КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

_____

(факультет)

Інформаційних технологій

_____

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Автоматизована інформаційна система обліку та управління особистими фінансами»

Кулик Антон Олегович

(прізвище, ім’я та по батькові студента повністю)

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

(факультет)

Інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри Інформаційних технологій
Гончаренко Т.А.
_____
„___” _____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Автоматизована інформаційна система обліку та управління особистими фінансами»

Виконав: студент 4–го курсу, групи КНс–21
Спеціальності: 122 «Комп`ютерні науки»
Спеціалізація: _«Інформаційні управляючі системи і технології»
(шифр і назва напряму підготовки, спеціальності)
_____Кулик А.О.____.
(прізвище та ініціали)
Керівник _____Гончаренко Т.А.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Київ 2024 р

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: __автоматизації і інформаційних технологій
Кафедра:__ __інформаційних технологій
Освітній рівень: «бакалавр» за ОП
Спеціальність:_122 «Комп`ютерні науки»
Спеціалізація:_.Інформаційні управляючі системи і технології.

ЗАТВЕРДЖУЮ

Завідувач кафедри Інформаційних технологій
Гончаренко Т.А.

_____
„_19_” ___січня___2024 року

ЗАВДАННЯ
ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

_____Кулик Антон Олегович_____

Тема роботи: Автоматизована інформаційна система обліку та управління особистими фінансами
затверджена наказом ректора КНУБА № ____ від «17» листопада 2024 р.
2. Керівник роботи: Гончаренко Тетяна Андріївна
кафедри інформаційних технологій проектування і прикладної математики
3. Строк подання студентом роботи до захисту:_____
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз та дослідження проблеми
Р.2. Проєктуванняінформаційного забезпечення
Р.3. Практична реалізація
Р.4. Бізнес-план

5. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз та дослідження проблем	Лютий 2024 р.
Р. 2. Проєктування інформаційного забезпечення забезпечення	Квітень 2024 р.
Р. 3. Практична реалізація	Квітень 2024 р.
Р. 4. Бізнес-план	Травень 2024 р.
Остаточне оформлення роботи	Травень 2024 р.
Направлення роботи на рецензування	Червень 2024 р.
Попередній захист роботи на кафедрі	Червень 2024 р.

6. Дата видачі завдання: 19 січня 2024 р.

Керівник		Гончаренко Т.А.
	(підпис)	(прізвище та ініціали)
Бакалавр		Кулик А.О.
	(підпис)	(прізвище та ініціали)

АНОТАЦІЯ

Кулик А.О. Автоматизована інформаційна система обліку та управління особистими фінансами.

Дипломна бакалаврська робота за спеціальністю – «Комп'ютерні науки» – Київський національний університет будівництва та архітектури, Київ, 2024 рік.

Бакалаврську роботу присвячено дослідженню та використанню основних тенденцій, принципів та реалізацій інформаційних систем. В контексті дипломного проекту було розроблено систему обліку та управління особистими фінансами, та база даних для цієї системи.

У роботі досліджено і використано основні принципи та технології розробки автоматизованих інформаційних систем обліку та управління особистими фінансами, розроблено дизайн програмного продукту, розроблено базу даних для зберігання інформації та програмний код для обробки і візуалізації збереженої інформації.

Ключовими словами є: "інформаційне забезпечення", "особисті фінанси", "база даних", "управління фінансами", "функціональні можливості".

SUMMARY

Kulyk A.O. Automated information system for personal finance accounting and management.

Bachelor's thesis in Computer Science – Kyiv National University of Construction and Architecture, Kyiv, 2024.

The bachelor's thesis is dedicated to the exploration and utilization of fundamental trends, principles, and implementations of information systems. Within the scope of the diploma project, a personal finance tracking and management system, along with a corresponding database, has been developed.

The thesis explores and applies the fundamental principles and technologies for developing automated information systems for accounting and managing personal finances, designs the software product, develops a database for storing information, and creates software code for processing and visualizing the stored information.

Key words: "information support", "personal finances", "database", "finance management", "functional capabilities".

ЗМІСТ

ВСТУП

Актуальність дослідження. Тема "Автоматизована інформаційна система обліку та управління особистими фінансами" є дуже актуальною в сучасному світі, де технології стають все більш інтегрованими в повсякденне життя людей.

Облік особистих фінансів необхідний для контролю особистих витрат, прибутків, змін у бюджеті. Ці інструменти допомагають планувати бюджет, виявляти зайві витрати, що в результаті покаращує економічне становище людини. Розробка автоматизованої інформаційної системи для обліку та управління особистими фінансами є рішенням для допомоги в управлінні особистим бюджетом.

Отже, автоматизована інформаційна система для обліку та управління особистими фінансами має великий потенціал для покращення якості життя людей та забезпечення ефективного використання особистих фінансів. Така тема дипломного проєкту є дуже актуальною та має потенціал для розвитку в майбутньому.

Для визначення напрямку даної роботи, необхідно спочатку визначити об’єкт дослідження і предмет дослідження, а також надати опис, за допомогою яких методів планується проводити дослідження і вирішення проблеми.

Отже, для дослідження і побудови імітаційної моделі підсистеми виділяємо наступні сутності:

- об’єктом дослідження даної роботи є інформаційне забезпечення автоматизованої системи обілку та управління особистими фінансами.
- предмет дослідження роботи є розробка інформаційного забезпечення автоматизованої системи обліку та управління особистими фінансами. В основі інформаційного забезпечення лежить модель обліку та управління особистими фінансами.

- методи дослідження – на початку необхідно провести дослідження існуючих систем для обліку та управління особистими фінансами, які є розповсюдженним у використанні. Провести аналіз найкращих практик для побудови зручного, інформаційного та інтуїтивного інтерфейсу користувача. Виділити основі критерії функціональності інформаційної системи для забезпечення потреб у обліку та управлінні особистими фінансами.

Практична значимість. Результати отримані під час дослідження можуть бути використані для реалізації програмного продукту для обліку і управлінням особистих фінансів із використанням найкращих практик і тенденцій.

Результати дослідження. Розробка та реалізація автоматизовної інформаційної системи для управління і облку особистими фінансами, що дозволяє легко вести облік особистих фінансів і отримувати звітну інформацію у зручному, зрозумілому графічному вигляді. Результатом є також розробка оптимальної структури баз даних, що забезпечують ефективну обробку, зберігання та взаємодію різних даних в системі.

1 АНАЛІЗ ТА ДОСЛІДЖЕННЯ ПРОБЛЕМ

1.1 Загальні та теоретичні відомості, концепції автоматизованої системи для обліку та управління особистими фінансами

Зв'язок інформаційних технологій зі змінами в суспільстві та побуті людей необхідно розглядати у контексті тенденцій, що склалися в останні роки. Питання особистих фінансів не є новим. Людина, що вміє вести облік особистих фінансів зазвичай вокористовувала традиційні інструменти для обліку різного роду інформації: ручка і папір. На ранніх етапах розвитику інформаційних технологій для виконання обліку певного типу даних була необхідність у наявності кваліфікованого оператора, тобто людини, що безпосередньо виконує облік за допомогою інформаційної системи. З часом інформаційні системи конкретно для обліку фінансів перейшли у, як мінімум, два шляхи - одні для корпорацій, де все ще є необхідність кваліфіковного робітника, бухгалтера, необхідного для виконання облікових операцій і другий шлях - системи для обліку особистих фінансів, що розраховані на звичайного користувача. Інформаційні системи для обліку особистих фінансів повинні мати, перш за все, інтуїтивний інтерфейс, що є простим і зрозумілим для пересічного користувача. Сам облік фінансів теж не повинен створювати складнощів для його виконання. Звіти по фінансовим рухам мають бути зображені у графічному вигляді та надавати мінімальний, але повний об’єм інформації.

Автоматизована інформаційна система для обліку та управління особистими фінансами – це технологія, яка надає можливість автоматизувати та контролювати рух особистих фінансів у максимально спрощеному вигляді.

Однак, під час використання інформаційних систем для обліку та управління особистими фінансами гостро стає питання безпеки даних, що надає користувач. Особисті дані користувача, фінансові операції, що зберігаються

інформаційною системою, можуть бути використані зловмисниками для різних цілей. Ці аспекти програмного продукту повинні бути розглянуті детально.

Тому, метою даної дипломної роботи є детальний аналіз вимог щодо системи для управління та обліку особистих фінансів для звичайного користувача та розробка відповідної інформаційної системи.

Для досягнення поставленої мети, в роботі є намір розглянути основні аспекти обліку та управління особистими фінансами, архітектурні особливості, системи управління, типи та реалізації баз даних, комунікації та безпеки. Також проаналізовані вже існуючі інформаційні системи для обліку та управління особистими фінансами, що призначені як для звичайного користувача, так і для корпорацій, компаній.

Загальна мета дипломної роботи – дослідити методики та тенденції для реалізації системи для обліку та управління особистими фінансами, а саме її інформаційного забезпечення, проаналізувати її особливості та можливості.

1.2 Постановка та аналіз проблеми.

Облік особистих фінансів необхідний для контролю особистих витрат, прибутків, змін у бюджеті. Ці інструменти допомагають планувати бюджет, виявляти зайві витрати, що в результаті покаращує економічне становище людини. Розробка автоматизованої інформаційної системи для обліку та управління особистими фінансами є рішенням для допомоги в управлінні особистим бюджетом.

Поетапний процес розробки інформаційного забезпечення автоматизованої системи обліку та управління особистими фінансами можливо представити у вигляді життєвого циклу, серед яких виділяють наступні основні етапи:

1. етап планування;
2. етап збору вимог;
3. етап проєктування;
4. етап розробки програмного забезпечення;
5. етап тестування програмного забезпечення;
6. випуск готового продукту на ринок;
7. етап експлуатації та технічного обслуговування.

На першому етапі узгоджуються всі деталі майбутньої автоматизованої інформаційної системи для обліку та управління фінансами, відповідаючи на важливі питання. Яка мета розробки? Які проблеми воно має вирішувати? Навіщо створюється система? Після відповіді на ці питання формується єдине бачення майбутньої системи. Що полегшує розробку та мінімізує ризики при створенні ПЗ.

Етап аналізу та збору вимог є одним із найважливіших етапів життєвого циклу розробки автоматизованої інформаційної системи для обліку та управління особистими фінансами. На цьому етапі збираються всі необхідні дані для успішної розробки та впровадження системи.

Після того, як стали зрозумілими цілі та завдання системи, а також технології, які будуть використовуватися для її розробки, можна переходити до проектування та дизайну архітектури. На цьому етапі проєктується майбутня архітектуру проекту у вибраній технології. Оформлення UML-діаграм системи, окремих модулів, опис принципу взаємодії компонентів системи. Дизайн сутностей бізнес-моделі, визначення зв'язків між окремими сутностями. Проєктування та дизайн адаптивного графічного інтерфейсу користувача, визначення вимог безпеки для системи.

Стадія розробки - це етап, на якому відбувається створення програмного продукту. На цьому етапі береться проєктна документація, прототипи, дизайн та архітектура, і на основі їх створюють програмний код, який реалізує всі необхідні функціональні можливості системи. Розробка ділиться на дві частини: розробка інтерфейсу користувача програми та розробка серверної частини програми. На етапі розробки серверної частини перш за все розробляється бізнес-частина системи, що відповідає за функціонування основних потреб системи, зазначених у вимогах. На етапі розробки інтерфейсу користувача реалізовується веб-інтерфейс користувача для взаємодії із системою. Адміністратори бази даних відповідають за створення та наповнення бази даних системи. Результатом етапу розробки є готовий програмний продукт, який відповідає вимогам, визначеним на етапі аналізу та збору вимог.

На етапі тестування команда перевіряє, чи відповідає розроблена система вимогам, визначеним на етапі аналізу та збору вимог. Тестування включає в себе такі основні завдання:

- Перевірка функціональності системи;
- Перевірка ефективності системи;
- Перевірка безпеки системи.

Після того, як система протестована і готова до використання, її запускають в експлуатацію. Система стає доступною для користувачів, які можуть її використовувати для обліку та управління особистими фінансами.

Замвоник збирає відгуки від користувачів, щоб зрозуміти, чи відповідає система їхнім потребам. Якщо в результаті збору відгуків виявляються помилки в системі, їх виправляють розробники.

Технічне обслуговування - це етап, на якому система підтримується в робочому стані і забезпечується її ефективне використання. На цьому етапі система постійно перевіряється на наявність помилок і проблем. Також на цьому етапі можуть вноситися зміни в систему для її поліпшення. Ці зміни можуть бути спрямовані на:

- усунення виявлених проблем;
- розширення функціональності системи;
- вдосконалення інтерфейсу користувача;
- підвищення продуктивності системи;
- покращення безпеки системи.

1.3 Дерево основних цілей

Основною метою є розробка інформаційного забезпечення автоматизованої системи обліку та управління особистими фінансами. Дерево цілей опису даної мети наведено на рисунку 1.1.

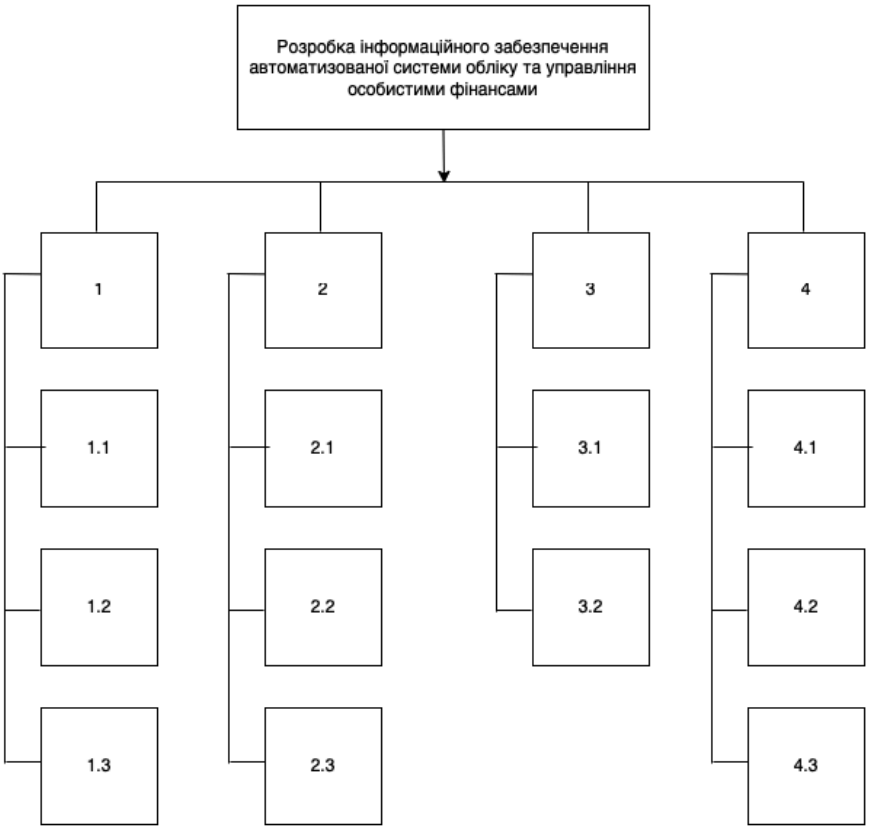

Рисунок 1.1 – Дерево цілей

1. Функціональні вимоги:

1.1 . Зберігання інформації про користувачів: електронна пошта, дані для авторизації, ім’я, прізвище, номер телефону.

1.2 . Зберігання інформації про фінансові операції користувача.

1.3 .Аналіз бюджету для генерації динамічних графіків із застосуванням певних параметрів для інформаційного відображення стану особистих фінансів.

1.4 Функціональна можливість розподілення фінансових операцій за користувальницькими категоріями.

2. Надійність та безпека

2.1. Розробка системи автентифікації та авторизації користувачів.

2.2. Захист даних користувачів та надану ними інформацію.

2.3 Виявлення та виправлення можливих вразливостей системи безпеки та захисту даних.

3. Масштабованість та розширюваність

3.1 Розширення функціональності бази даних.

3.2 Розширення кількості даних.

4. Підтримка та обслуговування

4.1 Забезпечення підтримки та обслуговування бази даних протягом всього її життєвого циклу

4.2 Забезпечення можливості оновлення програмного забезпечення бази даних

4.3 Вирішення помилок у разі їх виявлення.

1.4 **Вимоги та особливості проєктування системи**

Даний проєкт має на меті розробку інформаційного забезпечення для такої системи, яка має включати в себе функціонал управління особистими фінансами, відображення бюджетних змін у графічному вигляді для проведення ефективного аналізу бюджету. Забезпечення відображення інформації за певними критеріями, такими як дата, категорія і т.д. Основною метою системи є надання даних у вигляді, що допомогає краще зрозуміти загальний стан бюджету.

Основним елементом будь-якої автоматизованої системи є база даних, яка забезпечує збереження, обробку та доступ до інформації. Для цієї системи ставиться ряд важливих завдань щодо інформаційного забезпечення:

- Збереження текстових та числових даних;
- Представлення даних у зручному та структурованому вигляді;
- Швидкий доступ до даних;
- Можливість внесення корегувань та доповнень;
- Забезпечення цілісності та конфіденційності даних.

Для вирішення цих завдань обрана реляційна база даних. Відмінність між реляційними і нереляційними базами даних полягає в підході до зберігання інформації. Реляційна база даних зберігає інформацію у вигляді таблиць. Кожна таблиця має чітко описану структуру і представляє собою окрему сутність. Реляційна база даних надає можливість створення зв'язків між окремими таблицями на рівні самої бази даних, що дає можливість не концентрувати увагу на реалізації зв'язків між сутностями на рівні бізнес-моделі.

Основним фактором вибору саме реляційної бази данних є те, що реляційні бази даних зарекомендували себе часом і підходять для реалізації сховища структурованих даних. Використання реляційної бази даних в обліку та управління особистими фінансами дозволить забезпечити ефективне та оптимізоване зберігання та обробку даних, що є важливим для успішної роботи системи.

Щодо забезпечення конфіденційності даних, впровадження системи автентифікації та авторизації є ключовим елементом. Вона гарантує, що доступ до конфіденційної інформації буде здійснюватись лише авторизованими користувачами.

Такий комплекс вимог та особливостей проєктування системи інформаційного забезпечення спрямований на ефективної системи для обліку та управління особистими фінансами. Застосування реляційної бази даних покриває всі описані в архітектурі вимоги для забезпечення зберігання інформації.

1.5 Аналіз існуючих рішень

1.5.1 Аналіз існуючих розробок автоматизованих інформаційних систем для обліку та управління особистими фінансами

У сучасному світі автоматизовані інформаційні системи для обліку та управління фінансами займають певну нишу серед додатків, що спрямовані на покращення життя у повсякденні. Такі системи дозволяють швидко і зручно вести облік особистих фінансів і отримувати зрозумілий та чіткий вигляд стану бюджету. У даному аналізі розглянуто найбільш відомі системи обліку та управління особистими фінансами.

1. Система «Monefy»

Система «Monefy» - це інформаційна система, що дозволяє зберігати свої фінансові операції за певними категоріями для подальшого відображення даних у графічному вигляді. Система дає можливість аналізувати бюджет за певний період часу або за певними категоріями. Система представлена у вигляді мобільного додатку та розрахована на середньостатистичного користувача.

2. Система «tidely»

Система «tidely» представляє собою більш просунутий інструмент для обліку та управління особистими фінансами. Система надає більш детальний графічний вигляд загального стану бюджету.

3. Система «Precoro»

Система «Precoro» розроблена для професійного використання. Система надає можливість розподіляти фінансові операції за певними робочими відділами, враховувати майбутні платежі, прив’язувати банківські рахунки для синхронізації фінансів. Також система надає зовнішній програмний інтерфейс, що дає можливість іншим розробникам використовувати певний спектр готових рішень для обліку та управління особистими фінансами.

Таблиця 1.1 – опис існуючих розробок автоматизованих інформаційних систем обліку та управління особистими фінансами.

Назва	Країна	Аудиторія	Особливості
Monefy	США	Звичайний користувач	Мобільний додаток, просте управління особистими фінансами, графічний вигляд бюджету.
tidely	Німеччина	Продвинутий користувач	Більш продвинутий інструмент для управління особистими фінансами. Детальний графічний вигляд бюджету
Precoro	США, Німеччина, Польща, Литва	Для професійного використання	Інструмент для обліку фінансів, що використовується підприємствами

Порівнюючи різноманітні системи обліку та управління особистими фінансами можна визначити декілька ключових аспектів, що варто враховувати при розробці власної автоматизованої системи обліку та управління особистими фінансами. Майбутня система повинна в повній мірі задовольнити потреби користувача у зручному та швидкому обліку особистих фінансів. Графічний інтерфейс повинен бути інтуїтивно зрозумілим та нести в собі максимально корисну інформацію для ефективного здійснення аналізу бюджетного стану. Система повинна мати можливість створювати власні категорії фінансових операцій та здійснювати пошук і відображати запитані дані у зрозумілому форматі. Для обіймання більш широкої аудиторії, система повинна бути

крос-платформенною, для цього було прийнято рішення розроблювати систему у вигляді веб-додатку.

1.5.2 Аналіз існуючих баз даних

В даному розділі розглянуті деякі з найбільш поширених та ефективних рішень баз даних, доступних на ринку. Бази даних є невід'ємною частиною багатьох програмних продуктів та інформаційних систем, і різні системи баз даних надають користувачам можливість зберігати, організовувати та оптимізовувати дані. Ось огляд декількох з них:

Oracle Database: Це потужна та розширювана система управління базами даних, яка підтримує широкий спектр додатків, включаючи бізнес-аналітику та високошвидкісні транзакційні системи. Вона володіє можливостями масштабування та високої доступності.

MySQL: Це одна з найпопулярніших відкритих реляційних систем управління базами даних, яка підтримує широкий спектр додатків, включаючи веб-додатки та електронну комерцію. MySQL відзначається масштабованістю та високою доступністю.

MongoDB: Це нереляційна система управління базами даних, яка дозволяє зберігати та обробляти нереляційні дані. MongoDB підтримує масштабування та високу доступність з використанням вбудованого механізму шарування.

PostgreSQL: Це відкрита реляційна система управління базами даних з розширюваним архітектурним підходом. Вона підтримує масштабування, високу доступність та надає різноманітні інструменти для розробки та управління базами даних.

Таблиця 1.2 – опис рішень баз даних для розробки автоматизованої інформаційної системи обліку та управління особистими фінансами.

Назва бази даних	Тип бази даних	Мова запитів	Підтримувані операційні системи	Особливості
Oracle Database	Relational	SQL	Windows, Linux, macOS	Надійна платформа, підтримка великих обсягів даних, можливість використовувати більшість функцій SQL,
MySQL	Relational	SQL	Windows, Linux, macOS	Відкритий код, підтримка транзакцій, гарна підтримка індексів
MongoDB	NoSQL	MongoDB	Windows, Linux, macOS	Гнучкість у роботі зі структурованими та неструктурованими даними, горизонтальне масштабування
PostgreSQL	Relational	SQL	Windows, Linux, macOS	Відкритий код, підтримка транзакцій, можливість використовувати різні типи даних, розширені можливості безпеки

За результатами порівняльної характеристики баз даних, що можуть використовуватися в розробці автоматизованої системи обліку та управління особиситими фінансами, сформований висновок про те, що кожна база даних має свої переваги та недоліки і обирається в залежності від конкретних потреб проекту. Наприклад, якщо потрібна висока продуктивність та підтримка великої кількості користувачів, то більш доцільно використовувати PostgreSQL або MongoDB. У свою чергу, якщо важливо максимально спростити розробку та налаштування бази даних, то можна обрати SQLite. Крім того, вибір бази даних залежить від характеру даних, що зберігаються, та вимог до безпеки та зберігання історії змін. Отже, при розробці автоматизованої системи управління та обліку особистих фінансів важливо враховувати всі ці фактори і обирати базу даних, яка краще підходить для конкретного проекту.

1.6 **Постановка задачі**

Метою дипломного проекту є створення інформаційної системи та бази даних для автоматизованої системи управління та обліку особистих фінансів, яка дозволить користувачеві легко і швидко зберігати дані про власні фінансові операції. На основі цих даних система будує інформативні графіки з урахуванням критерій. Реалізація проекту насамперед передбачає розробку бізнес-моделі системи. Проектування сутностей для зв’язків між ними. Розробка сервісного шару, який надає методи для роботи із системою. Також розробка проекту передбачає проектування схеми бази даних для зберігання інформації про дані, які використовуються для управління і обліку особистих фінансів. База даних повинна містити дані про користувачів, їх фінансові операції, користувальницькі категорії, допоміжні структури для зберігання статистичних даних, що використовуються при побудові графіків. Очікуваними результатами є розроблена автоматизована система для управління та обліку особистими фінансами.

Основні завдання дипломного проекту:

1. Аналіз вимог користувачів у галузі управління особистими фінансами для визначення потреб і функціональності системи, визначення типів даних, які потрібно зберігати в базі даних;
2. Вибір технології, вивчення особливостей та можливостей обраної технології;
3. Проектування архітектури системи, створення бізнес-моделі системи.
4. Проектування схеми бази даних в залежності від моделі, визначення структури таблиць, які будуть зберігатися в базі даних;
5. Створення бази даних;
6. Розробка серверної частини системи з урахуванням архітектурних вимог.
7. Розробка графічного веб-інтерфейсу користувача.
8. Оптимізація, налаштування індексів для підвищення швидкодії пошуку даних;
9. Забезпечення безпеки;
10. Тестування системи;
11. Впровадження системи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Ключові питання постановки завдання проєктування інформаційного забезпечення

Для написання інформаційного забезпечення автоматизованої системи обліку та управління особистими фінансами необхідно розглянути декілька ключових питань, таких як:

1. Визначити тип розроблюваної системи, а саме - веб-додаток. Вибір веб-додатку обґрунтовується тим, що такий додаток може бути запущений, як на смартфоні, планшеті так і на персональному комп’ютері. Система може бути використана на будь-якому пристрої, що підтримує роботу веб-браузера.
2. Вибір технології та мови програмування для розробки автоматизованої інформаційної системи. В це питання входить також вибір фреймворку, що буде використаний для розробки.
3. Забезпечення безпеки: необхідно розглянути питання забезпечення безпеки особистих даних, та даних, що надає користувач у контексті обліку та управління особистими фінансами. Для цього можна використовувати різні технології, такі як шифрування, автентифікація користувача тощо.
4. Розробка архітектури системи в цілому, з урахуванням шару представлення, шару бізнес-логіки та шару бази даних.
5. Розробка інтерфейсу користувача: для забезпечення зручності ведення обліку та управління особистими фінансами необхідно спроектувати та розробити інтерфейс користувача. Інтерфейс користувача повинен забезпечувати простоту та зрозумілість вводу даних та графічне, зрозуміле відображення звітів.
6. Проектування моделі бази даних (концептуальна, фізична та логічні представлення): для правильної реалізації інформаційного забезпечення та баз даних автоматизованої інформаційної системи обліку та управління особистими фінансами, стоїть задача визначення правильної концептуальної, логічної та

фізичної форми представлення бази даних, щоб забезпечити оптимальну продуктивність та безпеку даних. Завдання оперує такими даними як:

- Концептуальна форма представлення бази даних описує відносини між сутностями в системі та їх атрибутами. Це допомагає визначити, які сутності повинні бути включені до бази даних та як вони пов'язані між собою. Концептуальна модель допомагає зрозуміти, які дані необхідні для системи, та допомагає уникнути зайвих даних.
- Логічна форма представлення бази даних описує відносини між таблицями та структуру даних, які використовуються для зберігання даних. Логічна модель дозволяє визначити, які таблиці повинні бути створені та які взаємозв'язки повинні бути встановлені між ними.
- Фізична форма представлення бази даних описує, як дані фактично зберігаються на диску або в інших засобах зберігання. Фізична модель визначає, які типи даних використовуються для зберігання даних, які індекси та ключі повинні бути встановлені та як дані можна розмістити на диску для оптимальної продуктивності.

7. Тестування та налагодження: після розробки програмного забезпечення та баз даних необхідно провести тестування та налагодження системи. Це допоможе виявити та виправити помилки та недоліки системи перед її використанням.

2.2 **Тип програмного забезпечення для інформаційної системи**

Автоматизована інформаційна системи для обліку та управління особистими фінансами передбачає можливість використовувати систему звичайному користувачеві, а отже і платформа, на якій використовуватиметься програмний засіб, повинна бути доступною. Тому було вирішено розробити програмне забезпечення у вигяді веб-додатку, що може бути запущений у будь-якому середовищі, що підтримує веб-браузер.

Веб-додаток передбачає, перш за все, мобільну версію системи. Оскільки багато фінансових операцій, що здійснюються користувачем, можуть бути зроблені у час, коли доступ до персонального комп'ютера обмежений, пріоритетом для веб-додатку є мобільна версія. Система також передбачає сторінку для персонального комп'ютера у випадку, коли користувачеві потрібно зробити облік фінансів, використовуючи комп'ютер.

Для реалізації доступності до даних у будь-якому місці необхідний варіант клієнт-серверної архітектури, коли сервер є віддаленим. Для досягнення найвищого показника доступності і зручності використання, сервер необхідно розмістити на віртуальному сервері, використовуючи хмарні технології. Клієнтом виступає веб-браузер, що є запущений на будь-якій підтримуваній системі.

2.3 Вибір технології та мови програмування для розробки автоматизованої інформаційної системи

Для проектування інформаційного забезпечення автоматизованої інформаційної обліку та управління особистими фінансами необхідною задачею є вибір мови програмування, яка найкраще підходить для вирішення поставлених задач. Тож, вибір мови програмування залежить від різних факторів, таких як складність проекту, доступність ресурсів, вимоги до продуктивності, масштабованості та безпеки.

Даний розділ проектування інформаційного забезпечення для автоматизованої інформаційної системи обліку та управління особистими фінансами описує ресурсну базу наступних завдань:

- Вибір мови програмування для розробки додатків і систем.
- Визначення технологій та фреймворку для розробки інформаційної системи.

- Розробка структури бази даних та вибір системи управління базами даних (СУБД).
- Вибір системи зберігання даних та забезпечення їх безпеки.

Перед вибором мови програмування для реалізації автоматизованої інформаційної системи для обліку та управління особистими фінансами потрібно розглянути доступні варіанти:

Python – це інтерпретована мова програмування високого рівня, яка зазвичай використовується в розробці веб–додатків та наукових обчислень. Однак, Python також може бути використаний для створення програмного забезпечення для автоматизованої інформаційної системи обліку та управління особистими фінансами. Python має читабельний та простий синтаксис, що робить його добрим вибором для розробки прототипів та швидкого прототипування.

JavaScript – це мова програмування, яка часто використовується в розробці веб–додатків, але може бути використана й для розробки додатків для автоматизованої інформаційної системи обліку та управління особистими фінансами. JavaScript дозволяє створювати динамічні та інтерактивні веб–сторінки та додатки. JavaScript має велику кількість бібліотек та фреймворків, що полегшують розробку програмного забезпечення.

Java – це об'єктно–орієнтована мова програмування, що має широке застосування в розробці додатків. Java є платформонезалежною, тому програми, написані на Java, можуть працювати на будь–якій операційній системі. Java підтримує багато фреймворків, бібліотек та інших інструментів, що зроблюють його відмінним вибором для розробки автоматизованої інформаційної системи для обліку та управління особистими фінансами.

Таблиця 3.2 – порівняння мов програмування для використання в практичній реалізації системи

Мова програмування	Переваги	Недоліки
Python	Простий синтаксис, велика кількість бібліотек, хороша продуктивність	Менш ефективний для високопродуктивних додатків
Java	Велика кількість фреймворків, хороша масштабованість, висока продуктивність	Вимагає великої кількості ресурсів для запуску, менш простий синтаксис.
JavaScript	Легка взаємодія з компонентами веб–додатків, широкі можливості для розробки інтерфейсів	Менш ефективний для розробки масштабних додатків

Тож, вибір мови програмування для реалізації інформаційного забезпечення автоматизовної системи обліку та управління особистими фінансами залежить від різних факторів. Оскільки програмний засіб буде реалізовано у вигляді веб-додатку і наявність клієнт-серверної архітектури необхідна - потрібно визначитись із технологіями та мовою програмування, що буде використана для реалізації як клієнтської, так і серверної частини системи. Крім того, важливим фактором є наявність бібліотек та фреймворків для розробки додатків, які можуть значно полегшити процес розробки.

Під час вибору мови програмування для реалізації автоматизованої інформаційної системи для обліку та управління особистими фінансами, враховано вимоги до продуктивності, функціональність та зручності розробки,

щоб забезпечити ефективну та швидку розробку інформаційного забезпечення та баз даних.

Отже, для реалізації клієнтської частини було обрано мову програмування JavaScript для реалізації динамічних змін на веб-сторінках, HTML для реалізації структури та маркування контету на веб-сторінках, CSS для стилізації веб-сторінок, розроблених за допомогою HTML. Для реалізації серверної частини було обрано мову програмування Java із використанням фреймворку Spring. Вибір фреймворку Spring обґрунтований тим, що фреймворк допомагає розробнику сфокусуватись на розробці бізнес-частини програмного засобу, не витрачаючи час на формування певної базової архітектури програми на технічному рівні, середовище Spring надає велику кількість готових рішень для розробки веб-додатків “із коробки”.

2.4 **Забезпечення безпеки**

Автоматизована інформаційна система обліку та управління особистими фінансами буде реалізована за використання клієнт-серверної архітектури, що передбачає налаштування віддаленого ресурсу із публічним доступом. Тому необхідна реалізація розподіленого доступу із використанням облікових записів. Кожен користувач повинен зареєструвати свій обліковий запис, після чого всі збережені фінансові операції будуть прив’язані до зареєстрованого облікового запису. Авторизація у систему вібдувається за використанням електроної пошти та паролю. Пароль не повинен зберігатись в базі даних у звичайному, “сирому” вигляді. Перед зберіганням паролю для користувача, він повине бути зашифрованим. Для шифрування паролів використовується BCrypt, що входить разом із модулем фреймворку Spring - Spring Security.

BCrypt - це алгоритм хешування паролів, призначений для збереження паролів у безпечному форматі. Головна його перевага полягає в тому, що він ускладнює атаки на паролі шляхом повільного обчислення хешу. Це робить BCrypt надійним вибором для збереження паролів у системах, які вимагають

високого рівня безпеки. У документації дипломного проекту можна вказати, як саме використовувати BCrypt для збереження та перевірки паролів користувачів у вашій програмі або веб-додатку.

Отже, для забезпечення безпеки даних користувачів і реалізації розподіленого доступу, реалізовується система облікових записів, коли кожен користувач має доступ до свого облікового запису і всі операції відбуваються у контексті окремого користувача. Авторизація у обліковий запис відбувається за вокористанням електроної пошти та паролю. Пароль зберігається у зашифрованому вигляді, що означає, що навіть під час атаки на базу даних, дані користувачів для вхід лишаться у безпеці, в зашифрованому вигляді.

Оскільки система побудована на клієнт-серверній архітектурі і є веб-додатком, то необхідне забезпечення безпеки на рівні комунікації клієнту із сервером. Оскільки система є веб-додатком, то комунікація клієнту із сервером відбувається із використанням протоколу HTTPS. HTTP (Hypertext Transfer Protocol) і HTTPS (Hypertext Transfer Protocol Secure) — це протоколи, що використовуються для передачі даних через мережу Інтернет.

HTTP — це стандартний протокол для передачі гіпертекстових документів інформації на веб-сайтах. Він використовується для передачі різних видів даних, таких як текст, зображення, відео, аудіо та інші ресурси між веб-серверами і веб-клієнтами (наприклад, веб-браузерами).

HTTPS — це захищена версія протоколу HTTP. Вона використовує шифрування SSL (Secure Sockets Layer) або його спадкоємця TLS (Transport Layer Security), щоб захистити конфіденційні дані, які передаються між веб-сервером і веб-клієнтом. HTTPS забезпечує конфіденційність, цілісність та автентичність даних, що передаються через Інтернет, і робить з’єднання більш безпечним для користувачів. Такий протокол особливо важливий при передачі чутливої інформації, такої як паролі, особисті дані, банківська інформація тощо.

2.5 **Архітектура інформаційної системи**

Автоматизована інформаційна система обліку та управління особистими фінансами є веб-додаток, побудований на клієнт-серверній архітектурі. Отже для реалізації клієнтської і серверної частини необхідний певний шаблон, а саме MVC (Model - View - Controller).

MVC, або Model-View-Controller, - це шаблон проектування для розробки програмного забезпечення, особливо популярний у веб-розробці. Він розділяє компоненти програми на три основні частини:

Модель (Model): Модель представляє дані програми та бізнес-логіку, яка опрацьовує ці дані. Модель не залежить від інших частин шаблону і може взаємодіяти як з контролером, так і з представленням.

Вид (View): Вид відповідає за відображення даних користувачу та інтерфейс користувача. Він отримує дані з моделі та відображає їх у вигляді, зрозумілому користувачу. Вид також може надсилати користувачеві вхідні дані до контролера для обробки.

Контролер (Controller): Контролер взаємодіє з користувачем та відповідає за обробку вхідних даних. Він отримує вхідні дані від користувача через представлення, виконує необхідну логіку та змінює стан моделі. Крім того, контролер також відправляє дані у відображення для подальшого відображення користувачеві.

Основна перевага MVC полягає у тому, що він дозволяє розділити логіку програми на окремі компоненти, що спрощує розробку, тестування та підтримку коду. Крім того, цей підхід дозволяє змінювати один компонент (наприклад, вигляд) без впливу на інші компоненти (модель або контролер), що полегшує розширення та зміни програми.

Отже система є веб-додатком, побудованим за шаблоном MVC і цей же додаток грає роль серверу. Клієнтом виступає веб-браузер. Архітектура системи у загальному вигляді має наступний вигляд:

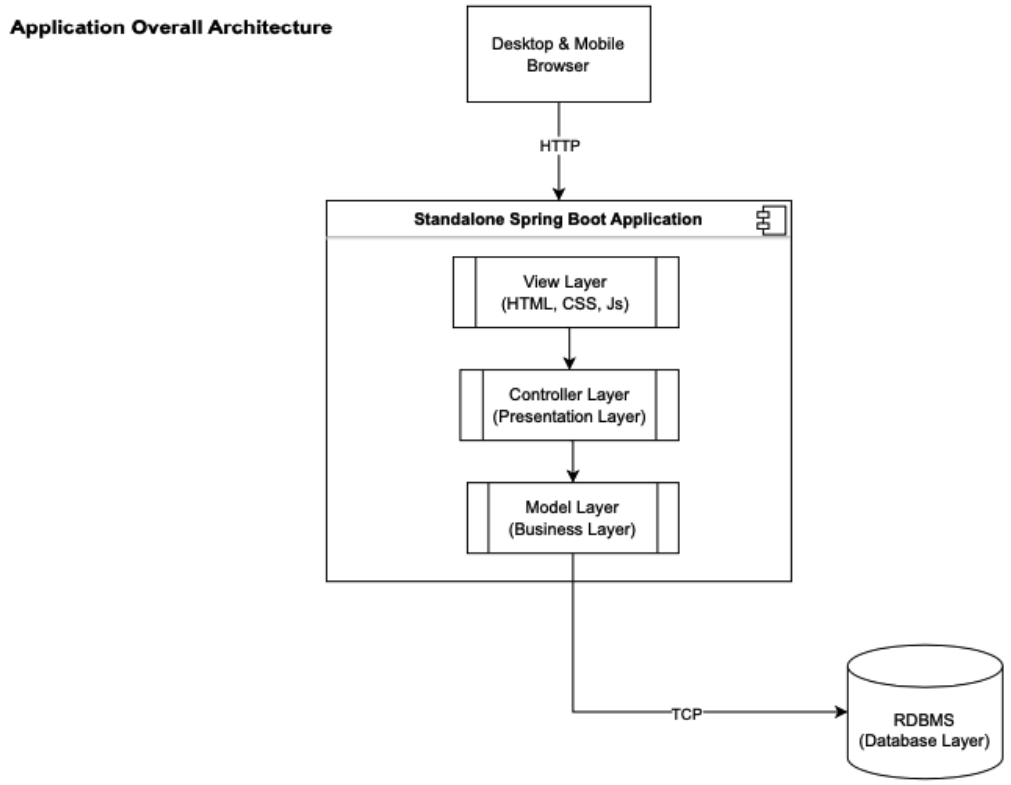

Рисунок 2.1 - Загальна архітектура інформаційної системи

Отже на рисунку 2.1 відображено загальну архітектуру інформаційної системи. Схема починається із клієнту (веб-браузеру), що звертається до серверу за допомогою HTTPS протоколу.

Сервером є самостійний Spring Boot додаток із екосистеми фреймворку Spring. Spring Boot спрощує процес розгортання та конфігурування застосунків, надаючи вбудовані засоби для швидкої розробки. Він використовує конвенції над конфігурацією, що дозволяє розробникам швидше створювати додатки, орієнтовані на бізнес.

Основні переваги Spring Boot включають:

- **Швидке створення застосунків**: Spring Boot дозволяє швидко створювати додатки з мінімальною конфігурацією.
- **Вбудований контейнер**: Він постачається з вбудованими контейнерами сервлетів, такими як Tomcat або Jetty, що спрощує розгортання застосунків.

- **Автоконфігурація**: Spring Boot надає автоматичну конфігурацію на основі залежностей та звичайних конфігураційних стандартів.
- **Компоненти**: Він має багато вбудованих компонентів для різних задач, таких як кешування, безпека, доступ до даних тощо.
- **Підтримка вбудованих баз даних**: Spring Boot має підтримку для вбудованих баз даних, таких як H2, що дозволяє швидко розпочати розробку без необхідності налаштування зовнішнього сервера баз даних.

Узагальнюючи, Spring Boot - це потужний інструмент для розробки Java-додатків, який спрощує процес розгортання та розвитку застосунків, дозволяючи розробникам швидше створювати ефективні програми.

Рухаючись по архітектурі системи далі можна виділити три основних шари веб-додатку: шар вигляду, шар контролеру та шар моделі.

Шар вигляду представляє собою набір HTML-сторінок із використанням CSS для стилізації та JavaScript для реалізації динамічних змін на веб-сторінках. Фактично, шар вигляду - це графічний інтерфейс користувача.

Шар контролеру - це шар представлення. Сфера відповідальності контролеру була описана вище, у детальному розгляді шаблону MVC. На даному рівні архітектури контролер також відповідає за об’єкти, що відображаються на веб-сторінках та об’єкти, що надсилаються клієнтом для оновлення стану моделі. На рівні представлення також реалізовано валідацію даних, що відправляє клієнт та перетворювання бізнес-даних у дані, що використовуються для відображення у графічному вигляді. Отже, шар представлення - це проміжний шар між інтерфейсом користувача та шаром бізнес-логіки. Більш детальна архітектура шару представлення:

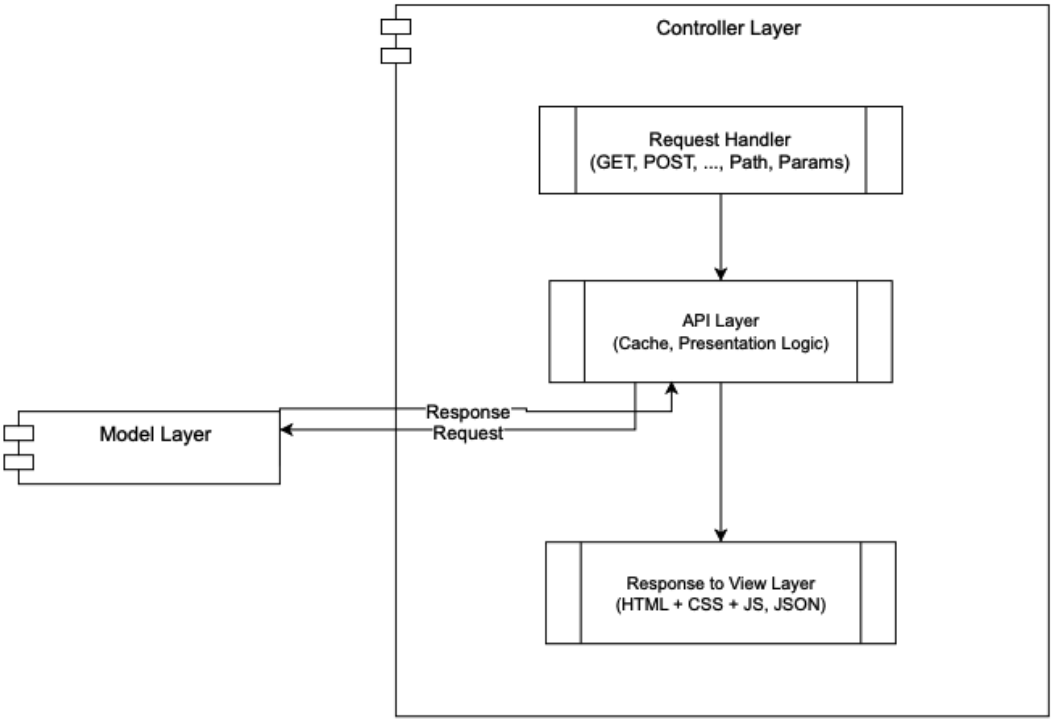

Рисунок 2.2 - Архітектура шару представлення

Шар моделі - це шар бізнес-логіки. Даний рівень побудований на сервісно орієнтованій архітектурі.

Сервісно орієнтована архітектура (SOA) - це підхід до розробки програмного забезпечення, в якому функціональність програми організована у вигляді набору незалежних компонентів, які виконують обмежені функції та мають чітко визначений інтерфейс для взаємодії з іншими компонентами. Кожен компонент (або "сервіс") може бути розглянутий як окрема функціональна одиниця, яка може бути використана іншими компонентами програми через мережу або інші засоби комунікації.

Основна ідея SOA полягає в тому, щоб розбити програму на менші, самодостатні сервіси, які можна легко розвивати, масштабувати та повторно використовувати. Кожен сервіс може бути реалізований, використовуючи будь-яку технологію чи мову програмування, і мати свою власну базу даних та бізнес-логіку. В нашому випадку, сервісно орієнтована архітектура реалізована на логічному рівні.

Переваги SOA включають підвищену гнучкість, орієнтованість на бізнес-потреби, зменшення залежності між компонентами, можливість повторного використання, а також легшу інтеграцію з існуючими системами. Однак для успішної реалізації SOA необхідно правильно спроектувати сервіси та їх інтерфейси, а також забезпечити ефективну систему управління конфігураціями та моніторингу.

Саме на цьому рівні реалізований шар бази даних та безпосередньо зберігання даних та реалізація підключення до системи управління базами даних. Архітектура шару бізнес-логіки:

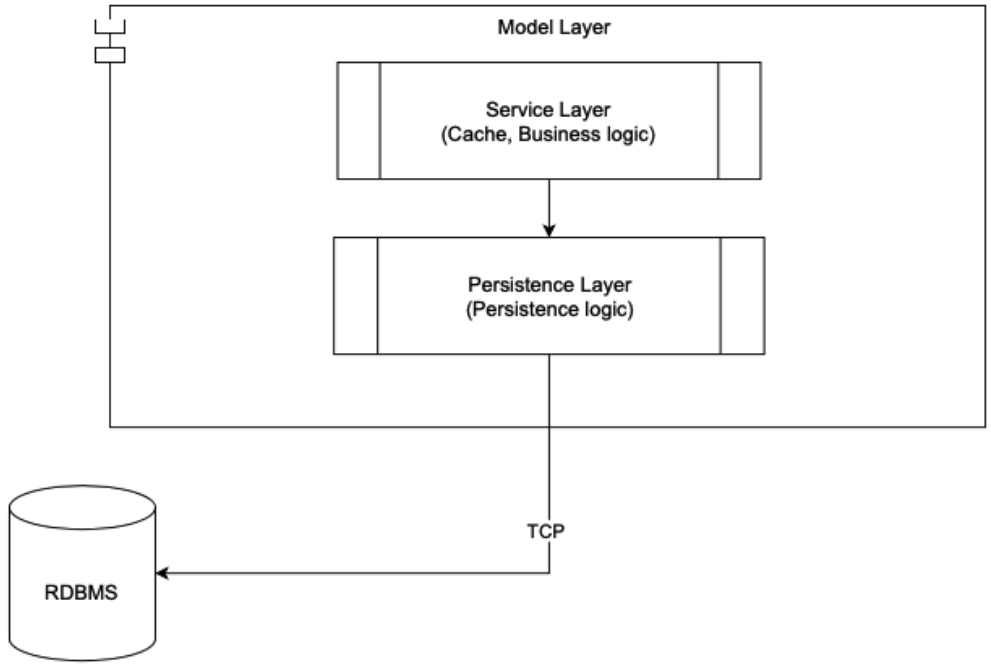

Рисунок 2.3 - Архітектура шару моделі

2.6 Проєктування та розробка графічного інтерфейсу користувача

2.6.1 Вимоги до інтерфейсу

Для розробки інформаційного забезпечення та баз даних системи обліку та управління особистими фінансами необхідний інтерфейс, який би дозволяв легко та зручно взаємодіяти з даними та переглядати звіти у зручному графічному вигляді. Основні вимоги до інтерфейсу створення інформаційного забезпечення мають бути такі:

1. Легкість використання: інтерфейс повинен бути простим та зрозумілим для будь–якого користувача.
2. Функціональність: інтерфейс повинен забезпечувати можливість додавання, видалення та редагування даних про фінансові операції, переглядати зміни у вигляді графіків.
3. Зручність відображення даних: інтерфейс повинен мати зручну та логічну структуру відображення даних, що дозволить користувачеві легко орієнтуватися та знайти потрібну інформацію.
4. Адаптивність: інтерфейс повинен бути адаптивним і мати правильний вигляд на будь-якому пристрої, що підтримує роботу веб-браузеру.
5. Підтримка стандартів: інтерфейс повинен дотримуватися стандартів програмування, що дозволить його легко інтегрувати з іншими системами та розробляти додаткові функції.

Загалом, інтерфейс створення інформаційного забезпечення має бути зрозумілим, зручним та наділяти додаткові функції.

Окрім цього, важливо, щоб інтерфейс був адаптивним. Веб-сторінка, відкрита на будь-якому девайсі, що підтримує веб-браузер, повинна виглядати правильно та бути повною в функціональному об’ємі. На даному етапі було вирішено розробити прототип дизайну інтерфейсу користувача для дизайну саме структури додатку. Для кожного компоненту дизайн буде реалізовано більш детально вже безпосередньо під час його фактичної реалізації.

2.6.2 **Загальні можливості структури інтерфейсу**

Для забезпечення зручного та інтуїтивно зрозумілого інтерфейсу розробки інформаційного забезпечення автоматизованої інформаційної системи обліку та управління особистими фінансами, його структура може включати наступні компоненти:

1. Панель навігації: це важливий елемент інтерфейсу, що дозволяє користувачам легко переключатись між різними розділами програми та виконувати необхідні дії. Панель навігації може містити посилання на такі розділи, як "Домашня сторінка", "Пошук", "Профіль", "Параметри". Панель навігації розміщена знизу і є зафіксованою. Прототип дизайну панелі навігації:

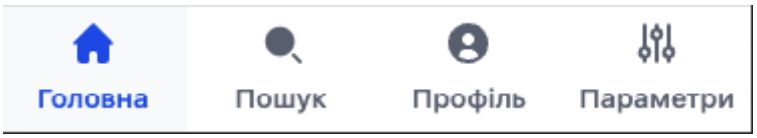

Рисунок 2.4 - Прототип дизайну для компоненту "Панель навігації".

2. Розділ "Домашня сторінка": цей розділ містить загальний вигляд фінансового стану за поточний або обраний місяць у вигляді кругової діаграми. Також користувач може переглянути загальний баланс. На цій сторінці доступно дві дії - операція додавання фінансів і операція віднімання фінансів. Операція додавання фінансів веде у розділ "Додавання фінансів". Прототип дизайну домашньої сторінки:

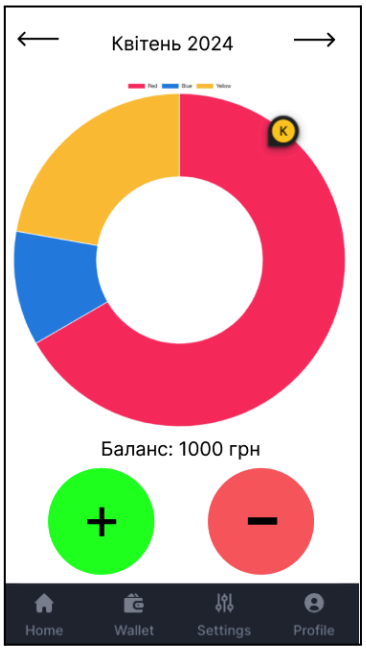

Рисунок 2.5 - Прототип дизайну для компоненту "Домашня сторінка"

3. Розділ "Додавання фінансів" містить форму для створення інформації про фінансові надходження, а саме поле для вводу суми надходження, поле для додавання власної примітки та вибір категорії. Розділ "Віднімання фінансів" має схожий інтерфейс, але приймає контекст фінансових видатків. Прототип дизайну для додавання та віднімання фінансів:

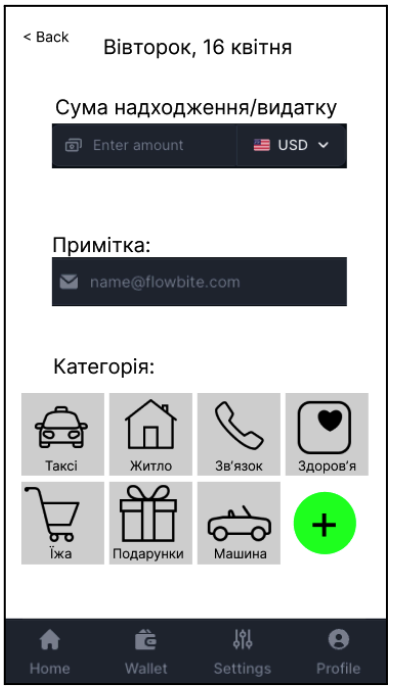

Рисунок 2.6 - Прототип дизайну для компоненту Додавання фінансів

4. Розділ "Пошук": цей розділ містить три підрозділи: "Баланс", "Надходження", "Видатки". Кожен з цих розділів має графік та два поля для вводу: "дата з", "дата до" та "список включених категорій". Після чого користувач може здійснити пошук за заданими критеріями, що спричинить оновлення графіку. Підрозділ "Баланс" має лінійний графік, що відображає зміну загального балансу у вибраному діапазоні дат та із урахуванням включених категорій. Прототип дизайну для підрозділу "Баланс":

Рисунок 2.7 - Прототип дизайну для підрозділа "Баланс".

Підрозділ "Надходження" та "Видатки" має графік у вигляді кругової діаграми, що відображає відповідні витрати або надходження у вибраному діапазоні дат з урахуванням включених категорій. Прототип дизайну для підрозділа "Надходження":

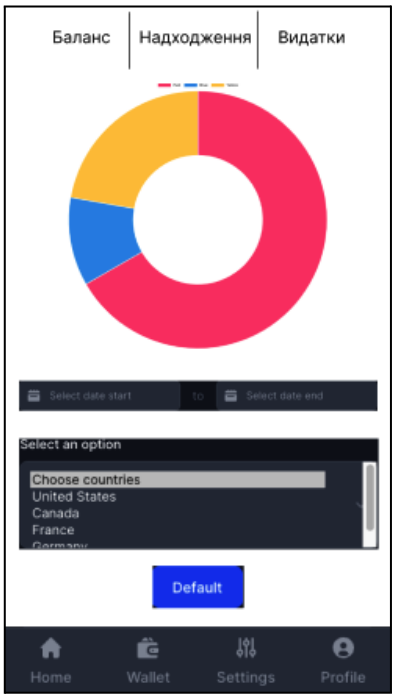

Рисунок 2.8 - Прототип дизайну підрозділу "Надходження"

5. Розділ "Профіль": цей розділ містить дані про користувача - електрона пошта. Також розділ передбачає можливість зміни паролю, а отже містить форму із двома полями: "старий пароль" і "новий пароль" - відправка форми перевіряє старий пароль на відповідність, і у разі успіху відбувається зміна паролю для користувача. Прототип дизайну розділу "Профіль":

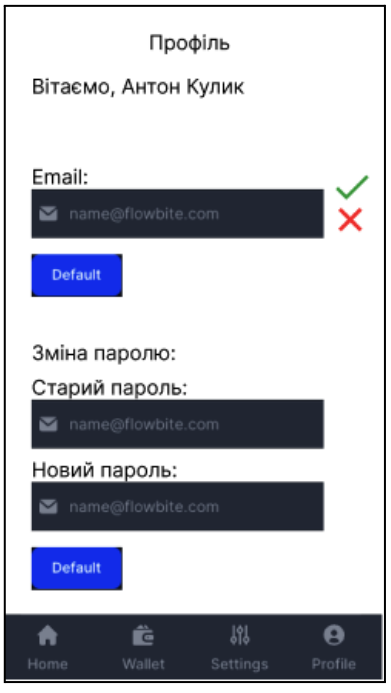

Рисунок 2.9 - Прототип дизайну розділу "Профіль"

6. Розділ "Параметри": цей розділ дає можливість користувачеві перемкнути тему додатку: світла або темна. Даний функціонал необхідний для розробки, оскільки певна частка людей надає перевагу використанню темної теми. Розділ також містить поле на згоду отримання розсилки на електрону пошту або відміна цієї згоди. Останнім компонентом даного розділу є управління категоріями, що може використовувати користувач під час додавання фінансових операцій. Прототип дизайну розділу "Параметри":

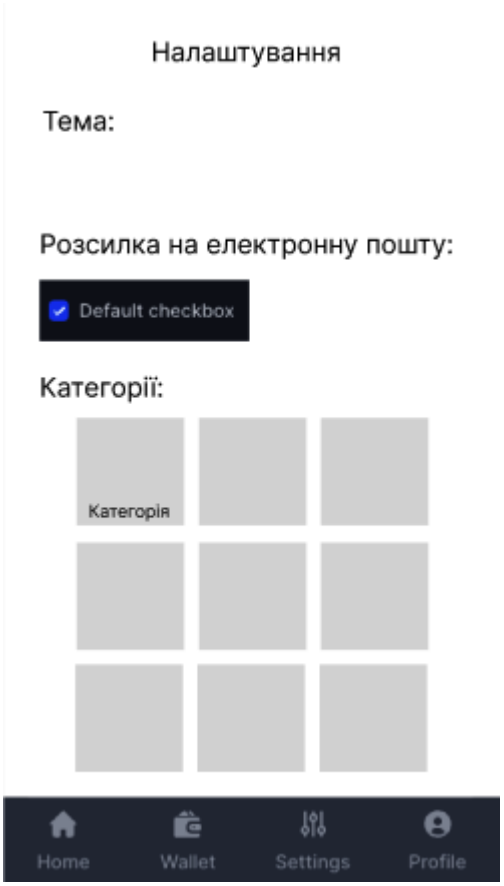

Рисунок 2.10 - Прототип дизайну розділу "Параметри"

2.7 Проектування моделі бази даних

База даних автоматизованої інформаційної системи повинна містити три рівні представлення: концептуальний, фізичний та логічний.

A. Концептуальне представлення бази даних – це високорівневе описання структури та залежностей між даними в предметній області обліку та управління особистими фінансами. Наприклад, можна визначити таблиці, які містять інформацію про операції фінансових надходжень, видатків та категорій. Кожна фінансова операція має свою категорію, у одної категорії може бути декілька операцій. Концептуальне представлення допомагає зрозуміти взаємозв'язки між різними об'єктами в системі та визначити ключові аспекти бази даних.

B. Фізичне представлення бази даних – це опис технічних аспектів зберігання даних, таких як формати файлів, типи дискових пристроїв, що використовуються для зберігання інформації системи обліку та управління особистими фінансами. Фізичне представлення бази даних включає в себе детальні відомості про розташування та організацію даних на дисках, а також процеси забезпечення безпеки і зменшення ризиків втрати даних.

C. Логічне представлення бази даних – це опис способів, якими дані можуть бути доступні та опрацьовуватися за допомогою запитів. Логічне представлення включає в себе опис схем бази даних, запитів, що дозволяють виконувати різні операції з даними, та інших елементів, що дозволяють взаємодіяти з інформацією системи обілку та управління особистими фінансами. Логічне представлення бази даних дозволяє здійснювати пошук, фільтр тощо.

Наступні розділи роботи наводять детальний опис кожного рівня представлення бази даних для автоматизованої інформаційної системи обліку та управління особистими фінансами.

2.7.1 Створення концептуальної моделі

Концептуальне представлення бази даних для інформаційної системи обліку та управління особистими фінансами почалось з аналізу предметної області. Основне питання розбору є визначення об'єктів та їх взаємозв'язків в систем. Оскільки система передбачає сервісно-орієнтований підхід, то і об’єкти існують в контексті свого сервісу. Тому можна виділити наступні основні сервіси: сервіс облікових записів, сервіс авторизації, сервіс профілю користувача, сервіс фінансів.

Сервіс облікових записів містить сутність “Обліковий запис” (Account). Сервіс авторизації містить сутність “Пароль авторизації” (Authentication Password). Сервіс профілю користувача містить сутність “Профіль” (Profile). Сервіс фінансів містить наступні сутності: “Категорія операції” (Operation Category), “Фінансова операція” (Finance Operation).

Далі, створена модель, яка описує зв'язки між різними об'єктами в системі.

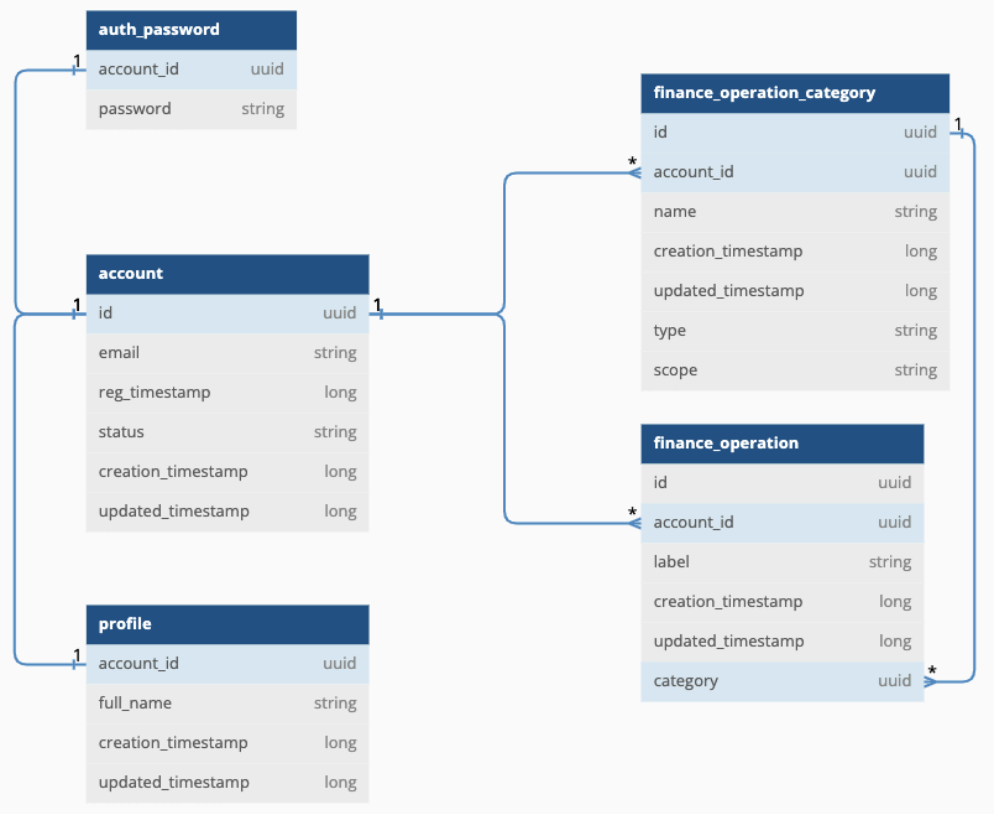

Рисунок 2.11 – концептуальна модель опису зв’язків розроблювальної бази даних для системи

Отже, сутність “Обліковий запис” є репрезентацією одного користувача. Один користувач має зв’язок із одним об’єктом сутності “Пароль авторизації” та із одним об’єктом сутності “Профіль”. Користувач має зв’язок із множиною об’єктів сутності ”Категорія операції”. Користувач має зв’язок із множиною об’єктів сутності “Фінансова операція”. Об’єкт сутності “Фінансова операція” має зв’язок із одним об’єктом сутності “Категорія операції”, в той час коли об’єкт сутності “Категорія операції” має зв’язок із множиною об’єктів сутності “Фінансова операція”.

Крім того, необхідно визначити ключові атрибути кожного об'єкту. Наприклад, для сутності користувача це стали такі атрибути, як унікальний ідентифікатор користувача, електронна пошта користувача, дата реєстрації, статус користувача тощо.

Детальний опис кожного атрибуту визначено та показано на рисунку 2.12 нижче.

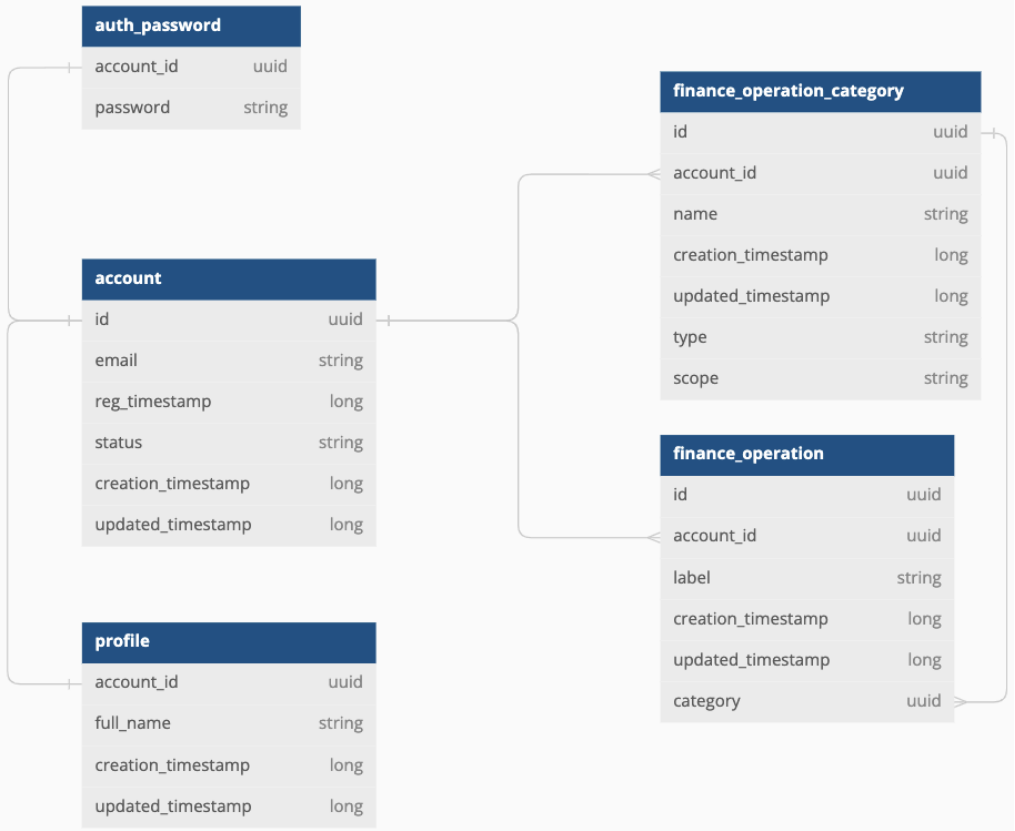

Рисунок 2.12 – концептуальна модель опису атрибутів розроблювальної бази даних інформаційної системи

Завершуючи створення концептуального представлення бази даних автоматизованої інформаційної системи обліку та управління особистими фінансами, створено та описано всі знайдені об'єкти та зв'язки між ними, а також визначені ключові атрибути. Це надало загальний огляд предметної області та визначило структуру бази даних для подальшої розробки фізичного та логічного представлень.

2.7.2 Створення логічної моделі

Після створення концептуального представлення бази даних автоматизованої інформаційної системи обліку та управління особистими фінансами, наступним етапом стало створення логічного представлення. Логічне представлення полягає в перетворенні концептуальної моделі в модель, яка враховує особливості конкретної системи управління базою даних.

Детальний опис визначено та показано на рисунку 2.13 нижче.

Рисунок 2.13 – описана логічна модель з урахуванням взаємозв’язків.

На етапі створення логічного представлення визначено сутності, атрибути та зв'язки між ними, відповідно до концептуальної моделі. При цьому, враховані особливості використовуваної системи управління базою даних. Враховано, що в базі даних фінансові операції надходження і видатку є різними сутностями, а тому вони мають окремі таблиці. Таке ж саме правило застосовується і для категорій.

Також на етапі створення логічного представлення були визначені обмеження цілісності даних, такі як обов'язковість заповнення певних атрибутів, обмеження на діапазон значень атрибутів та заборона на видалення записів, що мають зв'язки з іншими записами в базі даних.

Завершуючи створення логічного представлення бази даних автоматизованої інформаційної системи обліку та управління особистими фінансами, були описати всі таблиці та їх зв'язки, а також визначені обмеження цілісності даних застосовані в програмуванні самого проєкту. Це дозволило визначити структуру бази даних, яка відповідає концептуальній моделі та потребам системи управління базою даних і перейти до створення фізичного представлення.

2.7.3 **Створення фізичної моделі**

Після створення логічного представлення бази даних, наступним етапом стало створення фізичного представлення. Фізичне представлення полягає в перетворенні логічної моделі в модель, яка відображає реальні характеристики фізичної реалізації бази даних та їх типів даних.

На етапі створення фізичного представлення була визначена структура та параметри фізичної реалізації бази даних. Це включило в себе вибір типу сервера баз даних, налаштування параметрів бази даних та визначення способу зберігання даних.

Також на етапі створення фізичного представлення були визначені параметри забезпечення безпеки даних, такі як:

- Аутентифікація та авторизація – процеси ідентифікації користувачів та визначення їхніх прав доступу до даних. Для цього доцільно використовувати різні механізми, такі як логіни та паролі, двофакторна аутентифікація, біометричні технології тощо.
- Шифрування даних – процес перетворення звичайного тексту в зашифрований, що забезпечує конфіденційність даних. Для цього доцільно використовувати різні алгоритми шифрування, такі як AES, RSA, Blowfish тощо.
- Захист від SQL–ін'єкцій – це захист від вразливості, при якій зловмисник може використати SQL–запит для витягування або зміни даних в базі даних. Для захисту від цього доцільно використовувати параметризовані запити, а також фільтрацію введених користувачем даних.
- Резервне копіювання – забезпечення можливості відновлення даних в разі їх втрати або пошкодження. Для цього доцільно використовувати різні методи резервного копіювання, такі як повний бекап, інкрементальне копіювання тощо.
- Моніторинг та аудит – процес відстеження доступу до даних та змін в базі даних з метою виявлення потенційних загроз безпеці. Для цього доцільно використовувати системи моніторингу та аудиту, такі як Microsoft SQL Server Audit, Oracle Audit Vault тощо.

Завершуючи створення фізичного представлення бази даних автоматизованої інформаційної системи обліку та управління особистими фінансами, забезпечено її реалізацію згідно з встановленими параметрами та забезпечено правильну інтеграцію із системою управління базою даних. Це дозволило забезпечити ефективне та безпечне функціонування бази даних та системи в цілому.

2.7.4 **Загальний вигляд сутностей бази даних**

Після опису створення концептуального, логічного, та фізичного представлення бази даних, сформовано загальний вид сутностей(таблиць) для початку розробки, програмування та подальшої розробки системи.

Таблиця "Обліковий запис" (account)

- id (UUID, PRIMARY KEY) – унікальний ідентифікатор користувача
- email (VARCHAR) – електронна пошта користувача
- reg_timestamp (BIGINT) – дата реєстрації користувача
- status (VARCHAR) – статус користувача
- creation_timestamp (BIGINT) – дата створення об’єкту
- updated_timestamp (BIGINT) – дата останнього оновлення користувача

Таблиця "Пароль авторизації" (auth_password)

- account_id (UUID, PRIMARY KEY) – ідентифікатор користувача
- password (VARCHAR) – зашифрований пароль користувача

Таблиця "Профіль" (profile)

- account_id (UUID, PRIMARY KEY) – ідентифікатор користувача
- full_name (VARCHAR) – повне ім’я користувача
- creation_timestamp (BIGINT) – дата створення об’єкту
- updated_timestamp (BIGINT) – дата останнього оновлення об’єкту

Таблиця "Операція надходження" (finance_add_operation)

- id (UUID, PRIMARY KEY) – унікальний ідентифікатор операції
- account_id (UUID) – ідентифікатор користувача
- label (VARCHAR) – примітка операції
- creation_timestamp (BIGINT) – дата створення об’єкту
- updated_timestamp (BIGINT) – дата останнього оновлення об’єкту
- category (UUID) – ідентифікатор категорії

Таблиця "Операція видатку" (finance_remove_operation)

- id (UUID, PRIMARY KEY) – унікальний ідентифікатор операції
- account_id (UUID) – ідентифікатор користувача
- label (VARCHAR) – примітка операції
- creation_timestamp (BIGINT) – дата створення об’єкту
- updated_timestamp (BIGINT) – дата останнього оновлення об’єкту
- category (UUID) – ідентифікатор категорії

Таблиця "Категорія операції надходження" (finance_add_operation_category)

- id (UUID, PRIMARY KEY) – унікальний ідентифікатор категорії
- account_id (UUID) – ідентифікатор користувача
- name (VARCHAR) – назва категорії
- creation_timestamp (BIGINT) – дата створення об’єкту
- updated_timestamp (BIGINT) – дата останнього оновлення об’єкту
- type (VARCHAR) – тип категорії
- scope (VARCHAR) – область видимості категорії

Таблиця "Категорія операції видатку" (finance_remove_operation_category)

- id (UUID, PRIMARY KEY) – унікальний ідентифікатор категорії
- account_id (UUID) – ідентифікатор користувача
- name (VARCHAR) – назва категорії
- creation_timestamp (BIGINT) – дата створення об’єкту
- updated_timestamp (BIGINT) – дата останнього оновлення об’єкту
- type (VARCHAR) – тип категорії
- scope (VARCHAR) – область видимості категорії

Розділ проєктування інформаційного забезпечення став важливим етапом у розробці автоматизованої інформаційної системи обліку та управління особистими фінансами.

Розділ дозволив обрати мову програмування та технологію для безпосередньої реалізації інформаційної системи. Вибір мови програмування для реалізації системи залежав від вимог до продуктивності та функціональності, а також від наявності необхідних бібліотек та фреймворків.

Розділ дозволив визначити загальну архітектуру інформаційної системи, її необхідні компоненти, функціональності та взаємодії між ними. Декомпозиція компонентів системи є повною і завершеною. Для детальної реалізації окремих компонентів рівень декомпозиції залежить від самого компоненту.

В даному розділі також було розглянуто проектування бази даних, окремих сутностей та зв’язку між ними. Було розглянуто концептуальну, логічну та фізичну моделі даних. Виконаний аналіз щодо обґрунтованості вибору реляційної бази даних. Зпроектована база даних дозволить ефективно зберігати необхідні дані у контексті інформаційної системи.

Окрім того, у розділі проектування було створено прототип дизайну для графічного інтерфейсу користувача. Прототип дизайну задає загальну структуру кожного розділу. Детальніший дизайн графічного інтерфейсу користувача буде відбуватись паралельно із розробкою цього інтерфейсу.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Вибір програмного інструментарію

3.1.1 Бази даних

Дві популярні бази даних, які розглянуті для використовувати для розробки інформаційного забезпечення автоматизованої інформаційної системи обліку та управління особистими фінансами – PostgreSQL та MongoDB.

PostgreSQL: PostgreSQL – це потужна та надійна об'єктно-реляційна система керування базами даних (СКБД). Вона використовує мову запитів SQL для зберігання та обробки даних. PostgreSQL надає широкі можливості для розробки різноманітних додатків, від веб-сайтів до складних корпоративних систем.

MongoDB: MongoDB – це нереляційна база даних, яка використовує документи для зберігання даних. Вона підтримує широкий спектр операцій, включаючи збереження, оновлення, вилучення та пошук даних. MongoDB дозволяє зберігати дані у форматі JSON, що дозволяє більш легкий доступ до даних для розробників.

Таблиця 3.1 – порівняння баз даних для використання в практичній реалізації продукту

	PostgreSQL	MongoDB
Тип бази даних	Об’єктно-реляційна	Нереляційна
Мова запитів	SQL	MongoDB Query Language
Схема даних	Статична, може бути динамічною	Динамічна
Підтримка транзакцій	Повна підтримка транзакцій	Обмежена підтримка транзакцій
Підтримка реплікації	Так	Так

Продовження таблиці 3.1 - порівняння баз даних для використання в практичній реалізації продукту

Масштабованість	Горизонтальна та вертикальна	Горизонтальна
Доступність	Доступний на багатьох операційних системах	Доступний на багатьох операційних системах
Додаткові можливості	Повна підтримка ACID, гнучкість вибору типу індексів, розвинена система прав доступу	Вбудована підтримка геопросторових запитів, гнучкість у використанні складних документів

Для розробки інформаційного забезпечення та насамперед баз даних обрано PostgreSQL. Це пов'язано з тим, що PostgreSQL є реляцій на базою даних, яка підтримує транзакції ACID, що є важливим аспектом для систем з великим обсягом даних та високою надійністю. Також PostgreSQL має багато інструментів для оптимізації запитів та широко використовується в індустрії.

Загалом, вибір бази даних залежить від конкретних потреб проекту. Враховуючи особливості задачі дипломного проєкту, PostgreSQL є кращим вибором з точки зору надійності та організації даних.

3.1.2 **Програмне середовище**

Для розробки програмного забезпечення для автоматизованої інформаційної системи обліку та управління особистими фінансами на мові Java із використанням Spring Framework можна використовувати різні програмні середовища, такі як NetBeans, IntelliJ IDEA та Eclipse. Оскільки NetBeans є застарілим продуктом, у цій роботі порівняємо IntelliJ IDEA та Eclipse.

IntelliJ IDEA – розроблена компанією JetBrains, відома своєю потужністю та розширеними можливостями для розробки на мовах Java та Kotlin. Вона

включає в себе інтелектуальні функції, такі як автоматичне доповнення коду, аналіз коду, рефакторинг та багато інших. IntelliJ IDEA також підтримує широкий спектр інструментів для розробки веб-додатків, мобільних додатків та інших типів програм.

Eclipse – з іншого боку, є вільно розповсюджуваною платформою, розробленою фондом Eclipse. Вона відома своєю гнучкістю та розширюваністю завдяки широкому спектру плагінів, доступних для розширення її функціоналу. Eclipse також підтримує різні мови програмування та типи проектів, включаючи Java, C/C++, PHP, Python та інші.

Таблиця 3.2 – порівняння програмних середовищ для використання в практичній реалізації продукту

	IntelliJ IDEA	Eclipse
Вартість	Платне, є безкоштовна версія для навчання	Безкоштовне
Підтримка Git	Є	Є
Нативна підтримка Web-розробки	Є	Є, за використанням плагінів
Кількість плагінів	Значна кількість плагінів від самого розробника та спільноти	Значна кількість плагінів від різних розробників
Інструменти для тестування та налагодження	Багато інтегрованих засобів	Багато інтегрованих засобів

Оскільки для реалізації інформаційного забезпечення системи був обраний фреймворк Spring, а середовище розробки IntelliJ IDEA має інструменти для роботи із Spring Framework “із коробки”, то вибір стає саме на

IntelliJ IDEA. Це середовище розробки має велику спільноту і є передовим навідміну від Eclipse. IntelliJ IDEA відома своєю високою продуктивністю та широким спектром функцій, які полегшують розробку програмного забезпечення. Вона має потужні інтегровані інструменти для аналізу, автоматизації та підтримки розробки. IntelliJ IDEA надає широку підтримку для різних мов програмування, включаючи Java, Kotlin, JavaScript, TypeScript, PHP, Python та інші, що робить її привабливою для розробників з різних сфер. Команда розробників за IntelliJ IDEA активно веде розробку, випускаючи нові версії та вдосконалення. Також JetBrains надає хорошу підтримку користувачам.

Зважаючи на ці переваги, можна зробити висновок, що IntelliJ IDEA від JetBrains є оптимальним вибором програмного середовища для розробки інформаційного забезпечення для автоматизованої інформаційної системи обліку та управління особистими фінансами.

3.2 **Файлова структура проєкту**

Файлова структура проєкту розробки інформаційного забезпечення та баз даних розумного будинку включає наступні компоненти:

1. Коренева папка проекту: Це основна папка проєкту, яка містить всі інші підпапки та файли. Вона може мати назву, що відображає назву проєкту або ідентифікатор.
2. api: Згідно розробленої архітектури системи, ця папка містить реалізований код для шару представлення, що є наближчим шаром до контролеру у шаблоні MVC. Шар представлення перетворює об’єкти шару бізнес-логіки у об’єкти, що зможуть бути використані для легкого і зрозумілого подавання інформація на графічний інтерфейс користувача. Також шар представлення може мати реалізовану логіку для глобального чи локального кешування.
3. app: Ця папка містить програмний код для запуску самої інформаційної системи і програмний код, що відноситься до стадії запуску системи - сюди

входить: файли конфігурації системи та конфігурація на рівні програмного коду.

4. biz: Згідно розробленої архітектури системи, ця папка містить реалізований програмний код для шару бізнес-логіки. Цей шар є останнім у наведеній попередньо архітектурі. На цьому рівні реалізовано основні сутності системи - модель і методи взаємодії із ними. Кожна сутність на цьому рівні має свій сервіс і взаємодія із сутностями відбувається лише через відповідний сервіс.
5. diagrams: Ця папка містить всю документацію, пов'язану з проєктом. Вона може включати такі файли, як специфікації вимог, технічні описи, діаграми архітектури, звіти про тестування та будь-яку іншу документацію, необхідну для розуміння та розвитку проєкту.
6. ui: Згідно розробленої архітектури системи, ця папка містить реалізований програмний код для шару представлення, а саме шар контролерів. Тут реалізовано програмний код для сутності контролерів у шаблоні MVC. Також папка містить реалізований графічний інтерфейс користувача у вигляді комбінації HTML сторінок із CSS-стилями та програмним кодом, написаним на JavaScript.
7. pom.xml: Цей файл є основним файлом конфігурації проекту в середовищі розробки на основі інструменту управління проектами Maven для Java-програм. Основна його функція - це описувати конфігурацію проекту, включаючи залежності, плагіни, налаштування збірки тощо. Цей файл дозволяє Maven автоматизувати процес збірки, тестування та розгортання проекту. Завдяки структурованому підходу до конфігурації, команди можуть легко розуміти та керувати проектом.

8. Додаткові папки: Залежно від потреб проєкту, можуть бути додаткові папки для специфічних компонентів або модулів. Наприклад, папка для інтеграції з

іншими системами, папка для документації API, папка для додаткових статичних сторінок веб-сайту тощо.

9. Додаткові файли: Службові файли, що використовуються для інтеграції із іншими інструментами, такими як Git або файли, що відносяться до конкретного середовища розробки.

Ця файлова структура дозволяє організовувати проєкт у логічні групи файлів та директорій, забезпечуючи зручність розробки, управління та збереження даних. Важливо пам'ятати, що загальна структура повинна відповідати архітектурі інформаційної системи. Структура для детальної реалізації системи у свою чергу вже може варіюватися залежно від конкретних вимог та потреб проєкту, і можна адаптувати її відповідно до власних вимог і стандартів розробки. На рисунку нижче зображена схема вигляду файлової структури, що була описана вище.

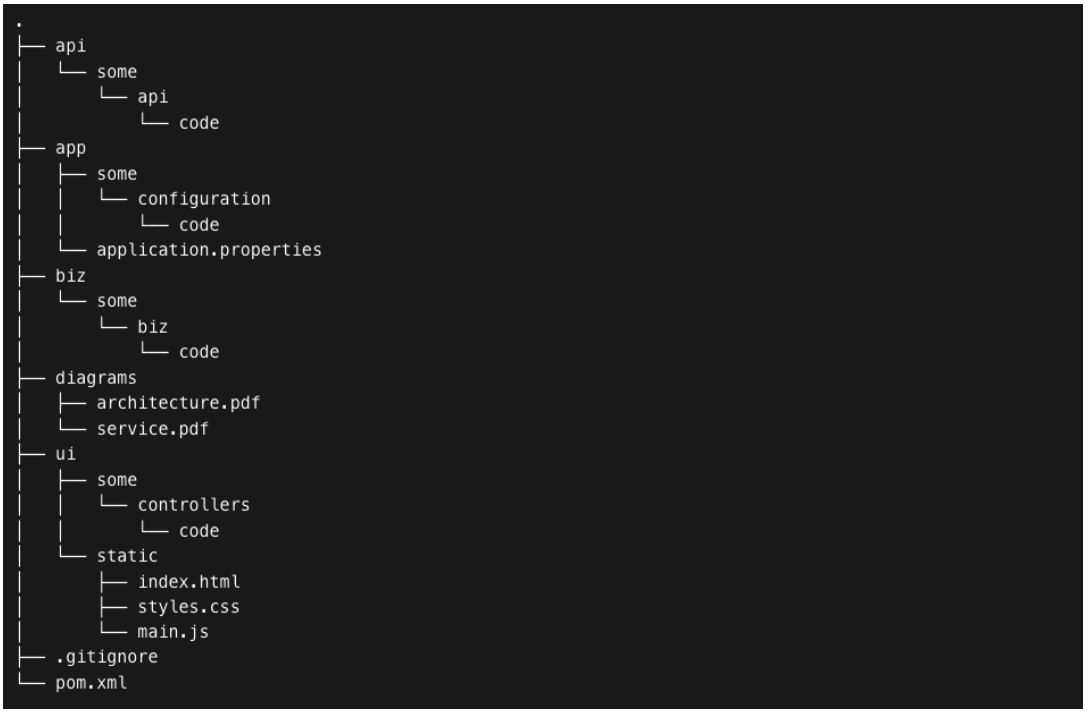

Рисунок 3.2 – Схема файлової структури для розроблюваної системи

3.3 **Шар бізнес-логіки**

Шар представленна компонований у вигляді сервісів. Кожен сервіс має свою сферу відповідальності і працює лише зі своїм набором сутностей.

Сервіс облікових записів - “AccountService”. Сфера відповідальності - облікові записи користувачів. Сутності, якими управляє сервіс: обліковий запис, налаштування акаунту. Методи, що надає сервіс для роботи із ним:

- createAccount – створення акаунту. Приймає у якості аргументу електрону пошту та набір статусів, що надаються обліковому запису. Повертає об’єкт створеного облікового запису.
- getAccountById – отримати акаунт за унікальним ідентифікатор. Приймає у якості аргументу унікальний об’єкт користувача. Повертає об’єкт облікового запису.
- getAccountByEmail – отримати акаунт за електронною поштою. Приймає у якості аргументу електрону пошту. Повертає об’єкт облікового запису.
- updateAccount – оновити акаунт. Приймає у якості аргументу об’єкт облікового запису із оновленими полями, що будуть збережені у базі даних. Повертає об’єкт оновленого облікового запису.
- getAccountSettings – отримати налаштування облікового запису. Приймає у якості аргументу унікальний ідентифікатор користувача. Повертає об’єкт, що містить інформацію про налаштування облікового запису.
- updateNewsCheckSetting – оновити налаштування про отримання новин електроною поштою. Приймає у якості аргументу унікальний ідентифікатор користувача та логічну зміну із варіантами “так” або “ні”, що і буде збережено в базі даних для конкретного облікового запису.

Сервіс авторизації - “AuthenticationService”. Сфера відповідальності - управління паролями користувачів. Сутності, якими управляє сервіс: пароль від облікового запису. Методи, що надає сервіс для роботи із ним:

- setPassword – встановити пароль. Приймає у якості аргументу пароль облікового запису. Метод не повертає жодного значення.
- getEncodedPassword – отримати зашифрований пароль. Приймає у якості аргументу унікальний ідентифікатор користувача. Повертає пароль у зашифрованому вигляді, у тому вигляді, в якому він зберігається у базі даних.

Сервіс фінансів - “FinanceService”. Сфера відповідальності - управління фінансовими операціями користувачів. Сутності, якими управляє сервіс: фінансова операція надходження, фінансова операція видатку, баланс. Методи, що надає сервіс для роботи із ним:

- createIncome– створити операцію надходження. Приймає у якості аргументів унікальний ідентифікатор користувача, примітку, суму надходження та категорію надходження. Повертає створений об’єкт операції надходження.
- createOutcome – створити операцію видатку. Приймає у якості аргументів унікальний ідентифікатор користувача, примітку, суму видатку та категорію видатку. Повертає створений об’єкт операції видатку.
- listIncomes – отримати список операцій надходження. Приймає у якості аргументу критерію для пошуку операцій надходження. Для критерії можливо задати унікальний ідентифікатор користувача, включені категорії, діапазон дат.
- listOutcomes – отримати список операцій видатку. Приймає у якості аргументу критерію для пошуку операцій видатку. Для критерії

можливо задати унікальний ідентифікатор користувача, включені категорії, діапазон дат.

- getBalance – отримати баланс. Приймає у якості аргументів унікальний ідентифікатор користувача та діапазон дат, за якими потрібно знайти баланс.

Сервіс профілів користувачів - “ProfileService”. Сфера відповідальності - управління профілями користувачів. Сутності, якими управляє сервіс: профіль користувача. Методи, що надає сервіс для роботи із ним:

- createProfile – створити профіль користувача. Приймає у якості аргументів унікальний ідентифікатор користувача та повне ім’я користувача. Повертає створений об’єкт профілю користувача.
- getProfile– отримати профіль користувача. Приймає у якості аргументів унікальний ідентифікатор користувача. Повертає об’єкт користувача.

3.4 **Шар представлення**

Майже кожен сервіс має свій варіант певного програмного інтерфейсу, але для шару представлення. Шар представлення - це найближчий шар до графічного інтерфейсу користувача, тому об’єкти, що повертаються програманими інтерфейсами, відповідають об’єктам, що фактично відображаться на графічному інтерфейсі користувача. Оскільки програмні інтерфейси найближчі до користувача, кожен має представлення про контекст поточного користувача, тобто методи можуть не приймати у якості аргументу унікальний ідентифікатор користувача, оскільки програманий інтерфейс вже має доступ до такого виду інформації. Методи програмного інтерфейсу облікових записів - AccountAPI:

- changeEmail – змінити електрону пошту облікового запису. Приймає у якості аругменту нову електрону пошту. Метод не повертає жодного результату.

Методи програмного інтерфейсу налаштувань облікового запису - AccountSettingsAPI:

- getAccountSettings – отримати налаштування облікового запису. Не приймає жодного аргументу. Метод повертає налаштування облікового запису для поточного користувача.

- updateAccountSettings – оновити налаштування облікового запису. Приймає у якості аргументу об’єкт із налаштуваннями облікового запису, що будуть використний для оновлення. Метод повертає оновлені налаштування облікового запису.

Методи програмного інтерфейсу аутентифікації - AuthAPI:

- updatePassword – оновити пароль для облікового запису. Приймає у якості аргументів старий пароль та новий пароль. Метод не повертає жодного результату.

Методи програмного інтерфейсу фінансових операцій - FinanceAPI:

- getTimeContext – отримати часовий контекст. Не приймає жодного аргументу. Повертає об’єкт часового контексту на поточний час.
- getTimeContext – отримати часовий контекст. Приймає у якості аргументів місяць та рік. Повертає об’єкт часового контексту, зважаючи на вказаний місяць і рік.
- getMyBalance – отримати мій баланс. Не приймає жодного аргументу. Повертає об’єкт балансу для поточного користувача.
- getMyBalance – отримати мій баланс. Приймає у якості аргументів місяць та рік. Повертає об’єкт балансу для поточного користувача, враховуючи вказаний місяць та рік.
- getMyBalance – отримати мій баланс. Приймає у якості аргументу діапазон дат. Повертає об’єкт балансу для поточного користувача, зважаючи на вказаний діапазон дат.
- createIncome – створити операцію надходження. Приймає у якості аргументів суму надходження, примітку, категорію надходження, місяць та рік. Повертає створену операцію надходження.
- createOutcome – створити операцію видатку. Приймає у якості аргументів суму видатку, примітку, категорію видатку, місяць та рік. Повертає створену операцію видатку.

- getIncomeOperations – отримати операції надходження. Не приймає жодного аргументу. Повертає об’єкт із інформацією для графічного представлення надходжень для поточного користувача за поточний місяць.
- getIncomeOperations – отримати операції надходження. Приймає у якості аргументу діапазон дат. Повертає об’єкт із інформацією для графічного представлення надходжень для поточного користувача за обраний діапазон дат.
- getIncomeOperations – отримати операції надходження. Приймає у якості аргументу критерію для отримання надходжень. Повертає об’єкт із інформацією для графічного представлення надходжень для поточного користувача, враховуючи надану критерію.
- getOutcomeOperations – отримати операції видатку. Не приймає жодного аргументу. Повертає об’єкт із інформацією для графічного представлення видактів для поточного користувача за поточний місяць.
- getOutcomeOperations – отримати операції видатку. Приймає у якості аргументу діапазон дат. Повертає об’єкт із інформацією для графічного представлення видатків для поточного користувача за обраний діапазон дат.
- getOutcomeOperations – отримати операції видатку. Приймає у якості аргументу критерію для отримання видатків. Повертає об’єкт із інформацією для графічного представлення видактів для поточного користувача, враховуючи надану критерію.
- getOucomeOperations – отримати операції видатку. Приймає у якості аргументів місяць та рік. Повертає об’єкт із інформацією для графічного представлення видатків для поточного користувача, враховуючи наданий місяць та рік.

- getFinanceOperations – отримати список фінансових операцій. Приймає у якості аргументів тип операцій та діапазон дат. Повертає список фінансових операцій для поточного користувача, враховуючи тип операцій та діапазон дат.

Методи програмного інтерфейсу для профілів користувачів - ProfileAPI:

- createProfile – створити профіль користувача. Приймає у якості аргументів унікальний ідентифікатор користувача та повне ім’я користувача. Повертає створений об’єкт профілю.
- getMyProfile – отримати мій профіль. Не приймає жодного аргументу. Повертає об’єкт профілю для поточного користувача.

Методи програмного інтерфейсу реєстрації - RegistrationAPI:

- registerAccount – зареєструвати обліковий запис. Приймає у якості аргументів електрону пошту, ім’я та пароль. Метод не повертає результату.

До шару представлення також відноситься спроектований та розроблений графічний інтерфейс користувача. Графічний інтерфейс користувача представлений у вигляді веб-сторінок із набором вихідних даних (графіків, списків) та елементів для вводу даних для здійснення фільтрацій, пошуків, редагування даних.

Нижче представлена сторінка створення облікового запису. Сторінка містить форму із полями для введення. Інформація з полей буде використана для створення облікового запису користувача. Поля: електрона пошта, прізвище ім’я по-батькові, пароль.

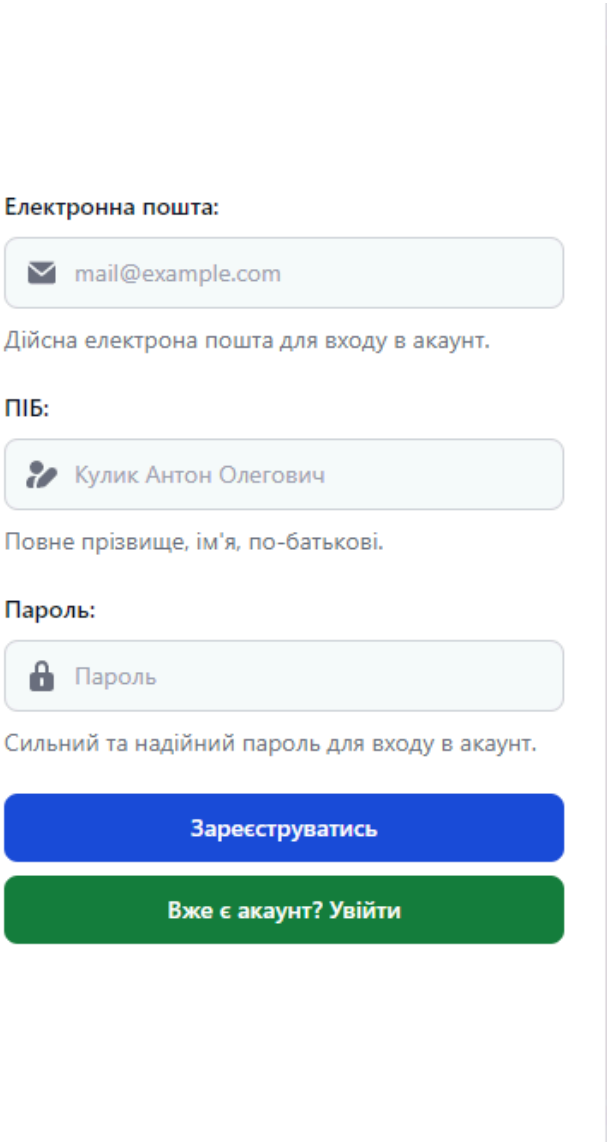

Рисунок 3.3 - Веб-сторінка для створення облікового запису

Веб-сторінка для входу в обліковий запис користувача. Сторінка містить форму із полями для введення. Інформація з полей буде використана для входу в обліковий запис користувача. Якщо електронна пошта чи пароль облікового запису неправильний, відповідна помилка виводиться на інтерфейс користувача.

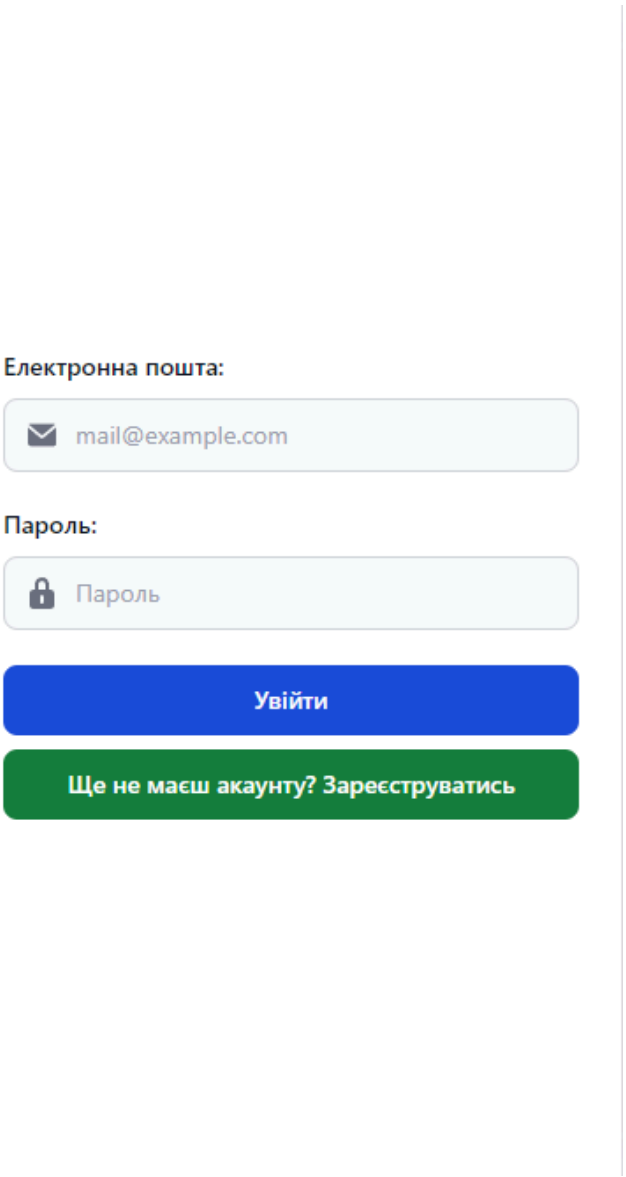

Рисунок 3.4 - Веб-сторінка для входу в обліковий запис користувача

Веб-сторінка із налаштуваннями. Сторінка містить перемикач теми. Тема може бути переключена на світлу або темну. На сторінці налаштувань користувач може підтвердити або скасувати дозвіл для розсилки на електрону пошту. Останньою секцією сторінки є список категорій видатку, що можуть бути використані для введення обліку особистих фінансів.

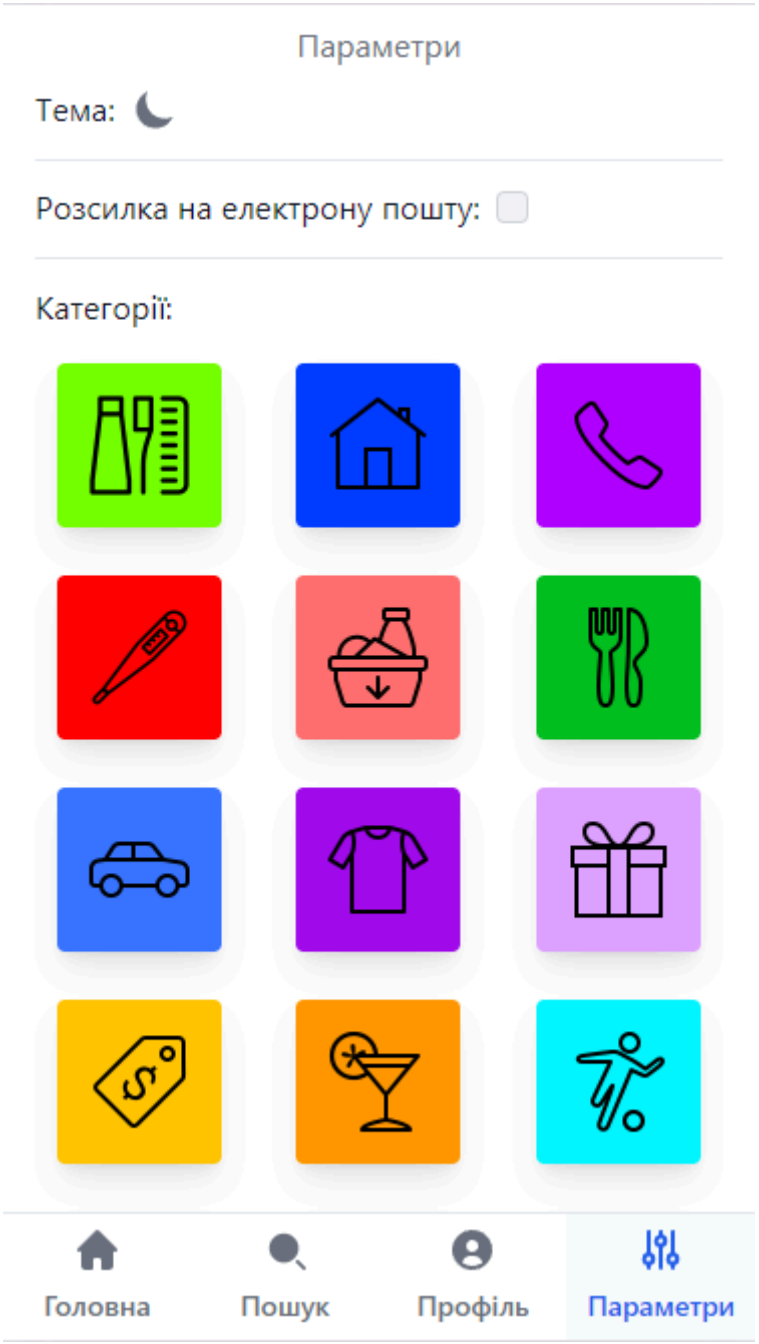

Рисунок 3.5 - Веб-сторінка із налаштуваннями

Веб-сторінка профілю користувача містить декілька форм для редагування данних. На даній сторінці користувач може змінити поточну електрону пошту або оновити пароль. Форма оновлення паролю містить два поля: старий пароль та новий пароль. У випадку, якщо старий пароль не співпадає, відповідна помилка буде показана на інтерфейсі. При зміні електронної пошти здійснюється перевірка на доступність електронної пошти. У випадку, якщо задана електронна пошта вже використовується іншим користувачем, відповідна помилка буде показана на інтерфейсі.

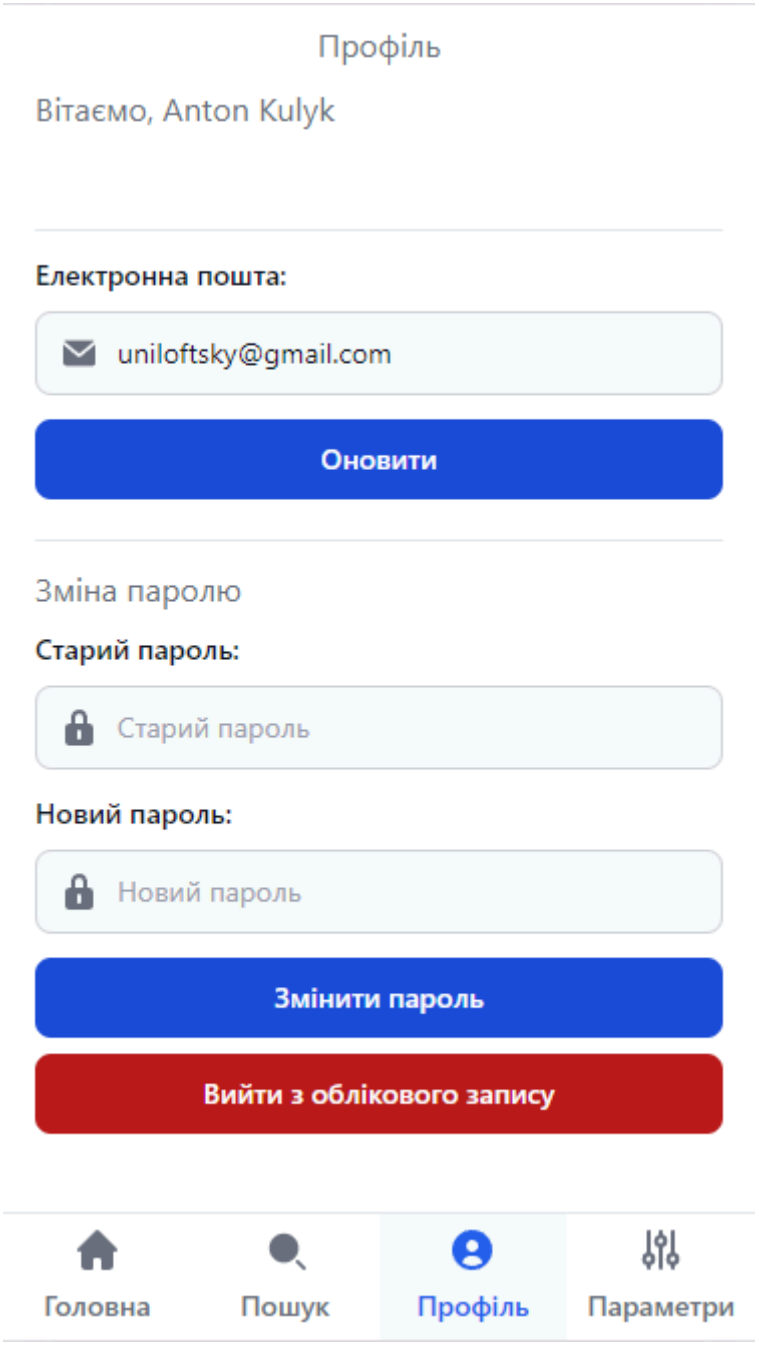

Рисунок 3.6 - Веб-сторінка профілю користувача

Головна сторінка системи містить діаграму з видатками за всіма категоріями за поточний місяць. Нижче від діаграми можна переглянути особистий баланс. Останньою секцією сторінки є кнопки додавання та віднімання, що означають створення операції надходження та операції видатку відповідно. У верхній частині сторінки присутня інформація про місяць, за який показані видатки. Присутні кнопки для переходу між місяцями зліва і справа від поточного місяця.

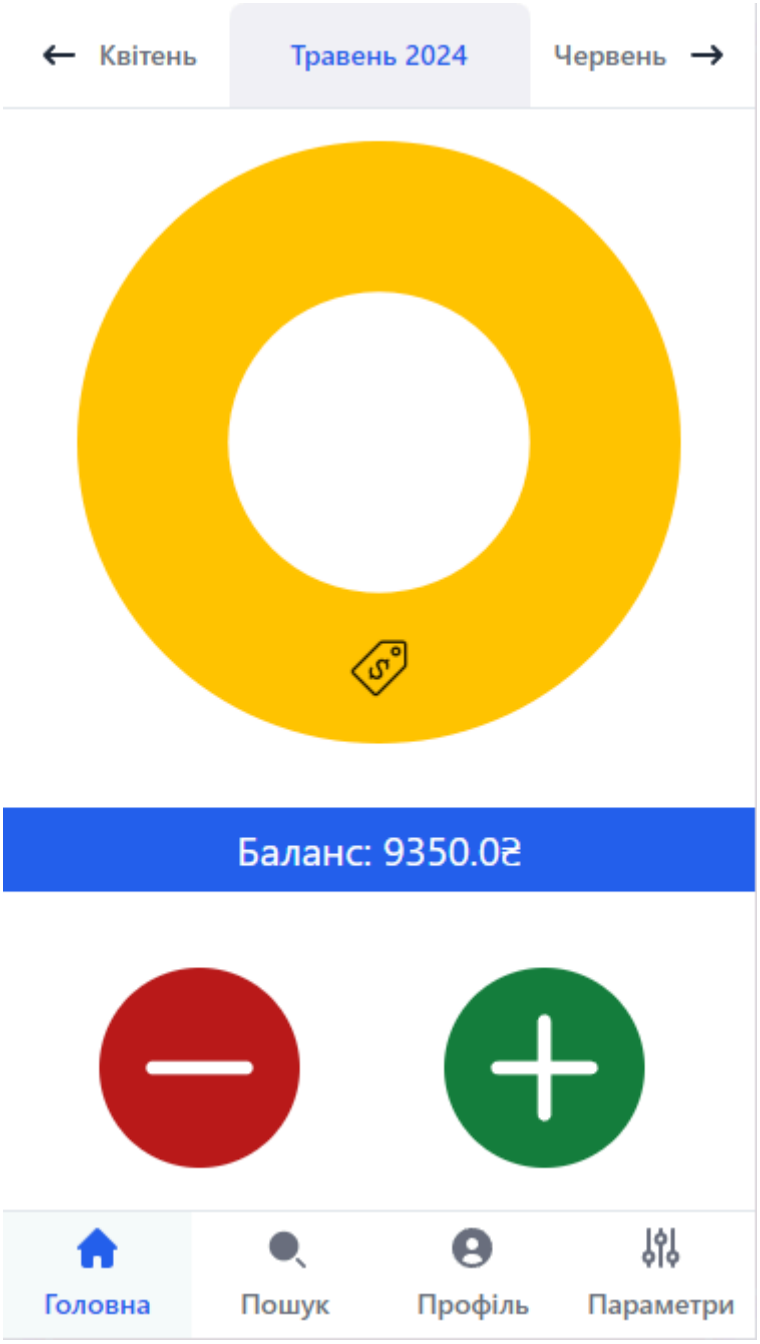

Рисунок 3.7 - Веб-сторінка головної сторінки

Натиснувши в середині діаграми та зачекавши, користувач відкриває веб-сторінку, що має список усіх фінансових операцій (надходження, видатки), згруповані за категоріями. Справа від категорії показано загальну суму надходження чи видатку, в залежності від категорії. Відкривши конкретну категорію, користувач може переглянути окремі операції із нотаткою, сумою та датою створення.

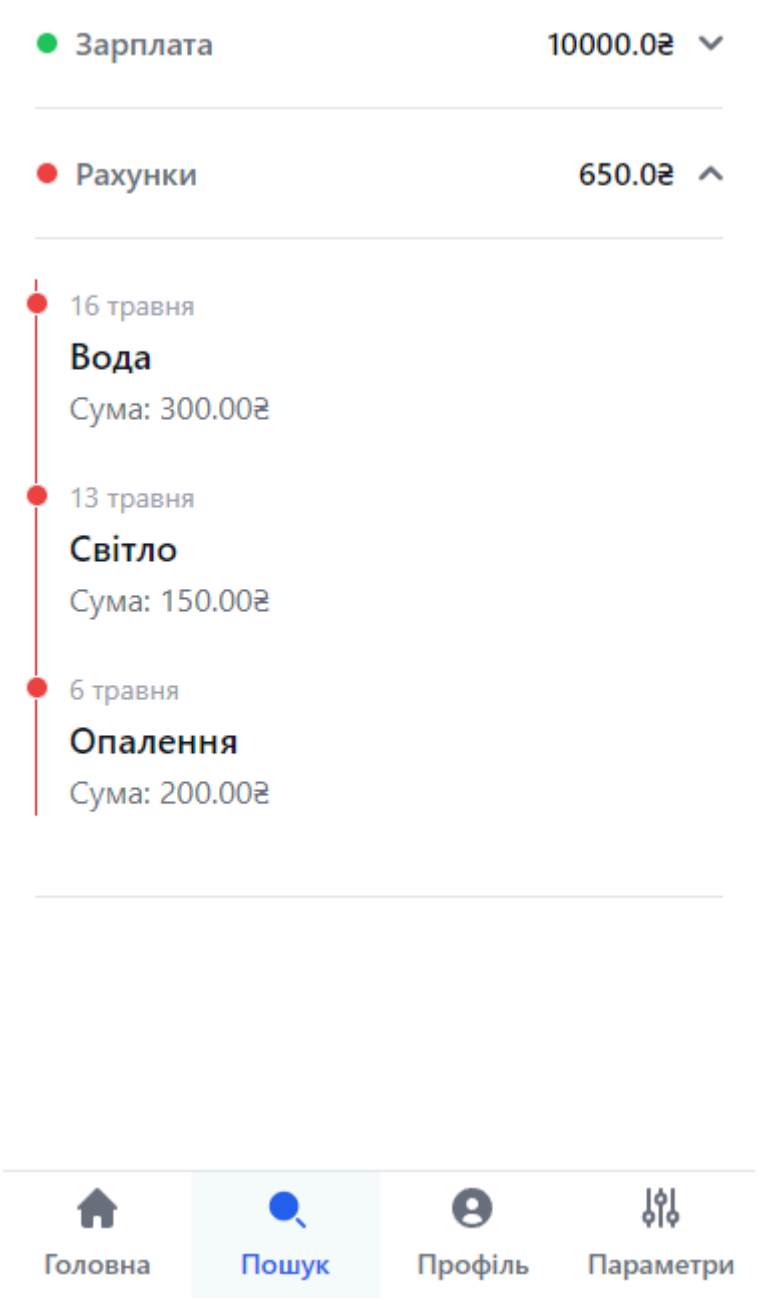

Рисунок 3.8 - Веб-сторінка із списком операцій

Веб-сторінка створення операції видатку містить форму із полями для введення. Форма містить наступні поля: сума видатку, примітка, категорія. Користувач повинен ввести суму видатку, за бажанням додати примітку та обрати категорію, до якої відноситься операція видатку.

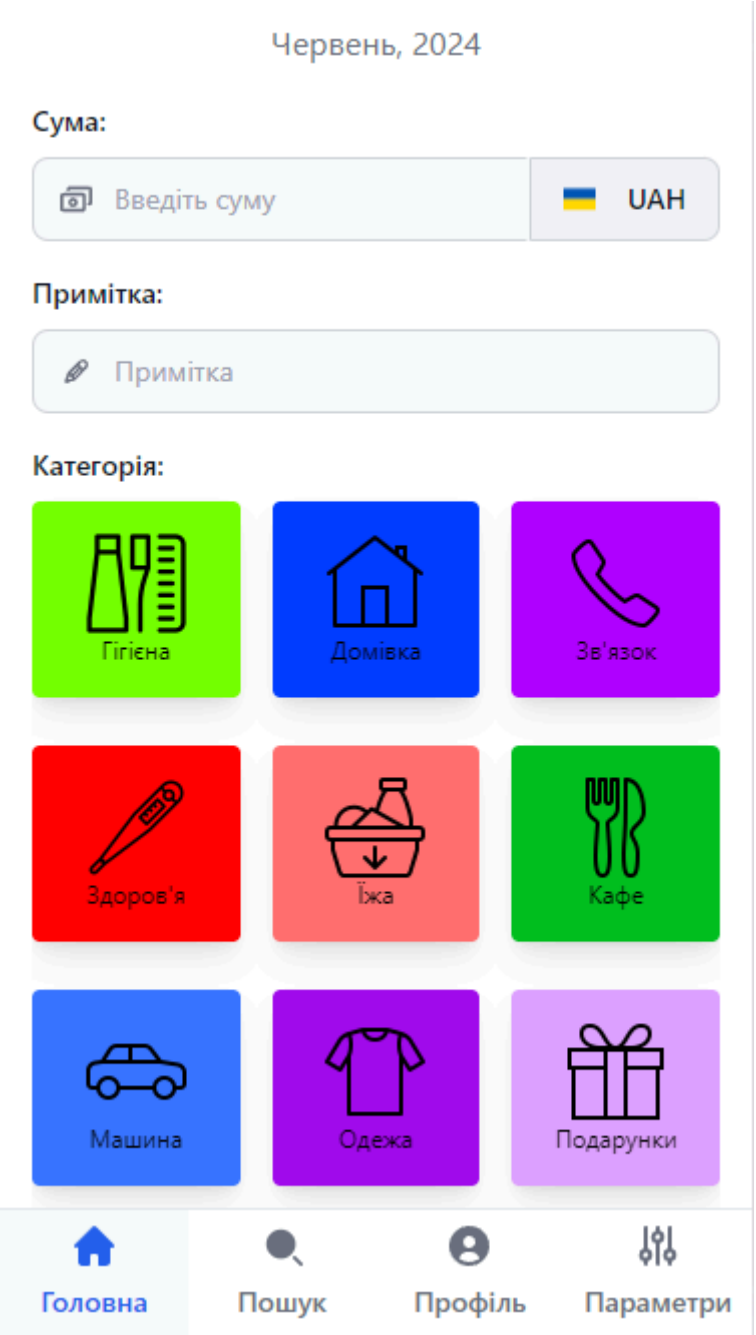

Рисунок 3.9 - Веб-сторінка створення операції видатку

Веб-сторінка створення операції надходження містить форму із полями для введення. Форма містить наступні поля: сума надходження, примітка, категорія. Користувач повинен ввести суму надходження, за бажанням додати примітку та обрати категорію, до якої відноситься операція надходження.

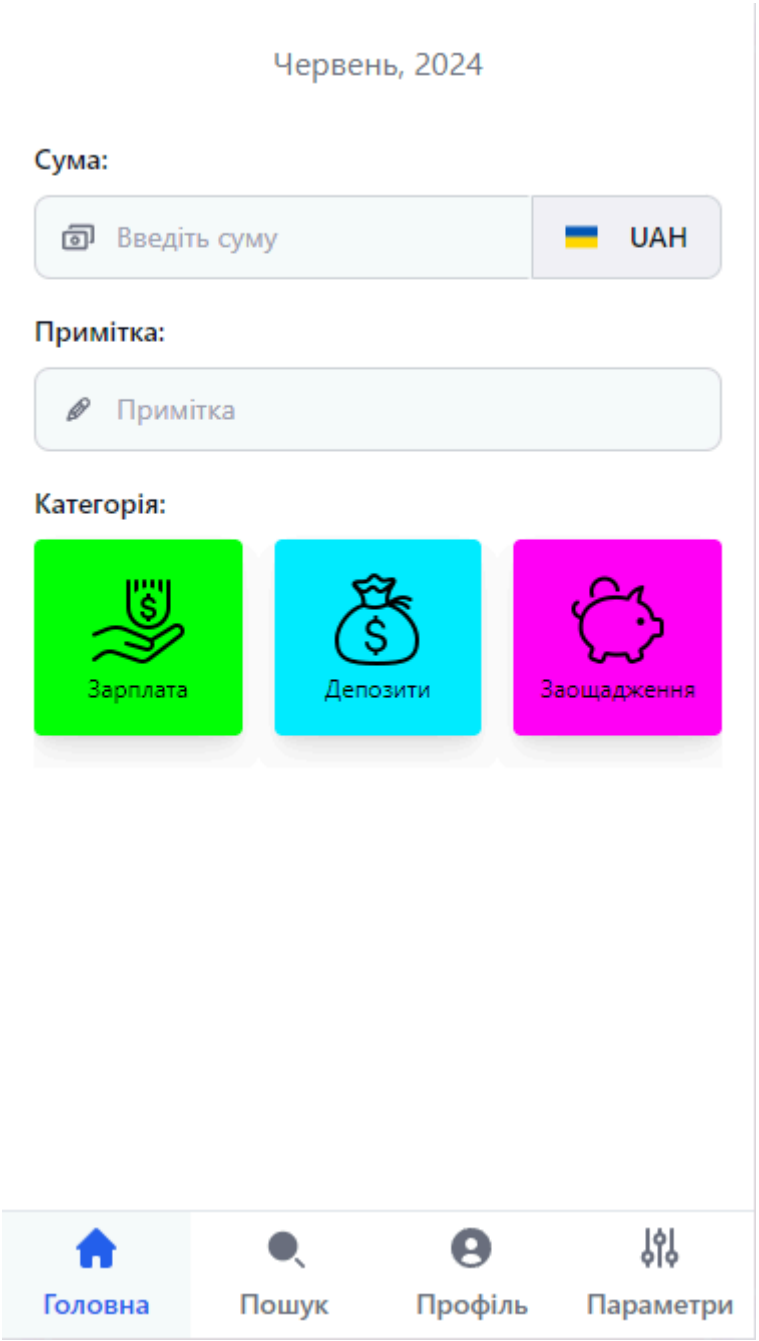

Рисунок 3.10 - Веб-сторінка створення операції надходження

Пункт меню “Пошук” веде до веб-сторінки, що містить інформацію про зміну балансу за обраний період часу. В нижній частині сторінки користувач може обрати діапазо дат та переглянути як змінювався його баланс впродовж обраного часу.

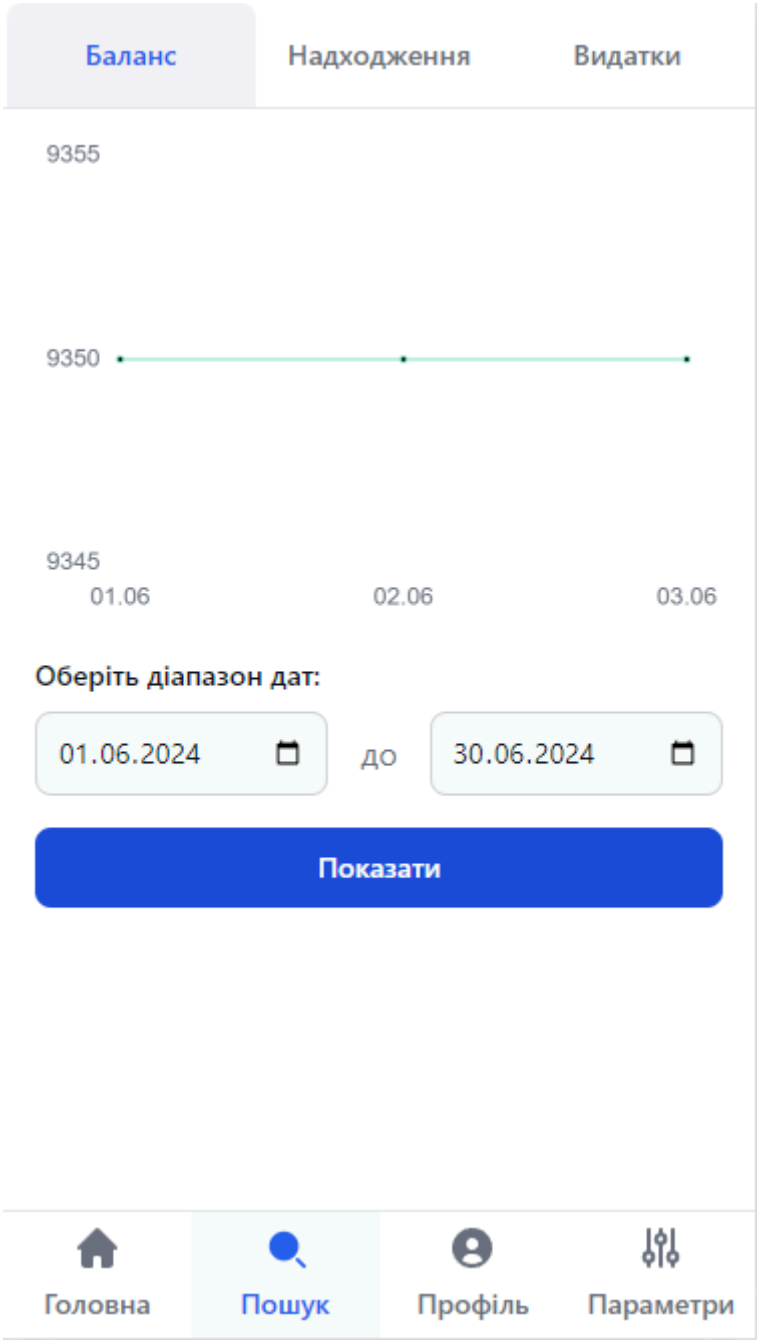

Рисунок 3.11 - Веб-сторінка зміни балансу

На поточній сторінці користувач може обрати іншу вкладку - Надходження або Видатки. В залежності від обраного типу операцій, система показує кругову діаграму із видатками або надходженнями. Користувач може задати діапазон дат та включені категорії, система відобразить кругову діаграму з урахуванням вхідних данних.

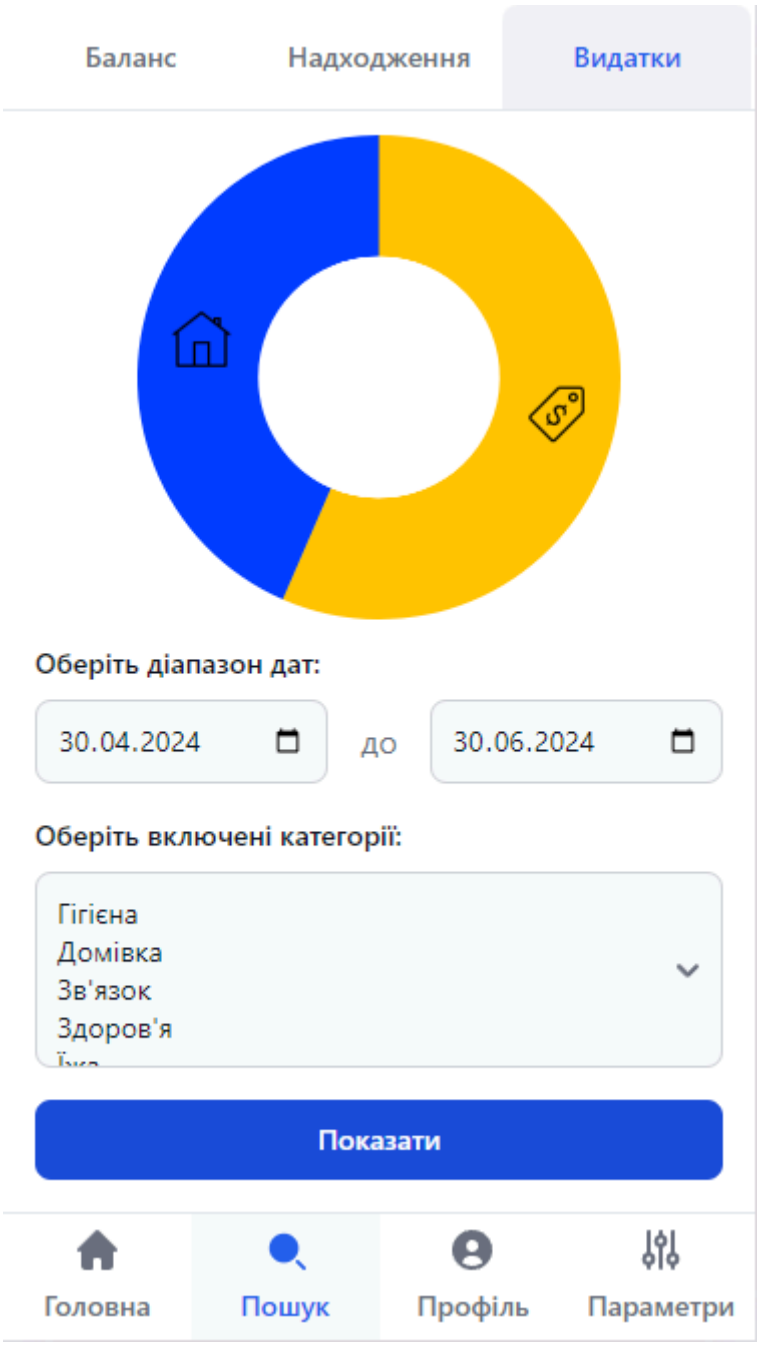

Рисунок 3.12 - Веб-сторінка із операціями видатку з урахуванням фільтрів

Розділ практичної реалізації інформаційного забезпчення для автоматизованої системи управління та обліку особистими фінансами є одним із найважливіших, оскільки висвітлює фактичну реалізацію програмного засобу.

У цьому розділі було обраного програмний інструментарій, а саме середовище розробки, що використовується для практичної реалізації застосунку, а також вибір конкретної СУБД, що задовільняє вимогам спроектованої інформаційної системи.

Створено файлову структуру проекту, де спроектовані функціональні компоненти і модулі розбиті на папки, що допомагає швидко і ефективно змінювати або розширювати код в залежності від потреб.

Окрім того, в розділі розглянуто детально практичну реалізацію кожного із шарів, шару бізнес-логіки, що включає у себе шар бази даних та шару представлення, що в певній мірі відповідає за інтерфейс користувача. В контексті шару бізнес-логіки було розглянуто сервіси та методи, що дозволяють працювати із сервісом. В контексті шару представлення було розглянуто програмні інтерфейси, що взаємодіють із сервісами для збереження або відтворення інформації.

Разом із шаром представлення було розглянуто реалізований графічний інтерфейс користувача, що представлений у вигляді веб-сторінок із набором полів для введення та сторінок, що відображають дані у графічному вигляді.

4 БІЗНЕС ПЛАН

4.1 Опис продукту

4.1.1 Вступ

Автоматизована інформаційна система для обліку та управління особистими фінансами представлена у вигляді веб-додатку. Облік особистих фінансів необхідний для контролю особистих витрат, прибутків, змін у бюджеті. Ці інструменти допомагають планувати бюджет, виявляти зайві витрати, що в результаті покаращує економічне становище людини.

Отже, автоматизована інформаційна система для обліку та управління особистими фінансами має великий потенціал для покращення якості життя людей та забезпечення ефективного використання особистих фінансів.

Монетизувати автоматизовану інформаційну систему можливо шляхом продажу доступу до нього або надаванням преміальних функцій за певну вартість.

4.1.2 Основні функції та переваги

Загальна мета автоматизованої системи обліку та управління особистими фінансами - це облік особистих фінансів. Основними функціями системи є:

- облік надходження фінансів за категоріями.
- облік витрат фінансів, використовуючи велику кількість категорій.
- генерація звітів у графічному, зручному вигляді.
- фільтрація і пошук фінансових рухів за певним діапазоном дат та категоріями.
- допомога в аналізі фінансових змін користувача.

Всі перелічені функції підкріплюються доступом до системи з будь-якої

точки, де є інтернет-з’єднання та з будь-якої платформи, що підтримує роботу веб-браузера. До переваг можна включити надійність та безпеку данних, зручний інтерфейс і простота використання.

4.1.3 **Цільова аудиторія**

Оскільки автоматизована інформаційна система розрахована на звичайного пересічного користувача, ця вимога була врахована під час проектування та розробки системи.

До основних користувачів можна віднести:

1. Фізичні особи:
    - **Молоді професіонали**: Люди у віці від 20 до 35 років, які тільки починають свою кар'єру і хочуть правильно управляти своїми фінансами з самого початку. Ця група зазвичай цікавиться технологічними новинками та активними користувачами мобільних додатків.
    - **Сімейні пари**: Люди, які мають сім'ї та бажають планувати спільний бюджет, відстежувати витрати на побутові потреби, освіту дітей та інші сімейні витрати.
    - **Пенсіонери**: Люди, які вийшли на пенсію і хочуть ретельно планувати свої витрати, щоб забезпечити собі фінансову стабільність. Вони можуть мати менше досвіду з технологіями, тому для них важлива простота та інтуїтивність інтерфейсу.
2. Малі підприємства
    - **Індивідуальні підприємці**: Власники невеликих бізнесів, які часто ведуть облік своїх особистих та бізнес-видатків самостійно. Вони можуть використовувати автоматизовану систему обліку та управління особистими фінансами для відстеження щоденних витрат, планування бюджету та аналізу фінансових результатів.

- **Мікропідприємства**: Невеликі компанії з кількістю працівників до 10 осіб. Власники та керівники таких компаній можуть використовувати інформаційну систему для контролю за фінансовими потоками, управління витратами і планування бюджетів.

Розуміння цільової аудиторії допомагає належним чином адаптувати функціонал автоматизованої інформаційної системи обліку та управління особистими фінансами та маркетингову стратегію для залучення користувачів. Важливо, щоб продукт відповідав потребам і очікуванням різних груп користувачів, забезпечуючи простоту використання для фізичних осіб і розширені можливості для малих підприємств.

4.2 Аналіз ринку

4.2.1 Огляд ринку

У сучасному світі автоматизовані інформаційні системи для обліку та управління фінансами займають певну нишу серед додатків, що спрямовані на покращення життя у повсякденні. Такі системи дозволяють швидко і зручно вести облік особистих фінансів і отримувати зрозумілий та чіткий вигляд стану бюджету. У даному аналізі розглянуто найбільш відомі системи обліку та управління особистими фінансами.

1. Система «Monefy»

Система «Monefy» - це інформаційна система, що дозволяє зберігати свої фінансові операції за певними категоріями для подальшого відображення даних у графічному вигляді. Система дає можливість аналізувати бюджет за певний період часу або за певними категоріями. Система представлена у вигляді мобільного додатку та розрахована на середньостатистичного користувача.

2. Система «tidely»

Система «tidely» представляє собою більш просунутий інструмент для обліку та управління особистими фінансами. Система надає більш детальний графічний вигляд загального стану бюджету.

3. Система «Precoro»

Система «Precoro» розроблена для професійного використання. Система надає можливість розподіляти фінансові операції за певними робочими відділами, враховувати майбутні платежі, прив’язувати банківські рахунки для синхронізації фінансів. Також система надає зовнішній програмний інтерфейс, що дає можливість іншим розробникам використовувати певний спектр готових рішень для обліку та управління особистими фінансами.

Таблиця 4.1 – опис існуючих розробок автоматизованих інформаційних систем обліку та управління особистими фінансами.

Назва	Країна	Аудиторія	Особливості
Monefy	США	Звичайний користувач	Мобільний додаток, просте управління особистими фінансами, графічний вигляд бюджету.
tidely	Німеччина	Продвинутий користувач	Більш продвинутий інструмент для управління особистими фінансами. Детальний графічний вигляд бюджету
Precoro	США, Німеччина, Польща, Литва	Для професійного використання	Інструмент для обліку фінансів, що використовується підприємствами

Порівнюючи різноманітні системи обліку та управління особистими фінансами можна визначити декілька ключових аспектів, що варто враховувати при просуванні і впровадженні власної автоматизованої системи обліку та управління особистими фінансами. Система повинна в повній мірі задовольнити потреби користувача у зручному та швидкому обліку особистих фінансів. Графічний інтерфейс повинен бути інтуїтивно зрозумілим та нести в собі максимально корисну інформацію для ефективного здійснення аналізу бюджетного стану. Система повинна мати можливість здійснювати пошук і відображати запитані дані у зрозумілому форматі. Для обіймання більш широкої аудиторії, система повинна бути крос-платформенною.

4.2.2 **Оцінка попиту**

Світовий ринок фінансових технологій (FinTech) постійно зростає. У 2023 році ринок оцінювався в кілька мільярдів доларів і продовжує зростати щороку. В рамках цього ринку, додатки для управління особистими фінансами також демонструють стабільне зростання, що свідчить про високий попит на такі рішення серед користувачів. Цей тренд обумовлений кількома факторами.

По-перше, зростаюча популярність мобільних додатків для управління фінансами особливо помітна серед молоді та активних користувачів смартфонів. Молоді професіонали, які тільки починають свою кар'єру, часто шукають інструменти для ефективного управління своїми фінансами, і вони віддають перевагу зручним і технологічно просунутим рішенням. Це робить їх однією з основних цільових груп для нашого продукту.

По-друге, збільшується увага до фінансової грамотності в суспільстві. Люди починають розуміти важливість контролю своїх витрат і планування бюджету. Ця тенденція стимулює попит на додатки, які можуть допомогти в цих питаннях. Особливо це актуально для сімейних пар, які планують свій спільний бюджет, та пенсіонерів, які бажають ретельно контролювати свої витрати для забезпечення фінансової стабільності.

Пандемія COVID-19 також мала значний вплив на фінансову поведінку людей. Економічна нестабільність і необхідність жорсткішого контролю над витратами призвели до того, що більше людей почало шукати ефективні інструменти для управління своїми фінансами. Це додатково збільшує попит на подібні додатки, оскільки користувачі хочуть мати чітке уявлення про свої фінансові потоки і планувати витрати більш обережно.

Для малих підприємств також існує значний попит на рішення для управління фінансами. Індивідуальні підприємці та мікропідприємства часто стикаються з необхідністю вести облік фінансів як для бізнесу, так і для особистих потреб. Використання додатку для управління фінансами дозволяє їм ефективніше контролювати витрати, планувати бюджети і приймати управлінські рішення на основі актуальної фінансової інформації.

Таким чином, аналіз ринку та поведінка цільової аудиторії свідчать про високий потенційний попит на автоматизовану інформаційну систему для управління та обліку особистих фінансів. Це підтверджується зростанням ринку фінансових технологій, підвищенням інтересу до фінансової грамотності та необхідністю ефективного управління фінансами в умовах економічної нестабільності.

4.3 Маркетингова стратегія

4.3.1 Ціноутворення

Процес ціноутворення для автоматизованої інформаційної системи управління та обліку особистих фінансів починається з детального аналізу витрат на його розробку та підтримку. Основними складовими вартості розробки є заробітна плата команди, яка складається з двох розробників, системного архітектора та тестувальника. Середня зарплата розробника становить 2000 доларів на місяць, що є стандартом для кваліфікованих спеціалістів у цій галузі. На проект буде залучено двох розробників, що складе

4000 доларів на місяць. Системний архітектор, відповідальний за загальний дизайн і структуру системи, отримує в середньому 4000 доларів на місяць. Тестувальник, який забезпечує якість продукту, отримує в середньому 1500 доларів на місяць.

Загальні місячні витрати на зарплату команди складають 9500 доларів. Якщо припустити, що процес розробки займає шість місяців, загальна вартість на зарплати становитиме 57,000 доларів. Це є основною частиною витрат на розробку.

Крім цього, важливим компонентом витрат є хостинг у хмарі. Для забезпечення надійної роботи системи необхідно використовувати хмарні сервери. Середня вартість одного серверу складає 150 доларів на місяць. Для оптимальної роботи та масштабованості системи передбачено використання двох серверів, що призведе до щомісячних витрат у розмірі 300 доларів. За шість місяців витрати на хостинг складуть 1800 доларів.

Разом витрати на розробку та хостинг за шість місяців становитимуть 58,800 доларів. Ці витрати є основою для визначення ціни доступу до системи. Важливо також врахувати додаткові витрати на маркетинг та підтримку, які можуть значно варіюватися в залежності від обраної стратегії просування продукту.

Для забезпечення рентабельності проекту та досягнення точки беззбитковості необхідно розрахувати очікуваний дохід. Пропонується встановити ціну доступу до додатку на рівні 5 доларів на місяць або 50 доларів на рік для кінцевих користувачів. Якщо передбачається залучення 1000 користувачів у перший рік, річний дохід складатиме 50,000 доларів. Це дозволить покрити початкові витрати протягом перших 1.2 років за умови стабільного росту користувацької бази.

Таким чином, запропоноване ціноутворення забезпечує конкурентоздатність продукту на ринку та дозволяє поступово окупити витрати

на розробку та підтримку, приваблюючи користувачів з різними фінансовими можливостями.

4.3.2 **Канали розповсюдження**

Для успішного розповсюдження системи для управління та обліку особистих фінансів необхідно використовувати різноманітні канали, які забезпечать максимальне охоплення цільової аудиторії та ефективне проникнення на ринок. Основним каналом розповсюдження буде цифрові платформи, такі як Google Play і Apple App Store. Ці платформи дозволяють легко знайти, завантажити та встановити додаток на мобільні пристрої користувачів, забезпечуючи зручність та доступність.

Розміщення додатку на таких популярних платформах не тільки спрощує процес його завантаження, але й забезпечує видимість серед мільйонів потенційних користувачів. Крім того, позитивні відгуки та висока оцінка додатку на цих платформах можуть значно підвищити його популярність і довіру користувачів.

Соціальні мережі також є важливим каналом розповсюдження. Використання платформ, таких як Facebook, Instagram, Twitter і LinkedIn, дозволить залучити широку аудиторію, створюючи рекламні кампанії, публікуючи корисний контент і взаємодіючи з користувачами. Соціальні мережі надають можливість таргетувати рекламу на основі демографічних і поведінкових характеристик, що допомагає ефективно досягати цільової аудиторії.

Крім того, важливу роль відіграють партнерські програми та співпраця з впливовими особами (інфлюенсерами). Партнерство з фінансовими блогерами, експертами та іншими впливовими особами у сфері фінансів може значно підвищити обізнаність про ваш продукт. Рекомендації від авторитетних осіб допомагають залучити довіру потенційних користувачів і стимулюють їх спробувати додаток.

Використання контент-маркетингу є ще одним ефективним каналом розповсюдження. Створення корисного і інформативного контенту, такого як статті, блоги, відео та інфографіки про управління фінансами, дозволить залучити органічний трафік до системи. Публікація такого контенту на власному блозі, а також на сторонніх ресурсах і в медіа, підвищить обізнаність про продукт і його переваги.

Електронна пошта (email-маркетинг) також відіграє важливу роль у розповсюдженні додатку. Збір електронних адрес потенційних користувачів через веб-сайт або соціальні мережі дозволяє проводити регулярні розсилки з інформацією про нові функції, оновлення, спеціальні пропозиції та інші важливі новини. Це допомагає підтримувати інтерес користувачів і стимулює їх до активного використання системи.

Таким чином, ефективне використання різноманітних каналів розповсюдження дозволить досягти широкої аудиторії та забезпечити успішне просування системи для управління та обліку особистих фінансів на ринку.

4.4 Фінансовий план

4.4.1 Прогноз доходів

Прогноз доходів для системи для управління та обліку особистих фінансів базується на оцінці потенційного попиту, обраній стратегії ціноутворення та використанні ефективних каналів розповсюдження. Основна модель монетизації передбачає щомісячну або річну підписку, що забезпечує стабільний потік доходів.

Розглянемо базовий сценарій, у якому ціна доступу до додатку встановлена на рівні 5 доларів на місяць або 50 доларів на рік для кінцевих користувачів. Виходячи з цього, визначимо очікуваний дохід за перший рік роботи.

Перший рік

Враховуючи поточний ринок і обрані канали розповсюдження, реалістично очікувати, що у перший місяць ми залучимо близько 100 користувачів. Завдяки активному маркетингу, соціальним мережам, партнерським програмам та рекомендаціям від впливових осіб, кількість користувачів буде збільшуватися щомісяця на 20%.

Таким чином, прогнозована кількість користувачів у кожному місяці першого року виглядає наступним чином:

- Січень: 100 користувачів
- Лютий: 120 користувачів
- Березень: 144 користувачів
- Квітень: 173 користувачів
- Травень: 207 користувачів
- Червень: 248 користувачів
- Липень: 298 користувачів
- Серпень: 358 користувачів
- Вересень: 430 користувачів
- Жовтень: 516 користувачів
- Листопад: 619 користувачів
- Грудень: 743 користувачів

Підрахуємо загальну кількість користувачів за рік та відповідний дохід. Середня кількість користувачів за рік буде приблизно 325 (у середньому).

Якщо всі користувачі обирають щомісячну підписку (5 доларів на місяць), річний дохід буде виглядати так:

Річний дохід = 3900 придбаних доступів × 5 доларів = 19,500 доларів

Якщо ж користувачі обирають річну підписку (50 доларів на рік), дохід буде виглядати так:

Річний дохід = 743 користувачів × 50 доларів = 37,150 доларів

Наступні роки

У другий рік очікується подальше зростання користувацької бази на 15-20% щомісяця завдяки підвищенню обізнаності про продукт і збільшенню числа рекомендацій. Припустимо, що середнє щомісячне зростання складе 15%, а кількість користувачів у грудні другого року може досягти близько 3000. Враховуючи цей темп зростання, середня кількість користувачів у другий рік може становити близько 1872 придбаних доступів.

Таким чином, прогнозований річний дохід у другий рік для щомісячних підписок може скласти:

Річний дохід = 22464 придбаних доступів × 5 доларів = 112,320 доларів

Якщо користувачі обирають річну підписку:

Річний дохід = 3000 користувачів × 50 доларів = 150,000 доларів

Прогноз доходів демонструє значний потенціал для зростання з року в рік. Перший рік допоможе покрити початкові витрати та встановити базу користувачів. У наступні роки очікується суттєве збільшення доходів за рахунок розширення користувацької бази і стабільного потоку підписок. Ця стратегія забезпечить стійкий розвиток та фінансову стабільність системи для управління та обліку особистих фінансів.

4.4.2 Початкові витрати

Початкові витрати включають у себе всі необхідні витрати, пов'язані з розробкою, запуском та початковим просуванням системи для управління та обліку особистих фінансів. Оцінка цих витрат є важливою для визначення загального обсягу інвестицій, необхідних для запуску проекту.

Розробка системи

Першим етапом є розробка автоматизованої інформаційної системи обліку та управління особистими фінансами. Для цього потрібна команда розробників, системний архітектор та тестувальник. Згідно зі зазначеними раніше заробітними ставками, вартість цієї команди складає 9500 доларів на місяць (2

розробника * 2000 + 4000 системний архітектор + 1500 тестувальник). Припустимо, що процес розробки займе 6 місяців, загальні витрати на зарплату команди складуть 57,000 доларів.

Хостинг та інфраструктура

Для забезпечення роботи системи необхідно мати хорошо налаштований сервер. Оптимальним варіантом є використання хмарних серверів. Припустимо, що для ефективної роботи потрібно два сервери, що коштують 150 доларів кожен на місяць. Це призведе до загальних витрат на хостинг у розмірі 1800 доларів за півріччя.

Маркетинг та реклама

Поступовий запуск та успішне виведення системи на ринок неможливе без ефективної рекламної кампанії. Витрати на маркетинг можуть включати в себе створення рекламних матеріалів, просування у соціальних мережах, рекламу на цифрових платформах, участь у виставках та інші маркетингові заходи. Приблизно оцінюється, що на маркетинг слід виділити близько 20% від загальних витрат на розробку, тобто 9,000 доларів.

Інші витрати

До інших початкових витрат можуть включатись витрати на правове консультування, створення логотипу та корпоративного стилю, оренду офісу, придбання офісного обладнання та інше. Ці витрати можуть варіюватися в залежності від потреб проекту та стратегії його розвитку.

Загальні початкові витрати

Загальні початкові витрати можна оцінити як суму всіх вищезазначених витрат. У нашому випадку, це:

45,000 (розробка) + 1800 (хостинг) + 9,000 (маркетинг) + інші витрати

Точна сума інших витрат може бути визначена на основі конкретних потреб та стратегій проекту. Після оцінки всіх цих витрат можна зробити точну оцінку загальних початкових витрат на запуск системи для управління та обліку особистих фінансів.

ВИСНОВОК

Розвиток автоматизованих інформаційних систем управління та обліку особистих фінансів стає важливою складовою для забезпечення ефективного фінансового планування та контролю. У сучасному світі, де фінансові рішення стають все складнішими, такі системи надають можливість збільшити ефективність управління фінансами та забезпечити безпеку особистих фінансів.

Цей проект спрямований на розробку інформаційного забезпечення для автоматизованої системи управління та обліку особистих фінансів. Використання сучасних технологій програмування дозволяє нам створити ефективну базу даних та програмне забезпечення, що забезпечують надійність та безпеку обліку фінансів.

Результатом цієї роботи є створення функціональних складових системи управління та обліку особистих фінансів, які можуть бути використані для базової реалізації концепції фінансового планування та далі розширені для вдосконалення фінансової обстановки.

Важливість створення інформаційного забезпечення для систем управління та обліку особистих фінансів полягає у тому, що воно є необхідною складовою для ефективного та надійного функціонування системи. Без відповідної бази даних та програмного забезпечення, система не може надати необхідний рівень контролю та безпеки фінансів користувачів.

Отже, створення інформаційного забезпечення для автоматизованих систем управління та обліку особистих фінансів є ключовим елементом для забезпечення їх продуктивності та безпеки. Застосування сучасних технологій та методів розробки дозволяє досягти максимальної ефективності та забезпечити користувачів необхідними функціями та можливостями.

Ці системи стають все більш популярними, оскільки вони надають можливість автоматизувати багато фінансових процесів та забезпечують безпеку та контроль над особистими фінансами. Такі системи допомагають користувачам зменшити ризики та забезпечити стабільність фінансового стану..

Тож у першому розділі було описано аналіз та дослідження проблеми розробки інформаційного забезпечення та баз даних для управління та обліку особистих фінансів, в основі чого досліджені питання: аналіз існуючих рішень обліку фінансів та баз даних, постановка задачі, проблематика взаємозв’язку баз даних з реалізаціями наявних систем.

Другий розділ описує процес проектування інформаційного забезпечення, а саме: розглянуті ключові питання постановки завдання проектування ІЗ, дерева функцій, вибору технології, вибору технології забезпечення безпеки, вибору інтерфейсу користувача (вимог до інтерфейсу, загальних можливостей структури інтерфейсу, вибору типу інтерфейсу користувача) та в подальшому проектування моделі бази даних.

Третій розділ виконує задачу вибору програмного інструментарію для подальшого створення та розробки бази даних та інформаційного забезпечення.

Четвертий розділ забезпечує інформацією щодо впровадження продукту на ринок. Описує задачу проектування продукту, конкуренції, плану виробництва, організаційний план та стратегії фінансування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. The complementary use of IDEF and UML modelling approaches / C.-H. Kim et al. *Computers in Industry*. 2003. Vol. 50, no. 1. P. 35–56.
2. Effective Java. 3rd ed. Addison-Wesley Professional, 2017. 416 p.
3. Abhishek, Soumili Chandra, Soumili Chandra. Analysis and Comparison of the Spring Framework, Struts Framework, Vaadin Framework, and Play Framework Performance, Used to Create Web Applications in Java. *International Journal of Advanced Research in Science, Communication and Technology*. 2023. P. 277–282.
4. Thakur R. N., Pandey U. S. The Role of Model-View Controller in Object Oriented Software Development. *Nepal Journal of Multidisciplinary Research*. 2019. Vol. 2, no. 2. P. 1–6.
5. DEVELOPMENT OF WEB APPLICATION USING MODEL VIEW CONTROLLER (MVC) ARCHITECTURE. *International Journal of Progressive Research in Engineering Management and Science*. 2023.
6. Dylan D. Schmorrow, Cali M. Fidopiastis (2009). "Foundations of Augmented Cognition: Directing the Future of Adaptive Systems".
7. Steven C. McConnell (2004). "Code Complete: A Practical Handbook of Software Construction".
8. Robert C. Martin (2008). "Clean Code: A Handbook of Agile Software Craftsmanship".
9. Robert M. Kaplan, Dennis P. Saccuzzo (2018). Psychological Testing: Principles, Applications, and Issues.