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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-застосунку для контролю витрат коштів

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Романишин Ю.Р.
(підпис) (прізвище та ініціали)

Керівник _____ Марценко С.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Микитишин А.Г.
(підпис) (прізвище та ініціали)

Тернопіль
2023

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

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

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Романишин Юлії Романівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-застосунку для контролю витрат коштів

Керівник роботи Марценко Сергій Володимирович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7-133

2. Термін подання студентом завершеної роботи 20.06 2023р.

3. Вихідні дані до роботи Література і інтернет джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз проблеми та постановка задачі. 1.1 Веб-додатки прости веб-сайтів. 1.2.1 Додаток "Monefy". 1.2.2 Додаток "Monobudget". 1.2.3 Додаток "S&E". 1.3 Принцип роботи веб-додатку. 1.4 Визначення вимог та структури веб-додатку. 1.5 Вибір технологій. 1.6 Висновок до першого розділу. Розділ 2. Опис програмної реалізаціх веб-додатку для контролю витрат коштів. 2.1 Загальна структура веб-додатку. 2.2 Розробка серверної частини 2.3 Розробка бази даних. 2.3.1 Структура бази даних. 2.3.2 Побудова зв'язків між таблицями 2.4 Проектування інтерфейсу користувача. 2.5 Тестування роботи системи. 2.6 Висновок до другого розділу. Розділ 3. Безпека життєдіяльності, основи охорони праці. 3.1 Вимоги до ергономічності робочого місця. 3.2 Психофізіологічне розвантаження для працівників. 3.3 Висновок до третього розділу. Висновки. Перелік джерел. Додаткок А. Додаток Б.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			08.06.2023

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про розробку веб-додатків та контроль	24.01.2023-26.01.2023	*Виконано*
	витрат в цифровому світі		
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо існуючих застосунків	01.02.2023-15.02.2023	*Виконано*
	та виявлення їх недоліків		
5.	Оформлення першого розділу « Аналіз проблеми та	16.03.2023-09.04.2023	*Виконано*
	постановка задачі»		
6.	Оформлення розділу « Опис програмної реалізації	10.04.2023-12.05.2023	*Виконано*
	веб-додатку для контролю витрат коштів»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	05.06.2023-06.06.2023	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	07.06.2023-08.06.2023	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	20.06.2023	

Студент _____ Романишин Ю.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марценко С.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-застосунку для контролю витрат коштів // Кваліфікаційна робота освітнього рівня «Бакалавр» // Романишин Юлія Романівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2023 // С. 41, рис. – 17, табл. – 2, кресл. – 0 , додат. – 2, бібліогр. – 37.

Ключові слова: веб-застосунок, витрати, розробка, планування, база даних, проект, система, проектування.

Кваліфікаційна робота присвячена дослідженню та розробці веб-застосунку для контролю витрат коштів. У першому розділі роботи проведено аналіз існуючих рішень та проблем фінансової нестабільності. Результатом цього аналізу є обґрунтування вибору конкретних технологій та формулювання вимог до веб-застосунку.

У другому розділі розглянуто структуру веб-застосунку та розроблено серверну частину та базу даних. Також було створено зручний інтерфейс користувача та протестовано роботу системи.

У третьому розділі кваліфікаційної роботи було розглянуто вимоги до створення зручного та безпечного робочого місця для працівників. Головна мета цих вимог - забезпечити працівникам оптимальні умови для ефективної та продуктивної праці, а також підтримувати їх фізичне та психологічне благополуччя

ANNOTATION

Web-Application Development to Control the Manage Costs // Qualification work of the educational level "Bachelor" // Yuliia Romanivna Romanyshyn // Ternopil National Technical University named after Ivan Pulyu, Faculty of Computer Information Systems and Software Engineering, Department of Computer Sciences, group SN-41 // Ternopil, 2023 // P. 41, fig. – 17, tab. – 2, armchair. – 0, add. – 2 , bibliography – 37

Keywords: web application, costs, development, planning, database, project, system, design.

The qualification work is devoted to the research and development of a web application for controlling the expenditure of funds. In the first part of the work, an analysis of existing solutions and problems of financial instability was carried out. The result of this analysis is the justification of the choice of specific technologies and the formulation of requirements for the web application.

The second chapter examines the structure of the web application and develops the server part and the database. A convenient user interface was also created and the system was tested.

In the third section of the qualification work, the requirements for creating a comfortable and safe workplace for employees were considered. The main purpose of these requirements is to provide employees with optimal conditions for effective and productive work, as well as to maintain their physical and psychological well-being

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CSS (від англ. Cascading Style Sheets) – каскадні таблиці стилів.

DB (від англ. Database) – база даних.

HTML (від англ. HyperText Markup Language) – мова розмітки.

IDE (від англ. Integrated development environment) – інтегроване середовище розробки.

URL (від англ Uniform Resource Locator) – унікальний ідентифікатор, який використовується для пошуку ресурсу в Інтернеті.

БД – база даних.

СУБД – система управління базами даних.

ЗМІСТ

ВСТУП

Актуальність теми. Протягом багатьох років технології еволюціонізували та змінили повсякденне життя і наш світ. Крім того, такий розвиток створив дивовижні ресурси та інструменти, надавши нам можливість користуватись усією інформацію будь-де.

Сучасні технології проклали шлях до таких багатофункціональних пристроїв, як розумний годинник і смартфон. Комп'ютери стають дедалі швидшими, потужнішими та портативнішими, ніж будь-коли раніше. Технології змінили те, як розважаємося, зустрічаємося один з одним і використовуємо всі види медіа. Більше не потрібно заходити в банк, щоб зняти гроші або переказати їх комусь. Багато банків уже зробили транзакції можливими в Інтернеті та доступними для людей у всьому світі.

За допомогою будь-якого пристрою можна робити миттєві перекази грошей, покупки: від одягу, доставки їжі, продуктів, меблів тощо, у будь-який час не в залежності до нашого місцезнаходження. Оплата рахунків також стала спрощеною завдяки технологіям. Можна автоматично планувати платежі, коли вони настануть, замість того, щоб забувати надсилати чек. За допомогою мобільного телефону та банківської програми є можливість керувати всіма необхідними платежами та рахунками онлайн. Це відкриває нові можливості, допомагаючи забезпечити безпеку, мобільність і підключення.

Проте у сучасному світі, де мобільність управління грошима та фінансова грамотність стають усе більш важливими аспектами нашого життя, потреба контролювати витрати набуває особливого значення. Знання грошових потоків допомагає зрозуміти, що кожен насправді робить зі своїми грошима, що може бути дуже корисним для найкращого використання ваших коштів. А при такій кількості банків та додатків зробити це стає все важче.

Тому ця дипломна робота має важливе практичне значення, оскільки розробка веб-застосунку може бути корисною у всіх сферах діяльності. Він може допомогти користувачам краще розуміти свої фінансові потоки, виявляти

непотрібні витрати, стежити за досягненням фінансових цілей та планувати майбутні витрати. Якщо кожен знає, що таке гроші, як ними користуватися, то в кінцевому підсумку надаємо можливості бути господарями не лише своїх грошей, але й — до певної міри — нашого майбутнього. У житті трапляється різне, але контроль над цими змінами настільки, наскільки це можливо, дає набагато більше шансів досягти успіху з нашими грошима.

Тому універсальність та можливість адаптації до різних потреб робить застосунок актуальним і цікавим об'єктом дослідження та розробки в рамках дипломної роботи.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення якості фінансових послуг та надання можливості цифрового контролю обігу коштів. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- проаналізувати та вивчити предметну область;
- розглянути потенціал для використання в різних сферах;
- проаналізувати існуючі веб-додатки, виявити їх недоліки;
- побудувати зручний доступ до фінансової інформації;
- забезпечити безпеку фінансових даних користувачів;
- розробити веб-застосунок з використанням розглянутих технологій і фреймворків
- протестувати застосунок.

Практичне значення одержаних результатів. Створення потужного веб-застосунку для контролю витрат коштів, що має велике значення для користувачів, оскільки надає зручний та ефективний спосіб ведення обліку фінансових операцій, категоризації витрат та аналізу особистих фінансів. Вони можуть більш усвідомлено керувати своїми фінансами, спостерігати за ростом або зменшенням витрат у певних категоріях, ідентифікувати недоцільні витрати та здійснювати раціональніші фінансові рішення. Такий веб-додаток може бути використаний як інструмент для особистого фінансового планування, так і для управління бюджетом сім'ї або навіть для фінансового управління в межах бізнесу.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Веб-додатки проти веб-сайтів

Розробка веб-додатків і веб-сайтів - це два терміни, що використовуються в IT-галузі. Вони схожі, але є деякі відмінності. Веб-додатки це веб-сайти, які працюють подібно до мобільних додатків. Вони доступні через браузер, але мають більше функцій, таких як сповіщення та інтерактивність. Веб-сайти більш спрямовані на надання інформації користувачеві. [1]

Веб-сайти - це, наприклад, сторінки компаній, де користувач може переглядати інформацію та зв'язуватись з компанією. Розробник просто створює сайт з відповідною функціональністю. З іншого боку, веб-додатки дозволяють користувачеві авторизуватись, працювати з даними та виконувати більше завдань.

Існує багато прикладів веб-додатків, таких як інтернет-магазини, соціальні мережі, банківські системи, платформи електронного навчання та інші. У всіх цих випадках розробка веб-додатків є частиною загального процесу веб-розробки.

Веб-додатки відрізняються від веб-сайтів головним чином за орієнтацією на взаємодію з користувачами. Вони створюються з метою надання функціональності, яка дозволяє користувачам виконувати різноманітні завдання та працювати з даними. Це можуть бути операції, такі як створення, редагування та видалення даних, а також розрахунки, аналізи, пошук і багато іншого.

Веб-сайти, з іншого боку, зазвичай створюються для інформування користувача. На таких сайтах відвідувачі можуть переглядати статичний контент, читати інформацію, переглядати зображення та відео, а також зв'язуватись з компанією за допомогою форми зворотного зв'язку або контактних даних. Вони слугують як онлайн-презентації компанії або організації, але не мають такої активної взаємодії з користувачем, як веб-додатки.

Таким чином, веб-додатки можуть бути потужними інструментами для автоматизації бізнес-процесів, розвитку електронної комерції, підтримки спільнот

та багатьох інших завдань, які вимагають активної взаємодії з користувачами та обробки даних. Вони можуть працювати на різних пристроях та операційних системах, забезпечуючи доступ до функціональності незалежно від місця перебування користувача.

Таблиця 1.1 – Порівняльна характеристика веб-сайтів проти веб-додатків

Особливості	**Веб-сайти**	**Веб-застосунки**
Характер взаємодії	Основна інформаційна передача	Інтерактивна взаємодія, обробка даних
Користувацький досвід	Зазвичай статичний і менш інтерактивний	Більш динамічний та інтерактивний
Доступність	Можна переглядати з будь-якого браузера	Можна доступатись як з браузера, так і з мобільного пристрою через спеціальні додатки
Функціональність	Обмежена функціональність	Розширена функціональність з можливістю авторизації, збереження даних, взаємодії з базою даних тощо
Комплексність	Зазвичай простіші і менш складні	Можуть бути складнішими та вимагати більшої розробки
Залежність від інтернету	Залежність від завантаження сторінок	Залежність від активного підключення до Інтернету
Обробка даних	Обмежена або відсутня	Можливість обробки, збереження та відображення даних

У залежності від конкретних потреб та цілей, веб-додатки можуть бути розроблені за допомогою різних технологій та інструментів, таких як HTML, CSS,

JavaScript, PHP, Python та багато інших. Розробники веб-додатків повинні мати глибокі знання веб-розробки та програмування, щоб створити потужні та ефективні рішення для веб-середовища. [7]

Отже, розробка веб-додатків розширює можливості веб-сайтів, надаючи більше функціональності та інтерактивності. Вона дозволяє створювати потужні інструменти для взаємодії користувачів і обробки даних.

1.2 Аналіз існуючих систем контролю витрат та їх недоліків

Аналіз ресурсів важливий етап у розробці веб-додатків. Він допомагає розробникам виявити потенційні можливості для покращення проекту або системи. Під час аналізу розглядаються різні аспекти, такі як функціональність, продуктивність, безпека та ефективність коду. [27]

Один з аспектів аналізу ресурсів - виявлення слабких місць у веб-додатку. Це можуть бути проблеми з процесами, які призводять до недостатньої продуктивності або ефективності. Наприклад, виявлення швидкодії проблем, завантаження сторінок, перевантаження сервера або неефективного використання ресурсів.

Також під час аналізу ресурсів можуть виявлятись проблеми з кодом веб-додатку. Це можуть бути недостатньо оптимізовані або погано написані фрагменти коду, які сповільнюють роботу додатку. Аналіз ресурсів допомагає виявити ці проблеми і зробити відповідні зміни для покращення продуктивності та швидкодії.

Додатково, аналіз ресурсів може допомогти виявити недостатню функціональність веб-додатку. Це означає, що додаток може не володіти певними функціями, які були б корисними для користувачів. Аналіз ресурсів дозволяє виявити ці проблеми та запропонувати відповідні рішення для розширення функціональності додатку.

У даному випадку, для аналізу ресурсів було обрано три мобільних додатки - 'Monefy', 'Monobudget' та 'S&E'. Аналіз цих додатків дозволить виявити їх переваги, недоліки та потенційні можливості для поліпшення. Це дозволить зробити

висновки і прийняти рішення щодо вдосконалення веб-додатку, який розробляється.

1.2.1 Додаток ‘Monefy’

Додаток "Monefy" привернув мою увагу своєю простотою використання та нескладним інтерфейсом. Функціонал додатку включає запис транзакцій, категоризацію витрат та доходів, а також відстеження балансу. Його сумісність з різними платформами та операційними системами, такими як iOS, Android і Windows, робить його доступним для широкого кола користувачів. Крім того, основні функції додатку розміщені в зручній боковій панелі, що сприяє зручності використання на будь-якому пристрої.

Проте, під час аналізу було виявлено кілька недоліків. Перш за все, присутність великої кількості реклами створює дискомфорт для користувача. Крім того, відсутність можливості створення резервних копій та нових категорій є обмеженнями, які можуть ускладнити користування додатком. Важливо також зазначити, що рівень безпеки фінансових даних користувачів є недостатнім, оскільки відсутнє шифрування та захист паролем.

На рисунках 1.1-1.2 наведено знімки головного меню та бокових панелей додатку.

Рисунок 1.1 – Головна сторінка додатку

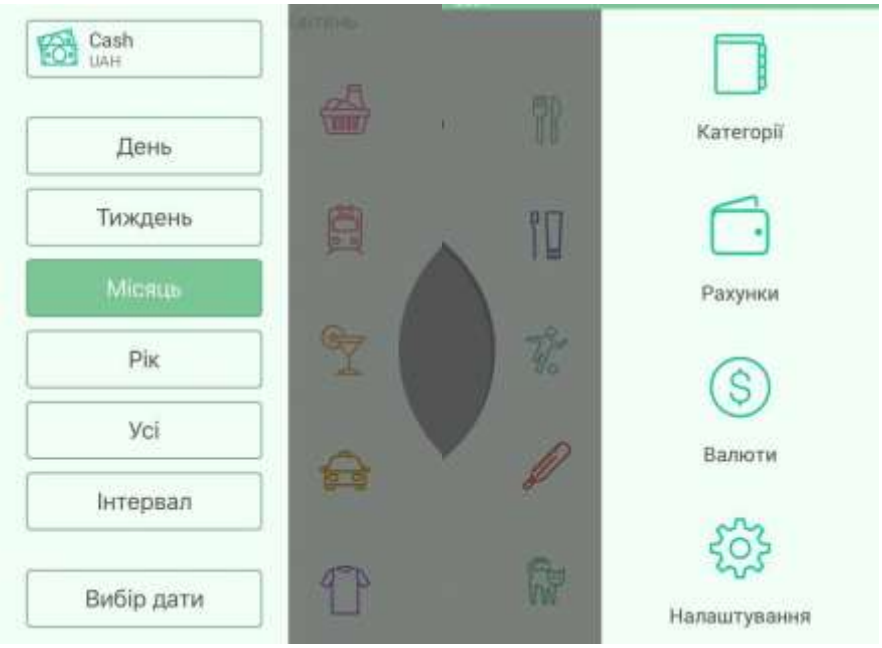

Рисунок 1.2 – Бокові панелі додатку

Бокові панелі представлені з двох сторін основного екрану, вони є динамічними, тому не створюють дискомфорту користувачеві.

1.2.2 Додаток ‘Monobudget’

Далі в аналізі звернено увагу на додаток "Monobudget", який є дочірнім від банку MonoBank. Інтерфейс користувача цього додатку вражає своєю зручністю та логічністю. Розробники приділили увагу якісному дизайну, що робить його привабливим для користувачів. Особливо варто відзначити, що "Monobudget" є інтуїтивно зрозумілим та приємним у використанні, що сприяє комфортному досвіду користування.

Один з ключових переваг цього додатку полягає в його швидкості та ефективності. Він працює без затримок і не заважає нормальному функціонуванню пристрою, що є важливим аспектом для задоволення потреб користувачів. На рисунку 1.3 наведено знімок головного меню додатку, що дає уявлення про його організацію та функціональні можливості.

Рисунок 1.3 – Головне меню додатку ‘Monobudget’

Серед важливих аспектів для додатків фінансового управління, наявність забезпечення синхронізації даних між різними пристроями та можливість створення резервних копій відіграють ключову роль. Синхронізація даних дозволяє користувачам отримувати доступ до своїх фінансових даних з будь-якого пристрою, забезпечуючи зручність і гнучкість використання додатку. Крім того, можливість створення резервних копій забезпечує безпеку даних та захист від втрати інформації у разі випадкового видалення або поломки пристрою.

Однією з унікальних особливостей цього додатку є прив'язка до вашої карти. Це означає, що всі фінансові транзакції будуть автоматично відображатися в додатку. Завдяки цьому, є можливість швидко і зручно відстежувати свої витрати без необхідності вручну вводити кожну транзакцію. Більше того, цей додаток надає можливість створювати необмежену кількість категорій витрат, що дозволяє вам деталізовано відслідковувати різні види витрат та керувати своїм бюджетом ефективніше.

Забезпечення частого оновлення додатку є ще одним важливим аспектом. Оновлення дозволяють вирішувати виявлені проблеми, вдосконалювати функціонал та надавати користувачам кращий досвід використання. Розробники

постійно працюють над покращеннями, враховуючи зворотний зв'язок користувачів і впроваджуючи нові функції, що забезпечують зручність. На рисунку 1.4 зображено вікно налаштувань.

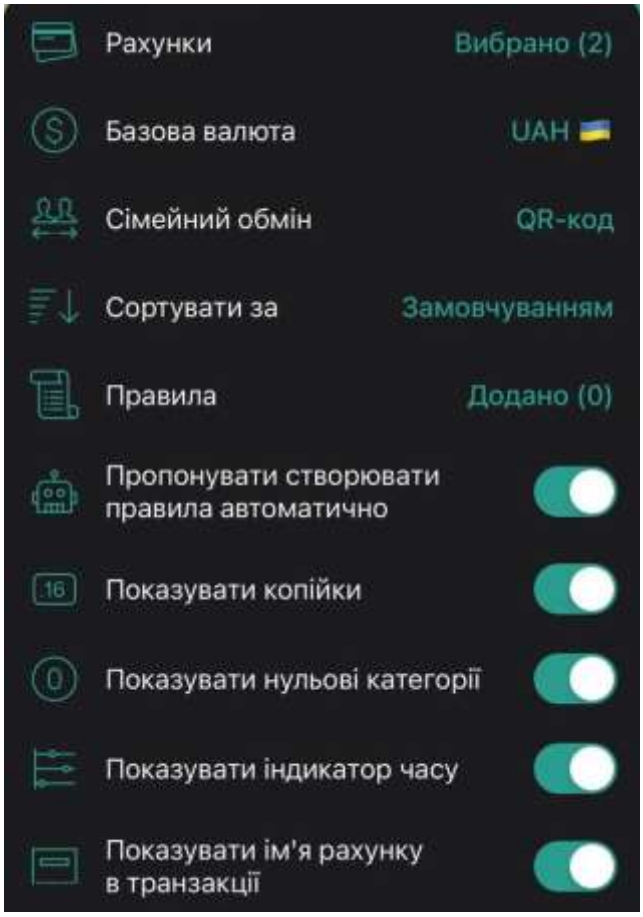

Рисунок 1.4 – Вікно налаштувань ‘Monobudget’

Загалом, "Monobudget" вражає своєю зручністю, швидкістю та ефективністю, що робить його привабливим вибором для користувачів, особливо тих, хто є клієнтами банку MonoBank.

1.2.3 Додаток ‘S&E’

Останнім аналогом, на який було звернено увагу, є додаток "S&E". Цей додаток також надає можливість відстежувати доходи та витрати, контролювати бюджет та планувати фінансові цілі, як і попередні два аналоги.

Проте, "S&E" вирізняється з-поміж інших додатків своєю можливістю вводити транзакції з детальною інформацією. Користувач може категоризувати свої витрати, відстежувати баланс та отримувати графіки і діаграми для візуального аналізу фінансових даних. Додаток також надає можливість встановлювати бюджети та отримувати сповіщення щодо їх дотримання, що допомагає користувачам краще контролювати свої фінанси.

Однією з особливостей "S&E" є наявність платної версії додатка, яка дозволяє отримати більш детальний аналіз витрат і доходів. Платна версія також надає можливість синхронізації даних з іншими пристроями, групування транзакцій за періодами (днями, місяцями, роками) і створення книги боржників або власних боргів. Ці функціональні можливості розширюють функціональність додатку і надають більше зручностей для користувачів у керуванні своїми фінансами.

Загалом, "S&E" є привабливим вибором для тих, хто бажає мати більше деталей і можливостей у відстеженні своїх фінансових показникі. На рисунку 1.5 зображено інтерфейс додатку.

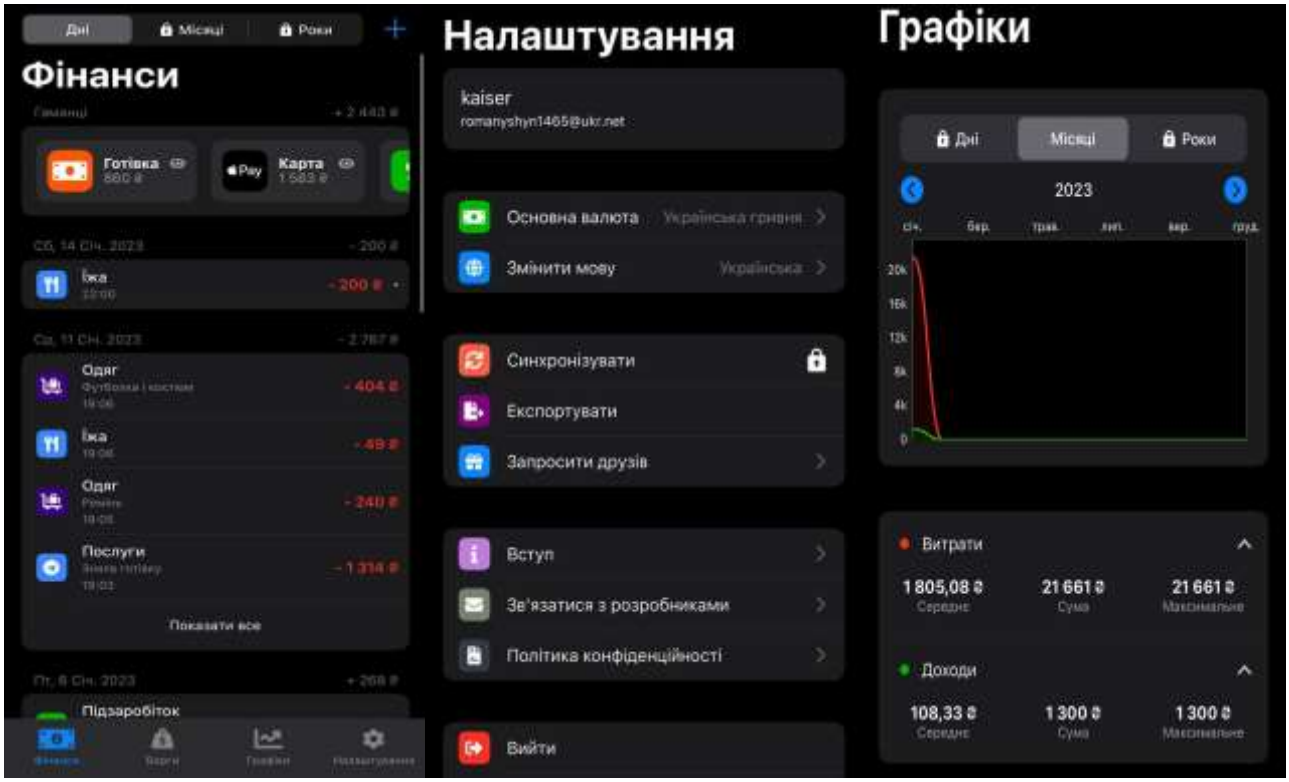

Рисунок 1.5 – Інтерфейс додатку

Враховуючи всі переваги "S&E", можна сказати, що цей додаток надає користувачам широкі можливості для контролю та аналізу їх фінансів, забезпечуючи зручний та інформативний інтерфейс

Нижче наведена порівняльна характеристика трьох аналогів додатків фінансового управління: Monefy, Monobudget і S&E.

Таблиця 1.2 – Порівняльна характеристика аналогів

Назва критерію	Назва ресурсу		
	‘Monefy’	‘Monobudget’	‘S&E’
Інтерфейс	Простий, нескладний	Зручний, логічний	Інтуїтивно зрозумілий
Функціональність	Запис транзакцій, категоризація витрат та доходів, відстеження балансу	Запис транзакцій, категоризація витрат та доходів, відстеження балансу	Запис транзакцій з детальною інформацією, категоризація витрат, відстеження балансу, графіки і діаграми
Безпека	Недостатній рівень	Присутня	Недостатній рівень
Підтримка	Відсутня	Присутня	+Присутня
Сумісність	iOS, Android, Windows	iOS, Android, Windows	iOS, Android, Windows

У таблиці представлені основні характеристики, функціональні можливості та особливості кожного додатку.

1.3 Принцип роботи веб-додатку

Веб-додаток є програмою, яка працює на веб-сервері та надає функціональні можливості користувачам через їх веб-браузери. Принцип роботи веб-додатка полягає в постійному обміні інформацією між браузером та сервером, що дозволяє взаємодіяти з додатком та використовувати його функціональність через Інтернет.

Коли користувач відкриває веб-додаток у своєму веб-браузері, він вводить адресу додатку, і браузер відправляє запит до сервера, що містить цю адресу. Сервер отримує цей запит і знаходить веб-додаток, який повинен обробити запит.

Веб-додаток обробляє запит, виконує різні дії та генерує відповідь, яку сервер надсилає назад до браузера. Браузер отримує цю відповідь і відображає її на екрані користувача. Користувач може взаємодіяти з веб-додатком, заповнюючи форми, натискуючи кнопки та виконуючи інші дії.

Кожна дія користувача викликає новий запит до сервера, який оброблює його та повертає відповідь. Така взаємодія триває протягом користування додатком.

Розробка веб-додатків використовує різні технології. Зазвичай вони базуються на комбінації мов програмування на стороні сервера (наприклад, PHP, Python, Ruby, Java), баз даних (наприклад, MySQL, PostgreSQL) та мов програмування на стороні клієнта (наприклад, HTML, CSS, JavaScript). [3]

Один з головних переваг веб-додатків полягає в тому, що вони працюють у веб-середовищі і не вимагають встановлення додаткового програмного забезпечення на пристроях користувачів. Веб-додатки також мають перевагу централізованого управління, оскільки оновлення та зміни можуть бути внесені на сервері і автоматично відображатися для користувачів без необхідності оновлювати сам додаток на кожному пристрої.

Таким чином, веб-додатки є зручним та широко використовуваним способом доступу до функціональності через веб-браузери, завдяки постійному обміну інформацією між браузером та сервером.

1.4 Визначення вимог та структури веб-додатку

При розробці нового веб-додатку для ведення фінансів було виявлено певні вимоги, що базуються на аналізі існуючих додатків. Ці вимоги будуть служити основою для подальшого проектування та розробки системи. До функціональних вимог можна віднести:

- реєстрація нових користувачів з обов'язковими полями, такими як ім'я, електронна пошта та пароль;
- авторизація та аутентифікація користувачів;
- можливість користувачам вносити інформацію про свої витрати, включаючи суму, категорію;
- відображення статистики про їхні витрати, такі як загальна сума витрат за певний період, розподіл витрат за категоріями;
- можливість встановлення бюджету або цілей витрат і відстеження прогресу досягнення цих цілей.

Чітко визначені функціональні вимоги визначають необхідний функціонал, що дозволить користувачам ефективно вести облік своїх витрат та отримувати необхідну інформацію.

Якщо говорити про нефункціональні вимоги, то я виділила такі:

- відстеження доходів та витрат;
- додаток повинен надати можливість користувачам планувати свої фінансові цілі, встановлювати цілі по заощадженнях або витратах, та відстежувати їх досягнення;
- додаток повинен дозволяти користувачам вводити транзакції з детальною інформацією, такою як опис, категорія та спосіб платежу;
- користувачам повинна бути надана можливість категоризувати свої витрати, щоб вони могли легко аналізувати, на що саме вони витрачають свої гроші;
- додаток повинен давати змогу користувачам відстежувати стан їх фінансового балансу - суму доступних коштів.

Ці вимоги виокремлюють новий додаток від інших. Нефункціональні вимоги забезпечують зручність, безпеку та продуктивність системи.

У результаті аналізу вимог до веб-застосунку для контролю витрат були визначені ключові функції, які включають реєстрацію користувачів, додавання витрат, відображення статистики та бюджетування.

1.5 Вибір технологій

При розробці веб-додатку було зроблено вибір певних технологій, які відіграють важливу роль у створенні функціонального та ефективного додатку. Нижче наведено список використаних технологій та їх взаємодію між собою.

Visual Studio є інтегрованою середою розробки (IDE), розробленою компанією Microsoft. Вона надає зручне середовище для написання, відлагодження та керування кодом програми. Visual Studio має багатий набір інструментів та функціональних можливостей, що спрощують процес розробки.

MS SQL Server є системою управління базами даних (СУБД), розробленою компанією Microsoft. Вона використовується для зберігання та керування даними в додатку. MS SQL Server надає надійність, швидкодію та розширюваність для ефективної роботи з базами даних.

Ruby on Rails (часто називається просто Rails) є веб-фреймворком, написаним мовою програмування Ruby. Він забезпечує швидку та зручну розробку веб-додатків, пропонуючи готові компоненти та конвенції розробки. Rails працює на основі моделі-вид-контролер (Model-View-Controller, MVC) і спрощує створення базової структури додатку.[12]

HTML (HyperText Markup Language) є мовою розмітки, використовуваною для створення структури та вмісту веб-сторінок. Він визначає структуру документа, включаючи заголовки, параграфи, списки, посилання та інші елементи.

CSS (Cascading Style Sheets) є мовою стилів, яка використовується для оформлення веб-сторінок. Він дозволяє задавати зовнішній вигляд елементів HTML, таких як кольори, шрифти, розміри, відступи тощо.

JavaScript є мовою програмування, яка використовується для створення інтерактивних елементів на веб-сторінках. Він дозволяє додавати функціональність, яка взаємодіє з користувачем, таку як анімація, валідація форм, взаємодія з сервером тощо.

Ці технології взаємодіють між собою таким чином:

Розробка додатку здійснюється у середовищі Visual Studio, де використовуються мови програмування Ruby on Rails, HTML, CSS та JavaScript.

Ruby on Rails використовується для створення серверної частини додатку, реалізації логіки, обробки запитів та доступу до бази даних.

HTML використовується для створення структури веб-сторінок, включаючи розмітку та розташування елементів.

CSS використовується для оформлення сторінок, задавання зовнішнього вигляду, кольорів, шрифтів та розмірів елементів.

JavaScript використовується для створення динамічних елементів, валідації форм, взаємодії з користувачем та сервером.

За допомогою цих технологій розроблений веб-додаток може ефективно взаємодіяти з користувачем, обробляти його запити, зберігати та відображати дані, а також забезпечувати зручний та естетичний інтерфейс користувача.

1.6 Висновок до першого розділу

У першому розділі дипломної роботи було проаналізовано проблеми, які виникають при контролі витрат коштів у цифровому середовищі. Потрібно було дослідити наявні ресурси, які вже існують для контролю витрат, і дізнатися, які переваги та недоліки мають ці ресурси. Також було розглянуто різницю між веб-сайтами і веб-додатками і вивчила принцип роботи веб-додатків.

Зрозуміла, як саме працюють веб-додатки, тому що це було важливо для мого подальшого дослідження. Вивчено послідовність дій, яка відбувається, коли користувач взаємодіє з веб-додатком, і вияснено, як інформація передається між веб-браузером користувача і веб-сервером, де розташований веб-додаток.

На основі цього аналізу, вирішено розробити новий веб-застосунок, який буде мати наступні можливості:

- Перегляд статистики про витрати, щоб користувачі могли бачити, куди йде їхній грошовий потік.
- Можливість створювати нові категорії для зручного моніторингу даних, а також редагувати та видаляти їх, щоб користувачі мали контроль над своїми витратами.
- Додавання можливості створення кошельків та відстежування їх балансу, щоб користувачі могли організовувати свої фінанси.
- Створення історії транзакцій з детальною інформацією, щоб користувачі могли переглядати свої попередні операції.

Окрім цього, проведено аналіз вимог і структури для мого веб-додатку, визначивши, які функціональні можливості має мати додаток для контролю витрат. Обрано технології для розробки веб-додатку, зокрема Visual Studio для написання програмного коду, MS SQL Server для зберігання даних, а також Ruby on Rails, HTML, CSS і JavaScript для реалізації функцій користувацького інтерфейсу та взаємодії з користувачем.

У цілому, цей перший розділ дипломної роботи дав необхідні знання і базу для подальшої розробки веб-застосунку для контролю витрат коштів.

РОЗДІЛ 2. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКУ ДЛЯ КОНТРОЛЮ ВИТРАТ КОШТІВ

2.1 Загальна структура веб-додатку

Мікросервіси — це архітектурне рішення, яке базується на розподілі модулів на окремі системи, які спілкуються між собою за допомогою повідомлень. Уся система — це набір маленьких систем, які пов’язані між собою, докладніше проілюстровано на рисунку 2.1 [5]

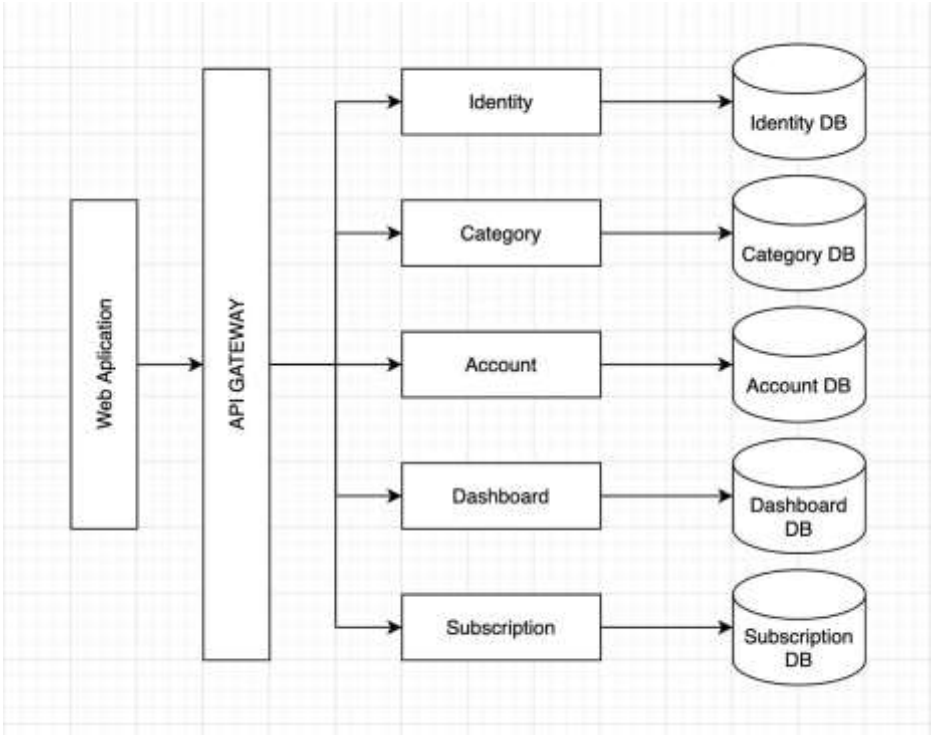

Рисунок 2.1 – Схема спроєктованої інфраструктури

Перевага використання мікросервісів порівняно з більш традиційним підходом полягає в тому, що групи розробників можуть швидко створювати нові компоненти додатків без негативного впливу на інші служби. Зміни, внесені однією командою, не призведуть до руйнування роботи всього додатка чи його частин. Мікросервіси підходять для розробки веб-додатків, особливо для великих та складних проектів. Вони дозволяють розподілити функціональність додатку на

окремі сервіси, які можуть бути розгорнуті на різних серверах та масштабовані незалежно один від одного. Це полегшує розробку нових функцій, оновлення та підтримку системи. Крім того, мікросервіси забезпечують більшу стійкість до відмов та можливість використання різних технологій для різних частин системи.

2.2 Розробка серверної частини

Моделі в Ruby on Rails - це основний компонент, який використовується для організації та роботи з даними у веб-додатку. Вони відображають структуру та взаємозв'язки між різними об'єктами додатку, такими як користувачі, категорії, транзакції тощо.

Моделі виконують декілька важливих функцій у веб-додатку. По-перше, вони забезпечують доступ до бази даних та дозволяють зберігати, зчитувати, редагувати та видаляти дані. Наприклад, якщо у нас є модель "Користувач", ми можемо використовувати її для створення нового користувача, зчитування інформації про існуючого користувача або оновлення даних користувача.

У лістингах 2.1 та 2.2 наведено приклад контролерів для реєстрації і входу користувача в систему.

Лістинг 2.1 – Контролери системи для реєстрації користувача

```
class AccountsController < ApplicationController
  before_action :set_account, only: [:show, :edit, :update, :destroy]
   def new
    @account = Account.new
  end

  def create
    @account = Account.new(account_params)
    @account.user = current_user

    if @account.save
      redirect_to accounts_path, notice: "Account was succesfully
created."
    else
      render :new, status: :unprocessable_entity
    end
  end
```

Лістинг 2.2 – Контролери системи для входу в систему

```
def account_params
    params.require(:account).permit(:name,              :account_type,
:account_number, :bank_name, :owner_name, :current_balance)
  end

  def set_account
    @account = Account.find(params[:id])
  End
```

Як можна замітити в лістингу 2.1 зазначений фільтр:

```
before_action :set_account, only: [:show, :edit, :update,
:destroy]
```

Цей фільтр встановлює змінну @account, викликаючи метод set_account, перед виконанням дій show, edit, update та destroy. Це дозволяє уникнути дублювання коду і забезпечує наявність актуальних даних про обліковий запис перед виконанням цих дій. Для кращого уявлення роботи фільтра, на лістингах 2.3 та 2.4, показано його реалізацію.

Лістинг 2.3 – Контролери системи для оновлення даних акаунту

```
class AccountsController < ApplicationController
  before_action :set_account, only: [:show, :edit, :update,
:destroy]

  def index
    @accounts = current_user.accounts
  End

def update
    if @account.update(account_params)
      redirect_to accounts_path, notice: "Account was
succesfully updated."
    else
      render :edit, status: :unprocessable_entity
    end
  end
```

Лістинг 2.4 – Контролери системи для видалення акаунта

```
def destroy
    @account.destroy
    redirect_to accounts_path, notice: "Account was deleted
successfully."
  end
```

2.3 Розробка бази даних

База даних - це структурована збірка даних, яка дозволяє зберігати, організовувати та отримувати доступ до інформації. Вона відіграє важливу роль у розробці програмного забезпечення та дозволяє зберігати дані про користувачів, транзакції, категорії та іншу інформацію, яка необхідна для вашого проекту. [23]

Перш за все, роботу розпочато на створенні таблиць і визначенні зв'язків між ними. Кожна таблиця має свої поля, що відображатимуть відповідну інформацію. Наприклад, таблиця "User" містить дані про користувачів, такі як ім'я, прізвище, вік, електронну пошту та хеш пароля.

У результаті створення бази даних отримано структуровану систему для зберігання та управління даними. Це дозволяє ефективно працювати з інформацією, забезпечувати цілісність даних та здійснювати різноманітні операції, такі як додавання, оновлення та видалення даних.

2.3.1 Структура бази даних

Як вже було раніше вирішено, то для проектування бази даних було використано MS SQL – Server.

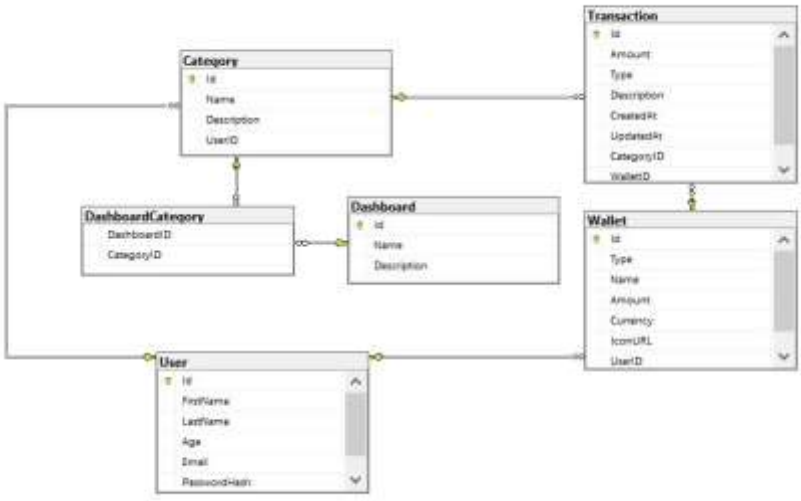

Рисунок 2.2 – Схема бази даних.

У лістингах 2.5-2.7 наведено код до таблиць User, Wallet і Category

Лістинг 2.5 – Створення таблиці User

```
CREATE TABLE [User] (
    Id INT PRIMARY KEY,
    FirstName NVARCHAR(255),
    LastName NVARCHAR(255),
    Age INT,
    Email NVARCHAR(255),
    PasswordHash NVARCHAR(255)
);
```

Лістинг 2.6 – Створення таблиці Wallet

```
CREATE TABLE Wallet (
    Id INT PRIMARY KEY,
    Type NVARCHAR(255),
    Name NVARCHAR(255),
    Amount DECIMAL(18, 2),
    Currency NVARCHAR(255),
    IconURL NVARCHAR(255),
    UserID INT,
    FOREIGN KEY (UserID) REFERENCES [User](Id)
);
```

Лістинг 2.7 – Створення таблиці Category

```
CREATE TABLE Category (
    Id INT PRIMARY KEY,
    Name NVARCHAR(255),
    Description NVARCHAR(MAX),
    UserID INT,
    FOREIGN KEY (UserID) REFERENCES [User](Id)
);
```

Ці таблиці утворюють базу даних для збереження і управління даними про користувачів, гаманці та категорії. З допомогою цих таблиць можна створювати зв'язки, здійснювати операції з даними та забезпечувати функціональність веб-застосунку.

2.3.2 Побудова зв’язків між таблицями

При розробці бази даних важливо встановити правильні зв'язки між таблицями, щоб забезпечити ефективну організацію та взаємозв'язок даних. Зв'язки

дозволяють нам встановлювати залежності та взаємодію між різними таблицями, що допомагає уникнути дублювання даних та забезпечує цілісність та консистентність інформації.

Таблиця "User" і "Wallet":

У таблиці "User" поле "Id" є первинним ключем, а в таблиці "Wallet" поле "UserID" є зовнішнім ключем, який посилається на "Id" таблиці "User". Це створює зв'язок один до багатьох між користувачем і його гаманцями. Користувач може мати багато гаманців, але кожен гаманець належить тільки одному користувачеві.

Таблиця "Transaction" і "Category":

У таблиці "Transaction" поле "CategoryID" є зовнішнім ключем, який посилається на "Id" таблиці "Category". Це створює зв'язок багато до одного між транзакцією і категорією. Одна категорія може мати багато транзакцій, але кожна транзакція належить тільки одній категорії.

Таблиця "Transaction" і "Wallet":

У таблиці "Transaction" поле "WalletID" є зовнішнім ключем, який посилається на "Id" таблиці "Wallet". Це створює зв'язок багато до одного між транзакцією і гаманцем. Одна транзакція може належати тільки одному гаманцю, але в одному гаманці може бути багато транзакцій.

Таблиця "Category" і "User":

У таблиці "Category" поле "UserID" є зовнішнім ключем, який посилається на "Id" таблиці "User". Це створює зв'язок багато до одного між категорією і користувачем. Одна категорія належить тільки одному користувачеві, але у користувача може бути багато категорій.

Таблиця "DashboardCategory" і "Dashboard":

У таблиці "DashboardCategory" поле "DashboardID" є зовнішнім ключем, який посилається на "Id" таблиці "Dashboard". Це створює зв'язок багато до одного між категорією і панеллю управління. Одна категорія може входити тільки в одну панель управління, але в панелі управління можуть бути багато категорій.

Таблиця "Subscription" і "SubscriptionSettings":

У таблиці "Subscription" поле "SettingsID" є зовнішнім ключем, який посилається на "Id" таблиці "SubscriptionSettings". Це створює зв'язок один до одного між підпискою і налаштуваннями підписки. Кожна підписка має тільки одні налаштування, але налаштування можуть використовуватись багатьма підписками.

Таблиця "UserSubscriptions" і "User" та "Subscription":

У таблиці "UserSubscriptions" поле "UserID" є зовнішнім ключем, який посилається на "Id" таблиці "User", а поле "SubscriptionID" є зовнішнім ключем, який посилається на "Id" таблиці "Subscription". Це створює зв'язок багато до багатьох між користувачем і підписками. Один користувач може мати багато підписок, а підписка може належати багатьом користувачам.

Всі ці зв'язки допомагають нам побудувати потужну та ефективну базу даних, яка дозволяє зберігати, організовувати та використовувати дані зручним та ефективним способом.

2.4 Проектування інтерфейсу користувача

Для реалізації панелі користувача було використано такі технології Ruby on Rails, HTML, CSS.

Дана панель повинна реалізовувати такі елементи як:

- авторизація для користувачів;
- панель для зображення статистичних даних;
- панель розпорядження категорій;
- панель гаманців;
- історія транзакцій;
- зміна теми інтерцейсу.

Весь наведений код можна знайти в додатку А. У цьому розділі представлено результат роботи коду.

Для нових користувачів, які ще не мають облікового запису в системі, сторінка авторизації може містити посилання на сторінку реєстрації. Це дозволить

їм створити новий обліковий запис та отримати доступ до системи. На рисунку 2.3 зображено сторінку реєстрації користувача.

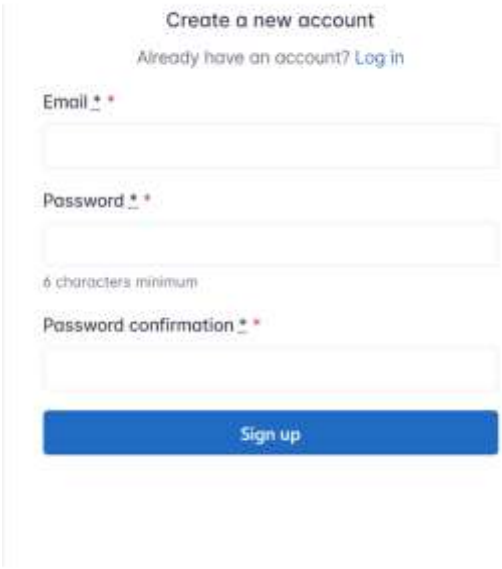

Рисунок 2.3 – Сторінка реєстрації користувача

Для користувачів, які вже були раніше зареєстровані, є меню для входу в систему. На рисунку 2.4 зображено сторінку авторизації користувача.

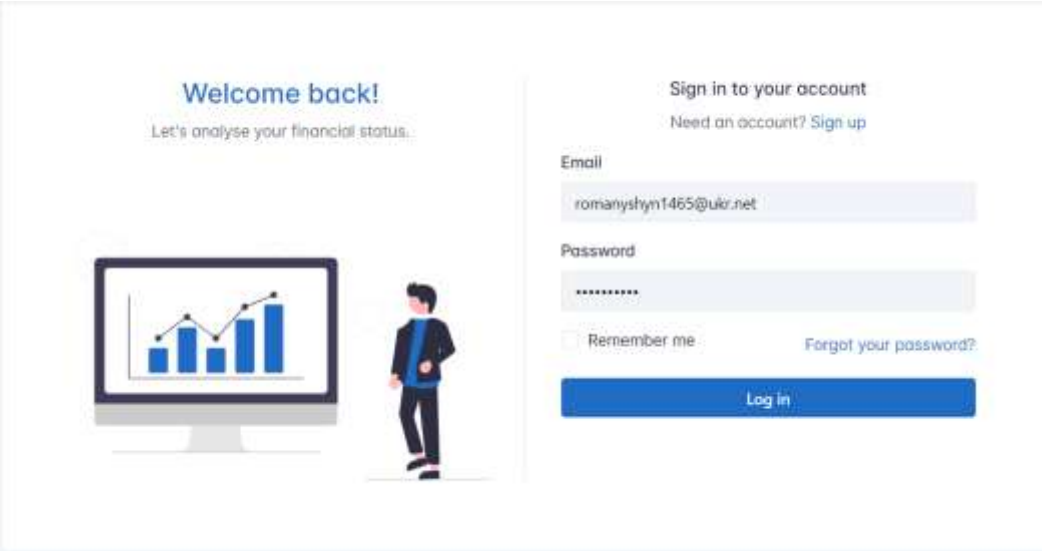

Рисунок 2.4 – Сторінка авторизації користувача

Для головного меню вирішено додати таблиці зі статистикою витрат та доходів, а також діаграму поділу витрат за категоріями. Також у правому верхньому куті сторінки можна вибрати період за який буде надана статистика (див. рис. 2.5)

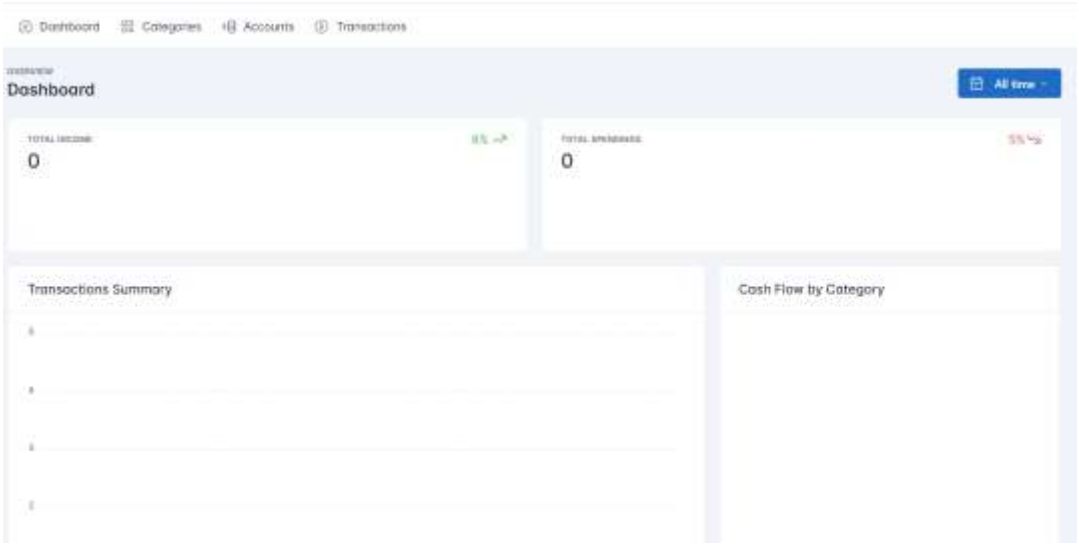

Рисунок 2.5 – Головна сторінка застосунку

Для зручності користування застосунком головне меню розміщено в верхній частині сторінки. Воно є статичним та при переході на іншу сторінку залишається незмінним. На сторінці категорій розміщено список, де будуть розміщені всі створені користувачем категорії. Також є можливість редагувати та видаляти їх. На рисунку 2.6 зображено сторінка управління категоріями.

Рисунок 2.6 – Сторінка управління категоріями

У розділі “Accounts” користувач може відслідковувати всі свої рахунки.

Крім цього можна додати інформацію про банк, номер карти, власника та тип кошилька. На рисунку 2.7 зображено вигляд гаманця.

Рисунок 2.7 – Перелік гаманців

На рисунку 2.8 зображено історію транзакцій користувача. Тут користувач додавати всі свої витрати та доходи, які будуть відображатись в діаграмах на головній сторінці застосунку.

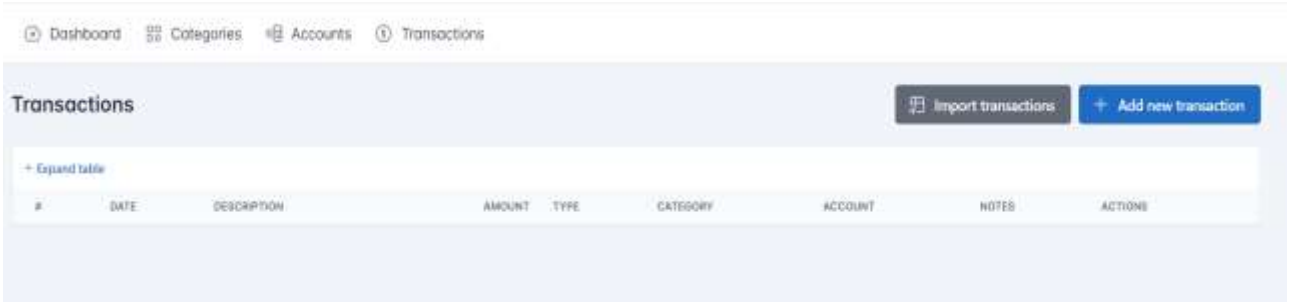

Рисунок 2.8 – Історія транзакцій

Під час проектування інтерфейсу було враховано потреби та очікування цільової аудиторії, використоно зрозумілі та логічні елементи управління, а також розроблено зручний дизайн і естетичний вигляд.

2.5 Тестування роботи системи

Етап тестування розробки допомагає забезпечити високу якість та надійність програмного забезпечення перед його випуском у виробництво. У цьому розділі

протестовано функціональності створення нових категорій та додавання нових транзакцій. А також на основі цього зможемо оглянути роботу діаграм та протестимо темну версію застосунку.

На рисунку 2.9 зображено функціональне меню додавання категорій. Тут достатньо ввести назву категорії і зберегти.

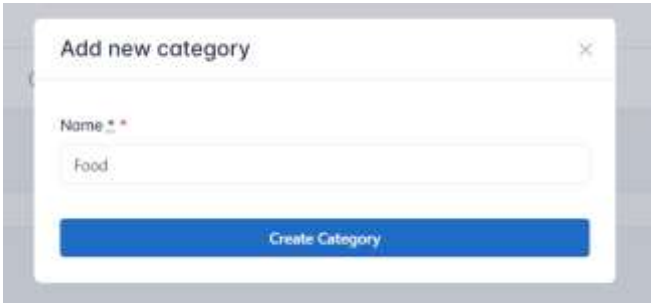

Рисунок 2.9 – Приклад додавання нової категорії

Щоб побачити список потрібно зайти в меню категорій. На випадок коли щось стало неактуально можна видалити або ж редагувати. На рисунку 2.10 показано результат.

Рисунок 2.10 – Список доданих категорій

Для комфортного відслідковування транзакцій добавлено пару пунктів, а саме: дату, опис, суму та категорію. Тому в списку ми будемо бачити все потрібне. На рисунку 2.11 зображено історію транзакцій.

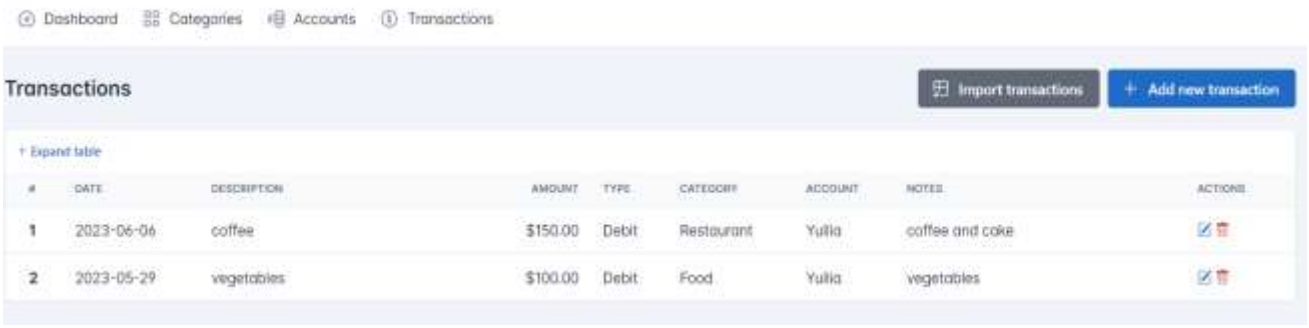

Рисунок 2.11 – Список транзакцій

На рисунку 2.12 зображено темна тема та аналітика витрат та доходів.

Рисунок 2.12 – Аналітика витрат та доходів

Під час тестування було здійснено перевірку правильності роботи функціоналу додатку. Користувач міг успішно взаємодіяти з інтерфейсом, додавати та редагувати дані, а також виконувати необхідні дії.

Загалом, тестування роботи системи показало, що веб-додаток працює відповідно до очікувань і надає користувачам зручний і функціональний інтерфейс для керування своїми обліковими записами, категоріями та транзакціями.

2.6 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було проведено значну роботу з розробки серверної частини, бази даних, проектування інтерфейсу користувача та тестування системи. Весь процес вимагав детального планування та уважної роботи над кожним етапом.

Основні досягнення включають успішну реалізацію серверної частини, що забезпечує взаємодію між клієнтами та базою даних, а також забезпечує безпеку та автентифікацію користувачів. Розроблено структуру бази даних, включаючи необхідні таблиці та зв'язки між ними, що дозволяє ефективно зберігати та організовувати дані. Було створено зручний та інтуїтивно зрозумілий інтерфейс користувача, що дозволяє легко керувати обліковими записами, категоріями та транзакціями. На завершення, система пройшла успішне тестування, що підтверджує її надійність та коректну роботу.

Результатом цієї роботи є функціональний веб-додаток, який допомагає користувачам контролювати свої витрати та керувати фінансовими операціями. Він забезпечує зручний та безпечний спосіб ведення обліку та аналізу фінансових даних. Важливо відзначити, що цей проект є вдалим кроком у поліпшенні фінансової грамотності та управління особистими фінансами.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Вимоги до ергономічності робочого місця

Вимоги до ергономічності робочого місця означають, що потрібно створити комфортне і безпечне середовище для працівників. Ергономіка ставить за мету забезпечити, щоб робота виконувалася з ефективністю і мінімальними негативними впливами на здоров'я.

Робоче місце – це ділянка простору, яка облаштована необхідним обладнанням, відповідно до трудової діяльності, для виконання поставлених завдань. Правильно побудоване робоче місце повинне забезпечувати:

• найкраще розміщення обладнання і предметів праці;

• не допускати дискомфорту;

• підвищувати продуктивність праці;

• зменшувати втому працівника.

Перш за все, важливо забезпечити зручну позицію для працівника. Розмір робочого місця повинне бути таким, щоб людина не виконувала лишніх рухів і не відчувала дискомфорту під час виконання роботи. Це означає, що стіл і стілець повинні бути належного розміру і підходити до фізіологічних особливостей людини. Стіл повинен бути достатньо просторим для розташування всіх необхідних робочих матеріалів, таких як комп'ютер, клавіатура, мишка і документи. Стілець повинен мати підлокітники і спинку, щоб підтримувати правильну позицію тіла і запобігати напруженням. [21]

Також для працівника важливо мати змогу змінити робочу позу, наприклад, положення тулуба, рук або ніг. Потрібно мінімізувати або звести до нуля всі незручності положення тіла. [37]

Робоча поза – це найбільш тривале положення тіла працівника протягом робочого дня. При зручній робочій позі забезпечується стійкість положення тулуба, ніг, рук, голови і витрачається мінімальний запас енергії та максимальну продуктивність.

Сидячки і стоячи – дві найпопулярніших пози у робочому процесі. При проектуванні робочого місця потрібно враховувати, що з фізичним навантаженням бажана поза стоячи, а при малих зусиллях – сидячи. При роботі стоячи, людина стомлюється більше ніж сидячи. У відсотковому еквіваленті це на 10% більше енергії. При додатковому навантаженні підвищується артеріальний і венозний тиск крові, розширення вен, пошкоджуються ступені та викривляється хребет.

У свою чергу при сидячій роботі нижня частина тіла розслаблена, а основне навантаження спрямоване на м'язи шиї, спини, таза, стегон. При неправильній сидячій позі розвивається застій крові у ногах, а якщо пальці виконують багато роботи можливе запалення суглобів.

Наступний аспект - освітлення. Робоче місце повинно бути яскраво освітлене, але при цьому уникати прямих блисків на монітор комп'ютера. Найкращий варіант - природне світло, але якщо його недостатньо, слід встановити достатню кількість штучного освітлення. Для запобігання напругам очей можна використовувати спеціальні фільтри на моніторах.

Третій аспект - правильна організація робочого простору. На робочому столі повинні бути місця для розташування робочих матеріалів так, щоб працівник не мусив постійно нахилятися або дотягуватися до них. Комп'ютерний монітор повинен бути розташований на відстані 50-70 см від очей, і його верхня частина повинна бути на рівні очей.

Важливо також забезпечити правильну підтримку тіла під час роботи. М'язи повинні бути розслаблені, а кістки та суглоби - у відповідному положенні. Наприклад, клавіатура повинна бути розташована на такій висоті, щоб зап'ястя були рівні ліктям, а при роботі з мишею важливо уникати надмірного напруження зап'ястя.

Додатковою вимогою є наявність перерв під час роботи. Рекомендується робити невеликі перерви кожні 30-60 хвилин, під час яких працівник може розтягнутися, погуляти або зробити фізичні вправи. Це допоможе запобігти накопиченню напруги в м'язах і знизити втому.

Ще одним важливим аспектом є вентиляція та якість повітря на робочому місці. Добре провітрюване приміщення забезпечує достатній потік свіжого повітря і допомагає уникнути накопичення високої вологості, запахів або шкідливих речовин. Це можна досягти шляхом встановлення відповідних систем вентиляції та використання повітряних очисників, якщо необхідно.

Організація кабелів та проводів також має велике значення для безпеки та зручності робочого місця. Дроти повинні бути закріплені та організовані таким чином, щоб не перешкоджати рухам працівника і не створювати ризику спотикання. Крім того, це також зменшує шанси на пошкодження або випадкове відключення кабелів.

Не слід забувати і про психологічний аспект ергономіки робочого місця. Важливо створити приємну атмосферу, де працівник почувається комфортно і мотивований до роботи. Це можна досягти шляхом декорування приміщення рослинами, фотографіями, використанням приємних кольорів та наданням можливості персоналізації робочого простору.

Усі ці аспекти в сукупності допомагають створити ергономічне робоче місце, яке сприяє здоров'ю, комфорту та продуктивності працівників. Роботодавці повинні пам'ятати, що інвестування в ергономічність має довгострокову вигоду, оскільки здорові та задоволені працівники здатні досягати кращих результатів та бути більш ефективними у своїй роботі.

3.2 Психофізіологічне розвантаження для працівників

У цьому розділі розглянуто методи та стратегії, які допомагають працівникам зняти напругу та стрес під час роботи. Це важливий аспект, оскільки довготривалий стрес може негативно вплинути на здоров'я та продуктивність працівників.

Одним із ефективних методів психофізіологічного розвантаження є фізична активність. Регулярні фізичні вправи та розтяжки під час робочого дня можуть допомогти зняти напругу з м'язів, поліпшити кровообіг та сприяти виробленню

ендорфінів - речовин, які сприяють почуттю щастя та релаксації. Навіть невеликі перерви на зарядку або прогулянку можуть значно підвищити настрій та зменшити стрес.

Ще одним способом розвантаження є впровадження практик медитації та дихальних вправ. Ці методи допомагають зосередитися на моменті, зняти напругу та відновити емоційний баланс. Короткі сеанси медитації або глибокого дихання можуть бути виконані протягом робочого дня, навіть за декілька хвилин. Вони сприяють зниженню рівня стресу та поліпшенню зосередженості та ефективності.

Також важливо враховувати вплив робочого середовища на психологічний стан працівників. Створення приємного та сприятливого робочого оточення може допомогти зняти напругу та покращити настрій. Розташування рослин, наявність природного світла, приємні кольори та звуки можуть створювати релаксуючу атмосферу. Крім того, можна розглянути можливість створення спеціальних зон для відпочинку або релаксації, де працівники можуть відпочити та відновити енергію.

Не слід забувати про важливість раціонального планування робочого часу та встановлення меж між роботою та особистим життям. Продуктивна робота вимагає часу для відпочинку та релаксації. Важливо забезпечити працівникам достатню кількість часу для сну, відпочинку та зайняття хобі. Також, планування регулярних перерв та відпусток сприяє відновленню енергії та попередженню вигорання.

Усі ці психофізіологічні методи та стратегії допомагають працівникам зберігати емоційний баланс, знижувати стрес та покращувати загальний стан здоров'я. Роботодавці повинні сприяти впровадженню таких практик на робочому місці, оскільки це приносить користь як працівникам, так і організації в цілому.

3.3 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи було розглянуто вимоги до ергономічності робочого місця і психофізіологічне розвантаження для працівників. Вимоги до ергономічності робочого місця включають такі пункти, як правильна

позиція тіла, оптимальне розташування обладнання, належна освітленість та вентиляція. Ці вимоги спрямовані на забезпечення комфорту, безпеки та продуктивності працівників.

Психофізіологічне розвантаження для працівників включає такі аспекти, як зменшення психологічного навантаження, періодичні перерви та використання спеціальних методів для зняття напруження. Ці заходи спрямовані на підтримку психічного та фізичного здоров'я працівників, запобігання стресу та вигорання.

Загальна мета вимог до ергономічності робочого місця та психофізіологічного розвантаження полягає в створенні сприятливого середовища праці, яке сприяє збереженню здоров'я працівників і підвищенню їхньої ефективності.

Врахування цих пунктів є критичним для будь-якого робочого місця, оскільки незабезпечення ергономічних умов та недостатнє психофізіологічне розвантаження можуть призвести до зниження продуктивності, збільшення ризику виникнення травм та погіршення фізичного та психічного здоров'я працівників.

ВИСНОВКИ

У рамках цієї дипломної роботи було проведено комплексне дослідження та розроблено веб-застосунок з метою створення функціонального та ефективного інструменту, який задовольняє потреби користувачів та відповідає сучасним стандартам розробки веб-додатків.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Подано загальну концепцію веб-застосунків.
– Розглянуто існуючі рішення для контролю витрат коштів.
– Проведено аналіз проблем фінансової нестабільності та неграмотності.
– Проаналізовано недоліки та переваги за стосунків.
– Обґрунтовано вибір конкретних технологій для розробки проекту.

На основі цього аналізу були сформульовані основні вимоги до майбутнього веб-застосунку та визначена його загальна структура.

В другому розділі кваліфікаційної роботи:

– Розроблена структура веб-застосунку з урахуванням вимог.
– Розроблено серверна частина додатку, включаючи необхідну базу даних, яка забезпечує зберігання та обробку інформації.
– Запропоновані ефективні рішення для вирішення проблем користувачів.
– Розроблено зручний інтерфейс користувача для зручного взаємодії з додатком.
– Протестовано роботу повноцінного веб-застосунку та його відповідності вимогам.

Результатом кваліфікаційної роботи бакалавра є веб-застосунок для контролю витрат коштів. Проте, цей застосунок потребує деяких покращень і у майбутньому планується вдосконалити його функціонал. Зокрема, планується додати можливість створення декількох облікових сторінок для одного користувача, впровадити платну підписку і створити можливість генерації статистичних звітів у форматі PDF.

Це означає, що користувачі зможуть мати більше можливостей та зручніший інструмент для ведення обліку своїх фінансів. Вони зможуть створювати окремі облікові сторінки для різних цілей, таких як особисті фінанси або сімейний бюджет. Також буде запроваджена платна підписка, яка надасть користувачам доступ до додаткового функціоналу та розширених можливостей.

Окрім цього, статистичні звіти у форматі PDF дозволять користувачам отримувати зрозумілу та зручну інформацію про свої фінансові показники та витрати. Це допоможе їм аналізувати свої фінанси більш ефективно та приймати обґрунтовані фінансові рішення.

Таким чином, покращення та розширення функціоналу веб-застосунку сприятимуть зручному та ефективному управлінню фінансами користувачів і покращать їх фінансову грамотність.

ПЕРЕЛІК ДЖЕРЕЛ

1. AltexSoft. "Web application architecture: how the web works." URL: https://www.altexsoft.com/blog/engineering/web-application-architecture-how-the-web-works/

2. AgingInPlace.org. "Technology in our life today and how it has changed." URL: https://aginginplace.org/technology-in-our-life-today-and-how-it-has-changed/

3. Build Offshore Technology Team in India. "Best web development technologies to use in 2023." URL: https://www.clariontech.com/blog/best-web-development-technologies-to-use-in-2022

4. GeeksforGeeks. "How web works - web application architecture for beginners." URL: https://www.geeksforgeeks.org/how-web-works-web-application-architecture-for-beginners/

5. Hillel IT School Blog. "Архітектура веб-додатків - як вибрати?" URL: https://blog.ithillel.ua/articles/web-application-architecture

6. Microsoft Support. "Основні відомості про бази даних." URL: https://support.microsoft.com/uk-ua/office/основні-відомості-про-бази-даних-a849ac16-07c7-4a31-9948-3c8c94a7c204

7. Monocubed. "14 web technologies list for web developers in 2023." URL: https://www.monocubed.com/blog/web-technologies-list/

8. Ruby-Doc.org. "Programming ruby: the pragmatic programmer's guide." URL: https://ruby-doc.com/docs/ProgrammingRuby/html/web.html

9. RubyGuides. "Ruby tutorial for complete beginners: learn ruby now!" URL: https://www.rubyguides.com/ruby-tutorial/

10. Ruby Programming Language. "Ruby in twenty minutes." URL: https://www.ruby-lang.org/en/documentation/quickstart/

11. SavchukIT. "Основи HTML & CSS для початківців #1 - Вступ." URL: https://www.youtube.com/watch?v=Rr9QmVLqoP4

12. Simplilearn. "What is ruby on rails? | ruby on rails for beginners | ruby programming language | simplilearn."URL: https://www.youtube.com/watch?v=6Vi_7R

13. Thi Bảo. "005 setting up a local ruby on rails web server." URL: https://www.youtube.com/watch?v=-GHKUGCyUAA

14. WoMo – Дайджест деловой женщины. "Топ-9 застосунків для контролю особистих фінансів." URL: https://womo.ua/top-10-dodatkiv-dlya-kontrolyu-osobistih-finansiv/

15. Вікіпедія. "База даних." URL: https://uk.wikipedia.org/wiki/База_даних

16. Вікіпедія. "HTML." URL: https://uk.m.wikipedia.org/wiki/HTML

17. Вікіпедія."Учасники проектів Вікімедіа." URL: https://uk.wikipedia.org/wiki/Учасники_проектів_Вікімедіа

18. Галина Лисенко. "Урок 18. Веб-сервер та бази даних, 2022." URL: https://www.youtube.com/watch?v=8DxslffFMs0

19. Галина Лисенко. "Урок 19. Взаємодія 'клієнт-сервер', 2022." URL: https://www.youtube.com/watch?v=6izSKqMmzXA

20. ДБНУ - Державні будівельні норми України - норми: ДБН, ДСТУ, СНиП, ГОСТ, СН, ВБН. "Ергономіка робочого місця і чому це Важливо? - Для дому та дачі - Техніка."URL: https://dbn.co.ua/publ/ergonomika_robochogo_miscja/39-1-0-1014

21. Державний університет телекомунікацій. "Сучасні ергономічні вимоги до організації робочих місць, обладнаних екранними пристроями." URL: https://dut.edu.ua/ru/news-1-0-7678-suchasni-ergonomichni-vimogi-do-organizacii-robochih-misc-obladnanih-ekrannimi-pristroyami

22. Общество. "Як організувати контроль витрат: керівництво до дії." URL: https://wol.org.ua/blog/uk/budzetirovanie-ak-organizuvati-kontrol-vitrat-kerivnictvo-do

23. Підтримка від Microsoft. "Основні відомості про бази даних." URL: https://support.microsoft.com/uk-ua/office/основні-відомості-про-бази-даних-a849ac16-07c7-4a31-9948-3c8c94a7c204

24. Puby-Doc.org: Documenting the Ruby Language. "Programming ruby: the pragmatic programmer's guide." URL: https://ruby-doc.com/docs/ProgrammingRuby/html/web.html

25. Ruby Programming Language. "Ruby in twenty minutes." URL: https://www.ruby-lang.org/en/documentation/quickstart/

26. RubyGuides. "Ruby tutorial for complete beginners: learn ruby now!." URL: https://www.rubyguides.com/ruby-tutorial/

27. WEZOM. "Як створити веб-додаток: типи, переваги, принцип роботи - Wezom." URL: https://wezom.com.ua/ua/blog/kak-sozdat-veb-prilozhenie

28. YouTube. "Best web development technologies to use in 2023." URL: https://www.clariontech.com/blog/best-web-development-technologies-to-use-in-2022

29. YouTube. "Основи HTML & CSS для початківців #6 - HTML-таблиці." URL: https://www.youtube.com/watch?v=5LCosRmmF3Y

30. YouTube. "Основи HTML & CSS для початківців #7 - CSS." URL: https://www.youtube.com/watch?v=dNwZsTTfVSE

31. YouTube. "Основи HTML & CSS для початківців #9 - Форми." URL: https://www.youtube.com/watch?v=nxRMeu9zCZg

32. YouTube. "Setting up a local ruby on rails web server." URL: https://www.youtube.com/watch?v=-GHKUGCyUAA

33. IT-УНІВЕР. "Організація та підтримка веб-сервера." URL: https://it-univer.com/wp-content/uploads/2017/02/Лабораторна-робота-№2-2.0-Організація-та-підтримка-веб-сервера.pdf

34. Інфорум. "Етапи розробки програмного продукту." URL: https://www.infourm.com.ua/novyny/etapy-rozrobki-programnogo-produktu

35. Інформаційний ресурс "Педагогічні технології". "Що таке веб-додаток." URL: http://plsi.org.ua/informatika/web-dodatok/

36. IT-ШКОЛА РОЗРОБНИКА. "Структура веб-проекта на Ruby on Rails." URL: https://ithillel.ua/study/courses/frontend-javascript-module-1

37. Ергономічні вимоги до організації робочих місць [Електронний ресурс] – Режим доступу до ресурсу: https://pidru4niki.com/14821111/bzhd/ergonomichni_vimogi_organizatsiyi

ДОДАТКИ

Додаток А

```
<div class="col-sm-6 col-lg-6">
  <div class="card d-flex flex-column">
    <div class="row row-0 flex-fill">
      <div class="col-md-3">
        <%= image_tag "westpac-logo.png", class: "w-100 h-100" %>
      </div>
      <div class="col border-start">
        <div class="card-body text-center">
          <h3><%= account.name %></h3>
          <div class="row">
            <div class="col">
              <samp           class="text-muted">Bank:           <%=
account.bank_name.capitalize %></samp>
            </div>
            <div class="col">
              <samp           class="text-muted">Type:           <%=
account.account_type.capitalize %></samp>
            </div>
          </div>
          <div class="row">
            <div class="col">
              <samp class="text-muted">Owner: <%= account.owner_name
%></samp>
            </div>
            <div class="col">
              <samp          class="text-muted">Number:          <%=
account.account_number %></samp>
            </div>
          </div>
          <div class="row">
            <div class="col mt-3">
              <strong>Balance:                                    $<%=
number_with_precision(account.current_balance,     precision:     2)
%></strong>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

<div class="page-header d-print-none">
  <div class="container-xl">
    <div class="row g-2 align-items-center">
      <div class="col">
        <h3 class="page-title">Accounts</h3>
      </div>
      <div class="col-auto ms-auto">
        <div class="btn-list">
```

```
          <%= link_to new_account_path, data: { turbo_frame: 'modal'
}, class: 'btn btn-primary' do %>
            <svg   xmlns="http://www.w3.org/2000/svg"   class="icon"
width="24"   height="24"   viewBox="0   0   24   24"   stroke-width="2"
stroke="currentColor"   fill="none"   stroke-linecap="round"   stroke-
linejoin="round"><path      stroke="none"      d="M0      0h24v24H0z"
fill="none"></path><path  d="M12  5l0  14"></path><path  d="M5  12l14
0"></path></svg>Add new account
          <% end %>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="page-body">
  <div class="container-xl">
    <div class="row row-deck row-cards">
      <%= render @accounts %>
    </div>
  </div>
</div>
</div>

<div class="page-header d-print-none">
  <div class="container-xl">
    <div class="row g-2 align-items-center">
      <div class="col">
        <h3 class="page-title">Categories</h3>
      </div>
      <div class="col-auto ms-auto">
        <div class="btn-list">
          <%=   link_to   new_category_path,   data:   {   turbo_frame:
'modal' }, class: 'btn btn-primary' do %>
            <svg   xmlns="http://www.w3.org/2000/svg"   class="icon"
width="24"   height="24"   viewBox="0   0   24   24"   stroke-width="2"
stroke="currentColor"   fill="none"   stroke-linecap="round"   stroke-
linejoin="round"><path      stroke="none"      d="M0      0h24v24H0z"
fill="none"></path><path  d="M12  5l0  14"></path><path  d="M5  12l14
0"></path></svg>Add new category
          <% end %>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="page-body">
  <div class="container-xl">
    <div class="col-12">
      <div class="card">
        <div class="table-responsive">
          <table  class="table  card-table  table-vcenter  text-nowrap
datatable">
            <thead>
```

```
              <tr>
                <th scope="col">Name</th>
                <th scope="col">Transactions</th>
                <th scope="col">Created at</th>
                <th scope="col">Updated at</th>
                <th scope="col">Actions</th>
              </tr>
            </thead>
            <tbody>
              <%= render @categories %>
            </tbody>
          </table>
        </div>
      </div>
    </div>
  </div>
  <div class="container-xl my-3">
    <%= paginate @categories, theme: 'bootstrap-5', pagination_class: "pagination-sm flex-wrap justify-content-center" %>
  </div>
</div>

<tr>
  <td><%= category.name %></td>
  <td><%= category.transactions.count %></td>
  <td><%= category.created_at.to_date %></td>
  <td><%= category.updated_at.to_date %></td>
  <td>
    <%= link_to edit_category_path(category), class: 'text-primary', data: { turbo_frame: 'modal' } do %><i class="bi bi-pencil-square"></i><% end %>
    <%= link_to category_path(category), class: 'text-danger', data: { turbo_method: :delete, turbo_confirm: 'Are you sure?' } do %><i class="bi bi-trash"></i><% end %>
  </td>
</tr>

<!DOCTYPE html>
<html>
  <head>
    <title>My Cash Flow</title>
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>
    <%= stylesheet_link_tag "application", "data-turbo-track": "reload" %>
    <link rel="preconnect" href="https://rsms.me/">
    <link rel="stylesheet" href="https://rsms.me/inter/inter.css">
    <script async src="https://www.googletagmanager.com/gtag/js?id=G-0SPWQM395B"></script>
```

```
    <%= javascript_include_tag 'analytics', async: true %>
    <%=  javascript_include_tag  "application",  "data-turbo-track":
"reload", defer: true %>
  </head>
  <body class="theme-light" data-controller="theme">
    <div class="page">
      <%= render 'shared/navbar' %>
      <%= render 'shared/flashes' %>
      <div class="page-wrapper">
        <%= yield %>
        <div
          class="modal fade"
          tabindex="-1"
          data-controller="modal"
          data-action="turbo:frame-load->modal#open    turbo:submit-
end->modal#close"
          >
          <div class="modal-dialog">
            <div class="modal-content">
              <%= turbo_frame_tag "modal" %>
            </div>
          </div>
        </div>
      </div>
      <%= render 'shared/footer' %>
    </div>
  </body>
</html>


<div class="page-header d-print-none">
  <div class="container-xl">
    <div class="row g-2 align-items-center">
      <div class="col">
        <h3 class="page-title">Transactions</h3>
      </div>
      <div class="col-auto ms-auto">
        <div class="btn-list">
          <%= link_to read_transactions_path, data: { turbo_frame:
'modal' }, class: 'btn btn-secondary' do %>
            <svg   xmlns="http://www.w3.org/2000/svg"   class="icon
icon-tabler   icon-tabler-table-import"   width="24"   height="24"
viewBox="0   0   24   24"   stroke-width="2"   stroke="currentColor"
fill="none" stroke-linecap="round" stroke-linejoin="round">
              <path      stroke="none"      d="M0      0h24v24H0z"
fill="none"></path>
              <path d="M4 13.5v-7.5a2 2 0 0 1 2 -2h12a2 2 0 0 1 2
2v12a2 2 0 0 1 -2 2h-6m-8 -10h16m-10 -6v11.5m-8 3.5h7m-3 -3l3 3l-3
3"></path>
            </svg>Import transactions
          <% end %>
          <%=  link_to  new_transaction_path,  data:  {  turbo_frame:
'modal' }, class: 'btn btn-primary' do %>
```

```
            <svg   xmlns="http://www.w3.org/2000/svg"   class="icon"
width="24"   height="24"   viewBox="0   0   24   24"   stroke-width="2"
stroke="currentColor"   fill="none"   stroke-linecap="round"   stroke-
linejoin="round"><path      stroke="none"      d="M0      0h24v24H0z"
fill="none"></path><path  d="M12  5l0  14"></path><path  d="M5  12l14
0"></path></svg>Add new transaction
          <% end %>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="page-body">
  <div class="container-xl">
    <div class="col-12">
      <div class="card">
        <div class="table-responsive" data-controller="cell-expand">
          <button   type="button"   data-cell-expand-target="button"
data-action="click->cell-expand#toggle" class="btn btn-ghost-primary
btn-sm m-2">
            <svg   xmlns="http://www.w3.org/2000/svg"   class="icon
icon-tabler icon-tabler-plus" width="24" height="24" viewBox="0 0 24
24"   stroke-width="2"   stroke="currentColor"   fill="none"   stroke-
linecap="round" stroke-linejoin="round">
              <path      stroke="none"      d="M0      0h24v24H0z"
fill="none"></path>
              <path d="M12 5l0 14"></path>
              <path d="M5 12l14 0"></path>
            </svg>Expand table
          </button>
          <table class="table card-table table-vcenter table-nowrap
datatable table-collapse">
            <thead>
              <tr>
                <th scope="col">#</th>
                <th scope="col">Date</th>
                <th scope="col">
                  Description
                </th>
                <th scope="col" class="text-end">Amount</th>
                <th scope="col">Type</th>
                <th scope="col">Category</th>
                <th scope="col">Account</th>
                <th scope="col">Notes</th>
                <th scope="col">Actions</th>
              </tr>
            </thead>
            <tbody>
              <%= render @transactions %>
            </tbody>
          </table>
        </div>
      </div>
```

```
    </div>
  </div>
  <div class="container-xl my-3">
    <%=    paginate    @transactions,    theme:    'bootstrap-5',
pagination_class: "pagination-sm flex-wrap justify-content-center"
%>
  </div>
</div>

<div class="modal-header">
  <%= turbo_frame_tag 'modal' do %>
    <h2 class="modal-header">New transaction</h2>
    <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
    <div class="modal-body">
      <%= render 'form', transaction: @transaction %>
    </div>
  <% end %>
</div>
```

Додаток Б

```
class AccountsController < ApplicationController
  before_action :set_account, only: [:show, :edit, :update,
:destroy]

  def index
    @accounts = current_user.accounts
  end

  def show
  end

  def new
    @account = Account.new
  end

  def create
    @account = Account.new(account_params)
    @account.user = current_user

    if @account.save
      redirect_to accounts_path, notice: "Account was succesfully
created."
    else
      render :new, status: :unprocessable_entity
    end
  end

  def edit
  end

  def update
    if @account.update(account_params)
      redirect_to accounts_path, notice: "Account was succesfully
updated."
    else
      render :edit, status: :unprocessable_entity
    end
  end

  def destroy
    @account.destroy
    redirect_to accounts_path, notice: "Account was deleted
successfully."
  end

  private
  def account_params
    params.require(:account).permit(:name, :account_type,
:account_number, :bank_name, :owner_name, :current_balance)
```

```
  end

  def set_account
    @account = Account.find(params[:id])
  end
end

class CategoriesController < ApplicationController
  before_action :set_category, only: [:show, :edit, :update,
:destroy]

  def index
    @categories = current_user.categories.page(params[:page])
  end

  def show
  end

  def new
    @category = Category.new
  end

  def create
    @category = Category.new(category_params)
    @category.user = current_user
    if @category.save
      redirect_to categories_path, notice: "Category was succesfully
created."
    else
      render :new, status: :unprocessable_entity
    end
  end

  def edit
  end

  def update
    if @category.update(category_params)
      redirect_to categories_path, notice: "Category was succesfully
updated."
    else
      render :edit, status: :unprocessable_entity
    end
  end

  def destroy
    @category.destroy
    redirect_to categories_path, notice: "Category was successfully
destroyed."
  end

  private
```

```
  def category_params
    params.require(:category).permit(:name)
  end

  def set_category
    @category = Category.find(params[:id])
  end
end

class DashboardController < ApplicationController

  def show
    @debits                                                    =
current_user.transactions.transaction_type_debit.group_by_month(:dat
e, format: "%b %Y").sum(:amount)
    @credits                                                   =
current_user.transactions.transaction_type_credit.group_by_month(:da
te, format: "%b %Y").sum(:amount)

    @labels = @debits.keys
    @series = [@debits.values, @credits.values]

    category_chart_data                                        =
current_user.transactions.group(:category).sum(:amount)
    @category_chart_labels = category_chart_data.keys.map(&:name)
    @category_chart_series = category_chart_data.values

    @transactions = current_user.transactions.first(10)
  end
end

class TransactionsController < ApplicationController
  before_action  :set_transaction,  only:  [:show,  :edit,  :update,
:destroy]

  def index
    @transactions                                              =
current_user.transactions.ordered.page(params[:page])
  end

  def show
  end

  def new
    @transaction = Transaction.new
  end

  def create
    @transaction = Transaction.new(transaction_params)
    @transaction.category     =     Category.find_or_create_by(name:
'Uncategorized',          user:          current_user)          if
params[:transaction][:category].nil?
    if @transaction.save
```

```
      Transaction.update_account_balance(@transaction)
      redirect_to transactions_path, notice: "Transaction was
succesfully created."
    else
      render :new, status: :unprocessable_entity
    end
  end

  def edit
  end

  def update
    if @transaction.update(transaction_params)
      redirect_to transactions_path, notice: "Transaction was
succesfully updated."
    else
      render :edit, status: :unprocessable_entity
    end
  end

  def destroy
    @transaction.destroy
    redirect_to transactions_path, status: :see_other, alert:
"Transaction was deleted successfully."
  end

  def read
    @transaction = Transaction.new
  end

  def import
    file = params[:transaction][:file]
    if file.content_type == 'text/csv'
      ImportTransactionsService.new.call(file,
params[:transaction][:account_id])
      redirect_to transactions_path, notice: 'Transactions imported
successfully.'
    else
      redirect_to new_transaction_path, notice: 'Only CSV files
accepted.'
    end
  end

  private

  def transaction_params
    params.require(:transaction).permit(:date, :description,
:transaction_type, :amount, :notes, :account_id, :category_id,
:file)
  end

  def set_transaction
    @transaction = Transaction.find(params[:id])
```