МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Економічний факультет
Кафедра економіко-математичного моделювання та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА/ПРОЄКТ
на здобуття освітнього ступеня бакалавра

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ МЕНЕДЖМЕНТУ ВИТРАТ ТА НАДХОДЖЕНЬ»»**

Виконав: студент 4 курсу, групи КН-4
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Чайніков Дмитро Андрійович

Керівник: *викладач кафедри ЕММІТ*
Красюк Богдан Віталійович

Рецензент: *Front-end Developer "DOODLE", LLC*
Місай Володимир Віталійович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ
Завідувач кафедри економіко-математичного моделювання та інформаційних технологій _____ (проф., д.е.н. Кривицька О.Р.)
Протокол № ____ від « ___ » _____ 2023 р.

Острог, 2023

Міністерство освіти і науки України

Національний університет «Острозька академія»

Факультет: *економічний*
Кафедра: *економіко-математичного моделювання та інформаційних технологій*
Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри економіко-математичного моделювання
та інформаційних технологій

_____ Ольга КРИВИЦЬКА
«____» _____20__ р.

З А В Д А Н Н Я
на кваліфікаційну роботу/проєкт студента

Чайнікова Дмитра Андрійовича

1. Тема роботи "Розробка інформаційної системи для менеджменту витрат та надходжень"

керівник проєкту Красюк Богдан Віталійович, викладач.

Затверджено наказом ректора НаУОА від 17 жовтня 2022 року №77.

2. Термін здачі студентом закінченої роботи/проєкту: 31 травня 2023 року.

3. Вихідні дані до роботи/проєкту: Visual Studio Code, GitHub, Python, FastAPI, SQLAlchemy, PostgreSQL, Pydantic, Passlib, PyJWT, React, Redux, Material UI.

4. Перелік завдань, які належить виконати: розробити схему бази даних, створити проекти клієнтської та серверної частини, розробити дизайн інтерфейсу, реалізувати функціонал для додання коштів, збережень, надходжень, категорій та витрат до них. Провести тестування інтерфейсу. Розробити механізм взаємодії з API для клієнтської частини.

5. Перелік графічного матеріалу: рисунки, таблиці.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Красюк Б. В.	01.12.2022	01.12.2022
2	Красюк Б. В.	01.12.2022	01.12.2022
3	Красюк Б. В.	01.12.2022	01.12.2022

7. Дата видачі завдання: 01.12.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1.	Затвердження теми роботи/проекту	до 31.10.2022.	
2.	Постановка технічного завдання	до 01.11.2022	
3.	Підбір та побудова архітектури проекту	до 01.12.2023	
4.	Проектування та розробка серверної частини	до 01.02.2023	
5.	Тестування серверної частини	до 01.03.2023	
6.	Створення клієнтської частини	до 01.04.2023	
7.	Інтеграція функціоналу у інтерфейс клієнтської частини	до 01.05.2023	
8.	Тестування системи	до 17.05.2023	
9.	Попередній захист кваліфікаційної роботи/проєкту	до 18.05.2023	
10.	Здача кваліфікаційної роботи/проєкту на кафедру	до 31.05.2023	

Студент: _____ Дмитро Чайніков

Керівник кваліфікаційної роботи: _____ Богдан КРАСЮК

АНОТАЦІЯ

кваліфікаційної роботи/проєкту

на здобуття освітнього ступеня бакалавра

Тема: *Розробка інформаційної системи для менеджменту витрат та надходжень*

Автор: *Чайніков Дмитро Андрійович*

Науковий керівник: *Красюк Богдан Віталійович, викладач.*

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: *57 с., 32 рис., 13 джерел..*

Ключові слова: *онлайн-навчання, веб-сервіс, розроблення платформи, клієнт-серверна архітектура.*

Короткий зміст праці:

Завданням кваліфікаційної роботи/проєкту, було розроблення інформаційної системи менеджменту витрат та надходжень. Зокрема було розглянуто такі питання як взаємодія з Python FastAPI на клієнтській частині, процес проектування бази даних за допомогою SQLAlchemy. При розробленні було використано чисту архітектуру, що дає змогу в подальшому використати існуюче API для створення, наприклад, мобільного застосунку. Серверна частина була розроблена на бібліотеці React з використанням MUI та JS. Результатом роботи є готова, швидка та проста для подальшого покращення платформа для моніторингу витрат та надходжень.

The task of the qualification work/project was to develop an expenditure and revenue management information system. Specifically, it involved addressing issues such as interaction with Python FastAPI on the client side and the process of designing a database using SQLAlchemy. The development utilized a clean architecture, which allows for the future use of existing APIs to create, for example, a mobile application. The server-side component was developed using the React library, incorporating MUI and JS. The result is a ready-made, fast, and easy-to-improve online learning platform

ЗМІСТ

ВСТУП

Більшість людей бажають витрачати менше та економити більше, але щоденна реальність роботи, сімейного життя та інших зобов’язань робить це непрактичним, навіть, неможливим. Погодьтесь, важко знайти час чи мотивацію для контролю особистих фінансів, а деякі люди взагалі цього не роблять.

Є такий феномен: хоч би скільки людина не заробляла — вона все витрачає. Разом із доходом зростають наші потреби і здається, що маневру для заощадження немає. Але відкладати частину заробітку цілком реально.

Можна нескінченно вивчати, де краще відкрити вклад чи яку купити криптовалюту, але справжня фінансова грамотність починається з планування особистого бюджету.

Завжди задумуючись після походу в магазин, куди ділись гроші, - це не вихід. Це схоже до того щоб вести зошит витрат але оскільки ми живемо у 21 столітті де настала ера діджиталізації найкращим варіантом є додаток для контролю фінансів, оскільки де б ви не були ваш телефон буде з вами а отже і всі дані про ваші витрати.

З цих та багатьох інших причин бюджетні програми стають все більш популярними протягом останнього десятиліття. Ці програми полегшують контроль за вашими витратами та накопиченнями, пов’язуючи облікові записи з додатком та відстежуючи ваш грошовий потік. Додатки, безкоштовні або платні, пропонують цілий ряд функцій для покращення ваших фінансів.

Програми для особистих фінансів поділяються на дві основні категорії це контроль особистих фінансів та управління інвестиційним портфелем. Оптимізація витрат – це те, до чого рано чи пізно приходять практично всі, кому не байдуже на власні кошти. Ці програми виконують роль персональних фінансових помічників: надають докладний звіт за доходами і витратами, допомагають дотримуватися плану видатків на місяць, нагадують про регулярні платежі

Проте у цій сфері є і свої мінуси так, так як вони оптимізовані лише для мобільного використання та не мають веб аналогів, що було б просто прекрасним рішенням за деяких умов.

Метою дослідження є розробка та прикладна реалізація системи моніторингу витрат.

Для досягнення мети дослідження потрібно виконати наступний ряд завдань:

1. Виконати аналіз конкурентних програмних рішень.
2. Ознайомитись з теоретичним матеріалом.
3. Спроектувати серверну частину.
    i. Проектування БД.
    ii. Вибір архітектури.
    iii. Вибір технологій.
4. Розробити серверну частину
    i. Спроектувати БД.
    ii. Створити Контролери.
5. Розробити клієнтську частину
    i. Розробити авторизацію та автентифікацію.
    ii. Розробити інтерфейс для зручного використання сервісу.

Дана робота описуватиме процес проектування та розробки «Системи моніторингу витрат». Вибір на користь цієї теми був зроблений завдяки бажанню зробити зручним та простим у використанні продукт для відслідковування витрат.

Отже підсумую що додатки для моніторингу витрат при адекватному використанні можуть зменшити ваші нерозважливі витрати та зберегти ваші кошти. Допомогти реально оцінювати матеріальне становище, підсумовувати загальні витрати та контролювати основні витоки коштів.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ФІНАНСОВОГО ОБЛІКУ

1.1. Постановка проблеми, опис базових понять бізнес-процесу

Актуальність обліку фінансів на мою думку здається недостатньо розповсюдженою хоча існує безліч різноманітних варіантів додатків. Проте не все так добре як може здатися на перший погляд. Фінансова грамотність починається з самого користувача, а не додатку. Ці сервіси в свою ж чергу намагаються максимально привернути до себе користувачів надаючи різноманітні послуги починаючи від звичайних витрат, надходжень, заощаджень до нагадування про регулярні платежі, встановлення лімітів на певні категорії витрат та можливості додання витрат по фотографії чеку.

Моніторинг витрат це максимально швидко та просто, для тих хто в цьому зацікавлений, та досить складний процес що потребує нагадування від додатку про додання витрат. Наразі я вважаю що проблема фінансової грамотності та моніторингу витрат є недооціненою, оскільки можна зробити спонтанні витрати на свої «хотілки» одразу ж після отримання доходу а потім думати на чому краще зекономити щоб дожити до наступного місяця не говорячи вже про фінансову свободу, відкладення коштів на якусь мрію або щоб була фінансова подушка безпеки.

Розглянемо основні визначення:

Фінансова грамотність - це набір навичок і знань, які дозволяють людині ефективно управляти своїми фінансами. Це включає вміння рахувати, заробляти, накопичувати, інвестувати гроші і навички фінансового планування.

Фінансова свобода - це стан, коли особа має достатньо особистих фінансових ресурсів, щоб не залежати від заробітної плати або продажу власної робочої сили. Це означає, що людина має фінансову незалежність і може зосередитися на досягненні своїх цілей та мрій.

Оптимізація витрат - це процес планування, обліку, аналізу та контролю витрат з метою досягнення ефективного використання грошей. Це означає раціональне розподілення коштів на основні потреби, заощадження та інвестиції.

Моніторинг - це систематичне спостереження за явищами і процесами. У контексті фінансів, моніторинг означає постійне слідкування за витратами, доходами і іншими фінансовими показниками з метою контролю та аналізу.

Інвестування - це процес розміщення грошових коштів з метою отримання прибутку в майбутньому. Інвестування може включати покупку акцій, облігацій, нерухомості та інших фінансових інструментів. Це важлива складова фінансової стратегії, яка може сприяти зростанню капіталу та досягненню фінансових цілей.

Кредитний скоринг - це числова оцінка кредитної історії особи, яка використовується фінансовими установами для визначення ймовірності відшкодування кредитних зобов'язань. Кредитний скоринг враховує різні фактори, такі як кредитна історія, платоспроможність, заборгованість тощо. Це важлива інформація для отримання кредиту або виконання фінансових угод.

Фінансовий план - це документ, який містить стратегію та цілі щодо управління фінансами особи, сім'ї або бізнесу. Фінансовий план включає в себе інформацію про доходи, витрати, заощадження, інвестиції та інші фінансові аспекти. Він служить як орієнтир для прийняття рішень та досягнення фінансових цілей.

Фінансовий консультант - це професійний фахівець, який надає клієнтам консультації та рекомендації з питань фінансів, вкладень, планування пенсії, оподаткування тощо. Фінансовий консультант допомагає людям розуміти їхню фінансову ситуацію, розробляти стратегію та планувати майбутнє з урахуванням їхніх фінансових цілей.

Фінансова стабільність - це стан, коли особа або організація має достатні ресурси та фінансову безпеку для забезпечення своїх потреб і зобов'язань у протязі тривалого періоду часу. Фінансова стабільність включає в себе належне управління

витратами, налагодження надходжень, наявність резервів та запасів, а також здатність впоратися з фінансовими негараздами чи несподіваними витратами.

У кожного продукту що випускається свій кінцевий споживач, так і з додатками моніторингу витрат, вони спрощують життя надаючи корисні дані для розуміння фінансового становища, можливостей та встановлення майбутніх цілей.

Розповсюдження систем для контролю над фінансами це ключовий фактор для збільшення фінансово грамотних людей, що призведе до розвитку людину, що коректно їх використовує.

Система моніторингу витрат – це ультимативний спосіб, що допомагає людям піклуватися про власні кошти та заставляє задуматися, а чи добре я живу.

1.2. Суть системи моніторингу витрат

Система моніторингу витрат є незамінним інструментом для ефективного контролю фінансів та забезпечення фінансової стабільності. Її основна мета полягає у збереженні інформації про всі витрати та надходження, а також нагадуванні про заплановані платежі. Однак, ця система не обмежується лише зберіганням даних, вона також забезпечує створення аналітики витрат.

Завдяки аналітичним звітам та статистиці витрат, користувач може отримати повну картину своєї фінансової ситуації. Це дає змогу оцінити, чи можна відкласти більше грошей, придбати кращі товари або збільшити суму заощаджень. Більш того, це може бути додатковою мотивацією для заробітку більшої суми грошей або розгляду можливостей зменшення витрат, особливо коли стикаєшся з нераціональними покупками.

Важливим аспектом успішного використання системи моніторингу витрат є регулярне внесення даних. Чим частіше користувач оновлює інформацію, тим точніша та корисніша буде аналітика. Незважаючи на те, що це може забирати невелику кількість часу, вкладений час повернеться з великими перевагами.

Наприклад, здається, що регулярний витрати на каву перед роботою можуть здаватися незначними, проте аналітика може показати, що це фактично складає близько 20% від загальних місячних витрат. Це усвідомлення може стимулювати зміни в поведінці, наприклад, приготування кави вдома для зниження витрат.

Отже, система моніторингу витрат допомагає людині відкрити багато нового про своє фінансове становище. Вона може розкрити проблеми та недоліки у власних фінансах, які раніше можуть бути ігноровані. Засоби контролю фінансів не мають магічної здатності знайти додаткові гроші, але вони служать нагадуванням про потенційні проблеми та допомагають зосередитися на важливих аспектах фінансового управління.

1.3. Аналіз продуктів-конкурентів на ринку

На даний момент на ринку є досить багато конкурентів з різним напрямком послуг, проте більшість з них мають платну підписку без якої додаток майже втрачає свій сенс. З головних конкурентів можна виділити:

1. Дзен-мані – це програма, яка допомагає вести облік фінансів з мінімальними зусиллями. Вона пропонує цілий набір функцій, які роблять фінансове управління легким та зручним для користувача.

Одна з ключових особливостей Дзен-мані - це функція синхронізації з банками. Вона дозволяє автоматично завантажувати транзакції з вашого банківського рахунку, що усуває необхідність ручного введення даних. Це значно економить ваш час і забезпечує точність обліку витрат та надходжень.

Окрім цього, Дзен-мані також надає нагадування про важливі фінансові події. Наприклад, вона може нагадувати вам відкласти необхідну суму для своєчасного погашення кредиту або рахунку за комунальні послуги. Це допомагає уникнути прострочення платежів та покарань у вигляді пені.

Ще одна перевага Дзен-мані - це можливість спільного ведення сімейного бюджету. Ви можете запрошувати інших членів родини до програми і об'єднувати свої фінансові дані. Це дозволяє бачити загальну картину витрат та внесків всієї родини і спільно приймати рішення щодо фінансового планування.

Таблиця 1.1 SWOT-аналіз програмного продукту «Дзен-мані»

Сильні сторони	Слабкі сторони
Нагадування про заплановані витрати Розподіл витрат за категоріями	Платна версія надає доступ до ключових функцій додатку
Можливості	Загрози
Синхронізація даних між пристроями	Кіберзлочинність Можливість продажу даних користувачів

Джерело: розроблено автором

2. Goodbudget: Budget & Finance - це додаток, спрямований на персональне планування бюджету та складання фінансового плану на місяць. Його головна особливість полягає в тому, що він дозволяє користувачам особисто визначати суму грошей та на що вони хочуть їх витратити, а додаток відстежує виконання цього наміченого плану.

При використанні Goodbudget ви можете створити категорії витрат і визначити бюджет для кожної категорії. Наприклад, ви можете встановити окремі ліміти на їжу, житло, транспорт, розваги та інші види витрат. Після того, як ви встановили свої категорії та бюджет, Goodbudget дозволяє відстежувати реальні витрати відносно запланованого бюджету.

Цей підхід до планування бюджету дозволяє користувачам бути більш свідомими щодо своїх фінансів. Вони самостійно приймають рішення про те, на що вони хочуть витратити свої гроші, і можуть відстежувати, наскільки ефективно вони

дотримуються свого плану. Це стимулює здорове фінансове планування, контроль над витратами і допомагає досягти фінансових цілей.

Goodbudget також надає інші корисні функції, такі як синхронізація з іншими пристроями, можливість додавати фінансові цілі та регулярні платежі, аналіз витрат і генерація звітів. Він створений для того, щоб зробити планування бюджету простим, зручним і ефективним для будь-якого користувача.

Таблиця 1.2 SWOT-аналіз програмного продукту «Goodbudget»

Сильні сторони	Слабкі сторони
Розподіл витрат за категоріями Звіт про витрати Інтуїтивно зрозумілий інтерфейс	Платна версія надає доступ до ключових функцій додатку
Можливості	Загрози
Синхронізація даних між пристроями Попередження про ліміт витрат	Кіберзлочинність Можливість продажу даних користувачів

Джерело: розроблено автором

3. Splitwise - це додаток, який допомагає точно розподілити спільні витрати між учасниками. Чи це оренда житла, організація вечірки або подорож, Splitwise забезпечує зручний та справедливий розрахунок витрат.

Для використання Splitwise достатньо додати всіх учасників до додатку і вказати, хто скільки витратив. Потім Splitwise автоматично обчислить, як розподілити витрати між учасниками, забезпечуючи справедливий розрахунок на основі внесків кожного учасника.

Наприклад, якщо ви і ваші друзі орендуєте житло разом, ви можете вести облік всіх платежів, пов'язаних з орендою, в Splitwise. Додаток автоматично розрахує, скільки кожен з вас повинен сплачувати, враховуючи індивідуальні внески та розподіл витрат.

Також, Splitwise надає інші корисні функції, такі як можливість додавати коментарі до витрат, завантажувати фотографії квитанцій, стежити за заборгованостями та отримувати нагадування про платежі.

Цей додаток допомагає уникнути непорозумінь та конфліктів між учасниками спільних витрат, забезпечуючи прозорість та точність в розрахунках. Ви можете легко вести облік своїх витрат та спільно розрахуватися з друзями чи колегами, що дозволяє економити час і уникати неприємностей.

Splitwise спрощує процес поділу витрат і забезпечує справедливість у розрахунках, роблячи спільні фінансові угоди більш зручними та безпроблемними.

Таблиця 1.3 SWOT-аналіз програмного продукту «Splitwise»

Сильні сторони	Слабкі сторони
Розподіл витрат між людьми Можливість додавати витрати на одну категорію в різних валютах	Платна версія надає доступ до ключових функцій додатку Довгий процес реєстрації
Можливості	Загрози
Синхронізація даних між пристроями Розподіл витрат за категоріями Коментування витрат	Кіберзлочинність Можливість продажу даних користувачів

Джерело: розроблено автором

1.4. Опис власного продукту на основі аналізу конкурентів

Система моніторингу витрат є інноваційним проєктом, який використовує передові технології для спрощення процесу внесення, збереження та аналізу витрат користувачів. Цей продукт надає можливість зручно вести облік різних видів активів та витрат, створювати категорії для легкого класифікування витрат, додавати нові витрати та отримувати зрозумілу аналітику за визначений період часу.

Одним із ключових елементів цієї системи є зручний інтерфейс, який дозволяє користувачам швидко та просто внести всю необхідну інформацію про свої витрати. Завдяки цьому, користувачі можуть легко стежити за своїми фінансами, знати, куди йдуть їх гроші і як ефективно вони розподіляються.

Окрім того, система моніторингу витрат є доступною з будь-якого пристрою з підключенням до Інтернету. Це означає, що користувачі можуть мати доступ до своїх фінансових даних і здійснювати необхідні операції з будь-якого місця і в будь-який час. Відсутність обмежень щодо типу пристрою робить цю систему зручною та універсальною для всіх користувачів.

Крім збереження і аналізу витрат, ця система може також надавати користувачам корисні поради та рекомендації щодо оптимізації витрат та досягнення фінансових цілей. За допомогою аналітики та статистики, користувачі можуть зрозуміти свої фінансові звички та прийняти кращі рішення щодо економії та планування свого бюджету.

Система моніторингу витрат відкриває нові можливості для кожного користувача, допомагаючи краще розуміти та керувати своїми фінансами. Вона стає надійним помічником у повсякденному фінансовому плануванні та допомагає досягати фінансової стабільності та успіху.

Таблиця 1.4 SWOT-аналіз програмного продукту «Система моніторингу витрат»

Сильні сторони	Слабкі сторони
Відсутність платної підписки Розподіл витрат за категоріями	Велика конкуренція Відсутність реклами продукту
Можливості	Загрози
Синхронізація даних між пристроями Аналітика витрат	Кіберзлочинність

Джерело: розроблено автором

РОЗДІЛ 2. ПРИКЛАДНА РОЗРОБКА ВЕБЗАСТОСУНКУ “СИСТЕМА МОНІТОРИНГУ ВИТРАТ”

2.1. Опис технологічного стеку та архітектури рішення

Стек технологій:

Backend:

Python

Fastapi

SQLAlchemy

PostgreSQL

Pydantic

Passlib

PyJWT

GitHub

Frontend:

ReactJS

HTML

CSS

JS

Material UI

Python — проста у використанні, та водночас повноцінна мова програмування, що надає набагато більше засобів для структурування і підтримки великих програм, ніж shell. Python дозволяє розбивати програми на модулі, що потім можуть бути використані в інших програмах. Python поставляється з великою бібліотекою стандартних модулів, які можна використовувати як основу для нових програм або

як приклади при вивченні мови. Стандартні модулі надають засоби для роботи з файлами, системними викликами, мережевими з'єднаннями і навіть інтерфейсами до різних графічних бібліотек. Python є однією з найпопулярніших мов програмування на сьогоднішній день, і вона має численні переваги, які роблять її зручною і ефективною для розробки програм. Ось деякі ключові особливості мови Python: простота використання, широкий спектр застосувань, модульність, велика стандартна бібліотека, підтримка спільноти, переносимість та розширюваність[6].

Fastapi — це сучасний, швидкий та високопродуктивний фреймворк, оскільки він повністю підтримує асинхронне програмування та може працювати з WSGI (інтерфейс шлюзу веб-сервера) та ASGI (інтерфейс шлюзу асинхронного сервера), веб-фреймворк для створення API, використовуючи Python 3.6+. Fastapi дозволяє декларувати додаткову інформацію та включає в себе перевірку параметрів, має дуже потужну, але інтуїтивно зрозумілу систему впровадження залежностей. За допомогою системи Dependency Injection ви також можете повідомити FastAPI , що ваша функція операції шляху також «залежить» від чогось іншого, що має бути виконано перед вашою функцією операції шляху , і FastAPI подбає про її виконання та «впровадження» результатів. Він створений для того, щоб бути дуже простим у використанні та полегшити будь-якому розробнику інтеграцію інших компонентів. Простота системи впровадження залежностей робить FastAPI сумісним із: всіма реляційними базами даних, базами даних NoSQL, зовнішніми пакетами, зовнішніми API, системами аутентифікації та авторизації, системи моніторингу використання API, системами введення даних відповіді, тощо. У Fastapi є декілька аналогів Flask, Django проте вони відрізняються між собою та кожен з них має власні переваги і недоліки. Основним недоліком даних аналогів є власна анотація коли Fastapi використовує стандартну анотацію Python[5].

SQLAlchemy — надає «повний набір добре відомих шаблонів корпоративного рівня стабільності, сконструйованих для високопродуктивного доступу до бази даних, написаних простою мовою Python». Філософія SQLAlchemy стверджує що

бази даних SQL поводяться тим менш подібно на колекції об'єктів чим більше починають важити розмір та продуктивність, і навпаки, колекції об'єктів починають поводитись тим менш подібно на таблиці і записи чим більш починає важити рівень абстракції. SQLAlchemy має безліч переваг до ключових відносяться: code first принцип що дозволяє визначити схему бази даних у коді. Автоматичну синхронізацію моделі та схеми (alembic) надбудова SQLAlchemy для керування базами даних. Коли ви вносите зміни в модель Python, Alembic може автоматично оновлювати схему бази даних. Це робить незначні зміни, наприклад додавання таблиці чи стовпця, швидкими та легкими. Стиль Pythonic робить ваш код приємним для читання. Простота створення запитів. Підтримка прозорого поліморфізму. Має можливість працювати із застарілими базами даних. Можливість встановлювати фільтри прямо у запитах. І заключною перевагою є деталізована документація[7].

Pydantic надає можливість валідації та серіалізації даних на основі цих оголошених моделей. Використовуючи Pydantic, можна визначити типи даних для атрибутів класу, встановити правила валідації, обмеження на значення, а також забезпечити перетворення даних в потрібний формат. Завдяки цьому підходу, Pydantic дозволяє розробникам створювати дуже стійкі та безпечні програми, оскільки він допомагає виявляти та уникати помилок, пов'язаних з некоректними даними. Крім того, Pydantic спрощує процес взаємодії з даними, забезпечуючи автоматичну серіалізацію та десеріалізацію об'єктів. Pydantic також інтегрується з різними фреймворками та бібліотеками Python, що дозволяє легко використовувати його в різних проектах. Він знайшов широке застосування в розробці веб-додатків, API, систем обробки даних та багатьох інших сферах. Узагальнюючи, Pydantic є потужним інструментом для статичної типізації та валідації даних в Python, що допомагає розробникам створювати надійні та ефективні програми з меншою кількістю помилок[8].

PostgreSQL — об'єктно-реляційна система керування базами даних (СКБД). Є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM

DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite). PostgreSQL - це розширена версія SQL, яка забезпечує підтримку різних функцій SQL, таких як зовнішні ключі, підзапити, тригери та різні користувацькі типи та функції. Перегляди в PostgreSQL можуть бути оновлені, але не автоматично на відміну від SQL-сервера. Користувач повинен написати правила проти різних поглядів, щоб оновити їх. Також складні види можна легко створити. PostgreSQL не надає обчислених стовпців. PostgreSQL, з іншого боку, має функціональні індекси, які працюють так само, як вид. Динамічні дії в SQL PostgreSQL надає цю функцію, і просто використовуючи вибрані оператори, користувач може виконувати дійсно всі операції, легко отримувати та виконувати всі інші роботи. SQL Server як і PostgreSQL є інструментами керування базами даних. Вони допомагають правильно та ефективно керувати всіма даними. Але коли справа доходить до різних функцій, PostgreSQL завжди на висоті. Це розширена версія SQL і, отже, надає багато додаткових функцій. Всі ці функції безкоштовні, на відміну від SQL server. Крім того, він кросплатформний і може використовуватися з будь-якою операційною системою.

Passlib — це бібліотека хешування паролів для Python 2 і 3, яка забезпечує кросплатформену реалізацію понад 30 алгоритмів хешування паролів, а також структуру для керування наявними хешами паролів. Його розроблено, для широкого кола завдань, від перевірки хешу, знайденого в /etc/shadow, до забезпечення повноцінного хешування пароля для багатокористувацької програми[9].

PyJWT — це бібліотека Python, яка дозволяє кодувати та декодувати веб-токени JSON (JWT). JWT — це відкритий промисловий стандарт (RFC 7519) для безпечного представлення претензій між двома сторонами. Веб-токени JWT використовуються для автентифікації та авторизації веб-додатків. Вони складаються з трьох частин: заголовка, корисної навантаження (payload) та підпису. PyJWT надає зручний спосіб генерувати JWT-токени та перевіряти їх справжність. Бібліотека підтримує різні алгоритми шифрування та підпису, такі як HMAC, RSA та ECDSA.

Вона також має вбудовану підтримку перевірки часу життя токена і обробки помилок. За допомогою PyJWT розробники можуть легко створювати токени для аутентифікації користувачів, передавати інформацію про ролі та дозволи, і надійно передавати ці дані між сервером і клієнтом. Бібліотека також надає можливість розширювати та налаштовувати JWT-токени відповідно до потреб конкретного додатка. PyJWT є популярним вибором для роботи з JWT-токенами у програмах на Python. Він допомагає забезпечити безпеку і цілісність даних, а також спрощує розробку систем автентифікації та авторизації[10].

GitHub є одним з найпопулярніших та найбільших веб-сервісів для спільної розробки програмного забезпечення. Він надає розробникам засоби для зберігання, керування та спільної роботи над проектами з використанням системи керування версіями Git. GitHub пропонує безкоштовні та платні тарифні плани для користувачів. Безкоштовний план дозволяє створювати та розгортати відкриті репозиторії, які доступні для публічного перегляду та співпраці. Платні плани, в свою чергу, надають додаткові можливості, такі як приватні репозиторії, управління доступом, інструменти для автоматизації робочого процесу та інші функції, спрямовані на командну роботу та підприємства. GitHub базується на системі керування версіями Git, яка дозволяє зберігати копії коду, здійснювати контроль версій та відстежувати зміни, зроблені в коді. Веб-інтерфейс GitHub надає зручний спосіб перегляду, редагування та коментування коду, ведення обговорень, відкриття проблемних звітів (issues) та запитів на злиття (pull requests). Компанія GitHub розробила цей сервіс на базі Ruby on Rails та Erlang, використовуючи їх для реалізації різних компонентів та функцій сервісу. Загалом, GitHub є незамінним інструментом для розробників програмного забезпечення, який дозволяє їм ефективно співпрацювати, відстежувати зміни в коді та керувати розробкою проектів з використанням системи керування версіями Git[4].

ReactJS — це відкрита JavaScript-бібліотека для створення інтерфейсів користувача, яка написана для вирішення проблеми часткового оновлення вмісту

веб-сторінки, з якими програмісти зустрічаються при написанні та розробці односторінкових додатків. Основна відмінність між багатосторінковими та односторінковими програмами полягає в тому, що односторінкові програми завантажують сторінку один раз після її виконання. Як тільки користувач взаємодіє з додатком, буде змінено лише необхідний компонент, а не повний додаток, що робить односторінковий додаток набагато швидшим з точки зору інтерактивності. Односторінкові програми пропонують набагато кращий користувальницький досвід (UX), а це означає, що користувачі можуть легко переміщатися між різними сторінками програми, не чекаючи завантаження сторінок. Невід'ємною частиною reactjs є react hooks - це функції, які дозволяють вам підключатися до стану реакції та “life-cycle features” з функціональних компонентів. React надає вбудовані хуки, такі як useState, useEffect, useReducer, useRef, useCallback, useContext, useMemo, а також ви можете створювати власні хуки. Функціональні компоненти як і компоненти класу також є частиною reactjs проте вони мають ключову відмінність - синтаксис. Функціональний компонент — це звичайна функція JavaScript, яка приймає props як аргумент і повертає елемент React. Компонент класу вимагає від вас розширення React.Component і створення функції рендерингу, яка повертає елемент React. Це вимагає більше коду, але також дасть деякі переваги. Одним з фундаментальних завдань будь-якого веб-додатка є зв'язок з серверами через протокол HTTP. Цього можна легко досягти за допомогою Fetch або Axios. Fetch і Axios дуже схожі за функціональністю. Деякі розробники віддають перевагу Axios перед вбудованими API за простоту використання. API Fetch чудово здатний відтворювати ключові особливості Axios. Fetch: API Fetch надає метод fetch(), визначений на об'єкті вікна. Він також надає інтерфейс JavaScript для доступу та маніпулювання частинами конвеєра HTTP (запитами та відповідями). Метод отримання має один обов'язковий аргумент - URL-адресу ресурсу, який потрібно отримати. Цей метод повертає Promise, яку можна використовувати для отримання відповіді на запит[11].

HTML (HyperText Markup Language) є стандартизованою мовою розмітки документів, призначеною для створення структури та вигляду веб-сторінок. Вона

використовується для передачі контенту на Інтернеті та відображення його веб-браузерами. Коли користувач відкриває веб-сторінку, браузер отримує HTML-код цієї сторінки від сервера, використовуючи протоколи HTTP або HTTPS. Браузер інтерпретує цей HTML-код і заснований на ньому побудовує відповідний інтерфейс, який відображається на екрані монітора користувача. HTML дозволяє визначати структуру документа, розміщувати текст, зображення, посилання, таблиці, форми та інші елементи, що складають веб-сторінку[2].

CSS (Cascading Style Sheets) є мовою стилів, яка використовується для визначення зовнішнього вигляду веб-сторінок, написаних мовами розмітки даних, такими як HTML. Вона відповідає за вигляд, розташування, кольори, шрифти та інші аспекти представлення веб-сторінок. CSS працює в парі з HTML, де HTML визначає структуру та зміст сторінки, а CSS відповідає за її стиль та оформлення. За допомогою CSS розробники можуть задавати стилі для різних елементів сторінки, таких як заголовки, параграфи, таблиці, посилання тощо. Вони можуть контролювати властивості, такі як розміри, межі, фонові кольори, відступи, вирівнювання, тіні та інші атрибути, щоб створити бажаний зовнішній вигляд сторінки. CSS працює за принципом каскаду, де стилі можуть успадковуватися та перезаписуватися від батьківських елементів до дочірніх. Це дає розробникам гнучкість та можливість швидко змінювати вигляд сторінок, застосовуючи стилі до великої кількості елементів одночасно. CSS є важливою технологією веб-розробки, оскільки вона дозволяє створювати привабливі, сучасні та користувачем-орієнтовані веб-сторінки. Вона працює у поєднанні з HTML та JavaScript, які забезпечують структуру та поведінку сторінок відповідно. Розробники використовують CSS для розмітки та стилізації веб-сторінок, забезпечуючи їм привабливий та консистентний вигляд на різних пристроях та браузерах[1].

JavaScript (JS) є динамічною, об'єктно-орієнтованою прототипною мовою програмування. Вона є реалізацією стандарту ECMAScript і широко використовується для створення сценаріїв на веб-сторінках. JavaScript найчастіше

використовується на боці клієнта, тобто в браузерах, і надає можливість взаємодії з користувачем. Завдяки JavaScript, розробники можуть створювати динамічні елементи на сторінці, реагувати на події, обробляти введення користувача та змінювати вигляд сторінки без перезавантаження. JavaScript також здатна керувати браузером, змінювати URL-адреси, створювати та керувати кукісами (cookies), виконувати валідацію даних на боці клієнта та виконувати інші операції, пов'язані з браузером. Одним з важливих аспектів JavaScript є асинхронне обмін даними з сервером. За допомогою AJAX (Asynchronous JavaScript and XML), JavaScript може взаємодіяти з сервером без перезавантаження сторінки, що дозволяє здійснювати динамічне оновлення вмісту сторінки та взаємодію з серверними даними у реальному часі. JavaScript також використовується для зміни структури та зовнішнього вигляду веб-сторінок, змінюючи атрибути, стилі, додавання або видалення елементів на сторінці. За допомогою JavaScript розробники можуть створювати веб-додатки, взаємодіяти з API, реалізовувати анімацію, валідацію форм, обробляти події та багато іншого[3].

Material UI є однією з найпопулярніших бібліотек компонентів для React. Вона має вражаючу кількість понад 55 тисяч зірок на GitHub і є широко використовуваною спільнотою розробників. Material UI надає набір готових компонентів та шаблонних елементів, розроблених відповідно до принципів дизайну Google Material Design. Ці компоненти пропонують чистий та сучасний дизайн, що сприяє створенню привабливих інтерфейсів користувача. Основною перевагою Material UI є те, що вона побудована на базі React. Це означає, що компоненти Material UI можна легко використовувати в проектах React без зайвих зусиль. Вони інтегруються з React-екосистемою, що дозволяє зручно керувати станом компонентів, реагувати на події та взаємодіяти з іншими елементами додатка. Компоненти Material UI пропонують широкий спектр можливостей, включаючи кнопки, форми, таблиці, навігаційні елементи, діалогові вікна, панелі, сповіщення та багато іншого. Кожен компонент має налаштовувані властивості та методи, що дозволяють змінювати їх зовнішній вигляд та поведінку. Material UI також надає

можливість налаштування теми додатка, що дозволяє змінювати стилі компонентів згідно з власними потребами та брендовими кольорами.Загалом, Material UI є потужним інструментом для швидкої розробки привабливих інтерфейсів користувача з використанням компонентного підходу та принципів Material Design. Вона дозволяє розробникам ефективно використовувати компоненти React, забезпечуючи зручну та консистентну роботу над проектами[12].

2.1.2 Опис архітектури фронтенду вебзастосунку

При розробці програмного забезпечення для інтерфейсу користувача (UI), вибір шаблону дизайну є важливим рішенням. Однак, це не повинно впливати на продуктивність програми. Серед шаблонів проектування інтерфейсу користувача, шаблон «Container – View pattern» є дуже популярним.

Цей шаблон дозволяє відокремити логіку представлення від логіки управління даними, забезпечуючи зручність супроводу та масштабованості. За допомогою контрактів, Контейнер і Представлення можуть взаємодіяти один з одним, розділяючи завдання та спрощуючи процес розробки.

Контейнер відповідає за управління даними та бізнес-логіку, включаючи отримання даних із зовнішніх джерел, обробку даних і контроль стану додатку. Він слугує сполучною ланкою між Представленням та основними джерелами даних або сервісами, надаючи API або інтерфейс для Представлення, щоб воно могло взаємодіяти з даними.

Представлення відповідає за рендеринг користувацького інтерфейсу та надання користувачеві даних. Вхідні дані від користувачів також приймаються і надсилаються до Контейнера для обробки. Вигляд коригує інтерфейс після отримання оновлень від Контейнера.

Використання шаблону «Container – View pattern» дозволяє розробникам програмного забезпечення ефективно працювати над інтерфейсом користувача та забезпечити якість та масштабованість продукту.

У процесі розробки програмного забезпечення використання шаблону дизайну є важливим етапом, який може вплинути на зручність використання продукту. Шаблон «Container – View pattern» є досить поширеним у сфері розробки інтерфейсу користувача, оскільки дозволяє забезпечити чіткий розподіл завдань між різними компонентами програми.

За допомогою «Container – View pattern» розробники можуть зробити програму більш масштабованою та зручною у супроводі. Зокрема, цей підхід дозволяє відокремити логіку представлення від логіки управління даними, що сприяє більш гнучкій розробці і підтримці програмного забезпечення.

Крім того, «Container – View pattern» забезпечує можливість використання різних технологій та підходів для реалізації окремих компонентів програми, що дозволяє підібрати оптимальні інструменти для кожного конкретного завдання.

У цілому, використання шаблону «Container – View pattern» є важливим етапом у розробці програмного забезпечення, оскільки дозволяє забезпечити чітку організацію коду та розподіл завдань між різними компонентами програми.

Atomic pattern є патерном проектування, який застосовується для побудови дизайну користувацького інтерфейсу. Він розділяє функціональні блоки інтерфейсу на більш дрібні компоненти, які можна змінювати і поєднувати, щоб створювати нові компоненти та макети. Atomic pattern має такі основні принципи:

Атоми (Atoms) - найдрібніші компоненти, які не можуть бути розділені на більш дрібні частини, наприклад кнопки, інпути, чекбокси тощо.

Молекули (Molecules) - компоненти, які складаються з декількох атомів, наприклад форми, списки, таблиці тощо.

Організми (Organisms) - компоненти, які складаються з декількох молекул та атомів, наприклад хедер, футер, меню тощо.

Шаблони (Templates) - збірки компонентів, які утворюють складніші структури, наприклад сторінки зі списком, формою та кнопкою.

Atomic pattern дозволяє розбити дизайн на менші блоки, які можуть бути легко змінені та повторно використані. Це полегшує розробку та підтримку дизайну, забезпечуючи більшу гнучкість та швидкість розробки.

Підсумовуючи, якщо порівнювати патерни «Container – View pattern» та «Atomic pattern», то можна зрозуміти, що вони мають різний підхід до розробки інтерфейсу користувача. Container – View pattern більше зосереджений на розділенні обов'язків та покращенні тестування, масштабування та підтримки інтерфейсу користувача. Atomic pattern, з іншого боку, зосереджується на роздрібненні дизайну на більш маленькі блоки, що полегшує повторне використання.

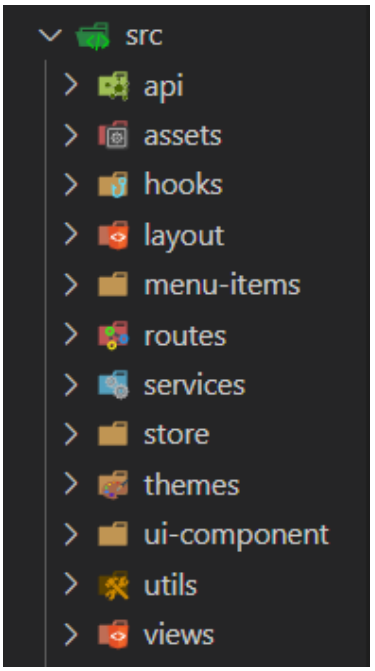

Рис. 1.0. Структура фронтенду проєкту

Джерело: [створено автором]

2.1.3. Сервіси серверної частини

На рисунку (рис. 1.0) зображено діаграму зміни станів реєстрації та авторизації користувача.

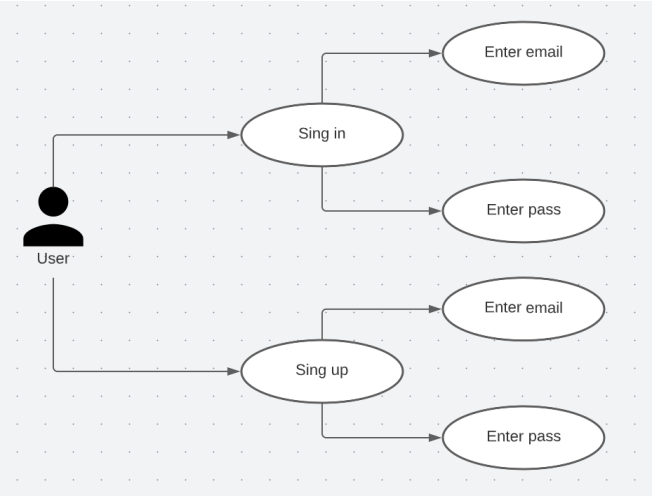

Рис. 1.1. Компонент авторизації та реєстрації

Джерело: [створено автором]

На рисунку (рис. 1.1) зображено діаграму зміни станів для зареєстрованого користувача

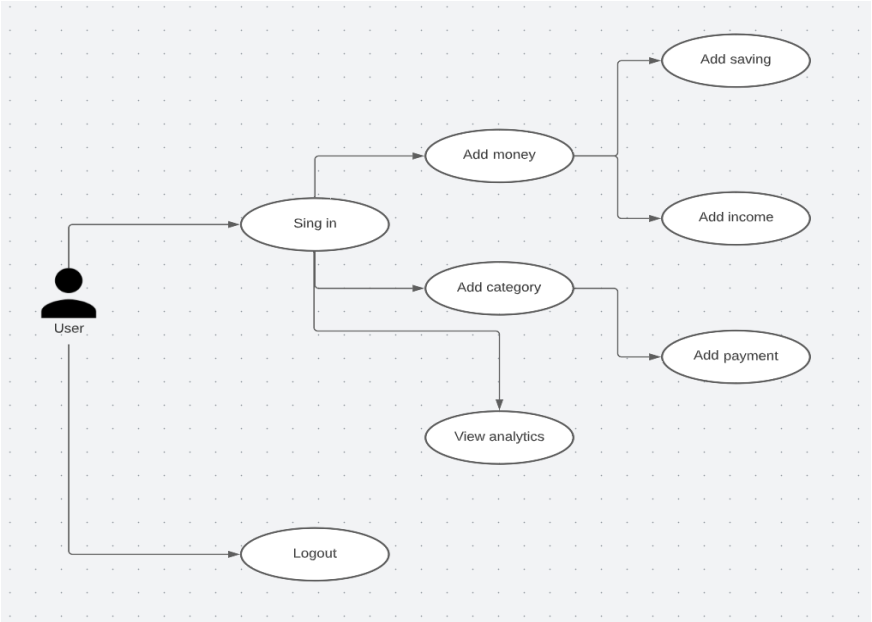

Рис. 1.2. Внесення даних та їх аналітика

Джерело: [створено автором]

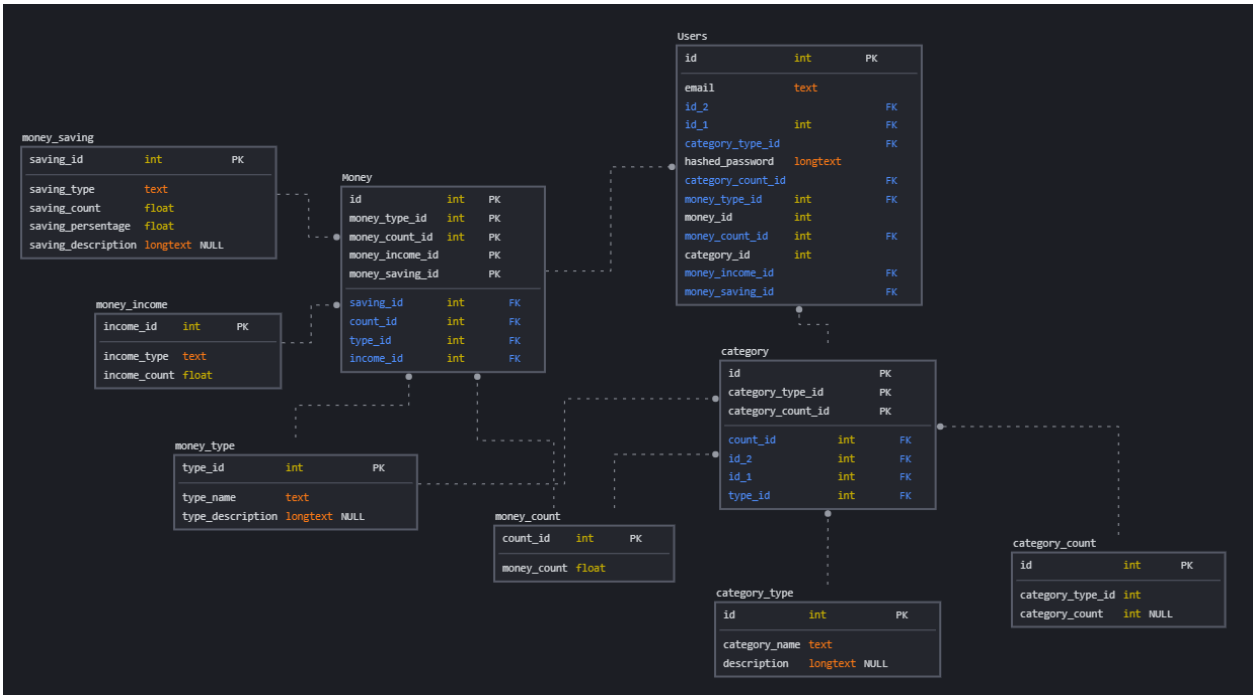

Рис. 1.3. Структура бази даних

Джерело: [створено автором]

Архітектура рішення - Clean Architecture є популярним підходом в світі програмного забезпечення, який ставить акцент на поділ обов'язків та забезпечення високої ступені залежності від деталей реалізації. Вона дозволяє створювати модульне, розширюване та тестируване програмне забезпечення[13].

Clean Architecture використовує концепцію кількох кіл, які накладаються одне на одне:

1. Найвнутрішнє коло представляє незалежність від фреймворків та зовнішніх бібліотек. Воно містить найбільш основні компоненти програми, такі як сутності (entities) та використовувані в програмі правила бізнес-логіки.
2. Коло, що знаходиться навколо внутрішнього, представляє використання фреймворків та зовнішніх бібліотек. Його завдання - адаптувати зовнішні

інтерфейси до внутрішніх компонентів, наприклад, шари доступу до даних та інтерфейси користувацького інтерфейсу.

3. Наступне коло - це шар використання (use cases), який містить бізнес-правила та додаткову логіку, специфічну для додатку. Він залежить від внутрішнього кола, але не залежить від зовнішніх фреймворків або бібліотек.
4. Зовнішне коло - це шар інтерфейсу, який включає компоненти, що забезпечують взаємодію з користувачем або зовнішніми системами, такі як графічний інтерфейс користувача, API тощо. Він залежить від шару використання, але не залежить від внутрішніх деталей.

Ця архітектура забезпечує чітку границю між різними компонентами програми, полегшує тестування та розширення, а також забезпечує більшу незалежність від зовнішніх змін. Кожен шар має свої відповідальності та рівень абстракції, що дозволяє змінювати або замінювати один шар без впливу на інші.

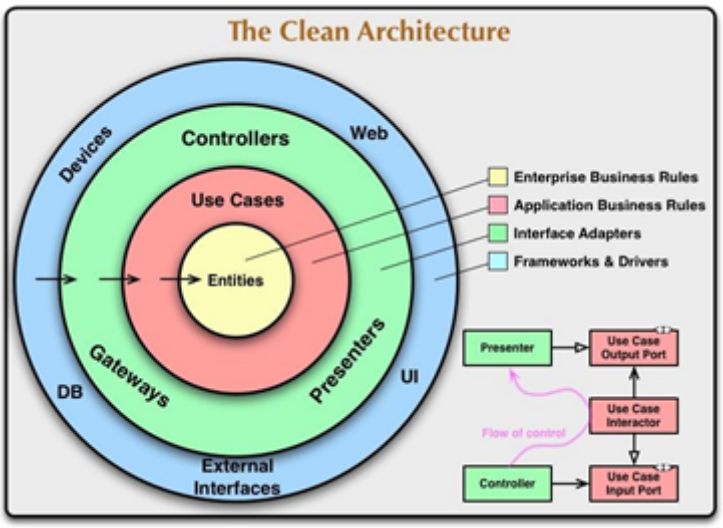

Рис. 1.3. Діаграма clean architecture

Swagger (також відомий як OpenAPI) - це набір інструментів для створення, визначення та документування веб-сервісів API. Він надає зручний спосіб описати структуру та функціональні можливості API, що дозволяє автоматично генерувати документацію, створювати клієнтські бібліотеки та навіть тестувати API.

Swagger дозволяє створювати файл специфікації API у форматі JSON або YAML, де вказується інформація про шляхи (endpoints), параметри, типи даних та можливі операції. З цього файлу можна автоматично згенерувати документацію в зручному для перегляду форматі, яка описує всі доступні ендпоінти, їх параметри, відповіді та приклади використання.

Swagger також надає інтерактивне середовище Swagger UI, яке дозволяє відображати документацію API у зручній веб-інтерфейсі. Користувачі можуть переглядати доступні ендпоінти, вводити значення параметрів та випробовувати API безпосередньо з браузера. Це дозволяє зробити процес розробки та тестування API більш зручним і ефективним.

Використання Swagger для документування та перегляду сервісів та контролерів допомагає команді розробникам та іншим зацікавленим сторонам швидко зрозуміти функціональність API та використовувати його згідно з документацією.

Основою будь яких сервісів є акаунт користувача, він повинен бути надійно захищеним, та містити в собі всю необхідну інформацію для роботи сервісу.

/api/users – Реєстрація користувача

/api/token – Перевірка токену та вхід в систему

/api/users/me – Отримання токену

/api/users/currency – Отримання базової валюти користувача

/api/users/edit – Редагування базової валюти користувача

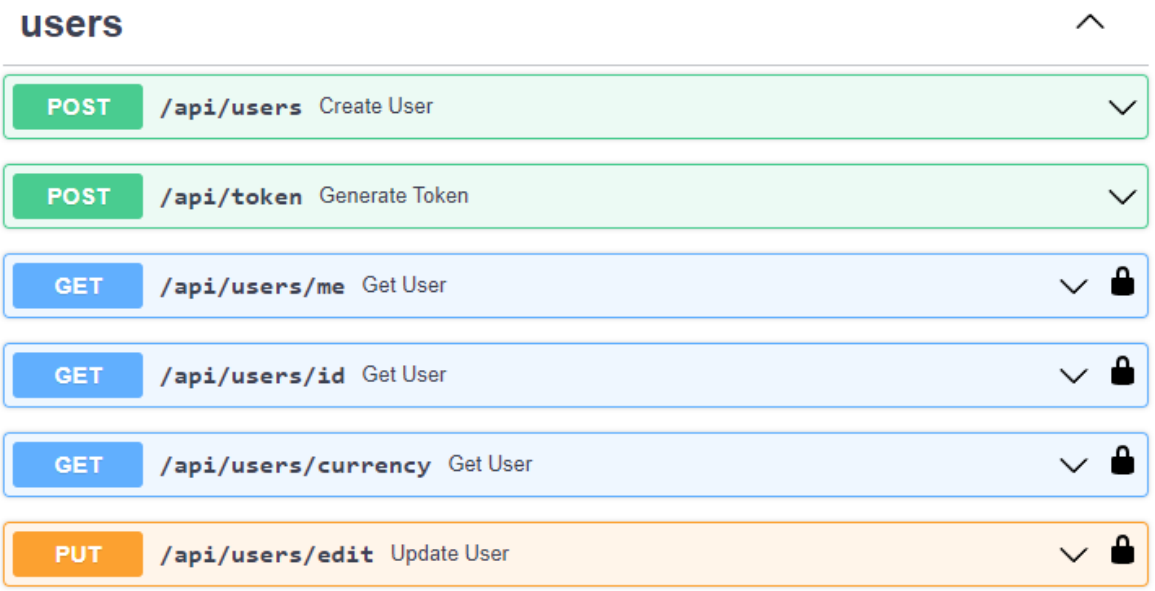

Рис. 1.4. Документація в Swagger контролера користувача

Джерело: [створено автором]

Чи не малу роль у захисті користувача відіграє шифрування паролю, а саме JWTtoken(Рис. 1.5)

JWT (JSON Web Token) - це стандарт відкритого формату для передачі безпечної інформації між сторонами у форматі JSON. JWT складається з трьох частин, які розділені крапками: заголовка (header), пейлоаду (payload) та підпису (signature).

Заголовок містить тип токена та алгоритм шифрування, використовуваний для створення підпису.

Пейлоад містить корисну інформацію, таку як ім'я користувача, час дії токена та будь-яку додаткову інформацію, яку потрібно передати.

Підпис створюється за допомогою алгоритму шифрування, вказаного в заголовку. Він використовується для перевірки цілісності токена та підтвердження, що токен був створений довіреною стороною.

JWT-токени широко використовуються для аутентифікації та авторизації користувачів в додатках та API. Коли користувач аутентифікується, сервер створює JWT-токен та відправляє його клієнту. Клієнт зберігає токен та передає його з кожним запитом до сервера. Сервер перевіряє підпис токена та користувача, щоб переконатися, що запит приходить від довіреної сторони.

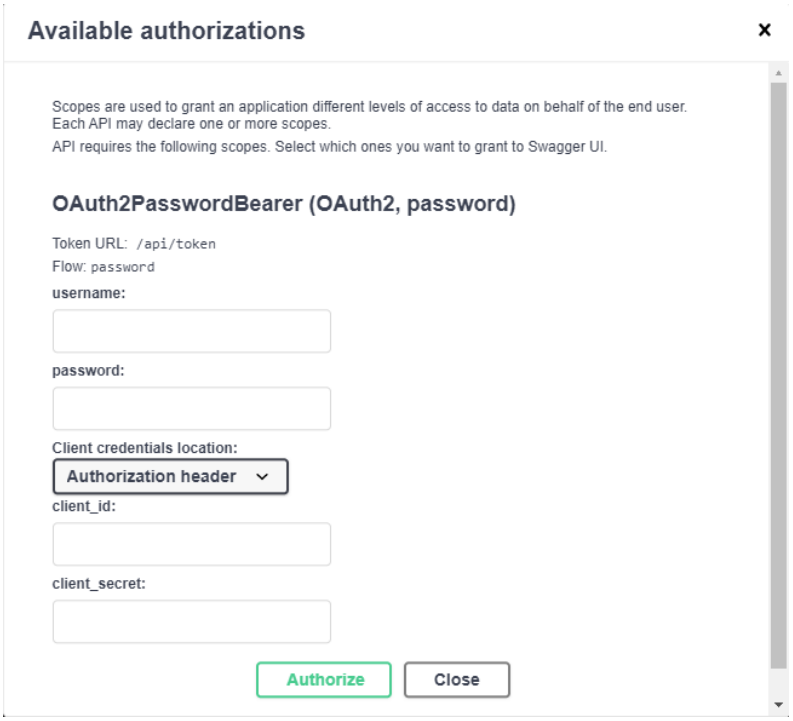

Рис. 1.5 Авторизація з допомогою JwtBearer

Джерело: [створено автором]

Не авторизований користувач може зареєструватися ввівши e-mail і пароль. Вже зареєстрований користувач, може увійти в додаток ввівши e-mail і пароль.

Сервіси керування доданням коштів та керування ними. Користувач може додати кошти на баланс, переглянути поточний баланс карти або готівки. Змінити суму коштів або ж взагалі видалити тип коштів з акаунту.

/money/add_money - Додання коштів на баланс

/money/get_balance - Отримання балансу всіх типів коштів

/money/get_total_balance - Отримання загального балансу

/money/get_money_type_id - Отримання id типу коштів для подальшого використання

/money/edit_money_type - Редагування типу коштів

/money/money_type_delete - Видалення типу коштів

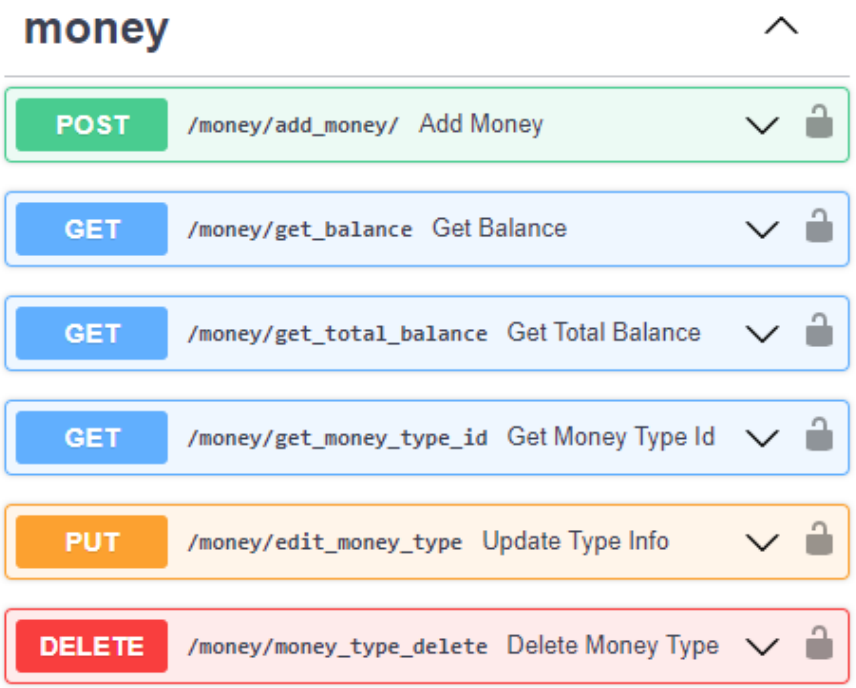

Рис. 1.6. Документація в Swagger контролера Money

Джерело: [створено автором]

Основну увагу хотілось би привернути до методу обчислення загальної суми на рахунку. Оскільки більшість API відкритого доступу надають обмежену кількість використань або не мають курсу для гривні було вирішено діставати JSON дані з API Національного банку України і лише потім конвертувати їх у інші валюти. Нажаль цей спосіб теж не є ідеальним оскільки довелося робити перевірку базової валюти користувача та додавати вручну обмінний курс якщо основна валюта гривня.(див. лістинг 1.0)

Лістинг 1.0 Конвертація валют

```
def get_exchange_rates():
    api_url = f'https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json'
```

```
    response = requests.get(api_url)
    if response.status_code == requests.codes.ok:
        exchange_rates = response.json()
        return exchange_rates
    else:
        print("Error:", response.status_code, response.text)
        return None


def get_total_amount(user_id: int, db: _orm.Session, api_key):
     data_array = db.query(_models.Money_type).filter(_models.Money_type.user_id == user_id).all()
                                    name_list = [{"type_quantity":item.type_quantity, "type_currency":item.type_currency } for item in data_array]
    user = db.query(_models.User).get(user_id)
    base_currency = user.main_currency
    exchange_rates = get_exchange_rates()
    exchange_rates.append(
        {"txt":"Гривня","rate":1,"cc":"UAH"}
    )
    sum = 0
    for item in name_list:
        if item["type_currency"] == "UAH" and base_currency != "UAH":
            data = list(filter(lambda u: u["cc"] == base_currency, exchange_rates))[0]["rate"]
            sum += item["type_quantity"] / data
        elif item["type_currency"] == "UAH" and base_currency == "UAH":
            sum += item["type_quantity"]
        else:
                        sum += (list(filter(lambda u: u["cc"] == item["type_currency"], exchange_rates))[0]["rate"]/list(filter(lambda u: u["cc"] == base_currency, exchange_rates))[0]["rate"]
            )*item["type_quantity"]
```

```
result = round(sum,2)
return result
```

Збереження коштів представляє собою ряд контролерів при використанні яких користувач має доступ до можливості додати кошти на зберігальний рахунок та відкладати їх на якусь конкретну ціль (Рис. 1.7.).

/money/add_money_saving - Внесення коштів на зберігальний баланс

/money/get_saving - Отримання зберігального балансу

/money/edit_saving_type - Редагування зберігального рахунку

/money/money_saving_delete - Видалення зберігального балансу

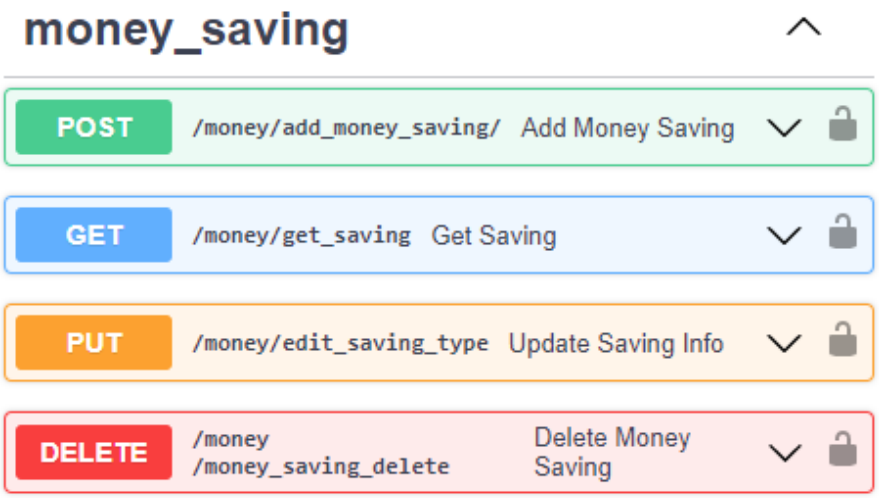

Рис. 1.7. Документація в Swagger контролера Money_Saving

Джерело: [створено автором]

В свою чергу контролер Money_income відповідає за надходження коштів на баланс по типу заробітньої плати, дивідендів (Рис. 1.8.).

/money/add_money_income - Внесення способів надходження коштів

/money/get_money_income - Отримання надходжень користувача

/money/edit_money_income - Редагування надходжень

/money/money_income_delete - Видалення типу надходження

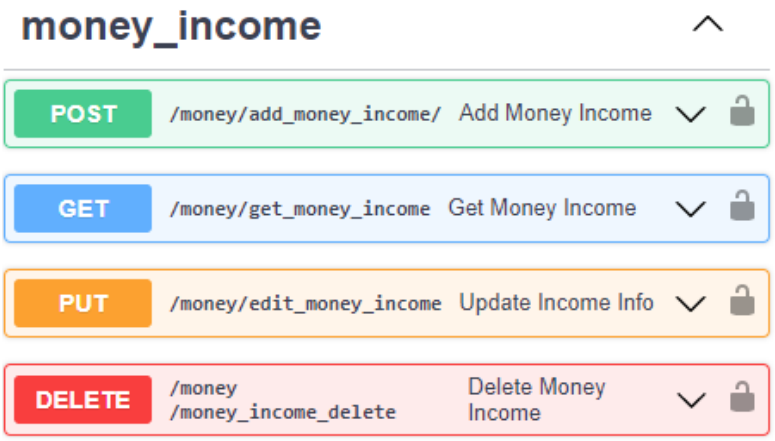

Рис. 1.8. Документація в Swagger контролера Money_income

Джерело: [створено автором]

Користувач має змогу додавати категорії та витрати в конкретні категорії за допомогою контролера Category.

/category/add_category - Внесення категорії витрат

/category/get_category - Отримання даних категорії

/category/edit_category - Редагування категорії витрат

/category/category_delete - Видалення категорії витрат

/category/add_category_money - Внесення коштів до категорії витрат

/category/get_costs – Отримання всіх витрат користувача

/category/get_costs_by_id – Отримання витрат користувача для конкретної категорії

/category/delete_category_money - Видалення витрат з категорії.

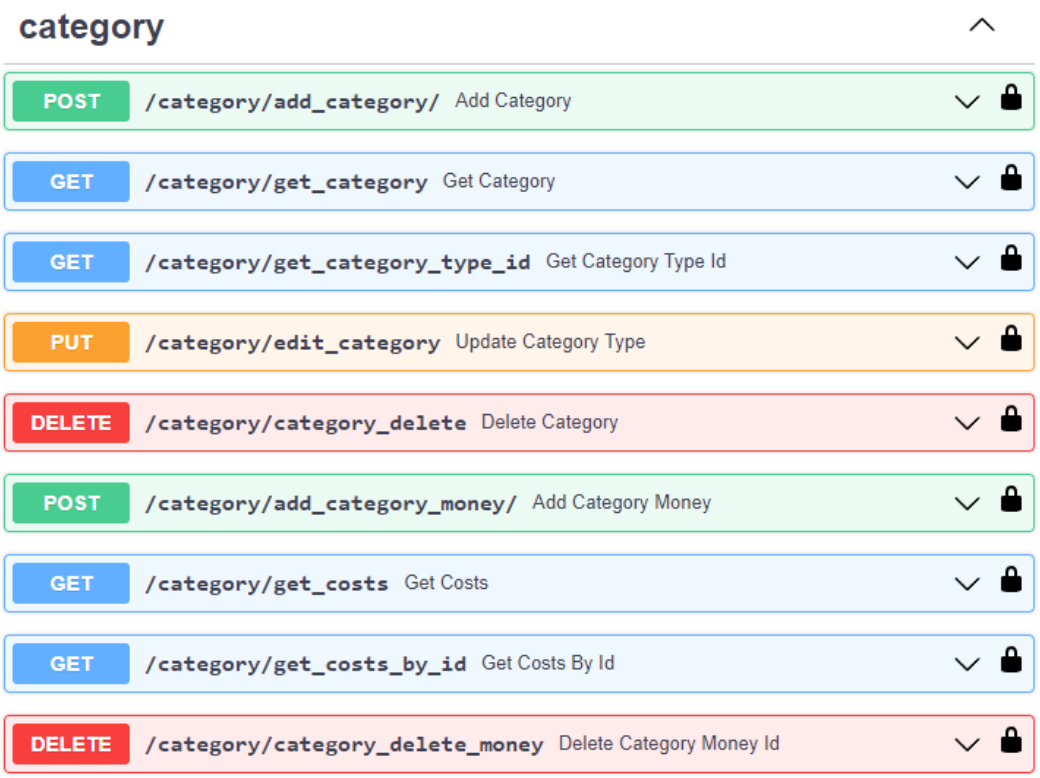

Рис. 1.9. Документація в Swagger контролера Category

Джерело: [створено автором]

2.1.4. Сервіси клієнтської частини

Програма була створена за допомогою ReactJS - надійної та масштабованої технології.

Перш за все, для розробки веб-додатку на основі React JS було необхідно налаштувати середовище розробки. Для цього було використано такі інструменти, як редактор коду Visual Studio Code та браузер Google Chrome. Після цього можна було приступати до написання коду і розробки програми.

Розробка фронтенду проєкту розпочалась з створення сторінок для входу та реєстрації користувачів

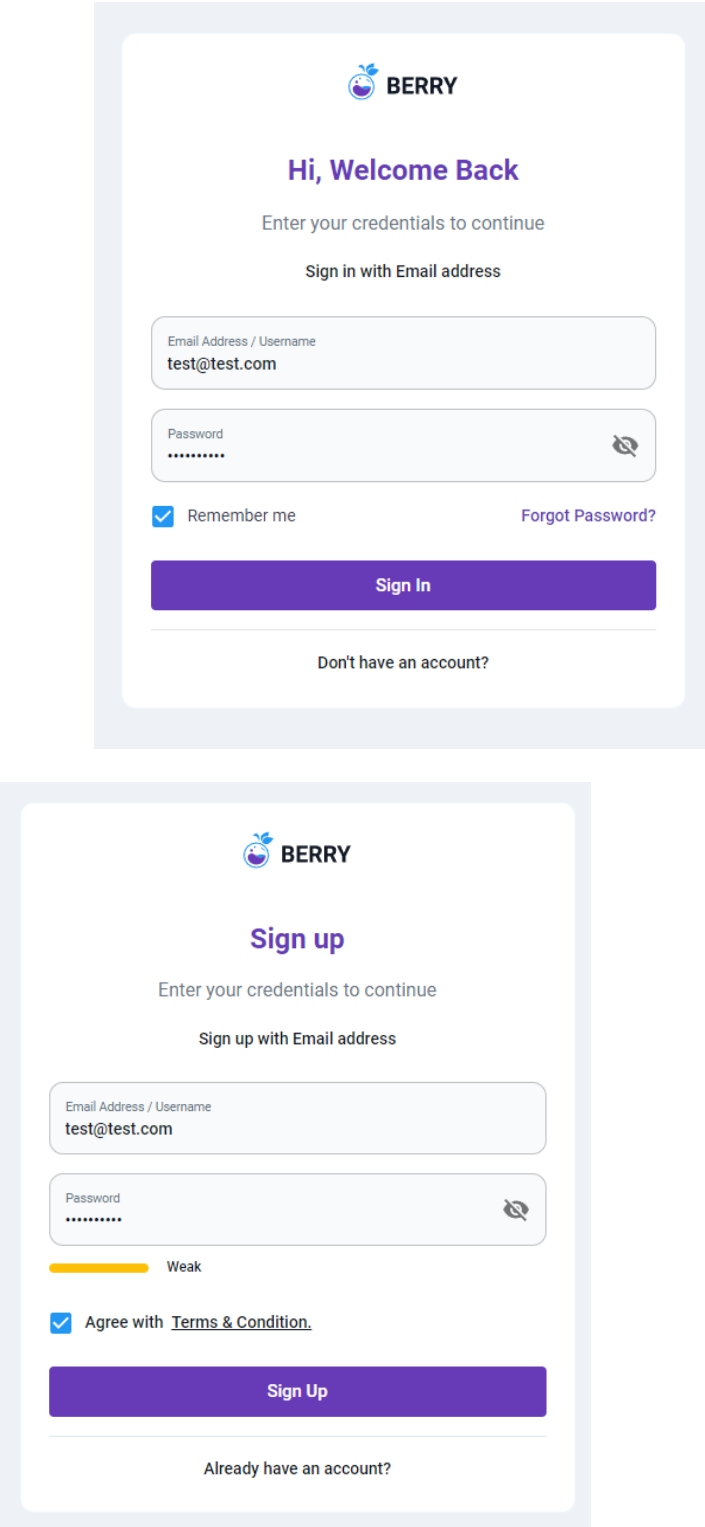

Рис. 2.0. Форма для входу та реєстрації акаунту

Джерело: [створено автором]

Алгоритм оцінки складності паролю(рис. 2.0) - це математична формула або програма, яка визначає, наскільки складним є пароль для вгадування. Для оцінки

складності паролю використовуються різні фактори, такі як довжина паролю, тип символів, які використовуються в паролі, і його ступінь унікальності.

Одним з найпростіших методів оцінки складності паролю є використання формули, що враховує довжину паролю та кількість унікальних символів, які використовуються в паролі. Ця формула може використовуватись для оцінки складності паролю відносно до відомих атак на паролі, таких як словникові атаки або атаки з використанням brute force.

Іншим методом оцінки складності паролю є використання програм, які аналізують різні фактори, такі як довжина паролю, склад символів, ступінь унікальності та інші параметри, які можуть впливати на складність паролю. Такі програми можуть давати точнішу оцінку складності паролю та рекомендувати певні зміни в паролі, щоб зробити його більш безпечним.

Один зі способів, які використовуються для забезпечення безпеки паролю, - це використання "солі". Сіль - це довільний рядок символів, який додається до паролю перед хешуванням. Вона додає додаткову складність до паролю, що робить атаки з використанням брутфорсу більш складними.

Всі ці методи оцінки складності паролю допомагають зробити його більш безпечним та унеможливити його вгадування. Однак, варто пам'ятати, що найкращим способом забезпечення безпеки пароля є використання сильного та унікального пароля, який складається з різноманітних символів та не повторюється для різних акаунтів. Також важливо використовувати двофакторну аутентифікацію та регулярно змінювати паролі, щоб уникнути можливих загроз безпеці в Інтернеті. За допомогою алгоритмів оцінки складності паролю можна зробити паролі більш складними та надійними, що допоможе забезпечити безпеку в Інтернеті.

Загальна робота функції для визначення сили пароля і встановлення кольорової позначки для показника сили пароля представлена у лістингу 1.1.

Функція strengthIndicator отримує рядок (імовірно, це пароль) і визначає силу пароля на основі різних умов, таких як довжина пароля, наявність цифр, спеціальних символів та комбінації різних типів символів. Вона повертає числове значення, яке відповідає загальній силі пароля.

Функція strengthColor отримує результат виклику функції strengthIndicator і встановлює кольорову позначку для показника сили пароля на основі отриманого значення. Вона повертає об'єкт з міткою (наприклад, "Weak", "Normal", "Strong") і відповідним кольором, який може бути використаний для візуального представлення сили пароля.

Цей код є загальною функціональністю для оцінки і відображення сили пароля з використанням певних критеріїв, таких як довжина, наявність символів різних типів та спеціальних символів.

Лістинг 1.1

```
import value from 'assets/scss/_themes-vars.module.scss';
const hasNumber = (number) => new RegExp(/[0-9]/).test(number);
const hasMixed = (number) => new RegExp(/[a-z]/).test(number) && new RegExp(/[A-Z]/).test(number);
const hasSpecial = (number) => new RegExp(/[!#@$%^&*)(+=._-]/).test(number);
export const strengthColor = (count) => {
  if (count < 2) return { label: 'Poor', color: value.errorMain };
  if (count < 3) return { label: 'Weak', color: value.warningDark };
  if (count < 4) return { label: 'Normal', color: value.orangeMain };
  if (count < 5) return { label: 'Good', color: value.successMain };
  if (count < 6) return { label: 'Strong', color: value.successDark };
  return { label: 'Poor', color: value.errorMain };
};

export const strengthIndicator = (number) => {
```

```
    let strengths = 0;
    if (number.length > 5) strengths += 1;
    if (number.length > 7) strengths += 1;
    if (hasNumber(number)) strengths += 1;
    if (hasSpecial(number)) strengths += 1;
    if (hasMixed(number)) strengths += 1;
    return strengths;
};
```

Додання коштів на баланс є основною функціональністю проєкту і здійснюється за допомогою модального вікна. При додаванні нового типу коштів користувачеві потрібно вказати наступну інформацію(див. рис. 2.2):

1. Назва: Користувач має вказати назву для нового типу коштів, яка найкраще описує їх природу або джерело. Наприклад, "Банківська картка", "Готівка", "Інвестиційний рахунок" тощо.
2. Еквівалент: числове значення, яке представляє еквівалент коштів у даному типі.
3. Валюта: Користувач обирає валюту, в якій знаходяться кошти даного типу. Вибір валюти дозволяє правильно відображати суми коштів у відповідній валюті на інтерфейсі та проводити конвертацію при необхідності.

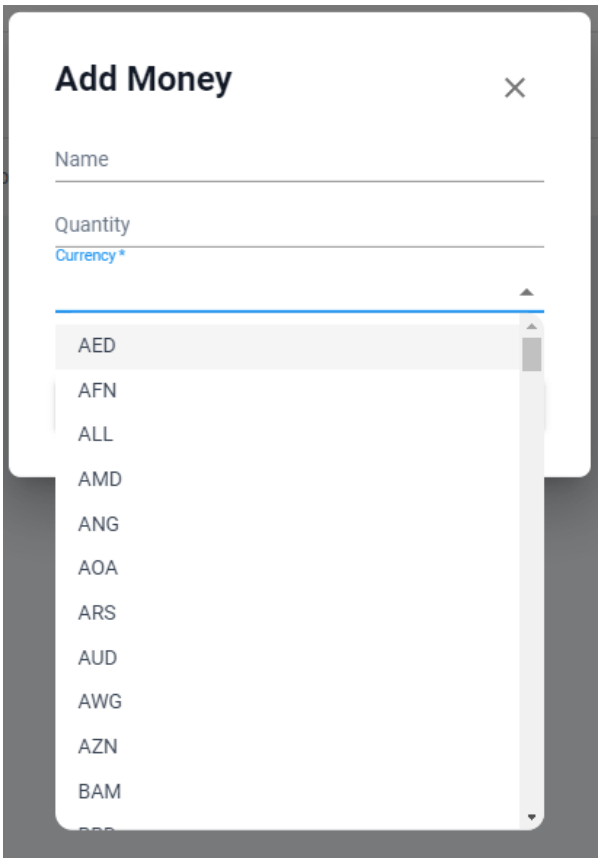

Рис. 2.2. Модальне вікно для додавання коштів

Джерело: [створено автором]

Можливість додавати більше ніж один варіант коштів це база для будь якого проєкту, саме це і реалізовано у власному(див. рис. 2.3)

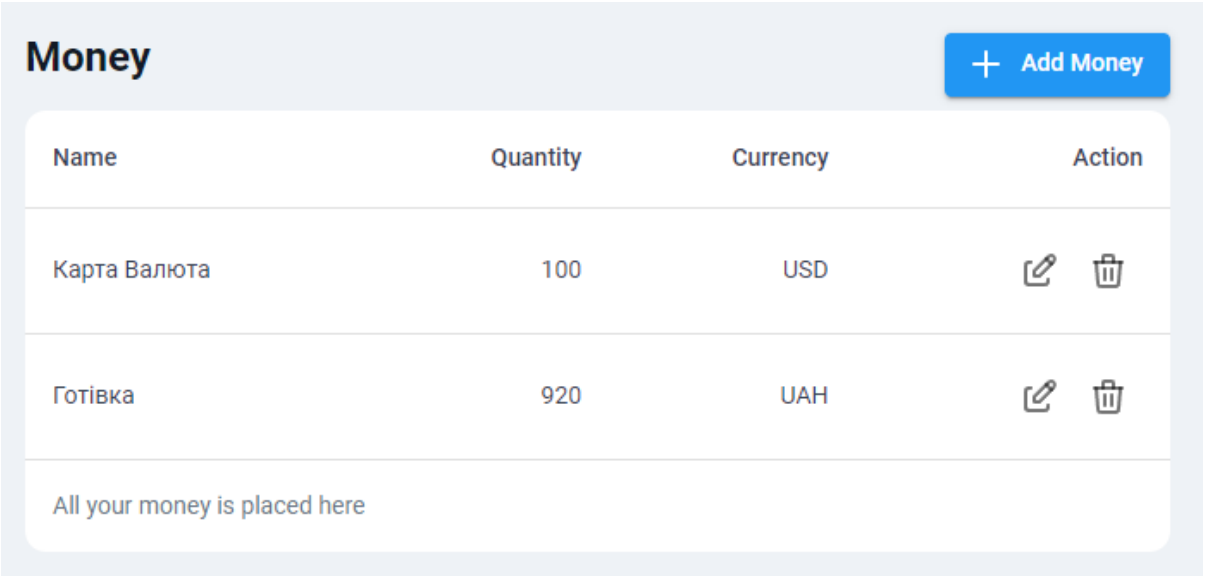

Рис. 2.3. Сторінка керування коштами

Джерело: [створено автором]

Функція "Money Saving" або збереження коштів, що працює як скарбничка, є корисним елементом додатку. Вона дозволяє користувачам збирати та відкладати кошти для майбутніх цілей або економічних потреб. (див. рис. 2.4)

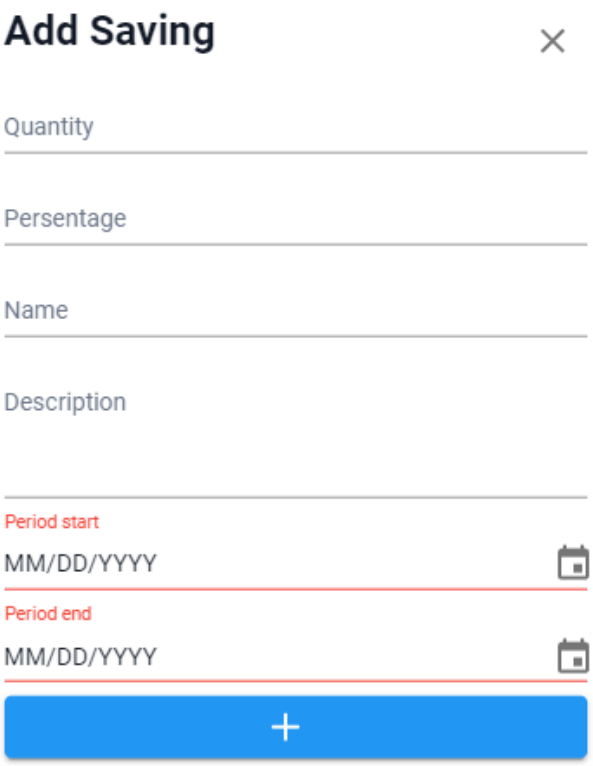

Рис. 2.4. Модальне вікно для додання збереження коштів

Джерело: [створено автором]

Вигляд таблиці збереження коштів (див. рис. 2.5)

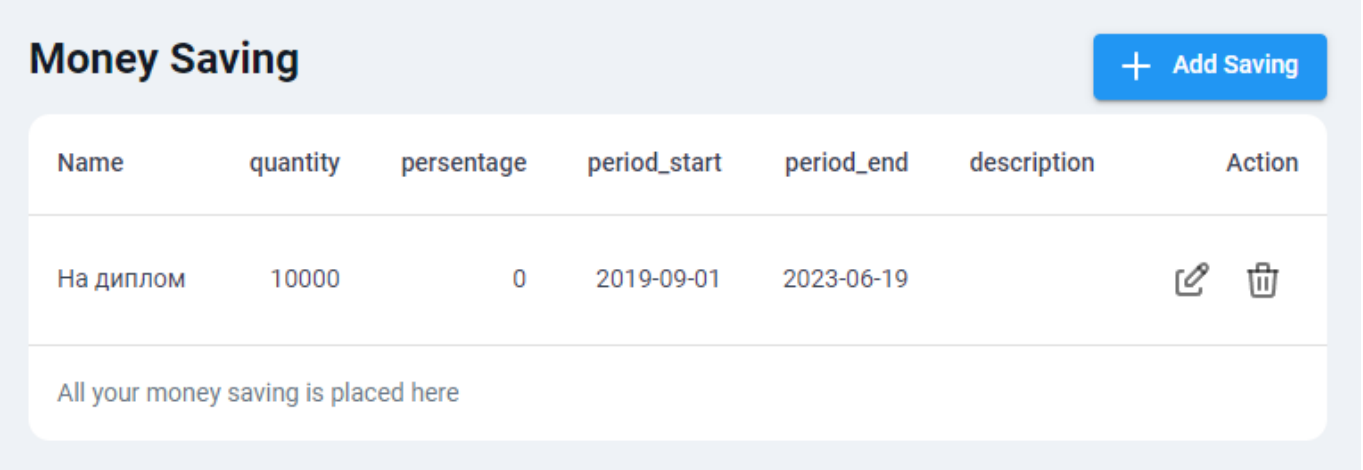

Рис. 2.5. Сторінка керування збереженнями

Джерело: [створено автором]

Також присутня можливість додавання надходжень які з певним періодом часу будуть надходити на основний баланс. Ці дані дозволяють системі автоматично додавати надходження до основного балансу відповідно до вказаного періоду. Це спрощує управління фінансами користувача та дозволяє точніше прогнозувати їх бюджет. Користувачі можуть з легкістю налаштувати та керувати своїми регулярними надходженнями в системі. (див. рис.2.6-7)

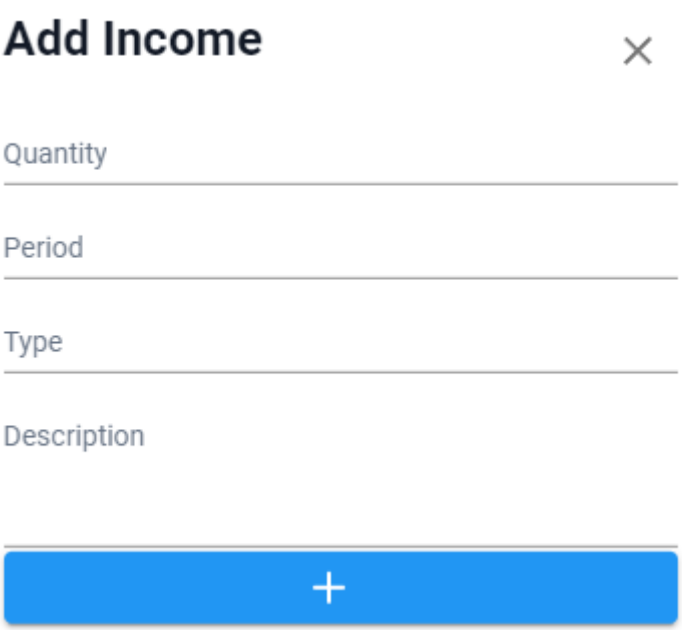

Рис. 2.6. Модальне вікно додання надходжень

Джерело: [створено автором]

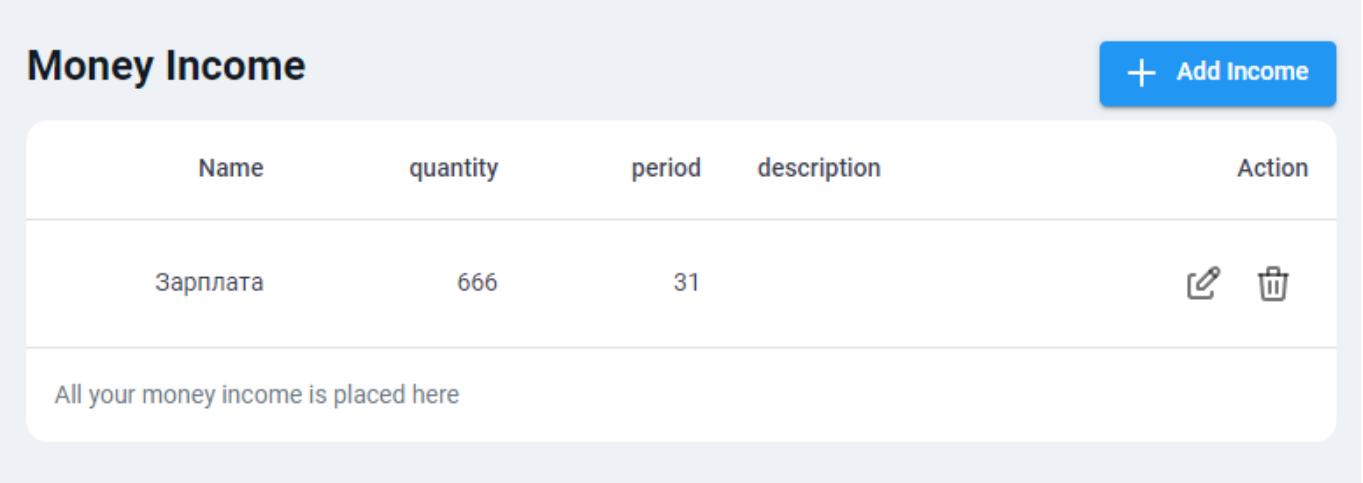

Рис. 2.7. Сторінка керування збереженнями

Джерело: [створено автором]

Також впроваджено додання категорій(див. рис.2.8-9)

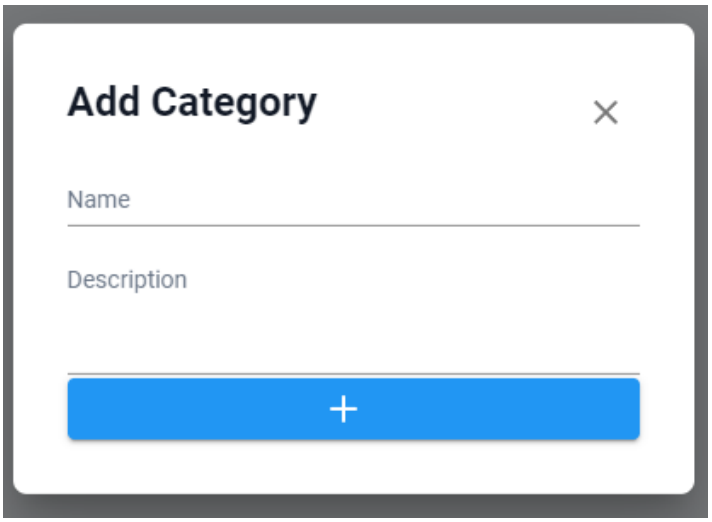

Рис. 2.8. Модальне вікно для додання категорій

Джерело: [створено автором]

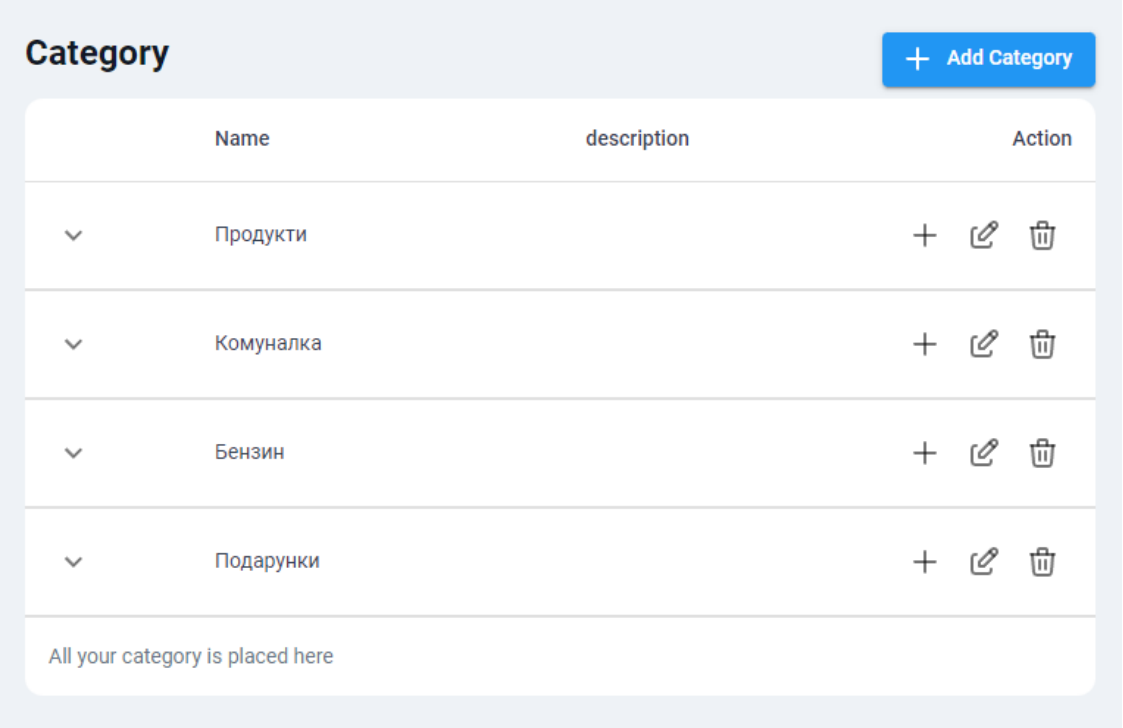

Рис. 2.9. Сторінка для категорій

Джерело: [створено автором]

Введення витрат для конкретних категорій через модальне вікно є зручним способом забезпечити користувачам легке та швидке додавання витрат. Після натискання плюсика, який представлений на інтерфейсі додатку, з'являється модальне вікно додання витрат (див. рис. 2.10). Модальне вікно має інтуїтивний та зрозумілий дизайн, де користувач легко зможе заповнити необхідні поля та зберегти витрату.

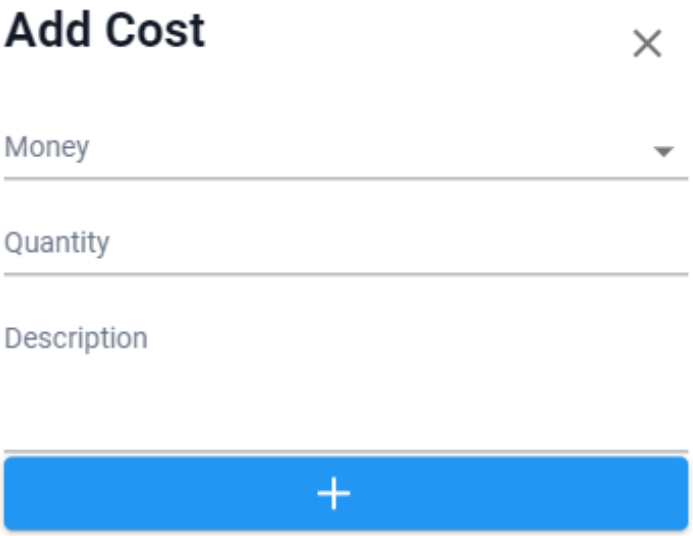

Рис. 2.10. Модальне вікно для додання витрат

Джерело: [створено автором]

Переглянути витрати можна можна натиснувши стрілку вниз у першій колонці таблиці (див. рис. 2.9) після чого з’являються наступні таблиці, що містять детальнішу інформацію про витрати. Ці таблиці дозволяють користувачам отримати докладну інформацію про їх витрати межах конкретної категорії. Користувачі можуть легко переглядати та аналізувати свої витрати, щоб краще управляти своїми фінансами. (див. рис. 2.11)

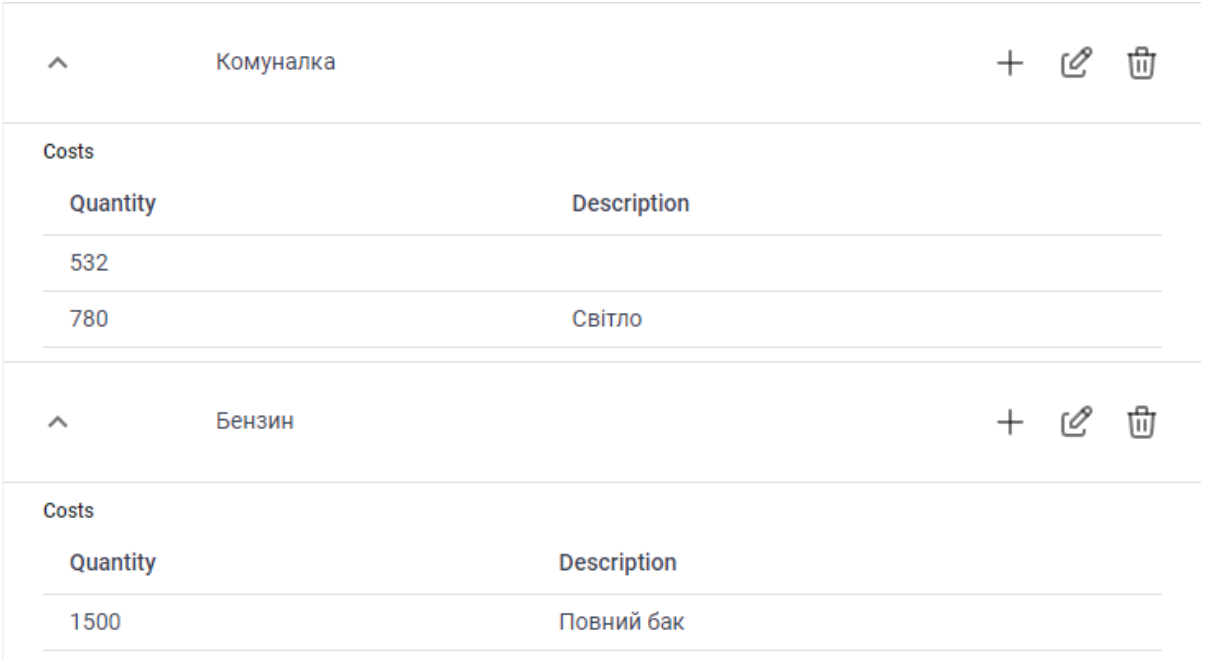

Рис. 2.11. Вигляд доданих витрат

Джерело: [створено автором]

На сайті присутня бічна панель навігації(див. рис. 2.12)

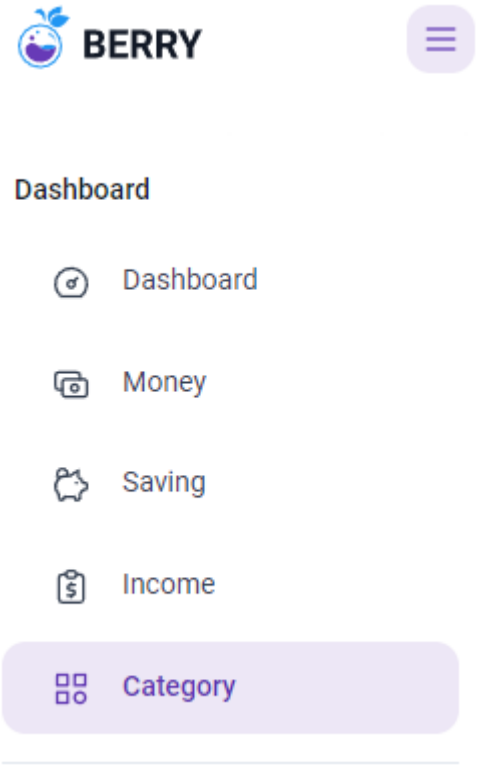

Рис. 2.12. Панель навігації

Джерело: [створено автором]

Інформаційне меню, яке містить кнопку виходу з облікового запису та можливість зміни базової валюти, є важливим елементом для управління обліковим записом користувача та налаштування проекту під його потреби. Це дозволяє користувачу керувати своїм обліковим записом та персоналізувати додаток.

Кнопка виходу з облікового запису дозволяє користувачеві безпечно вийти зі свого облікового запису, завершуючи сеанс роботи з додатком. Це важливо для забезпечення безпеки та конфіденційності даних користувача.

Можливість зміни базової валюти користувача впливає на коректне відображення інформації на головній сторінці проекту. Користувач може вибрати валюту, яка відповідає його місцезнаходженню або основній валюті, яку він

використовує для ведення фінансового обліку. Це дозволяє забезпечити зрозуміле та зручне відображення сум та інших фінансових даних в потрібній валюті.

Інформаційне меню з кнопкою виходу з облікового запису та можливістю зміни базової валюти доповнює функціонал додатку та надає користувачеві більше контролю та налаштувань, що сприяє зручному та персоналізованому досвіду використання системи моніторингу витрат. (див. рис. 2.13.)

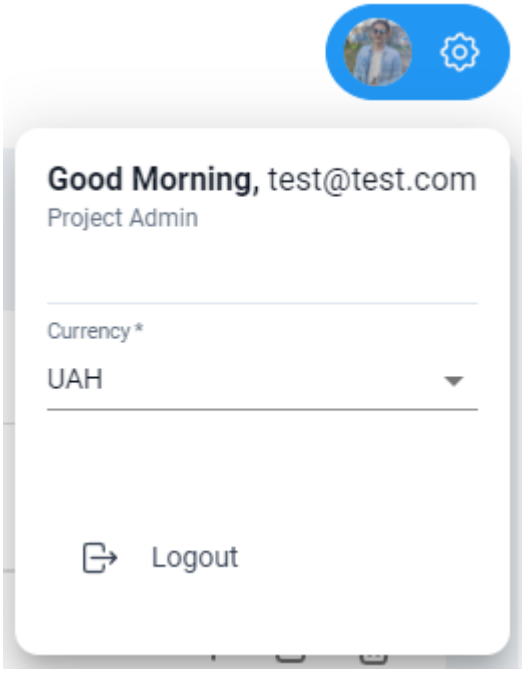

Рис. 2.13. Модальне вікно інформаційного меню

Джерело: [створено автором]

Головна сторінка (Dashboard) з аналітичними даними є важливим елементом додатку для менеджмента витрат та надходжень. Основна мета цієї сторінки - надати користувачеві швидкий та зрозумілий огляд їх фінансового стану. Нижче наведено компоненти, які включені на головній сторінці:

1. Загальний баланс: Цей компонент показує користувачеві їх поточний баланс - суму грошей, які вони мають на рахунку або відсотковій ставці. Це дає загальне уявлення про їх фінансову ситуацію.

2. Останній використаний платіжний засіб: Цей компонент відображає інформацію про останній використаний платіжний засіб користувача, наприклад, останню використану кредитну карту або банківський рахунок. Це дозволяє швидко переглянути деталі останніх транзакцій.
3. Загальні збереження та надходження: Цей компонент показує користувачеві загальну суму їх збережень та надходжень. Він може включати інформацію про внески на рахунок, інвестиційні портфелі, пенсійні фонди тощо. Це дозволяє користувачам відстежувати їх загальне фінансове благополуччя.
4. Останні витрати: Цей компонент показує користувачеві останні здійснені витрати. Він може включати інформацію про категорії витрат, суму витрати та дату. Це допомагає користувачам зрозуміти, куди йдуть їх гроші та контролювати свої витрати.
5. Діаграма витрат: Цей компонент представляє діаграму, на якій відображено, на що було здійснено найбільше витрат у поточному місяці. Це може бути кругова діаграма з розподілом витрат за категоріями, наприклад, їжа, транспорт, розваги, покупки тощо. Це дає користувачеві візуальне уявлення про те, на що вони витрачають найбільше грошей.

Важливо, щоб головна сторінка була зрозумілою, інтуїтивно зрозумілою та естетично привабливою. Користувач повинен швидко знаходити необхідну інформацію та легко користуватися функціями додатку.

(див. рис. 2.14-2.19)

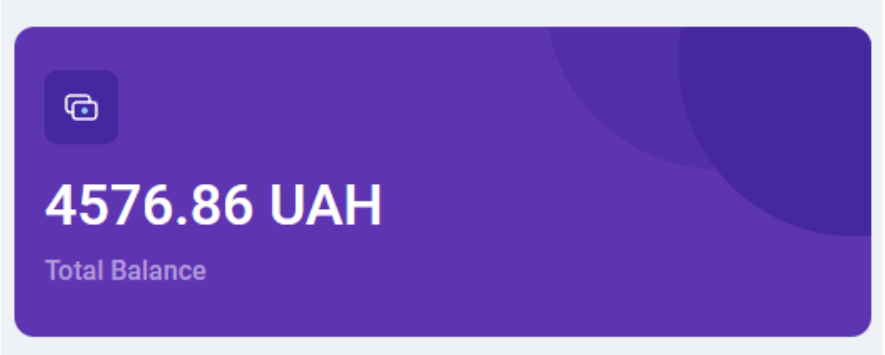

Рис. 2.14. Катка загального балансу

Джерело: [створено автором]

Рис. 2.15. Катка останнього використаного платіжного засобу

Джерело: [створено автором]

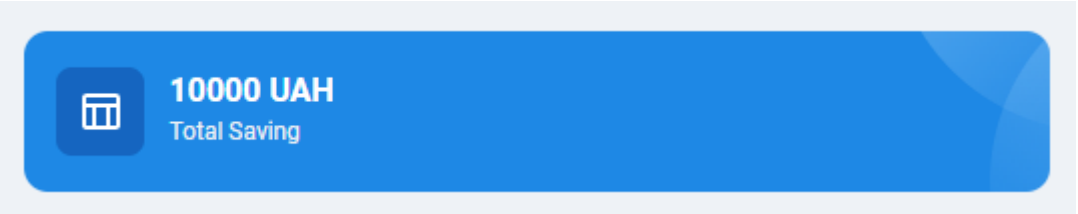

Рис. 2.16. Катка загальних збережень

Джерело: [створено автором]

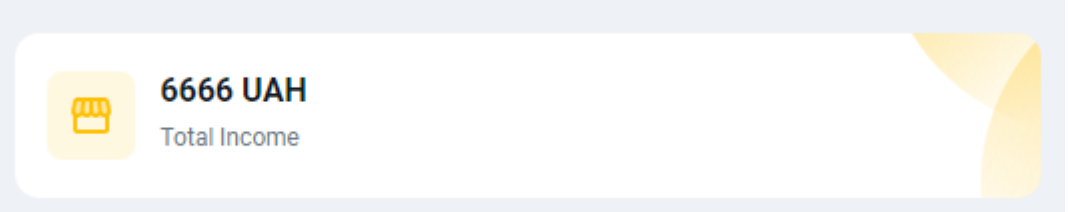

Рис. 2.17. Катка загального надходжень

Джерело: [створено автором]

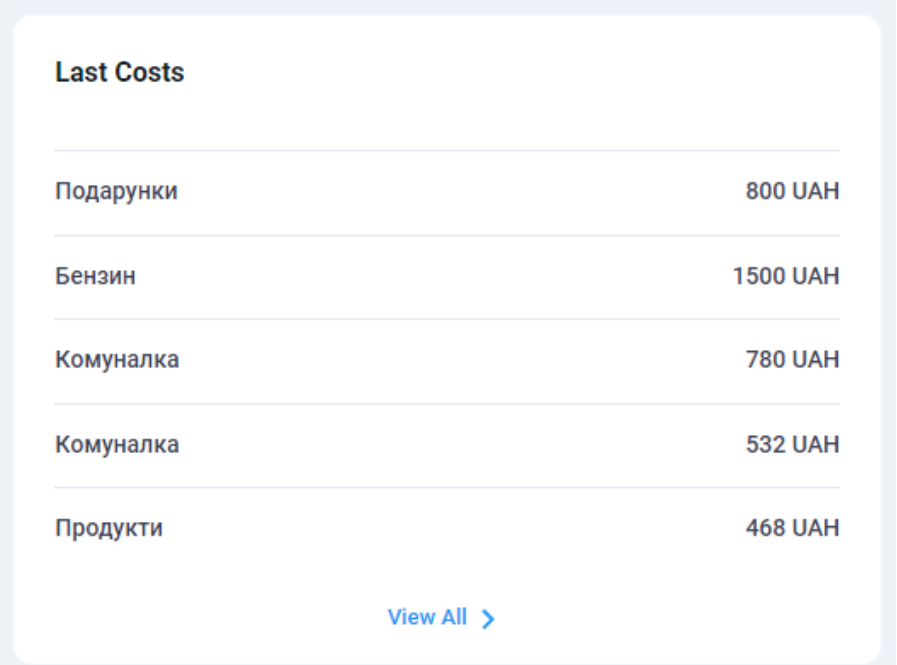

Рис. 2.18. Катка останніх витрат

Джерело: [створено автором]

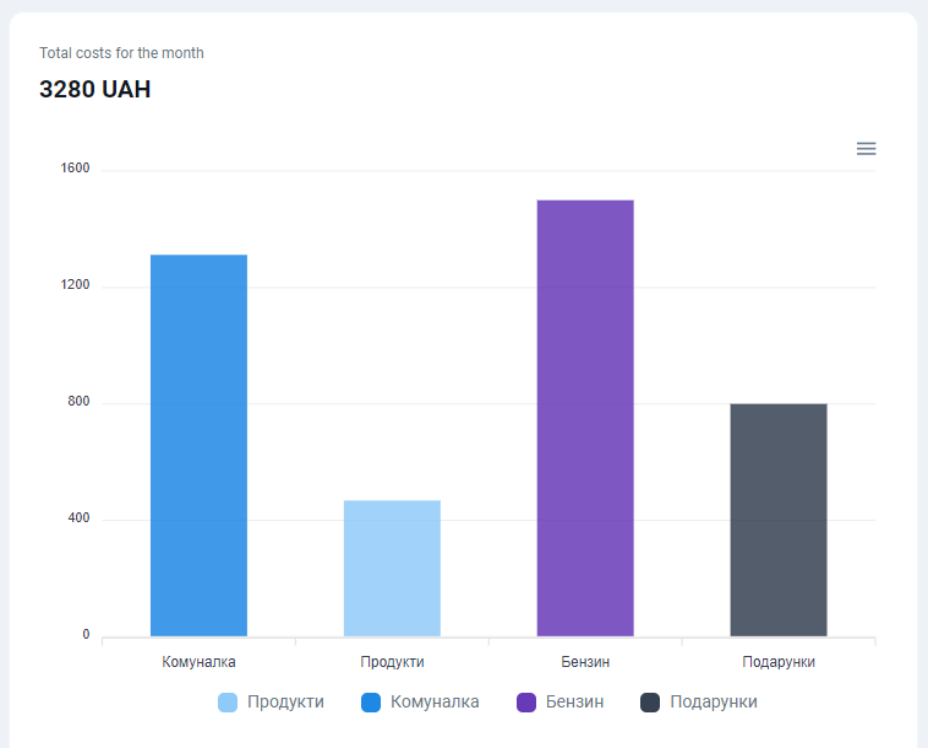

Рис. 2.19. Діаграма витрат

Джерело: [створено автором]

Загальний вигляд розміщення компонентів у дашборді (див. рис. 2.20)

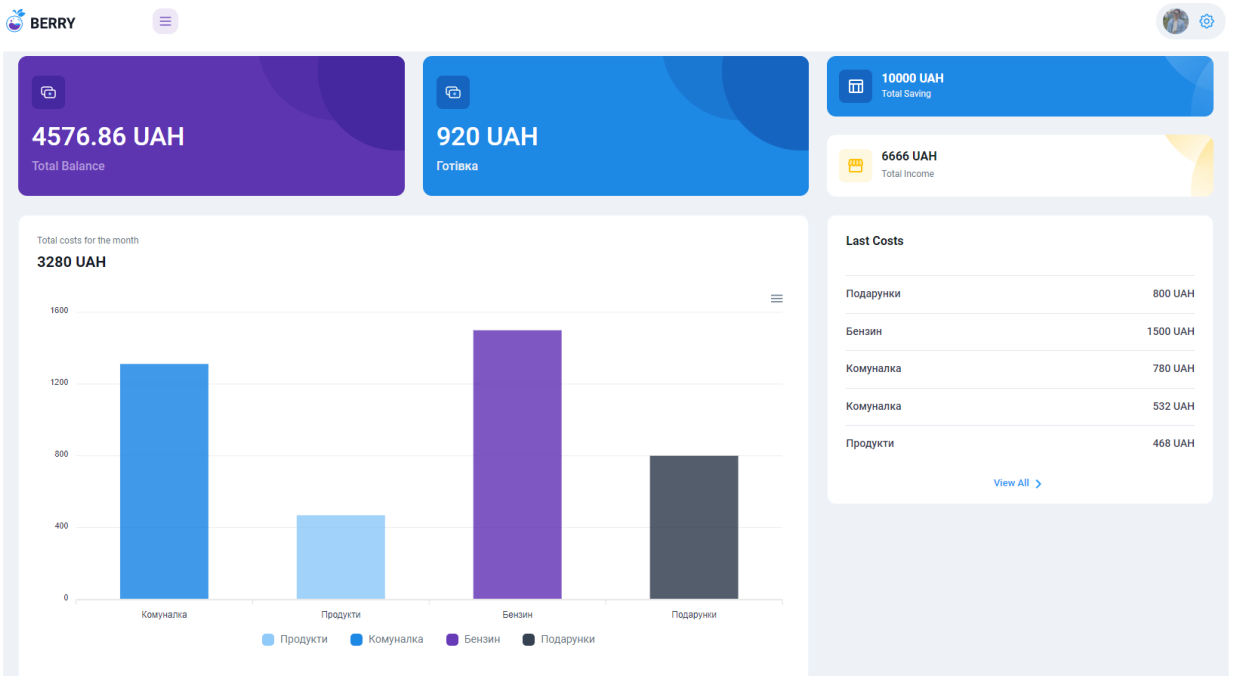

Рис. 2.20. Сторінка Dashboard

Джерело: [створено автором]

2.2. Перспективи розвитку та впровадження додатку на ринок веб-додатків

При розробці додатку для менеджменту витрат та надходжень варто звернути увагу на дизайн інтерфейсу, оскільки він має бути привабливим, зрозумілим та легким у використанні для користувачів. Потрібно ретельно продумати структуру додатку та розміщення функцій, щоб користувачам було зручно навігуватися та виконувати необхідні дії.

Щодо інтеграції з іншими фінансовими додатками та сервісами, це може стати великою перевагою для користувачів. Наприклад, можна розглянути можливість інтеграції з банківськими системами, електронними платіжними системами або іншими додатками для ведення фінансового обліку. Це дозволить користувачам автоматично отримувати дані про свої транзакції та оновлення балансу, спрощуючи процес введення інформації вручну.

Маркетингове дослідження перед впровадженням додатку на ринок веб-додатків є ключовим етапом. Воно дозволить визначити цільову аудиторію, її потреби та вимоги. Це допоможе адаптувати функціонал додатку під потреби користувачів та розробити ефективну стратегію продажів та просування. Важливо також знайти своє місце на ринку та виділитися серед конкурентів, наприклад, за допомогою унікальних функцій або привабливого дизайну.

Після випуску додатку на ринок важливо забезпечити його підтримку та обслуговування. Це включає випуск регулярних оновлень з виправленнями помилок та вдосконаленнями, а також надання користувачам можливості звертатися до служби підтримки з питань та проблем. Позитивний досвід користувача та задоволення його потреб є важливим аспектом успішності додатку.

Розробка додатку для менеджменту витрат та надходжень є перспективним напрямом, оскільки все більше людей розуміють важливість ефективного фінансового планування. Правильно розроблений додаток, який пропонує зручний інтерфейс, розширені функції та інтеграцію з іншими сервісами, може знайти свою аудиторію та стати успішним бізнесом. Забезпечення автономності додатку та синхронізації даних в режимі онлайн дозволить користувачам завжди мати доступ до своїх фінансових даних, незалежно від мережевого з'єднання. Підключення банківських API дозволить автоматично отримувати дані про транзакції та зміни балансів, забезпечуючи актуальну інформацію користувачам.

ВИСНОВКИ

Під час виконання даної роботи були розроблені сервісна і клієнтська частини вебдодатку «Інформаційної системи для менеджменту витрат та надходжень». Реалізовані всі необхідні сервіси для функціонування вебпрограми.

У ході роботи було проведено аналіз теми фінансової грамотності та важливості моніторингу витрат, що дало змогу обрати сферу, для якої буде відповідати наш додаток - моніторинг витрат та надходжень. Також був проведений аналіз ринку, виявлені основні конкуренти та здійснений SWOT-аналіз на основі цих даних.

Для оптимального архітектурного рішення було обрано Чисту архітектуру. Серверну частину реалізовано на Python з використанням таких технологій як FastAPI, SQLAlchemy, PostgreSQL, Pydantic, Passlib, PyJWT. Використання мови програмування Python та веб-фреймворка FastAPI сприяє швидкій розробці та забезпечує високу продуктивність системи. Бібліотеки SQLAlchemy та PostgreSQL дозволяють безпечно та ефективно працювати з базою даних, забезпечуючи високий рівень надійності та безпеки даних. Pydantic, Passlib та PyJWT виступають надійними інструментами для забезпечення безпеки системи та даних.

Клієнтську частину реалізовано на React з використанням таких технологій як React, Redux, Material UI та інші. Використання ReactJS, HTML, CSS та JS в Frontend розробці з використанням Material UI дозволяє створювати не тільки функціональний, але й естетичний інтерфейс для користувачів, що стає додатковим фактором успіху продукту.

Проєкт також розміщено на GitHub, що надає зручний інтерфейс для зберігання та керування версіями проекту, сприяючи зручній та безпечній розробці програмного продукту. Загальне поєднання цих технологій дозволяє створити потужний та надійний серверний компонент вебдодатку, що забезпечує швидку

розробку, ефективну роботу з базою даних, безпеку системи та надійність обробки даних.

Взаємодія вищезазначених технологій та інструментів дозволяє створити комплексну та продуктивну інформаційну систему для менеджменту витрат та надходжень з високим рівнем безпеки та зручним інтерфейсом користувача. Такий стек технологій може бути корисним та дієвим не тільки для цільової розробки інформаційної системи, але й для інших веб-продуктів, які мають високі вимоги до ефективності та безпеки.

Все це сформулювало міцний фундамент для нашого додатку. До повного завершення роботи ще потрібно зробити:

- розширити функціонал вебдодатку;
- розмістити проєк на хостингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація для CSS URL: https://www.w3schools.com/css/ (дата звернення: 12.04.2023 р)..
2. Документація для HTML URL: https://www.w3schools.com/html/ (дата звернення: 12.04.2023 р)..
3. Документація для JavaScript URL: https://www.w3schools.com/js/ (дата звернення: 12.04.2023 р)..
4. GitHub URL: https://github.com/ (дата звернення: 12.04.2023 р).
5. Документація для FastAPI URL: https://fastapi.tiangolo.com/ (дата звернення: 15.01.2023 р).
6. Документація для Python URL: https://docs.python.org/3/ (дата звернення: 15.01.2023 р).
7. Документація для SQLAlchemy URL: https://docs.sqlalchemy.org/en/14/ (дата звернення: 15.01.2023 р).
8. Документація для Pydantic URL: https://docs.pydantic.dev/ (дата звернення: 15.01.2023 р).
9. Документація для Passlib URL: https://passlib.readthedocs.io/en/stable/ (дата звернення: 15.01.2023 р).
10. Документація для PyJWT URL: https://pyjwt.readthedocs.io/en/stable/ (дата звернення: 15.01.2023 р).
11. Документація для React URL: https://uk.legacy.reactjs.org/tutorial/tutorial.html (дата звернення: 12.04.2023 р).
12. Документація для Material UI URL: https://mui.com/material-ui/getting-started/overview/ (дата звернення: 12.04.2023 р).
13. Чиста архітектура: Мистецтво розроблення програмного забезпечення / пер. з англ. І. Бондар-Терещенко. – Харків, 2019. 368 с.