботи визначено та проведено оцінку різнобічних аспектів. Перевірено виконання всіх вимог і очікувань, висунутих до завдання.

Виявлено, що ця програма добре виконує базові функції: фіксація бюджету та відповідних витрат, формування звітної інформації про фінансовий стан. Користувальницький інтерфейс розроблено зрозуміло та лаконічно, що дозволяє комфортно взаємодіяти з програмою.

Перевірено вірність опрацювання вхідних даних, відповідь програми на некоректне використання та власне наскільки вона є стабільною в роботі..

Результати тестування виявили деякі невеликі помилки та можливість для вдосконалення, які можна врахувати при подальшій розробці програми.

Переглядаючи та тестуючи завершену програму, тепер можна оцінити наскільки вона є придатною до використанн. Вважаю, що програма є функціональною, корисною та здатною задовільнити основні користувацькі потреби у фінансовому управлінні.

Ці результати є перспективою для подальшого розвитку та удосконалення програми для забезпечення більш задовільної та ефективної взаємодії з користувачем.

3. БЕЗПЕКА ЖИТТЄДІЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1. Значення адаптації в трудовому процесі

Праця людини безпосередньо пов’язана із виробничим середовищем. Працівник може нормально здійснювати трудову діяльність лише тоді, коли умови зовнішнього середовища відповідають оптимальним. Якщо вони змінюються, стають несприятливими, то на протидію їм організм людини включає спеціальний механізм, який зберігає постійність внутрішньою середовища, або змінює його в межах допустимого.

Такий механізм називається адаптацією. Адаптація є важливим засобом попередження травмування, виникнення нещасних випадків у трудовому процесі і відіграє значну роль в охороні праці.

Адаптація в трудовій діяльності поділяється на фізіологічну, психічну, соціальну та професійну.

Фізіологічна адаптація – це сукупність фізіологічних реакцій, які є в основі пристосування організму до змін зовнішніх умов, і направлені на збереження відносної постійності його внутрішнього середовища – гомеостазу.

Фізіологічна адаптація до праці має активний характер і за сприятливих умов виробничого середовища та оптимальних навантажень веде до підвищення стійкості та працездатності організму, збільшення його резервних можливостей, зменшення захворювань і травматизму. Проте коливання умов середовища, в яких відбувається фізіологічна адаптація, має певну межу, характерну для кожного організму. Якщо працівник потрапляє в умови, коли інтенсивність впливу чинників виробничого середовища переважає можливості його адаптації, настають патологічні зміни фізіологічних систем, захворювання організму.

Психічна адаптація – це процес встановлення оптимальної відповідності особистості до навколишнього середовища в процесі діяльності.

Зрозуміло, що такі властивості, як гальмування мислення та низька швидкість переробки інформації, обмежений діапазон сприйняття, порушення

функції пам’яті гальмують адаптацію; висока рухливість нервових процесів, навпаки, її підвищує.

Психічна адаптація в процесі праці залежить від психічних властивостей працівника, його психічного стану, психологічних реакцій на стреси, що виникають на роботі, кваліфікації та культури людини, особливостей професійної діяльності, конкретних умов праці тощо.

Соціальна адаптація – це пристосування працюючої людини до системи відносин у робочому колективі з його нормами, правилами, традиціями, ціннісними орієнтаціями. Під час соціальної адаптації працівник поступово отримує різнобічну інформацію про колектив, де він працює, про систему ділових та особистих взаємовідносин. При несприятливому протіканні соціальної адаптації підвищується рівень стресу на роботі, наслідки якого позначаються на поведінці працівника та можуть призвести до міжособових конфліктів, нещасних випадків.

Професійна адаптація – це адаптація до трудової діяльності з усіма її складовими: адаптація до робочого місця, знарядь та засобів праці, об’єктів та предметів праці, особливостей технологічного процесу, часових параметрів роботи тощо. Професійна адаптація виражається у розвитку стійкого позитивного ставлення працівника до своєї професії, певного рівня оволодіння ним специфічними навичками та уміннями, у формуванні необхідних для якісного виконання роботи властивостей.

Професійна адаптація визначається необхідним мінімумом знань та навичок, яких працівник набув при одержанні спеціальності, ступенем відповідальності, практичністю, діловитістю тощо. Адаптація вважається завершеною тоді, коли працівник досягає кваліфікації, відповідної існуючим стандартам.

Кожен із розглянутих видів адаптації впливає на працездатність та здоров’я працівника, формує у нього певний рівень чутливості та стійкості до психоемоційних перевантажень, внаслідок розвитку яких може істотно змінитися надійність професійної діяльності [37].

3.2 Оцінка технологічного процесу, обладнання, щодо умов електробезпеки, безпеки

При розробці програмного забезпечення інформаційної системи управління підприємством використано приміщення загальною площею 20 м^2, висотою стелі 3 м, де розташовано 6 робочих місць з ПК. Кожне робоче місце обладнане робочим столом площею 1,2 м^2, стільцем та персональним комп'ютером, що складається з монітора, системного блоку, клавіатури та миші. Слід відзначити, що площа одного робочого місця не повинна бути меншою за 6 м^2, а об'єм не менший за 20 м^3, тобто площі та об'єму даного приміщення вистачає для розташування шести робочих місць.

Аналіз умов праці показує, що на програміста можуть негативно впливати наступні фізичні та психофізіологічні фактори:

- підвищена або знижена температура повітря робочої зони;
- підвищена або знижена вологість повітря;
- недостатня освітленість робочого місця;
- підвищений рівень шуму на робочому місці;
- підвищена іонізація повітря;
- підвищений рівень електромагнітних випромінювань;
- нервово-психічні перевантаження (розумова перенапруга, перенапруга аналізаторів);
- фізичні перевантаження (одноманітна поза викликає статичну втому).

Робота програміста за енерговитратами відноситься до категорії легких робіт Іа, Іб, тому повинні дотримуватися наступні вимоги згідно ДСН:

- оптимальна температура повітря – 22°С (допустима – 20-24°С);
- оптимальна відносна вологість – 40-60% (допустима – не більше 75%);
- швидкість руху повітря не більш 0,1 м/с.

Виміряні за допомогою приладів (психрометр Августа) температура та вологість у лабораторії відповідають вказаним у таблиці для теплого періоду року.

Розташовані у приміщенні 6 ПК є джерелами тепловиділень, крім того для підтримання у приміщенні в холодний період року оптимальних параметрів мікроклімату використовуються нагріті поверхні опалювальної системи. Нормованим показником ІЧВ є гранично допустима густина потоку енергії $J_{г.д}$, Вт/м^2, яка встановлюється в залежності від площі опромінюваної поверхні тіла людини ($S_{опр}$). Нормовані рівні складають: $J_{г.д}$=35 Вт/м^2 при $S_{опр}$ > 50%; $J_{г.д}$=70 Вт/м^2 при $S_{опр}$ ~ 25-50%; $J_{г.д}$=100 Вт/м^2 при $S_{опр}$ < 25%.

Нормованим параметром природного освітлення згідно ДБН В.2.5-28-2019 є коефіцієнт природного освітлення (КПО). КПО встановлюється в залежності від розряду виконуваних зорових робіт. Робота програміста відноситься до робіт середньої точності (IV розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5-1,0мм), для яких при використанні бокового освітлення КПО=1,5%. Для штучного освітлення нормованим параметром виступає Емін – мінімальний рівень освітленості, та Кп – коефіцієнт пульсації світлового потоку, який не повинний бути більшим ніж 20%. Мінімальна освітленість встановлюється в залежності від розряду виконуваних зорових робіт. Для IV розряду зорових робіт вона складає 300-500 лк. Для забезпечення штучного освітлення використано світильники типу ЛПО. Кожен світильник комплектується двома лампами. Тобто необхідно використовувати 6 світильників із 12 працюючими лампами в них.

Оскільки при розробці програмного забезпечення інформаційної системи управління підприємством використовуються ПК, кожен з яких устаткований монітором, вінчестером в системному блоці, трьома вентиляторами системи охолоджування та клавіатурою. Крім того поряд працює периферійна техніка, то мають місце шуми механічного і аеродинамічного походження, широкосмугові із аперіодичним підсиленням (при роботі принтерів).

Допустимі значення параметрів неіонізуючих електромагнітних випромінювань від монітору комп'ютера представлені в таблиці 3.1. Нормованим параметром невикористаного рентгенівського випромінювання виступає потужність експозиційної дози. На відстані 5 см від поверхні екрану монітору її рівень не повинен перевищувати 100 мкР/год. Максимальний рівень рентгенівського випромінювання на робочому місці програміста зазвичай не перевищує 20 мкР/год.

На відстані 5-10 см від екрана і корпуса монітора рівні напруженості можуть досягати 140 В/м по електричній складовій, що значно перевищує допустимі значення.

Таблиця 3.1 – Допустимі значення параметрів неіонізуючих електромагнітних випромінювань

Найменування параметра	Допустимі значення
Напруженість електричної складової електромагнітного поля на відстані 50 см від поверхні відеомонітора	10 В/м
Напруженість магнітної складової електромагнітного поля на відстані 50 см від поверхні відеомонітора	0,3 А/м
Напруженість електростатичного поля не повинна перевищувати:	20 кВ/м

Таким чином, встановлено необхідність забезпечення вимог охорони праці та техніки безпеки на робочому місці програміста при розробці програмного забезпечення інформаційної системи управління підприємством [38].

ВИСНОВКИ

У цьому дослідженні розроблено програмний інструмент для моніторингу фінансового стану у формі веб-додатку, який є зручним інструментом для користувача і допомагає йому відстежувати, аналізувати та управляти своїм бюджетом.

При розробці веб-додатку для створення інтерфейсу користувача використовувалися сучасні веб-технології, JS, HTML, CSS і React framework. За допомогою них користувач зможе створити новий бюджет, вводити з можливістю подальшого перегляду витрати та надходження.

Розроблений веб-застосунок може в подальшому бути розширеним та удосконаленим.

Передбачена можливість розробити мобільну програму, яка дозволить вам легко отримувати доступ до свого фінансовго стану.

Користуватися такими додатками корисно тим, хто хоче бути більш поінформованим і впевненим у своєму фінансовому благополуччі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ReactJS [Електронний ресурс] – Режим доступу: https://react.dev/
2. Getting started with React [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started
3. TypeScript [Електронний ресурс] – Режим доступу: https://www.typescriptlang.org/docs/
4. JavaScript [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
5. Firebase [Електронний ресурс] – Режим доступу: https://firebase.google.com/docs
6. Firebase Storage JS SDK [Електронний ресурс] – Режим доступу: https://firebase.google.com/docs/reference/js/v8/firebase.storage.Storage
7. Clerk [Електронний ресурс] – Режим доступу: https://clerk.com/docs
8. Clerk (React integration) [Електронний ресурс] – Режим доступу: https://clerk.com/docs/reference/clerk-react/installation
9. Vite (Next generation frontend tooling) [Електронний ресурс] – Режим доступу: https://vitejs.dev/guide/
10. HTML (HyperText Markup Language) [Електронний ресурс] – Режим доступу: https://devdocs.io/html/
11. CSS (Cascading Style Sheets) [Електронний ресурс] – Режим доступу: https://w3schoolsua.github.io/css/index.html#gsc.tab=0
12. REST (Representational State Transfer) [Електронний ресурс] – Режим доступу: https://aws.amazon.com/what-is/restful-api/
13. Create React App (CRA) [Електронний ресурс] – Режим доступу: https://create-react-app.dev/

Auth0 [Електронний ресурс] – Режим доступу: https://auth0.com/docs

15. Firebase Authentication [Електронний ресурс] – Режим доступу:

https://firebase.google.com/docs/auth

16. OpenID [Електронний ресурс] – Режим доступу:https://openid.net/connect/

17. What is identity as a service? [Електронний ресурс] – Режим доступу: https://www.sailpoint.com/identity-library/identity-as-a-service/

18. Angular [Електронний ресурс] – Режим доступу: https://angular.io/docs

19. Vue.js [Електронний ресурс] – Режим доступу: https://vuejs.org/guide/introduction.html

20. Методичні вказівки до виконання дипломної роботи освітнього рівня ―бакалавр студентами усіх форм навчання для напряму підготовки 121 –Інженерія програмного забезпечення‖ / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

21. М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – “Інженерія програмного забезпечення” для усіх форм навчання [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.

22. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – Інженерія програмного забезпечення (Освітньо-професійна програма - «Інженерія програмного забезпечення») для студентів усіх форм навчання / Упор.: М.Р. Петрик, Д.М. Михалик, Я.І. Кінах, Г.Б. Цуприк - Тернопіль: ТНТУ, 2017-38 с

23. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник. Тернопіль: : ТНТУ 2020 – 62 с.

24. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Моделювання та видобуток даних (висопродуктивні обчислення у великих алгебраїчних та числових системах, комбінаторному аналізі): навчальний посібник. Тернопіль: : ТНТУ 2019 – 62 с.

25. Нога В.В., Цуприк Г.Б. Інформаційні технології при створенні автоматизованих систем для задач збору та збереження інформації. Збірник тез

доповідей VII Міжнародної науково-технічної конференції молодих учених та студентів. Актуальні задачі сучасних технологій – Тернопіль 28-29 листопада 2018. — Т. : ТНТУ, 2018. — Том 2. — С. 131.

26. І.В. Бойко, М.Р. Петрик, Г.Б. Цуприк. Дискретні структури (Алгебраїчні та числові системи, комбінаторний аналіз). Навчально-методичний посібник для студентів спеціальності 121 «Інженерія програмного забезпечення», аспірантів та викладачів вищих навчальних закладів. Конспект лекцій : Навчальний посібник / І.В. Бойко, М.Р. Петрик, Г.Б. Цуприк – Тернопіль : ТНТУ , 2017. – 64 с.

27. Якість та тестування ПЗ. URL: https://dl.tntu.edu.ua/index.php (дата звернення: 30.05.2023).

28. ICONIX - Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ICONIX (дата звернення: 30.05.2023).

29. Android Studio: переваги та особливості [Електронний ресурс] – Режим доступу до ресурсу: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/ (дата звернення: 30.05.2023).

30. Java - Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java (дата звернення: 30.05.2023).

31. Firebase - Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Firebase (дата звернення: 30.05.2023).

32. Лабораторна робота №3 [Електронний ресурс] – Режим доступу до ресурсу: https://dl.tntu.edu.ua/get.php/lab20172018/lab3.pdf.

33. Firebase Test Lab [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/test-lab.

34. What is automation testing? URL: https://www.globalapptesting.com/ blog/what-is-automation-testing.

35. Selenium IDE. URL: https://chrome.google.com/ webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd.

36. Запорожець О., Применко В. Безпека життєдіяльності. Навчальний посібник. – Київ: Центр навчальної і практичної літератури, 2024. 448 с.

37. ДБН В.2.5-28 : 2018. „Природне і штучне освітлення” – К.: Мінрегіон України, 2018. 133 с.

38. Желібо Є.П., Гандзюк М.П., Халімовський М.О. Основи охорони праці. Підручник. – К.: Каравела, 2023. 384 с.

ДОДАТКИ

ДОДАТОК А

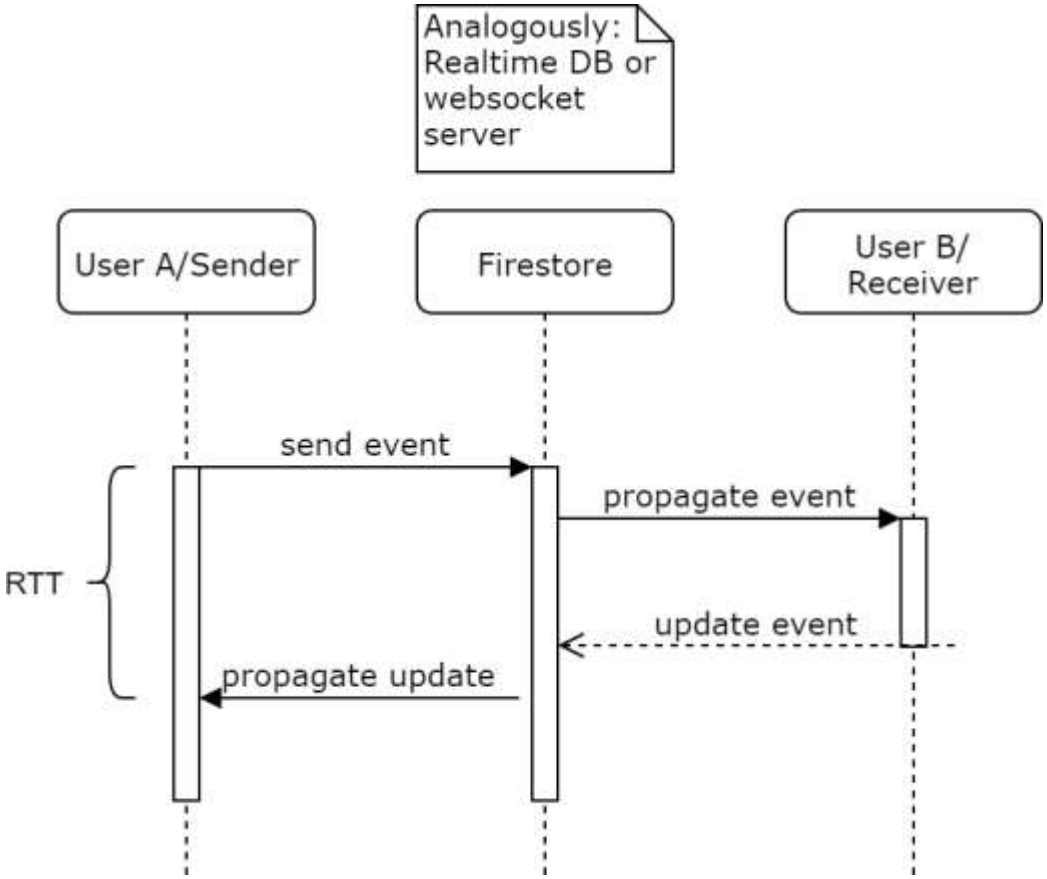


Рисунок А.1 – Структурна схема програмного засобу

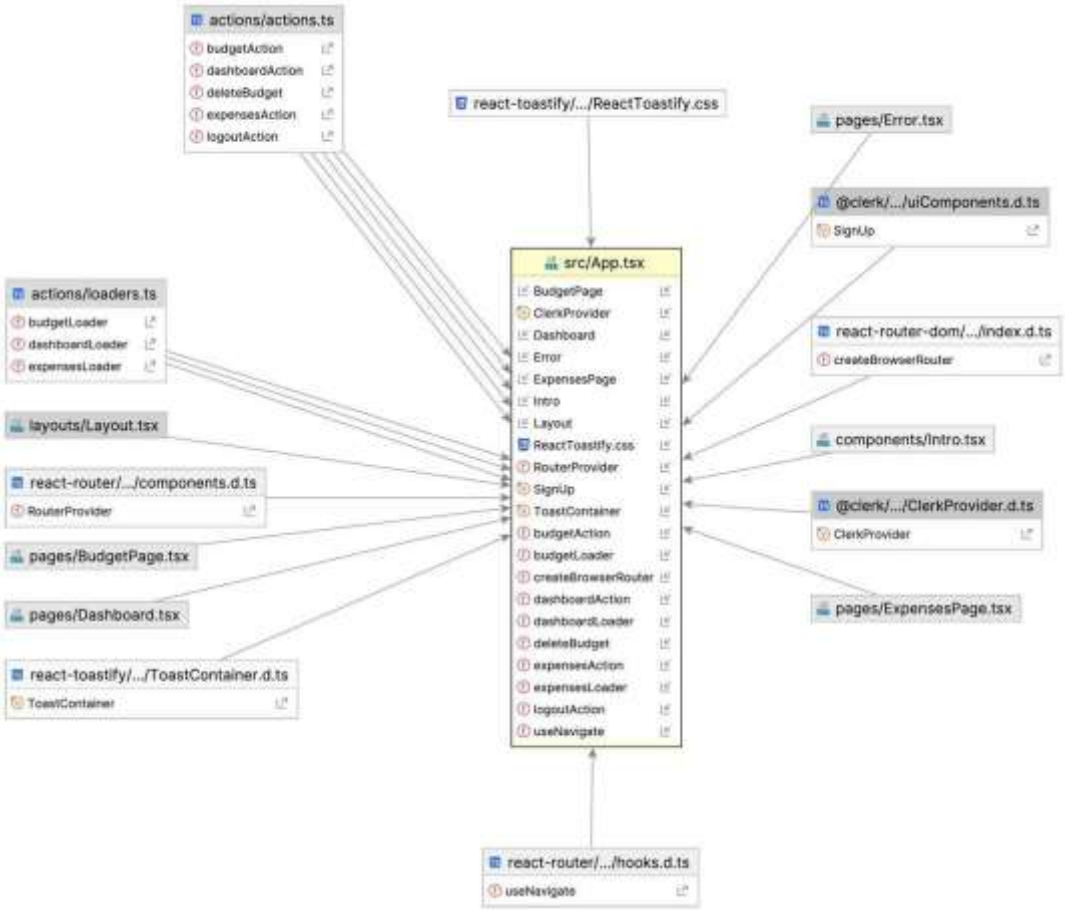


Рисунок А.2 – Функціональна схема засобу

Рисуноку А.3 – Алгоритм роботи застосунку

ДОДАТОК В – Диск із кваліфікаційною роботою бакалавра