ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОСОБИСТОГО ФІНАНСОВОГО МЕНЕДЖМЕНТУ МОВАМИ JAVA ТА JAVASCRIPT**»

Виконав: студент 4 курсу, групи ПД–44
спеціальності
121 Інженерія програмного забезпечення
Бацунов Д.С.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Нормоконтроль Трінтіна Н.А.
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного зебезпечення
Ступінь вищої освіти «Бакалавр»
Спеціальність підготовки - 121 «Інженерія програманого забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
зебезпечення

_____ Негоденко В.В.
“ ___ ” _____ 2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

БАЦУНОВА ДМИТРА СЕРГІЙОВИЧА
(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка програмного забезпечення особистого фінансового менеджменту мовами Java та JavaScript»
Керівник роботи: Жебка В.В., д.т.н., доц., зав. кафедри ТЦР

(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26

2. Строк подання студентом роботи «1» червня 2023 року

3. Вихідні дані до роботи:

3.1 Науково-технічна література з питань, пов‘язаних із застосування веб-додатків.

3.2 Середовище розробки веб додатку.

3.3 Методи та принципи побудови веб додатку.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1 Аналіз предметної області та постановка задачі розробки програмного забезпечення власного фінансового менеджменту.

4.2 Огляд засобів реалізації для створення веб додатку.

4.3 Розробка програмного забезпечення.

4.4 Опис реалізації веб додатку та тестування програмного забезпечення.

5. Перелік демонстраційних матеріалів.

5.1 Мета роботи. Об’єкт і предмет дослідження.

5.2 Результати аналізу предметної області.
5.3 Результати огляду засобів реалізації для створення веб додатку.
5.4 Архітектура додатку.
5.5 Результати дослідження та опис реалізації програми.
5.6 Апробація результатів дослідження.
5.7 Висновки.

6. Дата видачі завдання 25.02.2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02.2023 - 25.02.2023	виконано
2	Аналіз та дослідження існуючих аналогів	26.02.2023 - 10.03.23	виконано
3	Дослідження програмних об’єктів	11.03.2023 - 20.03.23	виконано
4	Моделювання об’єкту проектування	25.02.2023 - 01.03.23	виконано
5	Розробка веб-додатку	02.03.2023 - 20.03.23	виконано
6	Вступ, висновки, реферат	21.03.2023 - 24.04.23	виконано
7	Розробка обов’язкових демонстраційних матеріалів	25.04.2023 - 10.05.23	виконано
8	Попередній захист роботи	19.05.2023	виконано
9	Здача роботи	01.06.2023	виконано

Студент _____ Бацунов Д.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Жебка В.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: - с., 1 табл., 16 рис.,, 16 джерел.

Мета роботи – спрощення процесу контролю та управління особистими коштами за допомогою програмного забезпечення особистого фінансового менеджменту написану мовами Java та JavaScript

Об'єкт дослідження – процес контролю та управління особистими коштами

Предмет дослідження – програмне забезпечення для власного фінансового менеджменту

Методи дослідження: методи моделювання, методи проектування та розробки веб додатків, методи об'єктно-орієнтованого програмування

У ході роботи було проведено аналіз особливостей власного фінансового менеджменту. Проаналізовані наявні програмні та інші засоби для підрахунку оберту власних кошт. На основі аналізу було опрацьовано основні процеси фінансового менеджменту та вимоги до програмного забезпечення для побудови подальших функціональних та нефункціональних вимог. Здійснено проектування моделі за допомогою UML діаграм.

Були проаналізовані і вибрані відповідні технології та створено веб додаток для особистого фінансового менеджменту мовами Java та JavaScript

Галузь використання – фізичні особи які так чи інакше взаємодіють з фінансами.

Ключові слова: Java, JavaScript, веб розробка, Spring Boot, база даних, моно

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

SQL	-	Structured query language
HTML	-	HyperText Markup Language
CSS	-	Cascading Style Sheets
JS	-	JavaScript
REST	-	REpresentational State Transfer
ACID	-	Atomicity, Consistency, Isolation, Durability
JPA	-	Java Persistance API
ORM	-	Object-Relational Mapping
SSO	-	Single Sign-on
JWT	-	JSON web token

ВСТУП

Роль фінансового менеджменту в житті кожної людини набуває все більшого значення. Здатність ефективно керувати своїми фінансами стає ключовою навичкою для досягнення фінансової стабільності та забезпечення майбутнього благополуччя. Завдання особистого фінансового менеджменту включає в себе відстеження витрат та доходів, планування бюджету та підготовку фінансових звітів.

У сучасному цифровому світі розробка програмного забезпечення може великою мірою сприяти полегшенню цих завдань. Програмне забезпечення для особистого фінансового менеджменту дозволяє людям зручно та ефективно відстежувати свої доходи та витрати, планувати бюджет, аналізувати фінансові дані та приймати обґрунтовані фінансові рішення.

Метою даної бакалаврської роботи є розробка програмного забезпечення для особистого фінансового менеджменту з використанням мов програмування Java і JavaScript. Комбінація цих мов дозволяє створювати потужні та масштабовані програми, які забезпечують ефективну обробку фінансових даних та зручний інтерфейс для користувачів.

У процесі розробки будуть використані принципи об'єктно-орієнтованого програмування, а також фреймворки і бібліотеки, що допоможуть забезпечити швидку та надійну роботу програми. Результатом даної роботи буде функціональне програмне забезпечення, яке дозволить користувачам ефективно керувати своїми фінансами, аналізувати дані та планувати своє майбутнє.

Окрім того, дослідження в галузі особистого фінансового менеджменту та програмування може принести користь як академічній громадськості, так і практичним спеціалістам у сфері фінансів. Розробка програмного забезпечення для особистого фінансового менеджменту є актуальною та перспективною

темою дослідження, оскільки сприяє автоматизації та оптимізації фінансових процесів, підвищує фінансову грамотність та сприяє досягненню фінансової стабільності.

Отже, розробка програмного забезпечення для особистого фінансового менеджменту з використанням мов програмування Java і JavaScript має великий потенціал у полегшенні фінансових процесів та покращенні фінансової грамотності користувачів.

Об'єкт дослідження – процес контролю та управління особистими коштами

Предмет дослідження – програмне забезпечення для власного фінансового менеджменту

Мета роботи – спрощення процесу контролю та управління особистими коштами за допомогою програмного забезпечення особистого фінансового менеджменту написану мовами Java та JavaScript

Опираючись на поставлену мету, були поставлені наступні задачі:

1. Провести огляд предметної галузі.
2. Провести аналіз існуючих програмних забезпечень в області фінансового менеджменту.
3. Сформулювати функціональні і нефункціональні вимоги до веб додатку власного фінансового менеджменту.
4. Провести огляд програмних та технічних засобів реалізації веб-додатків.
5. Визначити найбільш доречні інструменти розробки.
6. Розробити відповідний до вимог функціонал.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особлисвості особистого менеджменту та поточна актуальність теми

Тема фінансів є вагомою у сучасному світі, де ефективне управління ресурсами, а саме, фінансами стає все більш критичним для досягнення успіху та досягнення особистих цілей.

Серед головних чинних важливості фінансового менеджменту можна виділити ефективний особистий фінансовий менеджмент який допомагає забезпечити фінансову стабільність. Також вміння контролювати та планувати свої доходи та витрати, управляти боргами та накопиченнями, також дозволяє досягти фінансової незалежності та фінансово забезпечити себе та свою сім'ю. Ефективний фінансовий менеджмент відкриває можливості для планування майбутніх витрат. З’являється можливість раціонально використовувати свої ресурси, отримувати вигоду від інвестицій та забезпечувати додаткові джерела доходу.

Саме тому особистий фінансовий менеджмент є ключовим аспектом нашого життя, який допомагає нам забезпечити фінансову стабільність, досягнути наших цілей та зменшити фінансовий стрес. Згідно нещодавного опитування більше 2-х мільйонів українців зареєстровані боржниками. Як повідомляє редакція “УКРІНФОРМ” протягом останього року їх кількість зросла на 18%. Саме тому питання фінансів є надважливим в наш час.

Торкаючись теми особистого фінансового менеджменту можна виділити декілька основних процесів менеджменту. По перше, це фінансовий облік – підрахунок витрат і доходів основний спососіб зрозуміти фінансове становище і реальні фінансові можливості. Підходи менеджменту відрізняються в залежності

від переваг людини. Наприклад деякі підраховують тільки великі витрати, а деякі віддають перевагу детальному запису всіх фінансових операцій вне залежності від суми операції.

Також однією з особливостей менеджменту є попередне прогнозування та плануваня – один з дійсно ефективних методів для досягненя фінансових цілей. Звичайно, планування так само залежить від конкретних можливостей людей та власних бажань але внезалежності від можливостей фізичних осіб, планування допомагає досягати цілі та зменшувати витрати. І на останок – дослідження, без якого практично не має сенсу в підрахунку. Дослідження фінансового обороту полягає у вивченні власних фінансових даних та подальшому маніпулюванню на основі виновків дослідження. Аналіз допомагає виявити всі доходи та витрати, зрозуміти, як використовуються ваші фінансові ресурси та чи досягаєця поставлені фінансові цілі.

Також виділяють декілька основних характеристик будь-яких фінансових операцій:

1. Тип операції: Операції можуть бути різними, такими як доходи (зарплата, інвестиційні доходи), витрати (купівлі, рахунки, оренда), інвестиції, погашення боргів тощо.
2. Сума операції: Це конкретна сума грошових коштів, пов'язаних з операцією, наприклад, дохід у розмірі 500 доларів або витрати у розмірі 1000 доларів.
3. Дата операції: Це дата, коли операцію було виконано або заплановано.
4. Категорія операції: Категорії допомагають класифікувати операції за характером. Наприклад, категоріями можуть бути "живлення", "транспорт", "розваги", "здоров'я" тощо.

5. Джерело чи одержувач операції: Це особа, компанія або рахунок, пов'язані з операцією. Наприклад, це може бути назва магазину, де було здійснено купівлю, або назву банку, де знаходиться рахунок для інвестицій.
6. Опис операції: Опис надає додаткові деталі чи коментарі, пов'язані з операцією. Це може бути корисним для запам'ятовування основних деталей операції.

Ці характеристики допомагають структурувати та систематизувати фінансові операції, а також аналізувати їх для кращого розуміння фінансового стану та прийняття усвідомлених рішень.

1.2 Особливості розробки веб додатків

Особливості розробки веб-додатків визначаються їхнім специфічним середовищем та вимогами. Розуміння цих особливостей дозволяє ефективно використовувати інструменти, технології та методики, що призводить до оптимізації роботи додатку і забезпечення його швидкого та ефективного функціонування. Виділяють декілька важливих критеріїв досягнення єфективності додатку: швидкість завантаження сторінок і відповіді додатку. Варто звернути увагу на оптимізацію розмірів зображень та інших медіафайлів, використання кешування і мінімізацію запитів до сервера. Також на ефективність можуть посприяти технології. Вибір правильних технологій та інструментів може суттєво покращити продуктивність та перфоманс веб-додатку.

Розробка з урахуванням можливості масштабування дозволяє додатку ефективно працювати навіть при збільшенні обсягу даних та навантаження.
Також правильна організація коду, використання ефективних алгоритмів та оптимізація запитів до бази даних можуть позитивно позначитися на продуктивності додатку. Застосування кешування, зменшення кількості запитів і

використання компіляції може допомогти досягти кращої швидкодії та продуктивності.

Також веб додатки мають працювати на різних платформах, браузерах та пристроях. Розглядання особливостей допомагає забезпечити сумісність і зручну навігацію для користувачів незалежно від їхнього обладнання або вибору програмного забезпечення. Тому обов’язково треба приділити ретельну увагу засобам реалзіації і їх сумісність з браузерами.

Більше того забезпечити відповідність стандартам веб-розробки, таким як HTML, CSS, JavaScript та інші. Це сприяє забезпеченню якості, сумісності та доступності додатку.

В цілому, розглядання особливостей розробки веб-додатків є необхідним для створення високоякісних, ефективних та безпечних продуктів, які забезпечують задоволення потреб користувачів та успішну реалізацію бізнес-цілей тому в роботі над проектом власного фінансового менеджменту були враховані головні зазначені особливості.

1.3 Аналіз аналогів

Було виконано ретельний аналіз наявних програмних рішень для вирішення проблем, що розглядаються у даній бакалаврській роботі. Нижче наведено огляд трьох програмних засобів, які були розглянуті:

1. BudgetSimple:

   Переваги:

   - Спрощений і інтуїтивно зрозумілий інтерфейс для створення та відстеження бюджету. Це робить його ідеальним варіантом для початківців або тих, хто не має багато досвіду з фінансовим

управлінням. Ви зможете швидко налаштувати свій бюджет і почати відстежувати свої фінанси.

- Забезпечує автоматичне підключення до банківських рахунків та витягів
- Розширена статистика у вигляді діаграм

Недоліки:

- Відсутність створення фінансових цілей
- Обмежена функціональність: Не має налаштувань для деталізації та категоризації витрат
- Вартість: Хоча BudgetSimple надає безкоштовний план, він може мати обмежену функціональність або обмеження в обсягу даних. Якщо ви бажаєте отримати більше розширені можливості або більший обсяг даних, ви, можливо, повинні розглянути платні плани, що можуть вплинути на ваши витрати.

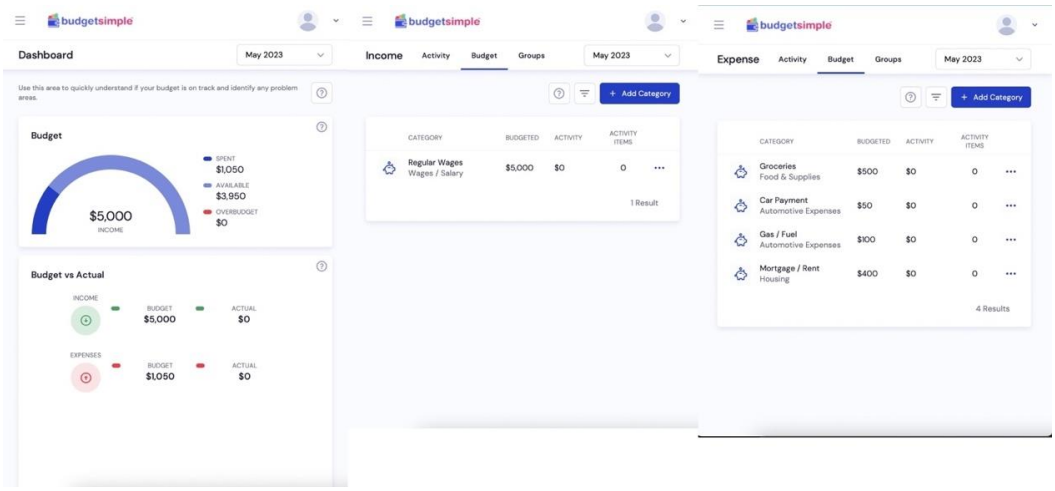


Рисунок 1.1 - Приклади інтерфейсу BudgetSimple

2. Empower:

   Переваги:

   - Загальний огляд фінансів: дозволяє підключити різні облікові записи, включаючи банківські рахунки, інвестиційні рахунки та пенсійні фонди, щоб забезпечити загальний огляд вашого фінансового стану.
   - Автоматичний імпорт даних: Додаток автоматично імпортує та категоризує ваші фінансові транзакції, що спрощує процес ведення обліку витрат і доходів.
   - Аналітика та інвестиційні інструменти: Empower надає засоби для аналізу витрат, планування пенсійного забезпечення, визначення цілей та аналізу інвестиційного портфеля.

Недоліки:

- Обмеження: неможливо почати використання без наявності банківського аккаунту з певного списку, який, в свою чергу, є доволі обмеженим.
- Не сповіщає користувачів перед зняттям підписки на сервіс
- Обмеження імпорту даних: Іноді може мати проблеми з точністю та повнотою імпортування фінансових даних з різних банків або фінансових установ. Це може призвести до потреби ручного внесення деяких транзакцій, щоб забезпечити точність вашого бюджету та звітності.

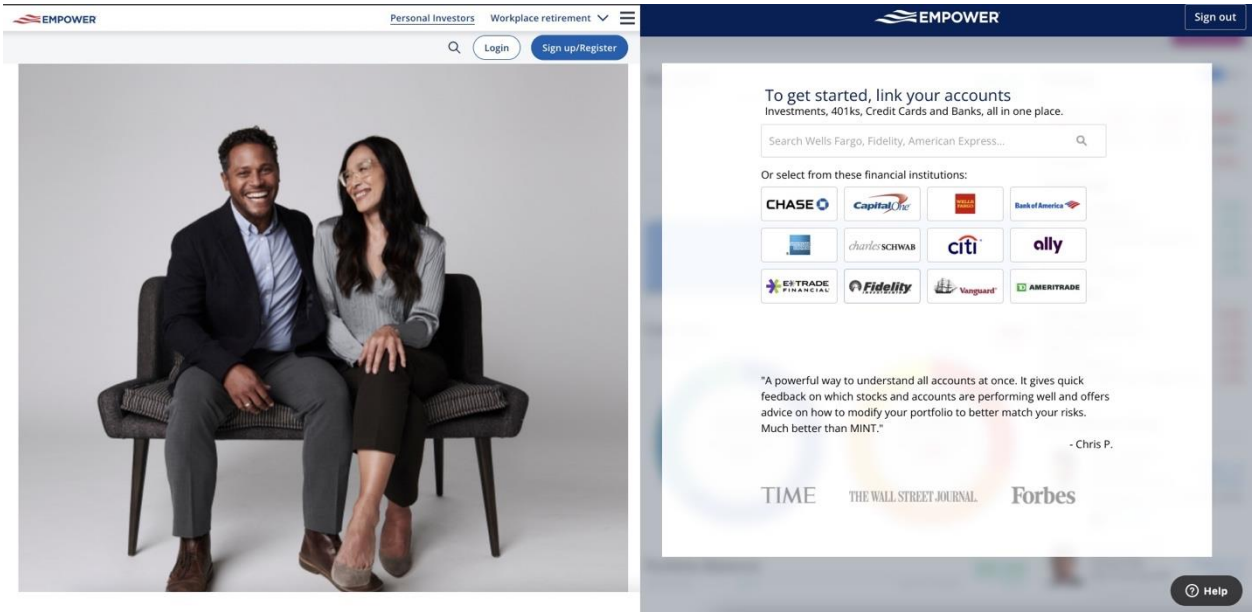


Рисунок 1.2 - Приклади інтерфейсу Empower

3. YNAB:

   Переваги:

   - Інтегрований підхід до бюджетування: YNAB пропонує систему "Кожен доллар має своє призначення", що допомагає вам активно керувати своїми витратами, сприяє зменшенню боргів та створенню ефективного бюджету.
   - Реалістичне планування: YNAB допомагає вам планувати витрати на майбутнє на основі доступних коштів, а не на основі передбаченого доходу. Це дозволяє більш реалістично управляти грошима та уникати надмірного використання кредиту.
   - Активна спільнота користувачів: YNAB має велику спільноту користувачів, яка активно обговорює стратегії бюджетування, надає поради та підтримку

Недоліки:

- Відсутність автоматичного імпорту даних: YNAB не надає автоматичний імпорт фінансових даних з банківських рахунків та кредитних карт. Користувачам потрібно вручну вводити та оновлювати інформацію про свої транзакції, що може бути часово витратним і не зручним.
- Платний підписний план: YNAB вимагає щомісячної або щорічної плати за користування своїми послугами(14 і 90 доларів відповідно). Для деяких користувачів це може бути не зручним або фінансово недоцільним, особливо якщо вони шукають безкоштовні альтернативи.

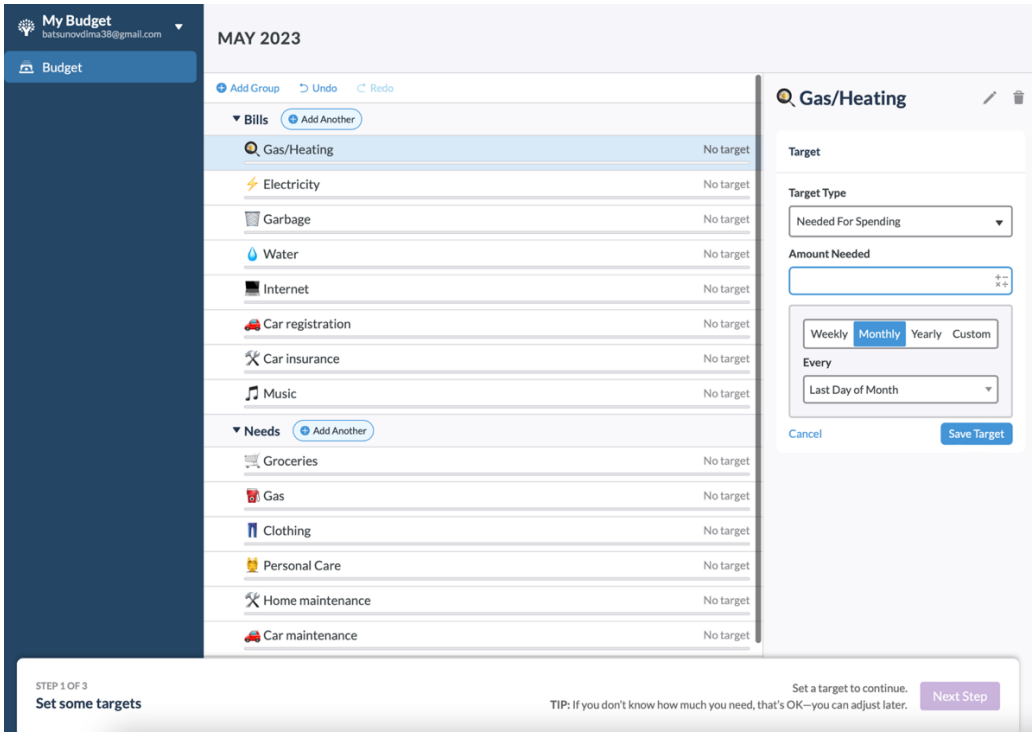


Рисунок 1.3 - Приклади інтерфейсу YNAB

Відповідно до вищезазначених характеристик була виведена результуюча таблиця(Таблиця 2.1) з інформаціє про наявність функціональних можливостей

Таблиця 2.1 – результуюча таблиця порівняння аналогів

Розглянуті додатки	BudgetSimple	Empower	Financial Geek (розроблений додаток)
Категоризація операцій	-	+	+
Інтеграція з банками	+	+	-
Перегляд статистики	+	+	+
Експорт статистики	-	-	+
Приблизне прогнозування витрат та доходів	-	-	+
Фінансові цілі	+	+	+
Фінансові рекомендації	-	-	+

Таблиця демонструє що наявність деякий функцій присутня тільки у розробленому додатку, та відсутня у аналогів. Наприклад: “Експорт статистичних даних”, “Приблизне прогнозування витрат та доходів” та “Фінансові рекомендації” саме ці особливості є ключовими в розробленому додатку

1.4 Визначення функціональних та нефункціональних вимог

Аналіз функціональних та нефункціональних вимог є необхідним етапом у процесі розробки системи, що вимагає уваги. Вимоги можуть охоплювати такі аспекти, як гнучкість масштабування, захист даних, інтуїтивно зрозумілий інтерфейс користувача, можливість створення звітів та аналітичних звітів, інтеграцію з іншими системами й багато іншого. Глибоке розуміння бізнес-процесів та особливостей фінансової діяльності людей допоможе виявити ключові вимоги та функціональні можливості системи.

Зважаючи на отримані результати дослідження та аналізу відповідної галузі, можна розробити план створення програмного забезпечення для власного фінансового менеджменту, використовуючи мову програмування Java + JavaScript та враховуючи виявлені особливості та вимоги.

1. Вхід в обліковий запис або реєстрація нового аккаунту
2. Ведення власного фінансового обліку :
    - Можливість додавати витрати.
    - Можливість додавати дохід.
    - Забезпечити варіант категоризації фінансової операції.
    - Додати функціонал по додаванню додаткового коментаря до операції
3. Статистика та попереднє прогнозування:
    - Система повинна нести в собі функціонал для перегляду всіх витрат та доходів.
    - Основуючись на попередньо введених даних, у випадку достатньої їх кількості, алгоритми повинні розраховувати приблизний прогноз фінансового становища на найближчий час.
4. Фінансові цілі:
    - Користувач повинен отримати можливість ставлення фінансових

цілей і йти до них.

5. Фінансові поради:
    - Реалізувати функціонал який буде в рандомному порядку видавати фінансові поради від людей які досягли фінансового успіху

2 ДОСЛІДЖЕННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ ТА АРХІТЕКТУРНИХ РІШЕНЬ ВЕБ ДОДАТКУ

2.1 Мови програмування

Мови програмування відіграють ключову роль у розробці програмного забезпечення і важливість правильного вибору мови не можна недооцінювати. Кожна мова має свої унікальні особливості, синтаксис, парадигми програмування і набір інструментів, що робить їх підходящими для певних завдань і проектів.

Один з ключових факторів, який впливає на вибір мови програмування, - це вимоги проекту. Також важливо враховувати величезну спільноту підтримки та ресурсів, пов'язаних з мовою програмування. Мови, які мають активну спільноту, часто надають доступ до багатої бази знань, бібліотек, фреймворків та інструментів, що полегшують розробку та забезпечують швидкий прогрес будь якого проекту.

2.1.1 Back-end

Мови серверного програмування так званий back-end відповідає за процесс обробки операцій на сервері та роботу з даними. Три популярні серверні мови програмування: Java, Python і Node.js. Кожна з них має свої унікальні особливості, плюси та мінуси. Однак Java виділяється як бажаний вибір для веб-розробки завдяки своїм винятковим функціям і розгалуженій екосистемі.

Java відома своєю стабільністю та масштабованістю, пропонує численні переваги. Його незалежність від платформи дозволяє розробникам написати код один раз і запустити його практично на будь-якій платформі, , що робить його надзвичайно універсальним. Java забезпечує надійну основу для створення

великомасштабних програм корпоративного рівня. Спільнота Java є великою та активною, пропонує вичерпну документацію, навчальні посібники та підтримку.

З іншого боку, Java має крутішу криву навчання порівняно з деякими іншими мовами через розширений синтаксис. Крім того, Java може бути більш докладним, вимагаючи більше рядків коду порівняно з іншими мовами але це дає можливість контролю над будь якою функцією яка в свою чергу саме завдяки великій кількості операцій відкрита до будь яких змін або виправлень.

Python, з іншого боку, відрізняється простотою та читабельністю, що полегшує написання та розуміння коду. Він широко використовується для завдань сценаріїв, автоматизації та аналізу даних. JavaScript (Node.js) — ще одна популярна серверна мова, яка використовує JavaScript на стороні сервера. Він пропонує чудову продуктивність для роботи з програмами в реальному часі та асинхронними операціями.

Однак домінування Java у веб-розробці є очевидним завдяки її надійності, масштабованості та розгалуженій екосистемі. Його незалежність від платформи дозволяє розробникам створювати програми, які можуть працювати на різних операційних системах, забезпечуючи широку сумісність. Крім того, фокус Java на продуктивності, безпеці та надійності робить її найкращим вибором для веб-додатків корпоративного рівня.

Підсумовуючи, універсальність, стабільність і розгалужена екосистема Java роблять її кращим вибором для веб-розробки. Його незалежність від платформи, масштабованість і широка підтримка спільноти сприяють його широкому застосуванню. Незалежно від того, створюєте ви невелику веб-програму чи складну систему корпоративного рівня, Java надає інструменти та ресурси, необхідні для створення потужних і надійних веб-рішень. Тому вибір для серверної мови програмування для додатку власного фінансового менеджменту пав саме на Java.

2.1.2 Front-end

На сьогодніщній день важко уявити програмування інтерфейсу користувача без мов програмування HTML, CSS і JavaScript. Кожна мова має свої унікальні характеристики, призначення та переваги, що робить їх основними інструментами для створення привабливих та інтерактивних веб-сайтів і веб-додатків.

HTML (мова розмітки гіпертексту). Він забезпечує структуру та вміст веб-сторінки, визначаючи різні елементи та їхні зв’язки. HTML простий у вивченні та широко підтримується всіма браузерами, що забезпечує сумісність на різних платформах. Це дозволяє створювати семантичні та доступні веб-сторінки, покращуючи пошукову оптимізацію та взаємодію з користувачем. Однак HTML сам по собі не може забезпечити складну інтерактивність і динамічну функціональність, він використовуються як база для подальшого наповнення.

CSS (каскадні таблиці стилів) був використанний для покращення презентації та візуальних аспектів веб-сайту. Це дозволило визначати кольори, шрифти та інші стилістичні елементи веб-сторінок. CSS забезпечує гнучкість і контроль над зовнішнім виглядом веб-сайту, забезпечуючи послідовне брендування та адаптивний дизайн. Він відокремлює візуальний рівень від рівня вмісту, що спрощує обслуговування та оновлення.

JavaScript — це потужна мова сценаріїв, яка додає веб-сторінкам інтерактивність і функціональність. Він забезпечує динамічний вміст, обробку подій, перевірку форм, анімацію та багато іншого. JavaScript широко підтримується всіма сучасними браузерами та має велику екосистему бібліотек і фреймворків, таких як React, Angular і Vue.js, які спрощують і прискорюють розробку. За допомогою JavaScript розробники можуть створювати адаптивний та інтерактивний веб-інтерфейс.

Поєднання HTML, CSS і JavaScript утворює потужне тріо для веб-розробки. HTML забезпечує структуру, CSS обробляє презентацію, а JavaScript додає інтерактивність і функціональність. Цей пакет пропонує комплексне та універсальне рішення для створення динамічних та привабливих веб-сайтів. Це дозволяє розробникам створювати адаптивний дизайн, налаштовувати користувацькі інтерфейси, обробляти взаємодію користувачів і спілкуватися з серверами для отримання та зберігання даних. Крім того, широка підтримка, величезні ресурси та постійний прогрес цих мов роблять їх ідеальним вибором для веб-розробки.

Підсумовуючи, поєднання HTML, CSS і JavaScript широко визнано найкращим рішенням для веб-розробки. Ці мови надають необхідні інструменти та можливості для створення візуально привабливих, інтерактивних та адаптивних веб-сайтів. Тому вибір мов для front-end пав саме на це тріо. Їхня універсальність, широка підтримка та велика спільнота гарантують, що розробники мають доступ до великої кількості ресурсів, фреймворків і бібліотек, що дозволяє їм створювати надійні та зручні веб-додатки. Будь то простий статичний веб-сайт чи складна веб-програма наприкладі власного фінансового менеджменту.

2.2 Фреймворки до обраних мов програмування

Фреймворки являє собою набір заздалегідь написаних і готових до використання програмних компонентів, які надають структуру та складні функціональні можливості для розробки програмного забезпечення. Вони надають розробникам готові інструменти та ресурси для спрощення процесу розробки, забезпечують стандартизацію та сприяють підвищенню продуктивності.

Основна мета фреймворків – забезпечити високий рівень повторного використання коду та спростити розробку програмного забезпечення, зокрема швидше і ефективніше створювати нові програми або розширювати існуючі. Вони надають готові рішення для типових завдань, таких як маршрутизація, бази даних, керування сесіями, автентифікація користувачів та багато інших.

2.2.1 Spring Java

Spring є доволі комплексною структурою, яка надає широкий спектр функцій і модулів для розробки програм Java. Він дотримується принципу інверсії управління (IoC) і впровадження залежностей, що робить його дуже гнучким і модульним. Spring пропонує чудову підтримку для створення програм корпоративного рівня, включаючи такі функції, як керування транзакціями, безпека, кешування та інтеграція з іншими фреймворками та технологіями. Він сприяє належним практикам розробки програмного забезпечення, таким як поділ проблем і слабкий зв’язок, що призводить до чистішого коду, який можна підтримувати. Крім того, Spring має активну спільноту та обширну документацію, що полегшує пошук допомоги та ресурсів. Серед різноманітних доступних підфреймворків Spring є Spring Boot, Spring Data JPA і Spring Security які виділяються як потужний набір стек технологій пропонуючи численні переваги перед своїми конкурентами які розкриті нижче.

2.2.2 Spring Boot Java

Spring Boot створений на основі фреймворку Spring, який спрямований на спрощення розробки програм Java. Він забезпечує конфігурації та автоматичне налаштування, що дозволяє розробникам швидко налаштовувати та розгортати програми з мінімальною конфігурацією.

Однією з ключових переваг Spring Boot є його здатність ефективно керувати залежностями. Він використовує концепцію під назвою «початкові залежності», які є попередньо налаштованими наборами залежностей для типових сценаріїв застосування. Ці початкові залежності постачаються з вирішеними версіями, що гарантує, що всі бібліотеки та компоненти програми мають сумісні версії. Це позбавляє розробників від необхідності вручну керувати залежностями та зменшує ймовірність конфліктів версій між різними бібліотеками. За допомогою Spring Boot розробники можуть легко включити необхідні залежності у свій проект, просто додавши відповідну початкову залежність.

Крім того, Spring Boot містить вбудований сервер Apache Tomcat, який дозволяє розробникам запускати свої програми як окремі виконувані файли. Це позбавляє від необхідності налаштування та налаштування окремого веб-сервера для розгортання. Вбудований сервер забезпечує зручне та легке рішення для запуску програм Spring Boot, що полегшує їх розробку, тестування та розгортання. Саме завдяки зазначеним вище перевагам було прийнято рішення використовувати Spring Boot для керування залежностями та запуску серверу

```java
@SpringBootApplication
public class DiplomaApplication {

    public static void main(String[] args) {
        SpringApplication.run(DiplomaApplication.class, args);
    }
}
```

Рисунок 1.2 - Фрагмент коду запуску додатку за допомогою Spring Boot

2.2.3 Spring Data JPA

Spring Data JPA — це потужний рівень абстракції, створений на основі JPA, який спрощує доступ до бази даних і збереження для програм Java. Він забезпечує зручний і узгоджений спосіб обробки відображення між об’єктами Java і таблицями реляційної бази даних.

Однією з ключових особливостей Spring Data JPA є підтримка відображення сутностей. Це дозволяє розробникам визначати класи сутностей, які представляють таблиці бази даних, і встановлювати зв’язки між ними. За допомогою анотацій, таких як @Entity, @Table і @Column, розробники можуть легко зіставляти класи Java із відповідними структурами баз даних. Ця абстракція зменшує потребу в написанні шаблонного коду для відображення та обробки операцій бази даних, роблячи розробку більш ефективною.

З точки зору пошуку даних і обробки асоціацій, Spring Data JPA пропонує підтримку лінивої та нетерплячої ініціалізації. Відкладене завантаження дозволяє відкладене отримання пов’язаних сутностей, що може допомогти підвищити продуктивність, завантажуючи дані лише за потреби. З іншого боку, швидке завантаження отримує пов’язані сутності разом із основною сутністю, гарантуючи, що всі пов’язані дані завантажуються одночасно. Вибір між відкладеним і нетерплячим завантаженням залежить від конкретного випадку використання та вимог до продуктивності програми.

Підсумовуючи, Spring Data JPA є цінним інструментом для спрощення доступу до бази даних і збереження в програмах Java. Він керує зіставленням між об’єктами Java і таблицями бази даних, підтримує відкладену та активну ініціалізацію, пропонує каскадні операції для керування асоціаціями та забезпечує автоматичне створення запитів SQL. Його інтеграція з екосистемою Spring і абстракція вищого рівня роблять його значно кращим вибором у порівнянні з Hibernate, дозволяючи розробникам більше зосереджуватися на

логіці програми, а не на низькорівневих операціях з базою даних.

Нижче наведений фрагмент коду для генерування запиту до бази даних задля отримання користувача яка явно вказує на простоту використання і легкість в роботі. Генерація SQL запитів повністю делегується фреймоврку.

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {

    Optional<User> findByUsername(String username);
}
```

Рисунок 2.2 - Фрагмент коду створення класу для запитів в базу даних

2.2.4 Spring Security

Коли справа доходить до аутентифікації у веб-додатках, існує два поширених підходи: автентифікація на основі сеансу та автентифікація на основі JWT. Обидва мають свої переваги, і вибір залежить від різних факторів архітектурм програми та вимог.

Аутентифікація на основі сеансу, яка зазвичай використовується в монолітних архітектурах, передбачає створення сеансу сервером для кожного аутентифікованого користувача. Сеанс зазвичай зберігається на стороні сервера, а ідентифікатор сеансу зберігається на стороні клієнта як файл cookie. З кожним наступним запитом сервер перевіряє ідентифікатор сесії для автентифікації та авторизації користувача.

Однією з переваг аутентифікації на основі сеансу є її простота та легкість використання в монолітних архітектурах. Оскільки всі запити обробляються

одним сервером, керування сеансами стає простим. Сервер може підтримувати стан сеансу та виконувати завдання, пов’язані із сеансом, наприклад закінчення терміну дії сеансу та визнання його недійсним.

Крім того, автентифікація на основі сеансу забезпечує вищий рівень безпеки, оскільки ідентифікатори сеансу надійно зберігаються на стороні сервера. Це зменшує ризик викрадення або втручання маркерів, які можуть виникнути з автентифікацією на основі JWT, якщо маркери скомпрометовані.

З іншого боку, автентифікація на основі JWT набула популярності з розвитком архітектур мікросервісів і API без збереження стану. У цьому підході сервер генерує веб-токен JSON (JWT) після успішної автентифікації, який потім надсилається клієнту та включається в наступні запити як заголовок авторизації. Сервер перевіряє підпис маркера для автентифікації та авторизації користувача.

Аутентифікація на основі JWT пропонує кілька переваг. Це забезпечує автентифікацію без стану, тобто серверу не потрібно зберігати дані сеансу або підтримувати стан сеансу. Це робить його масштабованим і придатним для розподілених систем. Крім того, JWT можуть містити спеціальні претензії, що дозволяє здійснювати гнучку та детальну авторизацію. Вони також полегшують інтеграцію зі сторонніми службами та сценаріями єдиного входу (SSO).

Тепер давайте обговоримо, чому Spring Security вважається найкращим рішенням для безпеки веб-додатків. Spring Security — це потужна та комплексна система безпеки для додатків Java. Він надає широкий спектр функцій і функцій для захисту програм від типових загроз безпеці.

Підсумовуючи, Spring Security вважається найкращим рішенням для безпеки веб-додатків завдяки бездоганній інтеграції з екосистемою Spring, комплексним функціям і дотриманню найкращих практик. Незалежно від того, чи це реалізує автентифікацію на основі сеансу для монолітних архітектур або автентифікацію

на основі JWT для розподілених систем, Spring Security надає необхідні інструменти та можливості для ефективного та ефективного захисту програм.

```java
@Service
public class UserDetailServiceImpl implements UserDetailsService {

    private final UserRepository userRepository;

    public UserDetailServiceImpl(UserRepository userRepository) { this.userRepository = userRepository; }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepository.findByUsername(username)
                .orElseThrow(() -> new UsernameNotFoundException("User not found username:" + username));

        return org.springframework.security.core.userdetails.User
                .withUsername(user.getUsername())
                .password(user.getPassword())
                .disabled(!user.isEnabled())
                .roles("USER")
                .build();
    }
}
```

Рисунок 2.3 - Фрагмент коду авторизації користувача

У додатку власного фінансового менеджменту використана базова аутентифікація на основі записів у базі данних. Тобто коли юзер робить запит на сторінку на яку потрібна авторизація программа робить запит до бази даних задля перевірки наявновсті юзера на основі його юзернейму, і відповідно у випадку якщо такого юзера нема то буде викинута помилка з повідомленням що запитунового юзера не існує. А якщо такий юзер існує - виконается ініціалізація та подальше зберігання сессії з такими параметрами: username, password, role, isAccountDisabled яка у подальшому буде зберігатись на сервері.

2.2.5 JavaScript frameworks

Фреймворки JavaScript набули значної популярності у веб-розробці завдяки своїй здатності спрощувати та оптимізувати процес розробки. Вони надають

розробникам готові до використання компоненти, ефективне зв’язування даних і підвищену продуктивність. Однак варто зазначити, що чистий JavaScript залишається простим і хорошим рішенням, особливо для монолітних архітектур.

Чистий JavaScript, не покладаючись на будь-яку конкретну структуру, дозволяє розробникам мати повний контроль над поведінкою та продуктивністю програми. Він забезпечує легкий і гнучкий підхід, що дозволяє розробникам адаптувати код відповідно до своїх конкретних потреб. За допомогою чистого JavaScript розробники мають прямий доступ до API браузера, маніпулювання DOM і низькорівневих операцій, що може бути вигідним у певних випадках використання.

У монолітних архітектурах, де вся програма розгортається як єдине ціле, складність і накладні витрати, які вводять фреймворки JavaScript, можуть не знадобитися. Оскільки програма не розподілена між кількома службами або мікросервісами, немає потреби у важкій структурі, яка керує складним керуванням станом або зв’язком компонентів.

Більше того, використовуючи чистий JavaScript, розробники можуть оптимізувати продуктивність і зменшити обсяг програми. Фреймворки JavaScript часто постачаються з додатковими залежностями та кодом, які можуть не знадобитися в монолітній архітектурі. Використовуючи чистий JavaScript, розробники мають повний контроль над розміром коду та можуть оптимізувати його для кращої продуктивності.

З огляду на це, фреймворки JavaScript дійсно пропонують переваги в певних сценаріях, особливо для складніших програм або тих, які вимагають розширеної взаємодії інтерфейсу користувача, керування станом або повторного використання компонентів. Такі фреймворки, як React, Vue.js або Angular, надають потужні абстракції та підтримку екосистеми для створення

масштабованих і підтримуваних програм. Саме тому для розробки веб додатку власного менеджменту було використано чистий JS

2.2.6 Інструменти розробки

IntelliJ IDEA та WebStorm — це два популярні інструменти розробки, які дуже допомагають у розробці веб-додатків. Обидва інструменти розроблені JetBrains і пропонують потужні функції та зручний інтерфейс для ефективного кодування.

IntelliJ IDEA — це інтегроване середовище розробки (IDE), яке в основному використовується для розробки на Java, але воно також підтримує інші мови, такі як JavaScript, HTML, CSS тощо. Він надає широкий спектр функцій, таких як інтелектуальне завершення коду, навігація по коду, налагодження та інтеграція контролю версій. Надійна підтримка таких фреймворків, як Spring, Hibernate і Maven, робить його кращим вибором для розробників Java. Крім того, IntelliJ IDEA пропонує велику бібліотеку плагінів, які розширюють її функціональність і дозволяють розробникам налаштовувати свій робочий процес.

З іншого боку, WebStorm — це спеціалізована IDE, спеціально розроблена для веб-розробки. Він зосереджений на JavaScript, HTML і CSS, надаючи розширені функції, адаптовані до веб-технологій. WebStorm пропонує інтелектуальне завершення коду, аналіз коду в реальному часі, потужні інструменти рефакторингу та повну інтеграцію з такими популярними веб-фреймворками, як React, Angular і Vue.js. Він також підтримує сучасні практики веб-розробки, такі як препроцесори CSS і програми запуску завдань, що робить його комплексним інструментом для веб-розробників.

DBeaver — це універсальний інструмент для баз даних, який дозволяє розробникам підключатися до різних систем керування базами даних (СУБД) і виконувати завдання, пов’язані з базами даних. Він підтримує широкий спектр

систем баз даних, включаючи MySQL, PostgreSQL, Oracle, SQL Server тощо. DBeaver надає зручний інтерфейс для виконання SQL-запитів, керування підключеннями до бази даних, вивчення схем баз даних і візуалізації даних. Він також пропонує такі функції, як імпорт/експорт даних, міграція схем і моделювання бази даних, що робить його цінним інструментом для розробки та адміністрування баз даних.

Підсумовуючи, IntelliJ IDEA та WebStorm є інструментами розробки, які підвищують продуктивність розробників, пропонуючи розширені функції та чудову підтримку для різних мов програмування та фреймворків. DBeaver, з іншого боку, служить універсальним інструментом для баз даних, що дозволяє розробникам підключатися до різних систем баз даних і ефективно керувати завданнями, пов’язаними з базами даних. Комбінація цих інструментів надає розробникам комплексне та ефективне середовище для веб-розробки та керування базами даних, значно полегшуючи процес розробки та забезпечуючи доставку високоякісних програм.

2.2.7 Бази даних

Серед популярних систем керування реляційними базами даних, таких як MySQL, Oracle і SQL Server, PostgreSQL виділяється як потужне та універсальне рішення, яке пропонує ряд переваг для проектів веб-розробки. PostgreSQL, також відома як Postgres, — це багатофункціональна система баз даних із відкритим вихідним кодом, яка приділяє велику увагу відповідності стандартам і можливості розширення.

Однією з ключових переваг PostgreSQL є його розширена функціональність і підтримка складних типів даних. Він надає широкий спектр типів даних, включаючи JSON, масиви, геометричні типи та повнотекстовий пошук, що дозволяє розробникам ефективно зберігати та запитувати різноманітні та

спеціалізовані дані. Крім того, PostgreSQL пропонує надійну підтримку просторових даних, що робить його чудовим вибором для додатків, яким потрібні можливості географічної інформаційної системи.

Іншою значною перевагою PostgreSQL є підтримка транзакцій, сумісна з ACID яка забезпечує цілісність і надійність даних. Це дозволяє розробникам виконувати складні операції в межах транзакцій, забезпечуючи послідовність і довговічність модифікацій бази даних навіть за наявності збоїв. Це робить PostgreSQL придатним для програм, які потребують високої узгодженості та надійності даних.

PostgreSQL також вирізняється підтримкою одночасного доступу та масштабованістю. Він надає розширені механізми керування паралелізмом, включаючи багатоверсійний контроль паралелізму, який дозволяє кільком транзакціям отримувати доступ до бази даних одночасно, не блокуючи одна одну. Це забезпечує високопродуктивні та масштабовані програми, що робить PostgreSQL чудовим вибором для обробки одночасних запитів користувачів у веб-середовищі.

Крім того, PostgreSQL має активну та активну спільноту з відкритим вихідним кодом, яка постійно робить внесок у його розвиток, покращення та безпеку. Природа PostgreSQL, керована спільнотою, забезпечує регулярне оновлення, виправлення помилок і доступність широкого спектру розширень і плагінів для покращення його функціональності.

PostgreSQL стає чудовим вибором для проектів веб-розробки завдяки розширеній функціональності, підтримці складних типів даних, сумісним з ACID транзакціям, механізмам керування паралелізмом, масштабованості та активній спільноті, яка підтримує це. Ці функції роблять PostgreSQL надійною та потужною системою баз даних, яка добре підходить для обробки різноманітних вимог до даних і забезпечує надійні, масштабовані та високопродуктивні веб-

додатки. Саме тому у веб додатку власного фінансового менеджменту була використана PostgreSQL.

2.2.8 Архітектурні рішення побудови веб додатків

При побудові веб-додатків існує кілька архітектурних рішень, залежно від потреб проєкту і специфіки застосування. Деякі з найпоширеніших архітектурних рішень включають монолітна архітектура та мікросервісна
Монолітна архітектура є традиційним підходом до розробки додатків, де весь додаток будується як єдине ціле, недільне на окремі компоненти. Зазвичай вона включає інтерфейс користувача на стороні клієнта, програму на стороні сервера і базу даних. Всі функції та обробка даних зосереджені в одному місці.

Монолітні програми мають зазвичай одну велику кодову базу і не мають модульності. Якщо розробники бажають зробити оновлення або зміни, вони мають доступ до цієї ж кодової бази. Таким чином, будь-які зміни вносяться до всього додатку одночасно і відповідно зміни на back-end та front-end сторони в більшості випадків виконуються одним full-stack розробником. Таким чином ми прискорюємо делівірі будь якого функціоналу або виправлення та замість двох розробників ми маємо можливість обмежитись одним.

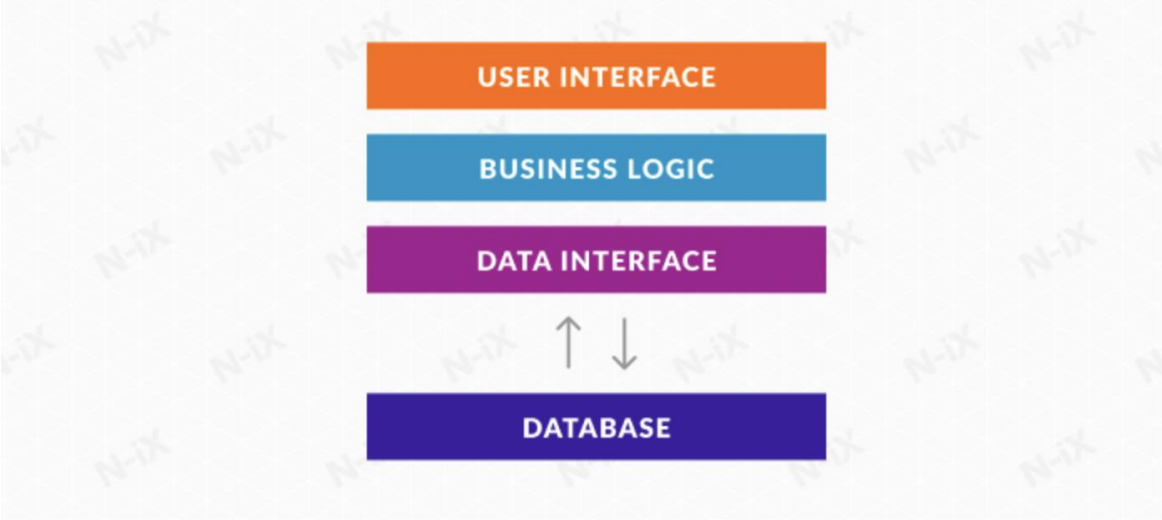


Рисунок 2.4 - Монолітна архітектура

Насупереч монолітній архітектурі, архітектура мікросервісів розбиває додаток на набір менших і незалежних одиниць. Кожен мікросервіс виконує окремі запити як самостійну послугу. У такій архітектурі кожен сервіс має свою власну логіку, базу даних і виконує конкретні функції.

Архітектура мікросервісів розбиває функціональність на модулі, які можна розгортати незалежно один від одного. Ці модулі взаємодіють один з одним за допомогою визначених методів, таких як API або брокери повідомлень.

Кожен сервіс охоплює певну область функціональності і може бути незалежно оновлений, розгорнутий та масштабований. Це дає багато переваг у випадку коли повина будуватись велика, вертикально та горизонтально масштабована система, а у замовника є достатні кошти і час для фінансування такого підходу до розробки. Оскільки в більшості випадках один розробник не може виконати зміни на обох сторонах саму у випадку мікросервісної архітектури, звідси витікає необхідність виділенню додаткового часу на коммунікацію між розробниками front-end та back-end сторони.

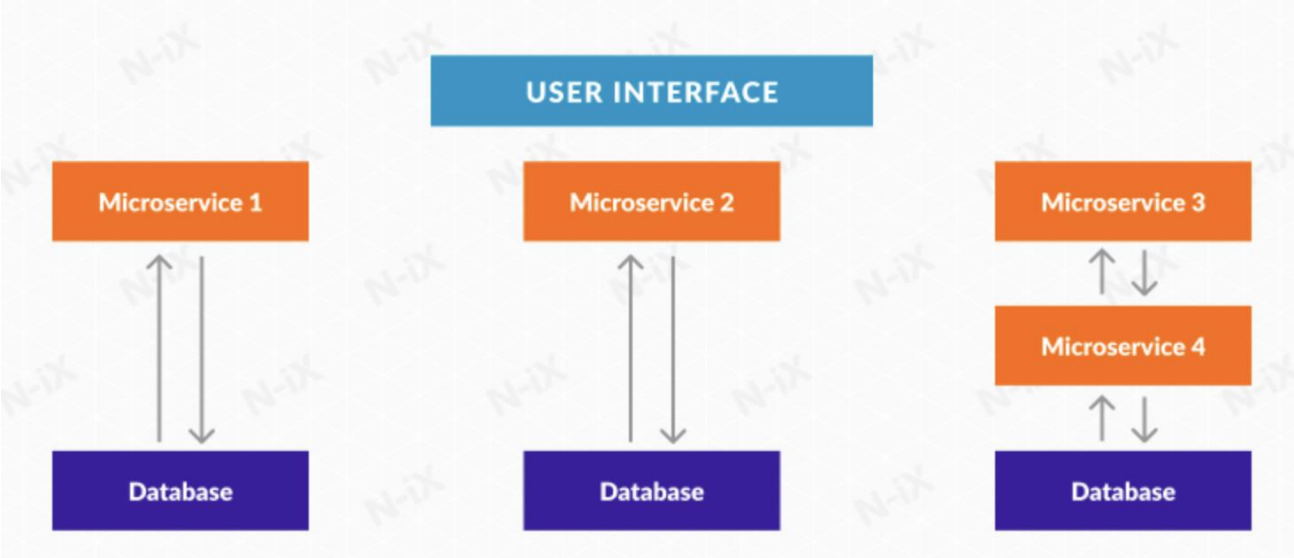


Рисунок 2.5 - Мікросервісна архітектура

Більше того, у випадку саме мікросервісної архітектури, доволі часто, зміни вносяться в 2-5 сервісів які треба окремо запускати локально для тесуваня змін і оновлювати кожен сервіс окремо. Тому все залежить від проекту і мети продукту, якщо головна мета - можливість швидкого внесення змін і делівері end-to-end функціоналу якнайшвидше - у цьому випадку перевагу надають монолітній архітектурі.

Загалом, монолітна архітектура є доволі розповсюдженим рішенням особливо коли розробляється перша версія продукту яку треба реалізувати за короткий проміжок часу. Мікросервісна архітектура дає можливість вертикального масштабування та ізольоване оновлення окремих модулів. Оскільки розробка ПЗ для особистого фінансового менеджменту несе в собі характер стартапу з обмеженими ресурсами - рішенням було використання саме монолітної архітектури.

3 РОЗРОБКА ДОДАТКУ ДЛЯ ВЛАСНОГО ФІНАНСОВОГО МЕНЕДЖМЕНТУ

3.1 Опис етапів розробки ПЗ

Розробка програмного забезпечення, зокрема веб-додатку для власного фінансового менеджменту включає кілька етапів:

1. Аналіз вимог: все почалось з детального аналізу наявних аналогів на ринку. Встановлюються функціональність, яку має виконувати веб-додаток, такі як ведення бюджету, відстеження витрат, створення звітів тощо. Також важливим є розуміння чим саме додаток відрізняється від вже існуючих і чому користувачі повинні почати використання саме розробленого додатку.
2. Проектування системи: на цьому етапі розроблялась архітектура веб-додатку для фінансового менеджменту. Визначились компоненти системи, взаємодія між ними та логіка обробки даних. Також дизайн бази даних для зберігання фінансових транзакцій та інших даних.
3. Реалізація: на цьому етапі ми почали написання коду. Використали Java, JS, HTML, CSS для реалізації функціональності, такої як додавання та видалення фінансових транзакцій, обчислення загального балансу тощо. Забезпечили інтеграція з базою даних та створення необхідних API для взаємодії з фронтендом.
4. Тестування: ретельне мануальне тестування веб-додатку. Мануальне тестування дозволяє виявляти дефекти, помилки та недоліки в програмному продукті. Тестер працює з програмою як кінцевий користувач і перевіряє, чи відповідає вона очікуванням та функціональним вимогам.

3.2 Моделювання додатку засобами UML

Моделювання додатку з використанням UML (Unified Modeling Language) є інструментом для візуалізації та проектування програмного забезпечення. UML надає стандартизовану нотацію та набір діаграм, які дозволяють представляти різні аспекти системи, їх взаємозв'язок та поведінку. Основні переваги моделювання засобами UML включають:

1. Візуалізація архітектури. За допомогою UML можна створювати діаграми, такі як діаграми класів, компонентів та пакетів, що дозволяють візуально представити архітектуру системи. Це допомагає розробникам та зацікавленим сторонам отримати загальне уявлення про структуру додатку.
2. Уточнення вимог. UML надає діаграми вимог, такі як діаграми варіантів використання, що допомагають уточнити та візуалізувати функціональні вимоги до системи. Це полегшує комунікацію з клієнтом та іншими зацікавленими сторонами.
3. Моделювання поведінки. UML має діаграми послідовностей, діаграми станів та інші діаграми, що дозволяють моделювати поведінку системи. Це допомагає зрозуміти, як взаємодіють компоненти системи та як вона змінює свій стан відповідно до подій.
4. Документація. Моделі UML можуть слугувати як документація для проекту. Вони надають зручний спосіб описати структуру та поведінку системи, що допомагає розробникам та іншим учасникам проекту легше розуміти код та зв'язки між його складовими.
5. Підтримка проектування: Використання UML дозволяє розробникам працювати на вищому рівні абстракції, проектуючи систему перед переходом до деталей реалізації. Це допомагає уникнути помилок та покращує розуміння взаємозв'язків компонентів системи.

Загалом, UML є корисним інструментом для моделювання додатків, оскільки він надає стандартизовану нотацію та діаграми для візуалізації архітектури, поведінки та вимог системи. Використання UML допомагає покращити якість проекту та допомогти відповісти на багато питань ще до початку розробки. Тому для ПЗ власного фінансового менеджменту ще до розробки було побудовано діаграми варіантів використання та діаграму діяльності і тільки після їх побудови була розпочата розробка.

3.3 Діаграми варіантів використання

Діаграма варіантів використання (Use Case Diagram) є одним з головних типів діаграм в аналізі та проектуванні програмного забезпечення. Вона використовується для моделювання функціональних вимог системи з точки зору її користувачів.

Діаграма варіантів використання надає високорівневий огляд процесів та взаємодії між користувачами (акторами) та системою. Вона допомагає зрозуміти, як користувачі взаємодіють з системою та які функціональні можливості системи вони використовують.

Основна мета діаграми варіантів використання - це відображення "випадків використання" або сценаріїв, які представляють конкретні дії та поведінку користувачів системи. Кожен випадок використання (use case) описує конкретну функцію, яку система здатна виконати, і може мати своїх акторів, які сприймають інформацію від системи або взаємодіють з нею.

Основні переваги використання діаграми варіантів використання:

1. Комунікація: Вона допомагає узгодити розуміння між розробниками програмного забезпечення та клієнтами, що стосується функціональних вимог системи.

2. Аналіз та проектування: Діаграма варіантів використання служить основою для подальшого аналізу, проектування системи та визначення вимог до неї.
3. Виявлення вимог: Дозволяє виявити та документувати вимоги до системи з точки зору користувачів.
4. Визначення тестових сценаріїв: Діаграма варіантів використання може служити основою для розробки тестових сценаріїв та перевірки відповідності системи вимогам користувачів.

Загалом, діаграма варіантів використання є потужним інструментом для моделювання функціональних вимог системи та визначення способів взаємодії між користувачами та системою. Вона допомагає зрозуміти, як система повинна працювати з точки зору кінцевих користувачів і створює основу для подальшого аналізу та проектування системи.

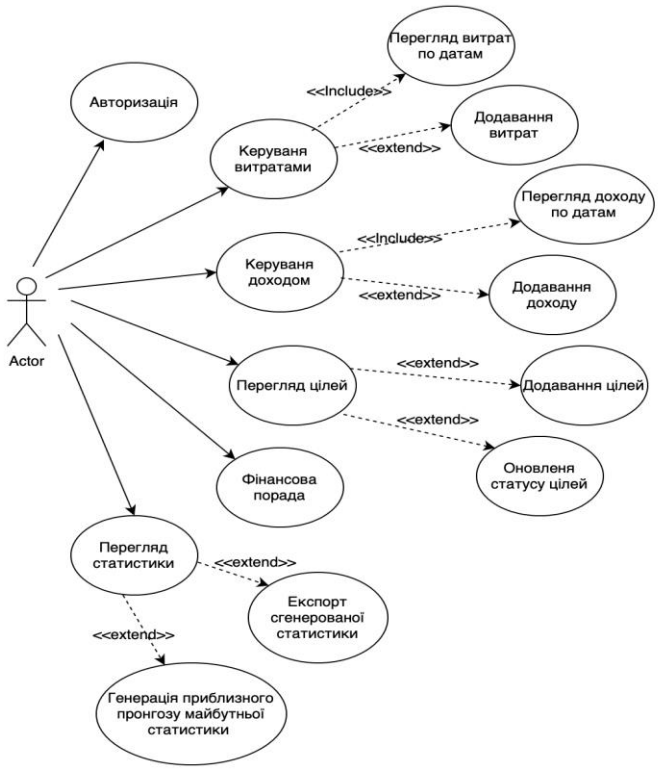


Рисунок 3.1 - Діаграма варіантів використання

3.4 Проектування інтерфейсу користувача для власного фінансового менеджменту

Проведено проектування інтерфейсу користувача для власного фінансового менеджменту. Головною метою проектування інтерфейсу було створення зручного та ефективного інструменту для управління фінансами користувачів.
Процес проектування інтерфейсу базувалось на аналізі потреб. Були проведені дослідження щоб з'ясувати, які функціональні можливості має містити програмний продукт, які елементи інтерфейсу будуть найбільш доцільними для користувачів, а також які вимоги до безпеки та конфіденційності даних є найважливішими.

На основі результатів аналізу було розроблено концепцію інтерфейсу, включаючи його структуру, компоненти та взаємодію з користувачем. Були розроблені макети інтерфейсу, що відображали розташування елементів, кольорову палітру, типографіку та інші важливі аспекти.
У процесі проектування була надана особлива увага забезпеченню простоти та зрозумілості інтерфейсу. Важливим було створення логічної структури, зручної навігації та інтуїтивно зрозумілих елементів керування. Користувачі мали мати змогу швидко зорієнтуватися у функціональних можливостях програмного продукту та здійснювати необхідні операції з мінімальними зусиллями.

Окрім того, було враховано адаптивність інтерфейсу для різних пристроїв та розмірів екранів. Дизайн повинен бути привабливим і функціональним як на настільних комп'ютерах, так і на мобільних пристроях.
Усі аспекти проектування інтерфейсу базувалися на сучасних принципах дизайну і враховували потреби користувачів. Були використані стандартні практики і рекомендації для створення зручного та ефективного інтерфейсу, що

максимально задовольняє потреби користувачів у власному фінансовому менеджменті.

Перша сторінка інтерфейсу яку бачить користувач відображена на Рисунку 3.2.

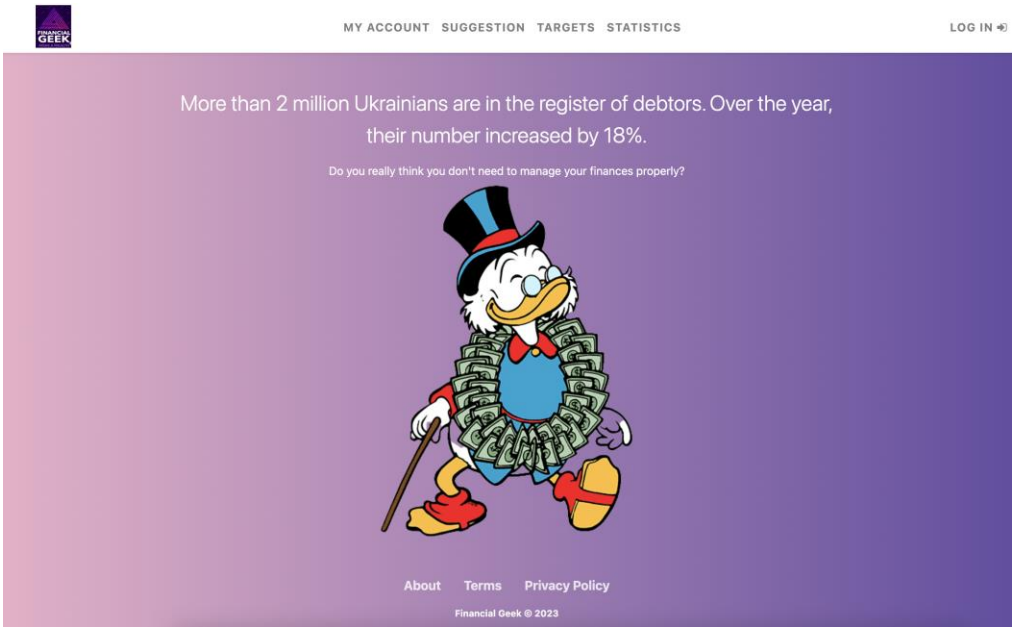


Рисунок 3.2 - Головна сторінка

У випадку якщо користувач переходить на сторінку Авторизації яка зображена на Рисунку 3.3, сторінка відображає форму вводу логіну та паролю. Якщо користувач переходить на сторінку власного аккаунту, цілей або статистики система автоматично переспрямує запит на сторінку авторизації і тільки у випадку ведення правильного логіну та паролю користувач буде можливість користуватись вищевказаним функціоналом.

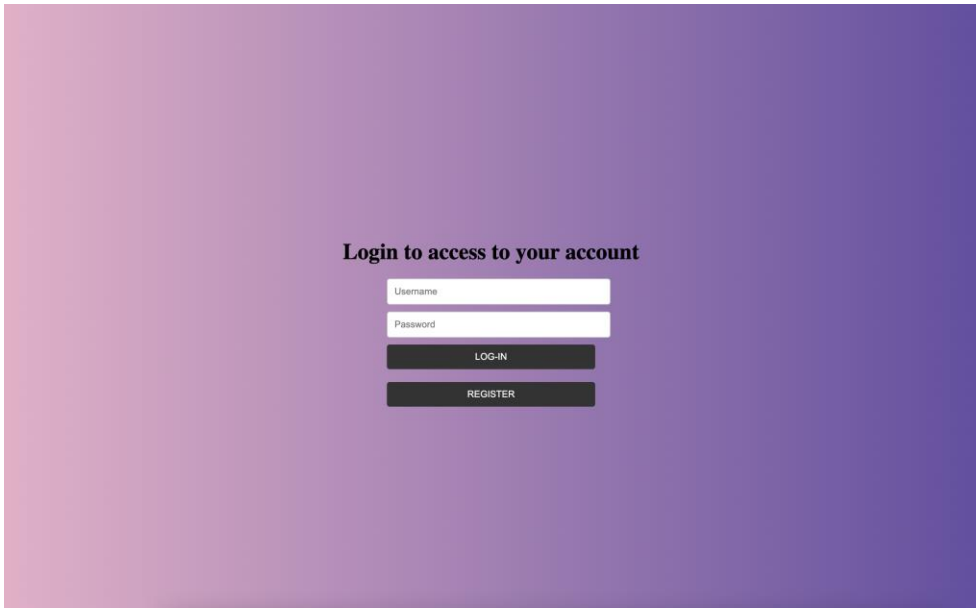


Рисунок 3.3 - Сторінка авторизації

Після успішної авторизації користувач буде переспрямований на головну сторінку яка зображена на Рисунку 3.2 з можливістю вільного користування всім наявним функціоналом. Наприклад на Рисунку 3.4 зображено додавання доходу користувачем.

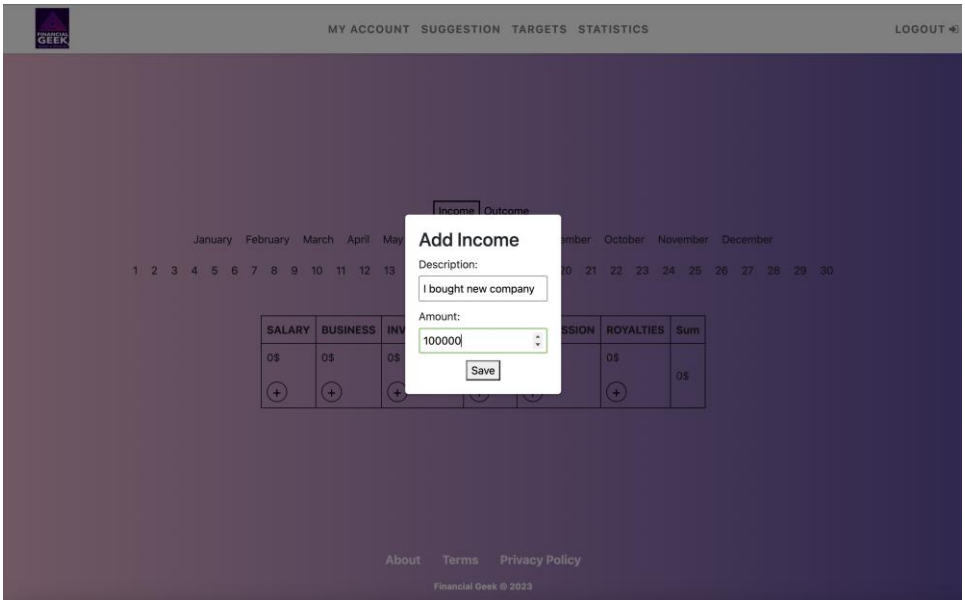


Рисунок 3.4 - Додавання доходу або витрати

Після введення достатньої кількості даних для перегляду статистики користувач має можливість перегляду статистики з подальшим експортом у різні формати або повномасштабним переглядом діаграми приклад якої наведений на Рисунку 3.5

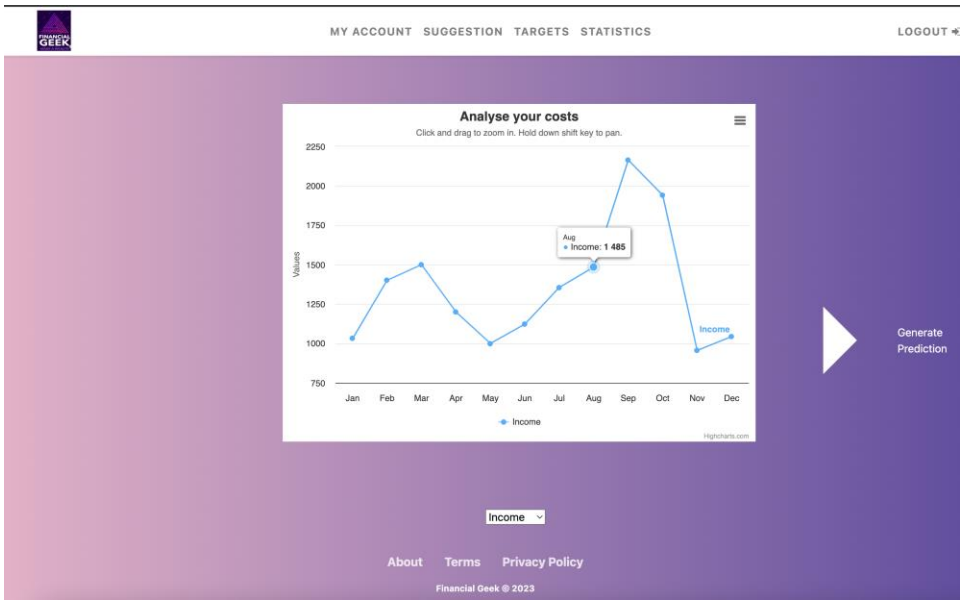


Рисунок 3.5 - Перегляд статистичних даних

Результатом проектування є створений інтерфейс користувача, який дозволяє зручно та ефективно управляти фінансами. Він включає в себе необхідні функції, інтуїтивно зрозумілі елементи керування та забезпечує безпеку та конфіденційність даних користувачів.

В подальшому, розроблений інтерфейс був взятий за основу для реалізації програмного продукту для власного фінансового менеджменту, що допоможе спростити контролювати та планувати свої фінансові ресурси.

3.5 Діаграма діяльності

Діаграма діяльності - це графічний інструмент моделювання, який використовується для візуалізації послідовності дій або процесів в системі, проекту або бізнес-процесу. Вона дозволяє детально описати кроки, дії та умови, які відбуваються під час виконання певного процесу або задачі. На Рисунку 3.6 спроектована діаграма діяльності

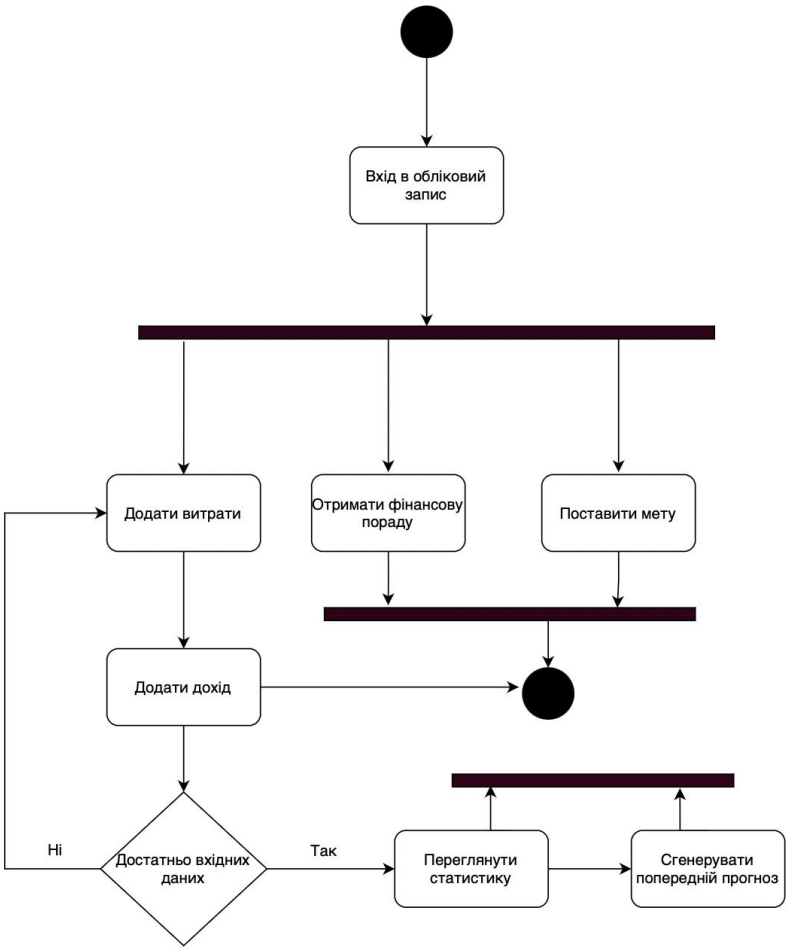


Рисунок 3.6 - Діаграма діяльності

3.6 Діаграма класів

Одним з основних видів діаграм в об'єктно-орієнтованому моделюванні є діаграма класів. Вона використовується для візуалізації класів, їх атрибутів та взаємозв'язків між ними. Основна мета діаграми класів полягає в моделюванні структури програмного коду або системи. На Рисунку 3.7 відображена остаточна діаграма класів для розробленого додатку власного фінансового менеджменту

Рисунок 3.7 - Діаграма класів

З лівої сторони Model – класи які розроблені для маніпулювання з необхідними для фінансового менеджменту даними. Посередині сервісні класи які розроблені для імплементації бізнес логікі. Вони в свою чергу мають зв'язок ассоціації с класами – контролерами, які розраховані на обробку запиту користувача і поверненню результату. Як можна бачити на діаграмі було використано суміш звичайних контролерів які повертають репрезентацію та REST API які допомагають з маніпуляцією даними.

3.7 Діаграма баз даних

Діаграма баз даних являє собою графічне зображення структури бази даних, що використовується для візуалізації взаємозв'язків між таблицями, атрибутами та їх типами даних. Вона надає візуальне подання схеми бази даних, де зображені таблиці, їх структура та зв'язки між ними.

Головною метою діаграми баз даних є уявлення логічної або фізичної структури бази даних перед її реалізацією. Вона допомагає розробникам програмного забезпечення та аналітикам зрозуміти, як дані організовані і як вони пов'язані між собою.

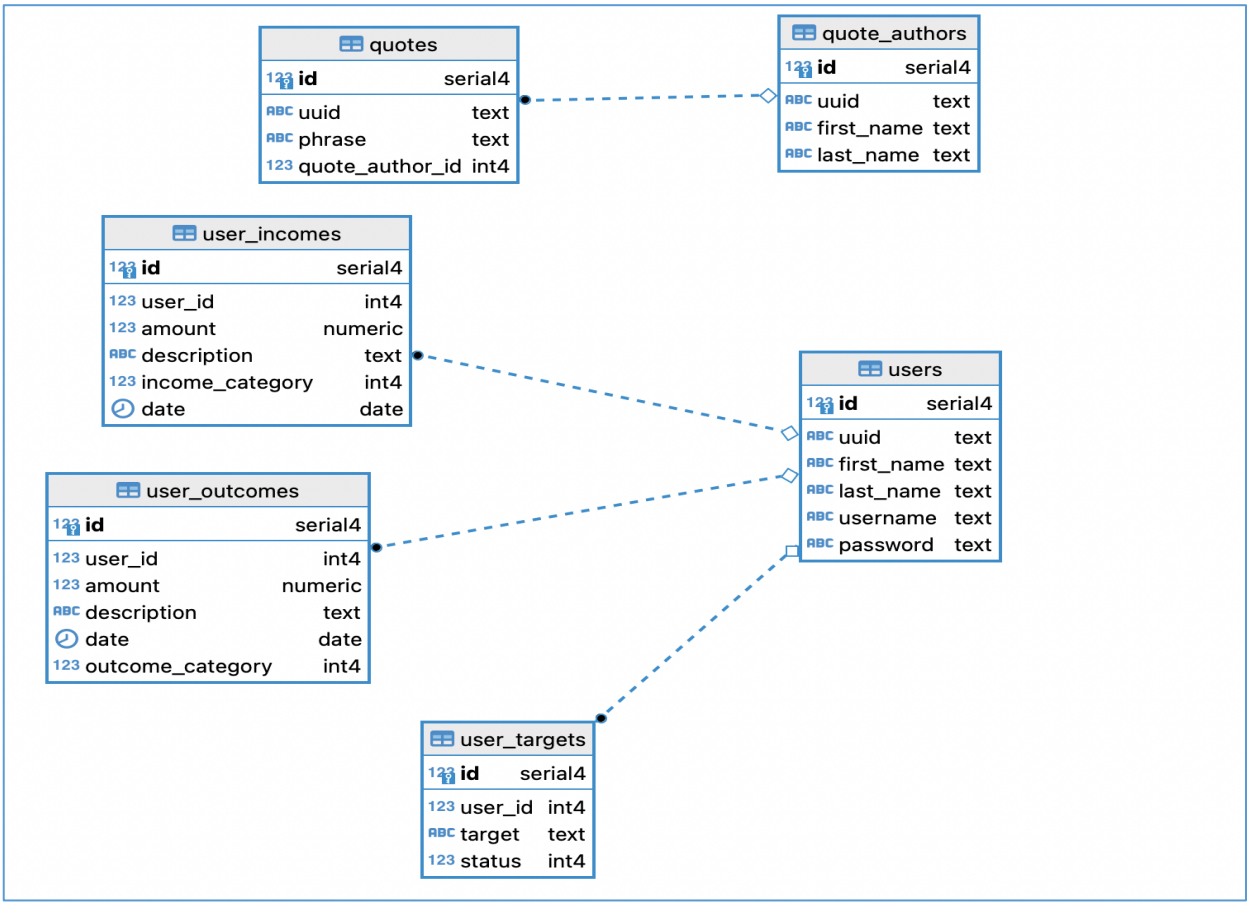


Рисунок 3.8 - Діаграма баз даних

Оскільки використана реляційна база данних тому зазнчаена схема зі звязками які в свою чергу були побудовані за допомогою так званого “зовнішнього ключа”

Наприклад quotes таблиця, тобто цитати, які в подальшому використовуються для фінансових порад мають звязок багато до одного до авторів. Відповідно один автор може мати багато цитат.

3.8 Тестування

Сутність тестування полягає в перевірці програмного продукту або системи з метою виявлення помилок, дефектів або неправильного функціонування. Тестування важливе для забезпечення якості програмного забезпечення та забезпечення задоволення користувачів. Основні мети тестування включають перевірку правильності роботи програми, виявлення помилок, забезпечення відповідності вимогам та підтримку надійності та стабільності системи. Для веб додатку власного фінансового менеджменту було прийняте рішення икористати мануальне тестування з наступними тест сценаріями:

Вхід в обліковий запис або реєстрація нового аккаунту:

1. Перевірено корректність обробки правильних та неправильні облікових даних.
2. Виконано тестування реєстрації нового аккаунту, перевірено правильність заповнення полів та валідацію введених даних.
3. Ведення власного фінансового обліку:
4. Проведено тестування функцій ведення доходів та витрат, переконанося, що дані коректно записуються та зберігаються в системі.

5. Випробування різних варіантів введення даних, включаючи суми, категорії та дати, щоб переконатися в правильності обробки цих даних.

Статистика та попереднє прогнозування:

1. Перевірено, чи правильно розраховуються статистичні показники, такі як загальний дохід, витрати, баланс тощо.
2. Виконано тестування функції попереднього прогнозування, переконанося в точності та достовірності прогнозованих даних.

Фінансові цілі:

1. Перевірено, чи можна встановити фінансові цілі, такі як накопичення певної суми грошей, оплата певних витрат тощо.
2. Випробування функції відстеження прогресу досягнення фінансових цілей, переконанося в правильності відображення та оновлення даних.

Фінансові поради:

1. Перевірено, чи правильно відображаються та оновлюються поради від успішних фінансистів або експертів.
2. Виконано тестування функції отримання персоналізованих порад на основі фінансової ситуації користувача.

Мануальне тестування були використано тому що це дозволяє людині яка тестує активно взаємодіяти з системою та динамічно перевіряти її реакцію на різні вхідні дані та сценарії. Це дозволяє отримати більш гнучкий та деталізований підхід до тестування та виявлення потенційних проблем.

ВИСНОВКИ

В процесі виконання дипломної роботи розроблено програмне забезпечення особистого фінансового менеджменту розробленому мовами Java та JavaScript для спрощення процесу контролю та управління особистими коштами, тобто мета дипломної роботи виконана в повному обсязі.

1. Проведено огляд предметної галузі фінансового менеджменту і виявлено доцільність розробки власного веб додатку.
2. Проаналізовано існуючі додатки у області фінансів такі як: BudgetSimple і Empower та виявлено їх недоліки які враховані при розробці веб додатку власного фінансового менеджменту.
3. Сформовані функціональні і нефункціональні вимоги. Ключовими функціональними вимогами є: експорт статистики, фінансові рекомендації та попереднє прогнозування.
4. Проведено огляд програмних та технічних засобів реалізації веб додатку. В розробці використано: Java, JavaScript, HTML, CSS, Spring, PostgreSQL.
5. Розроблений та протестований додаток який дозволяє вести фінансовий облік, приблизно прогнозувати витрати та доходи і отримувати фінансові рекомендації.
6. Результат дослідження бакалаврської роботи апробовані на МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ «СУЧАСНІ АСПЕКТИ ДІДЖИТАЛІЗАЦІЇ ТА ІНФОРМАТИЗАЦІЇ В ПРОГРАМНІЙ І КОМП'ЮТЕРНІЙ ІНЖЕНЕРІЇ» з темами « ЗАСТОСУВАННЯ МОНОЛІТНОЇ АРХІТЕКТУРИ ДО РОЗРОБКИ ПЗ ДЛЯ ОСОБИСТОГО ФІНАНСОВОГО МЕНЕДЖМЕНТУ» та «РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОСОБИСТОГО ФІНАНСОВОГО МЕНЕДЖМЕНТУ МОВАМИ ПРОГРАМУВАННЯ JAVA ТА JAVASCRIPT»

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Web Development Roadmap | W3School [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.w3schools.com/whatis/
2. Advantage of Java | IBM[Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.ibm.com/docs/en/java
3. Microservices vs monolith: Which architecture is the best choice for your business? – N-ix [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business
4. Bloch J. Effective Java. Addison-Wesley Professional, 2018. 416 с.
5. Intellij IDEA Features [Електронний ресурс] [Веб-сайт]. – електронні дані. – Режим доступу: https://www.jetbrains.com/idea/features
6. Advantages of JavaScript [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://codeinstitute.net/global/blog/advantages-of-javascript
7. WebStorm: Features. *JetBrains*. URL: https://www.jetbrains.com/webstorm/features(дата звернення: 29.05.2023).
8. Spring MVC Tutorial [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.baeldung.com/spring-mvc-tutorial
9. Walls C. Spring in Action. Manning Publications, 2018. 520 с.
10. Haverbeke M. Eloquent Javascript. San Francisco : No Starch Press, 2009.

11. HTML and CSS [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.w3.org/standards/webdesign/htmlcss

12. PostgreSQL 14.8 Documentation. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/14/index.html

13. Head First Design Patterns: A Brain-Friendly Guide / K. Sierra та ін. O'ReillyMedia, Incorporated, 2004. 694 с.

14. A brief history of web development [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://devdojo.com/tnylea/a-brief-history-of-web-development

15. Mcintosh H. Talk Java to me. Corte Madera, CA : Waite Group Press, 1996. 689 с.

16. Spring Security [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://docs.spring.io/spring-security/reference/index.html

ДОДАТОК

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення особистого фінансового менеджменту мовами Java та JavaScript

Виконав студент 4 курсу

Групи ПД-44
Бацунов Дмитро Сергійович
Керівник роботи

д.т.н., доц., зав. кафедри ТЦР Жебка Вікторія Вікторівна

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- Мета роботи – спрощення процесу контролю та управління особистими коштами за допомогою програмного забезпечення особистого фінансового менеджменту написану мовами програмування Java та JavaScript.
- Об’єкт дослідження – процес контролю та управління особистими коштами.
- Предмет дослідження – програмне забезпечення для власного фінансового менеджменту.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести огляд предметної галузі.
2. Провести аналіз існуючих програмних забезпечень в області фінансового менеджменту.
3. Сформулювати функціональні і нефункціональні вимоги до веб додатку власного фінансового менеджменту.
4. Провести огляд програмних та технічних засобів реалізації веб-додатків.
5. Визначити найбільш доречні інструменти розробки.
6. Розробити відповідний до вимог функціонал.
7. Провести тестування розробленого програмного забезпечення.

3

АНАЛІЗ АНАЛОГІВ

Розглянуті додатки	BudgetSimple	Empower	Financial Geek
Категоризація операцій	-	+	+
Інтеграція з банками	+	+	-
Перегляд статистики фінансових операцій	+	+	+
Експорт статистичних даних	-	-	+
Приблизне прогнозування витрат та доходів	-	-	+
Фінансові цілі	+	+	+
Фінансові рекомендації	-	-	+

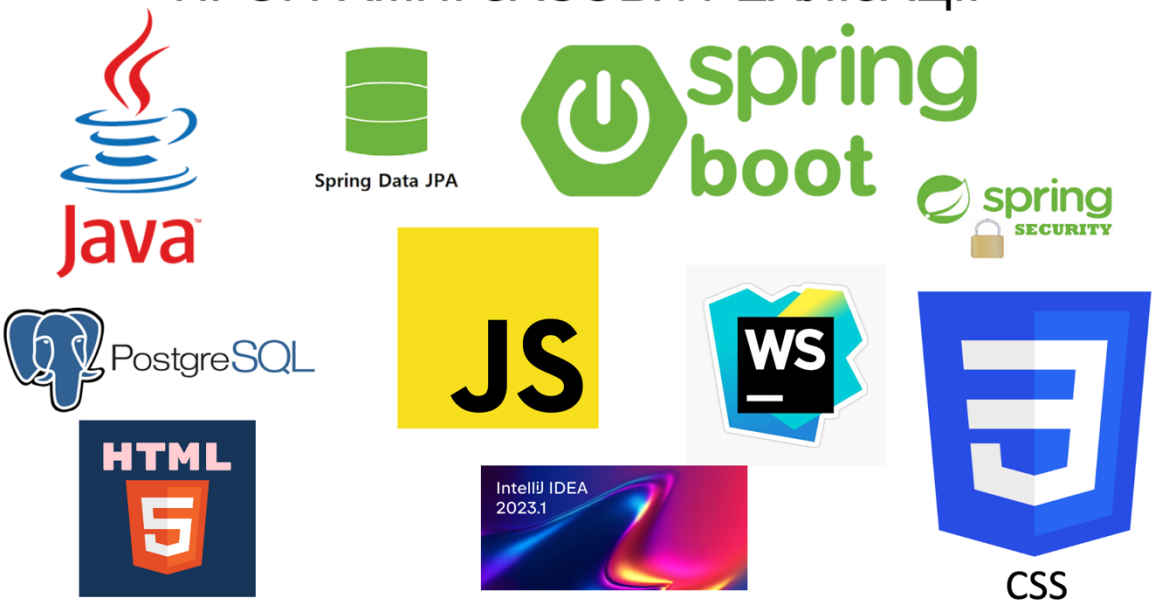
4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. Можливість мати власний обліковий засіб в системі.
2. Додавання та видалення даних щодо фінансових операцій.
3. Підрахунок статистики про грошові операції і попереднього прогнозування фінансових витрат.
4. Перегляд статистики та подальший експорт в форматах PDF, PNG, CSV.
5. Ставлення фінансових цілей та слідкування їх прогресу.
6. Можливість отримати фінансові рекомендації.

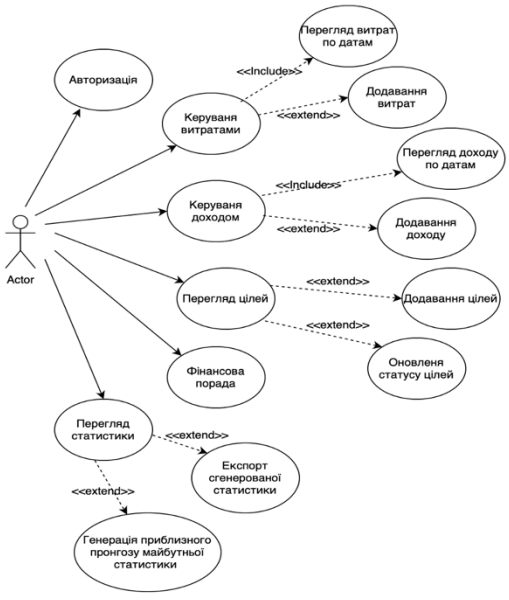
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



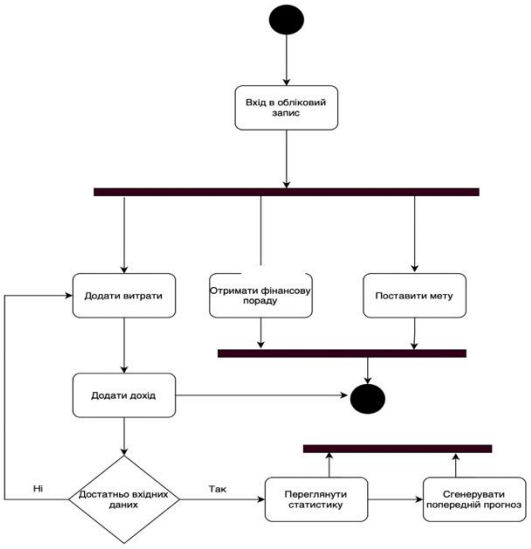
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



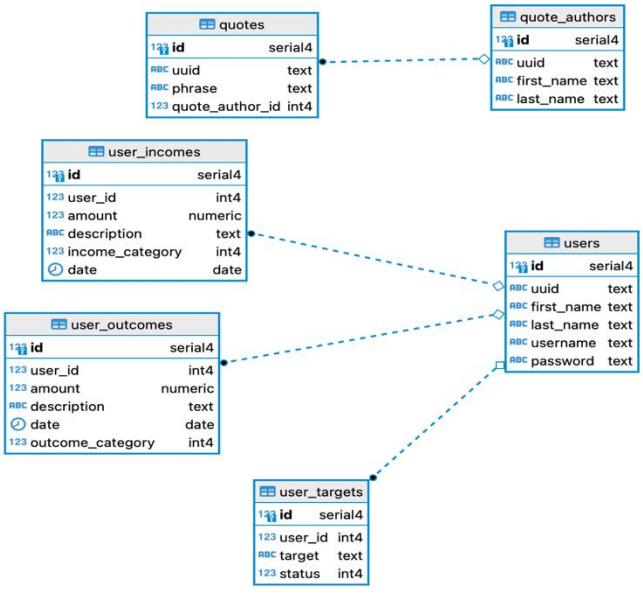
7

Діаграма діяльності



8

Схема бази даних

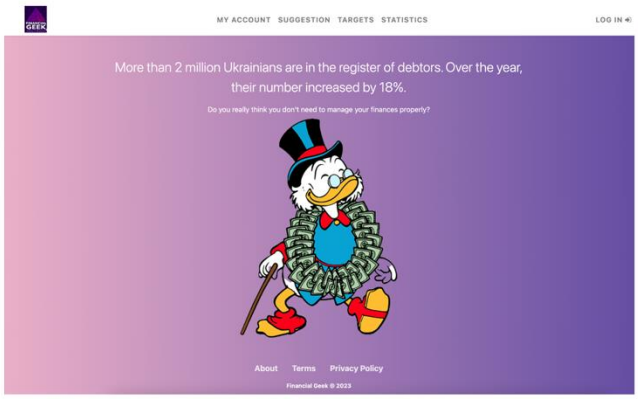


9

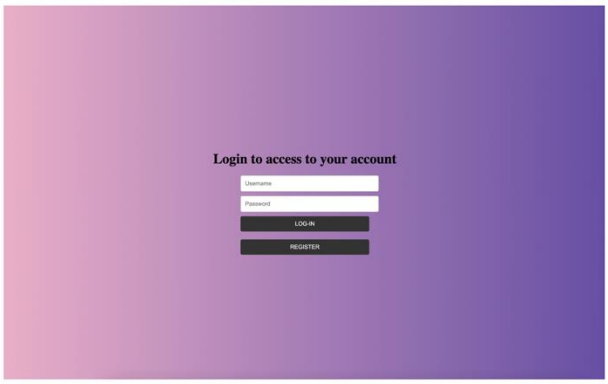
Діаграма класів

Model Service Controller

ЕКРАННІ ФОРМИ

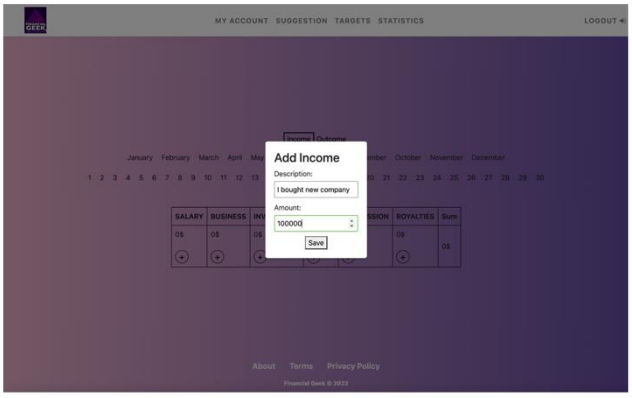


Головний екран додатку

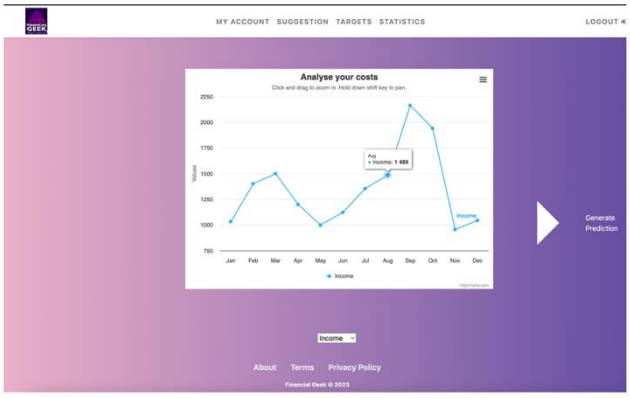


Авторизація

ЕКРАННІ ФОРМИ



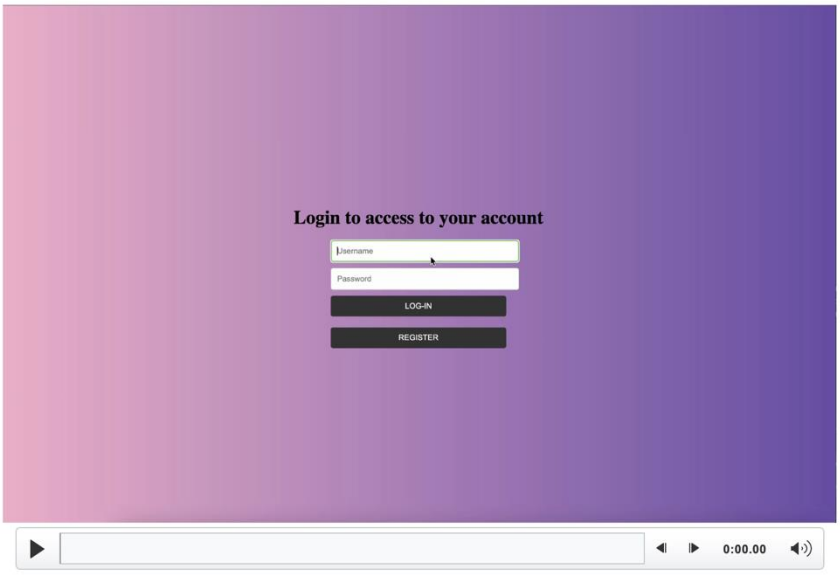
Додавання доходів/витрат



Перегляд статистики

12

ЕКРАННІ ФОРМИ



13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Бацунов. Д.С. Застосування монолітної архітектури до розробки пз для особистого фінансового менеджменту / Д.С. Бацунов // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії: Матеріали Міжнародної науково-практичної конференції. Збірник тез, 30.05.22, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 19 – 20.

- Бацунов. Д.С. Розробка програмного забезпечення особистого фінансового менеджменту мовами програмування Java та Javascript/ Д.С. Бацунов // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії: Матеріали Міжнародної науково-практичної конференції. Збірник тез, 30.05.22, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 25 – 26.

ВИСНОВКИ

1. Проведено огляд предметної галузі фінансового менеджменту і виявлено доцільність розробки власного веб додатку.
2. Проаналізовано існуючі додатки у області фінансів такі як: BudgetSimple і Empower та виявлено їх недоліки які враховані при розробці додатку власного фінансового менеджменту.
3. Сформовані функціональні і нефункціональні вимоги. Ключовими функціональними вимогами є: експорт статистики, фінансові рекомендації та попереднє прогнозування.
4. Проведено огляд програмних та технічних засобів реалізації веб додатку. В розробці використано: Java, JavaScript, HTML, CSS, Spring, PostgreSQL.
5. Розроблений та протестований додаток який дозволяє вести фінансовий облік, попередньо прогнозувати витрати та доходи і отримувати фінансові рекомендації.