МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра_____комп'ютерних_інформаційних_технологій_____

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО
«_____» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ "БАКАЛАВР"

Тема: _____«Розробка веб-сайту для управління особистими фінансами»_____

Виконавець: ___студент групи УС-414Б Годісь Павло Романович_____

Керівник: ____к.т.н., доцент_Харченко Олександр Григорович_____

Нормоконтролер: _____Олександр ШЕВЧЕНКО_____

Київ 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук та технологій

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 «Інформаційні технології», 122 «Комп'ютерні науки», «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО
«_____» _____2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи (проєкту)
Годіся Павла Романовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. **Тема кваліфікаційної роботи (проєкту):** «Розробка веб-сайту для управління особистими фінансами» затверджена наказом ректора від «05» 04 2024 р. № 517/ст.
2. **Термін виконання роботи (проєкту):** з 06.05.2024 по 10.06.2024
3. **Вихідні дані до роботи (проєкту):** PhpStorm, Apache2, Nginx, Docker, HTML, PHP, MySQL, CSS, Bootstrap, JavaScript.
4. **Зміст пояснювальної записки:** аналіз предметної області, проектування веб-сайту з використанням html, php, css та mysql, реалізація веб-сайту для управління особистими фінансами, висновки.
5. **Перелік обов'язкового графічного (ілюстративного) матеріалу:** зображення з виглядами використовуваних програмних забезпечень, зображення структури файлів системи, зображення діаграми бази даних, зображення елементів веб-сайту, зображення сторінок веб-сайту.

6. Календарний план-графік

№	Завдання	Термін виконання	Відмітка про виконання
1	Отримання завдання на дипломну роботу, створення плану дипломної роботи та побудова плану-графіку виконання робіт.	06.05.2024 – 07.05.2024	
2	Розроблення та затвердження календарного плану виконання дипломної роботи.	08.05.2024 – 09.05.2024	
3	Проведення консультацій з науковим керівником.	10.05.2024 – 12.05.2024	
4	Огляд та аналіз наукової літератури по темі дипломної роботи та написання Розділу 1.	13.05.2024 – 17.05.2024	
5	Написання Розділу 2 дипломної роботи.	18.05.2024 – 24.05.2024	
6	Написання Розділу 3 дипломної роботи. Завершення створення пояснювальної записки дипломної роботи.	25.05.2024 – 29.05.2024	
7	Оформлення та друк пояснювальної записки.	30.05.2024 – 01.06.2024	
8	Створення презентації, доповіді та підготовка до захисту дипломної роботи.	02.06.2024 – 04.06.2024	
9	Підготовка матеріалів дипломної роботи для передачі секретарю ДЕК (папка, конверт, диск із файлом диплому, рецензія, відгук).	05.06.2024 – 07.06.2024	

Дата видачі завдання:_____06.05.2024 р._____

Керівник дипломної роботи(проєкту): _____ _Олександр ХАРЧЕНКО
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання: _____ _____Павло ГОДІСЬ____
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Розробка веб-сайту для управління особистими фінансами»: 71 с., 16 рис., 4 табл., 26 літературних джерел.

Об’єкт дослідження: процес управління особистими фінансами.

Предмет дослідження: методи та інструменти веб-розробки для створення систем управління особистими фінансами.

Мета роботи: розробка веб-сайту для управління особистими фінансами, який надасть користувачам інструмент для ефективного планування, управління та моніторингу власних фінансових операцій.

Методи дослідження, технічні та програмні засоби: обробка літературних джерел, порівняльний аналіз систем управління фінансами, проектування архітектури веб-додатку, розробка інтерфейсів користувача за допомогою HTML, CSS і Bootstrap, програмування серверної частини на PHP, управління базами даних MySQL, інтеграція JavaScript для динамічного користувацького досвіду, тестування та оцінка якості програмного забезпечення за допомогою юніт- і інтеграційного тестування.

Наукова новизна роботи полягає у розробці інтегрованої системи управління особистими фінансами з трекінгом витрат, бюджетуванням, плануванням фінансових цілей та інтуїтивно зрозумілим інтерфейсом.

Результат роботи: створено веб-сайт для управління особистими фінансами, який надає користувачам можливість трекінгу витрат, бюджетування та планування фінансових цілей. Сайт забезпечує зручний та інтуїтивний інтерфейс користувача та інструменти для аналізу фінансової інформації.

Матеріали кваліфікаційної роботи можуть бути використані при розробці подібних систем управління фінансами, а також для подальших досліджень та вдосконалення існуючих рішень у цій галузі.

ВЕБ-САЙТ, ОСОБИСТІ ФІНАНСИ, ТРЕКІНГ ВИТРАТ, БЮДЖЕТУВАННЯ, ФІНАНСОВІ ЦІЛІ.

ЗМІСТ

ВСТУП

Актуальність теми кваліфікаційної роботи "Розробка веб-сайту для управління особистими фінансами" зумовлена стрімким розвитком інформаційних технологій та зростаючою потребою людей в ефективних інструментах для контролю та управління власними коштами. В умовах динамічних економічних процесів, збільшення кількості фінансових операцій та необхідності раціонального розподілу бюджету, питання оптимізації управління особистими фінансами набуває особливої значущості. Традиційні методи, такі як ведення паперових записів чи використання електронних таблиць, часто виявляються недостатньо зручними та функціональними. Саме тому розробка спеціалізованого веб-сайту, який би дозволив користувачам зручно та ефективно управляти своїми фінансами, є актуальним та затребуваним завданням.

Веб-сайт для управління особистими фінансами повинен надавати користувачам широкий спектр можливостей, таких як реєстрація та авторизація в системі, додавання та відстеження доходів і витрат, складання особистих бюджетів, візуалізація фінансових даних у вигляді графіків та діаграм, а також аналіз та порівняння фінансових показників за різними періодами. Такий функціонал дозволить користувачам отримати повну картину своїх фінансів, виявити проблемні зони та прийняти обґрунтовані рішення щодо оптимізації витрат та збільшення доходів.

Використання веб-технологій для створення такого сайту має низку переваг. По-перше, це забезпечує доступність фінансової інформації з будь-якого місця, де є підключення до інтернету. Користувачі зможуть вносити дані про свої транзакції та переглядати звіти в режимі реального часу, незалежно від свого місцезнаходження. По-друге, веб-сайт дозволяє автоматизувати багато рутинних операцій, таких як розрахунок балансу, формування звітів, нагадування про заплановані платежі тощо. Це заощаджує час користувачів та мінімізує ймовірність помилок, які можуть виникнути при ручному веденні фінансового обліку.

Для розробки веб-сайту було обрано стек технологій, що включає HTML, PHP, CSS та MySQL. HTML використовується для створення структури веб-сторінок та розмітки контенту. PHP є серверною мовою програмування, яка дозволяє реалізувати динамічну обробку даних та взаємодію з базою даних. CSS відповідає за візуальне оформлення сторінок, забезпечуючи привабливий та інтуїтивно зрозумілий інтерфейс користувача. MySQL, в свою чергу, виступає в ролі системи управління базами даних, забезпечуючи надійне зберігання та ефективну обробку фінансової інформації.

Об'єктом дослідження в рамках даної кваліфікаційної роботи є процес управління особистими фінансами за допомогою веб-технологій, а предметом дослідження виступає безпосередньо веб-сайт для управління особистими фінансами, розроблений з використанням HTML, PHP, CSS та MySQL.

Метою роботи є розробка функціонального та зручного у використанні веб-сайту для управління особистими фінансами, який дозволить користувачам ефективно контролювати свої доходи та витрати, планувати бюджет та аналізувати фінансові показники. Для досягнення поставленої мети необхідно вирішити ряд завдань, зокрема: провести аналіз предметної області та існуючих рішень, сформувати вимоги до розроблюваного веб-сайту, обґрунтувати вибір технологій, спроектувати архітектуру та структуру бази даних, реалізувати необхідний функціонал та провести тестування розробленого продукту.

В процесі виконання роботи були використані такі методи дослідження, як аналіз предметної області, моделювання архітектури веб-сайту, методи веб-програмування з використанням обраного стеку технологій, а також тестування та налагодження розробленого програмного забезпечення.

Практичне значення отриманих результатів полягає у створенні готового до використання веб-сайту для управління особистими фінансами, який може бути корисним широкому колу користувачів, які прагнуть ефективніше контролювати свої доходи та витрати, планувати бюджет та приймати обґрунтовані фінансові рішення. Розроблений веб-сайт може стати зручним інструментом для покращення фінансової грамотності населення та сприяти формуванню раціональних підходів до управління особистими коштами.

Особистий внесок випускника полягає в самостійному виконанні всіх етапів розробки веб-сайту, включаючи аналіз вимог, проектування архітектури, реалізацію функціональних можливостей та тестування готового продукту. Всі результати, представлені в роботі, отримані випускником особисто.

Апробація отриманих результатів відбувалась шляхом представлення окремих аспектів розробленого веб-сайту на студентських наукових конференціях та семінарах, де були отримані позитивні відгуки та цінні рекомендації щодо подальшого вдосконалення розробки.

Таким чином, розробка веб-сайту для управління особистими фінансами є актуальним та практично значущим завданням, яке потребує комплексного підходу та застосування сучасних веб-технологій. Запропоноване в рамках даної кваліфікаційної роботи рішення має потенціал для широкого практичного використання та може стати ефективним інструментом для покращення якості управління особистими фінансами для багатьох користувачів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд існуючих рішень для управління особистими фінансами

В сучасному світі управління особистими фінансами є важливою складовою життя кожної людини. Для того, щоб ефективно контролювати свої доходи та витрати, планувати бюджет та досягати фінансових цілей, люди використовують різноманітні інструменти та програмні рішення. В цьому розділі ми розглянемо деякі з найбільш популярних та ефективних рішень для управління особистими фінансами, які доступні на ринку.

Одним з найвідоміших додатків для управління фінансами є Mint. Цей додаток, розроблений компанією Intuit, пропонує користувачам можливість автоматичного імпорту фінансових даних з банківських рахунків, кредитних карток та інвестиційних рахунків. Mint дозволяє відстежувати доходи та витрати, категоризувати транзакції, складати бюджети та отримувати персоналізовані поради щодо покращення фінансового стану. Додаток має зручний та інтуїтивно зрозумілий інтерфейс, а також надає користувачам регулярні звіти та сповіщення про стан їхніх фінансів.

Рис. 1.1. Логотип компанії Mint

Ще одним популярним рішенням є YNAB (You Need A Budget). Цей веб-сайт та мобільний додаток базуються на власній методології бюджетування, яка допомагає користувачам розподіляти кожну зароблену гривню на конкретні цілі та категорії витрат. YNAB дозволяє користувачам вводити доходи та витрати вручну або імпортувати їх з банківських рахунків. Додаток надає детальні звіти та графіки, що відображають прогрес у досягненні фінансових цілей та дотриманні бюджету. YNAB також пропонує освітні ресурси та підтримку спільноти, щоб допомогти користувачам покращити свої навички управління грошима.

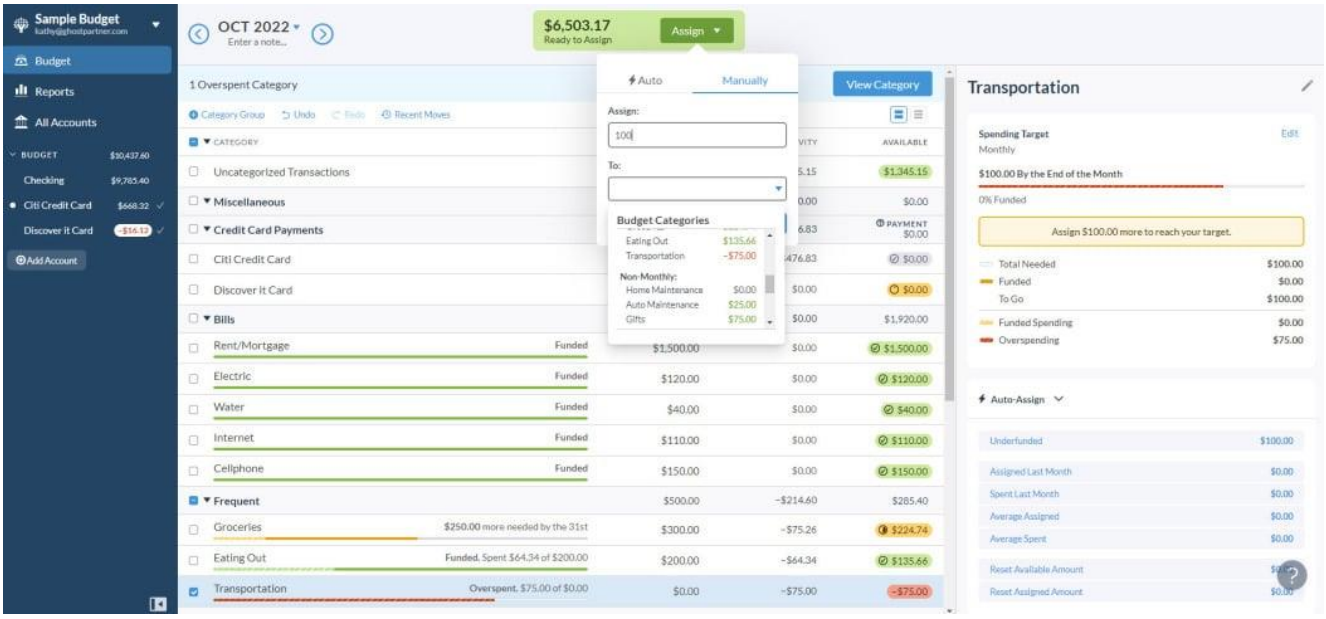

Рис. 1.2. Вигляд додатку YNAB

Для тих, хто надає перевагу простішим рішенням, існує додаток Goodbudget. Цей додаток використовує концепцію "конвертів" для розподілу коштів на різні категорії витрат. Користувачі можуть створювати віртуальні конверти для різних цілей, таких як житло, транспорт, їжа та розваги, і розподіляти свій дохід між цими конвертами. Goodbudget дозволяє відстежувати витрати в режимі реального часу та надає звіти, що показують, скільки грошей залишилося в кожному конверті. Додаток також підтримує синхронізацію даних між кількома пристроями, що дозволяє користувачам спільно управляти бюджетом з партнерами або членами сім'ї.

Рис. 1.3. Вигляд додатку Goodbudget

Для користувачів, які шукають більш потужні інструменти для інвестицій та управління капіталом, існує веб-сайт Personal Capital. Цей сервіс дозволяє користувачам об'єднати всі свої інвестиційні рахунки, пенсійні плани, банківські рахунки та кредитні картки в одному місці. Personal Capital надає детальний аналіз інвестиційного портфеля, включаючи розподіл активів, диверсифікацію та плату за управління. Веб-сайт також пропонує персоналізовані інвестиційні поради та консультації з фінансовими експертами. Окрім інвестиційних інструментів, Personal Capital також дозволяє користувачам відстежувати свої витрати, складати бюджети та планувати довгострокові фінансові цілі, такі як купівля нерухомості або вихід на пенсію.

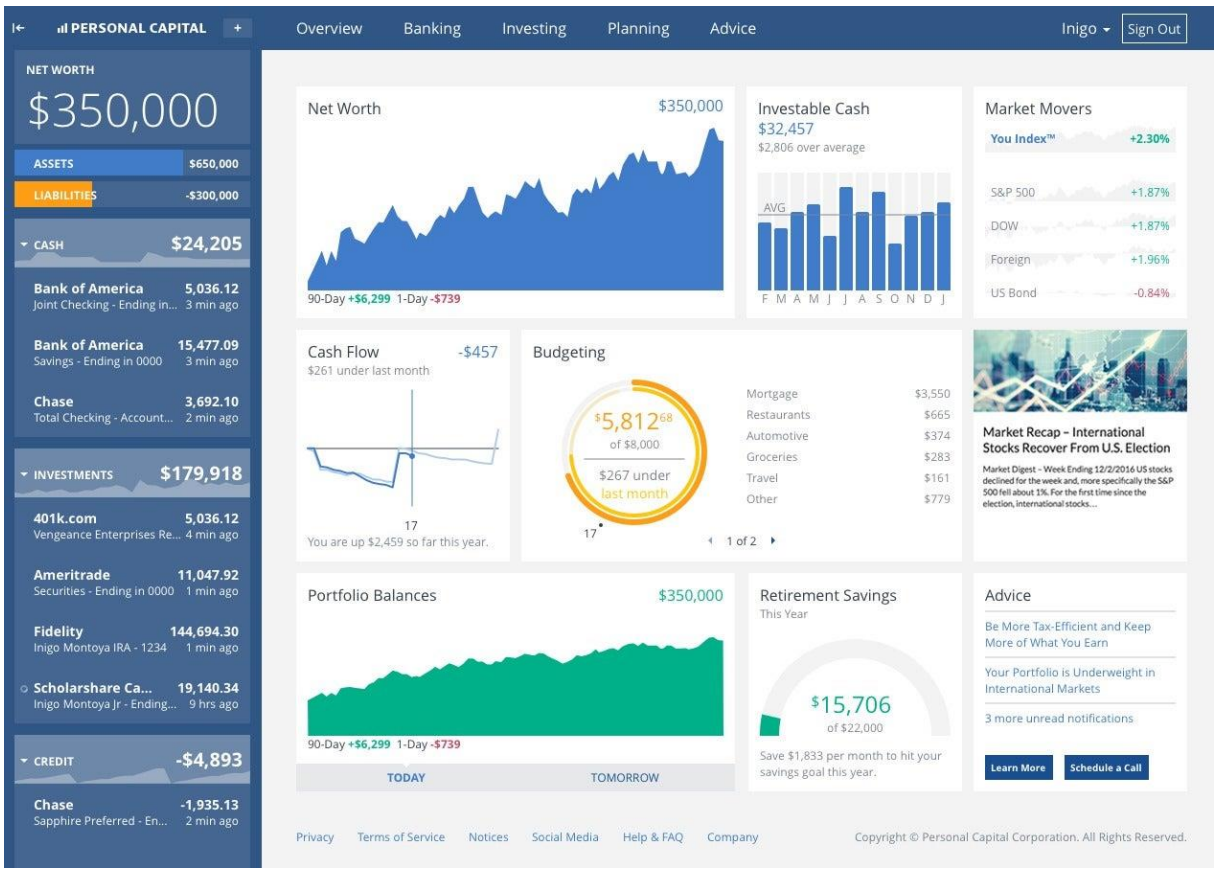

Рис. 1.4. Вигляд додатку Personal Capital

Ще одним цікавим рішенням є додаток Spendee. Цей додаток дозволяє користувачам легко відстежувати свої витрати за допомогою простого та привабливого інтерфейсу. Spendee автоматично категоризує транзакції на основі алгоритмів машинного навчання, що спрощує процес бюджетування. Додаток також дозволяє користувачам встановлювати ліміти витрат для кожної категорії та отримувати сповіщення, коли вони наближаються до цих лімітів. Spendee підтримує широкий спектр валют та може синхронізуватися з банківськими рахунками з різних країн світу. Додаток також надає детальні звіти та аналітику витрат, допомагаючи користувачам визначити області, де вони можуть заощадити гроші.

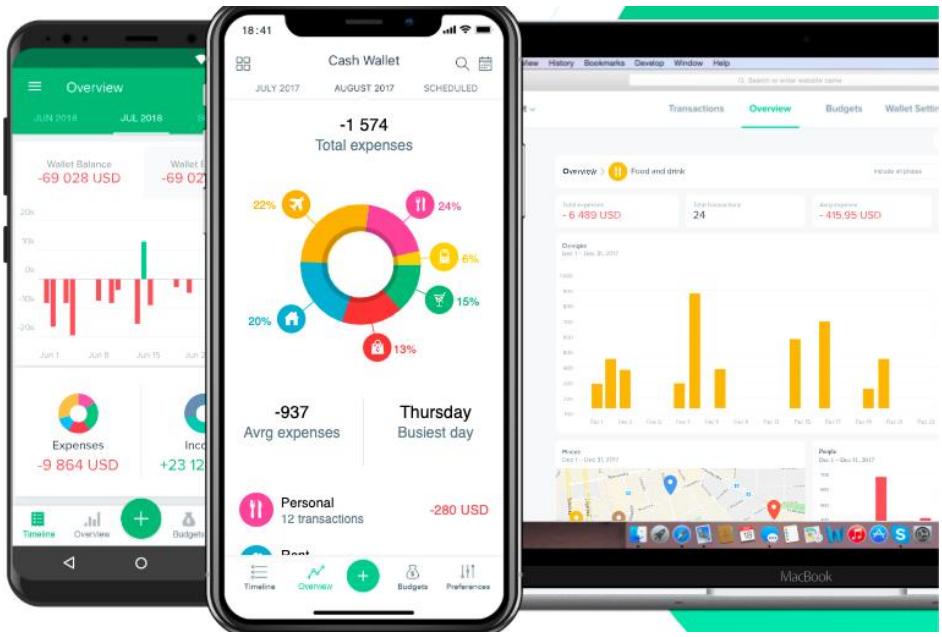

Рис. 1.5. Вигляд додатку Spendee

Незважаючи на те, що всі ці рішення мають свої переваги та особливості, вони всі спрямовані на те, щоб допомогти користувачам краще зрозуміти свої фінанси та приймати обґрунтовані рішення щодо управління грошима. Вибір правильного інструменту залежить від індивідуальних потреб та уподобань користувача. Деякі люди можуть віддати перевагу автоматизованим рішенням, які синхронізуються з їхніми банківськими рахунками, тоді як інші можуть надати перевагу ручному введенню даних для більшого контролю та усвідомлення своїх витрат. Незалежно від вибору, використання спеціалізованого програмного забезпечення для управління фінансами може допомогти користувачам заощаджувати гроші, уникати боргів та досягати своїх фінансових цілей більш ефективно.

В ході аналізу існуючих рішень для управління особистими фінансами було виявлено, що на ринку представлено багато якісних та функціональних додатків та веб-сайтів. Кожен з них має свої особливості та переваги, які можуть задовольнити потреби різних груп користувачів. Однак, розробка власного веб-сайту для управління фінансами все ще може бути актуальною, оскільки це дозволить врахувати специфічні вимоги та адаптувати функціонал під потреби конкретної аудиторії. В наступному розділі ми детальніше розглянемо функціональні та нефункціональні вимоги до нашого веб-сайту на основі аналізу існуючих рішень та потреб потенційних користувачів.

Таблиця 1.1

Порівняння характеристик рішень для управління особистими фінансами

Характеристика	**Mint**	**YNAB**	**Goodbudget**	**Personal Capital**	**Spendee**
Автоматичний імпорт даних	✓	✓	×	✓	✓
Ручне введення транзакцій	✓	✓	✓	✓	✓
Категоризація транзакцій	✓	✓	✓	✓	✓
Складання бюджетів	✓	✓	✓	✓	✓
Звіти та аналітика	✓	✓	✓	✓	✓
Сповіщення та нагадування	✓	✓	✓	✓	✓
Синхронізація між пристроями	✓	✓	✓	✓	✓
Можливості для інвестицій	×	×	×	✓	×
Освітні ресурси	✓	✓	×	✓	×
Підтримка різних валют	×	×	×	×	✓

З таблиці можна зробити висновок, що всі розглянуті рішення мають базовий набір функцій, необхідних для ефективного управління фінансами, такі як складання бюджетів, категоризація транзакцій та генерація звітів. Однак, деякі додатки, такі як Personal Capital, пропонують додаткові можливості для інвестицій та управління капіталом. Інші, як Spendee, підтримують різні валюти, що може бути корисним для користувачів, які часто подорожують або мають міжнародні транзакції.

При розробці власного веб-сайту для управління фінансами важливо врахувати ці особливості та визначити, які функції будуть найбільш затребуваними та корисними для цільової аудиторії. Це дозволить створити продукт, який буде виділятися на ринку та задовольняти потреби користувачів у зручному та ефективному управлінні особистими фінансами.

1.2. Аналіз функціональних та нефункціональних вимог до веб-сайту для управління фінансами

Для успішної розробки веб-сайту для управління особистими фінансами необхідно чітко визначити та проаналізувати функціональні та нефункціональні вимоги. Це дозволить створити продукт, який буде відповідати потребам користувачів та забезпечувати ефективне та зручне управління фінансами.

Функціональні вимоги описують основні функції та можливості, які повинен мати веб-сайт для управління фінансами. Ці вимоги базуються на потребах користувачів та аналізі існуючих рішень на ринку. Основні функціональні вимоги до нашого веб-сайту включають:

1. Реєстрація та авторизація користувачів: веб-сайт повинен надавати можливість створення особистих облікових записів та безпечної авторизації користувачів для доступу до їхніх фінансових даних.
2. Додавання та редагування транзакцій: користувачі повинні мати можливість вручну додавати та редагувати транзакції, вказуючи дату, категорію, суму та опис.
3. Автоматичний імпорт транзакцій: веб-сайт повинен підтримувати автоматичний імпорт транзакцій з банківських рахунків та кредитних карток користувачів для зручності та точності відстеження витрат.
4. Категоризація транзакцій: веб-сайт повинен автоматично категоризувати транзакції на основі попередньо визначених категорій або дозволяти користувачам створювати власні категорії для кращої організації фінансових даних.
5. Складання бюджетів: користувачі повинні мати можливість створювати та управляти особистими бюджетами, встановлюючи ліміти витрат для кожної категорії та відстежуючи прогрес у режимі реального часу.
6. Генерація звітів та аналітика: веб-сайт повинен надавати користувачам детальні звіти та аналітику щодо їхніх доходів, витрат та бюджетів, допомагаючи їм краще розуміти свої фінансові звички та прогрес.

7. Сповіщення та нагадування: веб-сайт повинен надсилати користувачам сповіщення та нагадування про наближення до лімітів бюджету, майбутні платежі або досягнення фінансових цілей.
8. Інтеграція з іншими сервісами: для розширення функціональності та зручності користувачів веб-сайт може інтегруватися з іншими фінансовими сервісами, такими як платіжні системи або інвестиційні платформи.

Окрім функціональних вимог, важливо також врахувати нефункціональні вимоги, які визначають якісні характеристики веб-сайту. Основні нефункціональні вимоги до нашого веб-сайту включають:

1. Безпека: веб-сайт повинен забезпечувати високий рівень безпеки для захисту фінансових даних користувачів, використовуючи шифрування, безпечні протоколи передачі даних та регулярні оновлення системи безпеки.
2. Продуктивність: веб-сайт повинен мати швидкий час завантаження сторінок, плавну навігацію та ефективну обробку даних для забезпечення позитивного користувацького досвіду.
3. Масштабованість: архітектура веб-сайту повинна бути розроблена таким чином, щоб він міг обробляти зростаючу кількість користувачів та даних без втрати продуктивності.
4. Зручність використання: інтерфейс веб-сайту повинен бути інтуїтивно зрозумілим, з чіткою навігацією та логічною структурою, щоб користувачі могли легко керувати своїми фінансами.
5. Адаптивний дизайн: веб-сайт повинен мати адаптивний дизайн, який забезпечує коректне відображення та функціональність на різних пристроях, включаючи настільні комп'ютери, планшети та смартфони.
6. Доступність: веб-сайт повинен відповідати стандартам доступності, щоб його могли використовувати люди з обмеженими можливостями, забезпечуючи альтернативні способи введення даних та навігації.

7. Локалізація: для охоплення ширшої аудиторії веб-сайт може підтримувати кілька мов та валют, адаптуючись до потреб користувачів з різних країн та регіонів.

Для кращої візуалізації та структурування вимог створимо таблицю, яка відображає основні функціональні та нефункціональні вимоги до веб-сайту для управління фінансами:

Таблиця 1.2

Основні функціональні та нефункціональні вимоги до веб-сайту для управління фінансами

Функціональні вимоги	**Нефункціональні вимоги**
Реєстрація та авторизація користувачів	Безпека
Додавання та редагування транзакцій	Продуктивність
Автоматичний імпорт транзакцій	Масштабованість
Категоризація транзакцій	Зручність використання
Складання бюджетів	Адаптивний дизайн
Генерація звітів та аналітика	Доступність
Сповіщення та нагадування	Локалізація
Інтеграція з іншими сервісами	

Аналіз функціональних та нефункціональних вимог є важливим етапом у процесі розробки веб-сайту для управління фінансами. Він дозволяє чітко визначити очікування та потреби користувачів, а також забезпечити відповідність веб-сайту високим стандартам якості та ефективності. Врахування цих вимог під час проектування та розробки веб-сайту дозволить створити продукт, який буде корисним, зручним та надійним інструментом для управління особистими фінансами.

1.3. Формування завдання на розробку веб-сайту на основі сформованих вимог

На основі аналізу предметної області та сформованих функціональних і нефункціональних вимог до веб-сайту для управління особистими фінансами, було визначено основні завдання та цілі розробки. У цьому підрозділі ми детально розглянемо процес формування завдання на розробку веб-сайту, враховуючи потреби користувачів та технічні аспекти реалізації.

Головною метою розробки веб-сайту для управління особистими фінансами є створення зручного та ефективного інструменту, який допоможе користувачам контролювати свої доходи, витрати, планувати бюджет та аналізувати фінансову діяльність. Веб-сайт повинен надавати користувачам можливість вести облік своїх транзакцій, категоризувати їх, генерувати звіти та отримувати візуальне представлення своїх фінансових даних.

Завдання на розробку веб-сайту включає наступні ключові аспекти:

1. Реєстрація та автентифікація користувачів:
    - Розробити систему реєстрації користувачів, яка дозволить створювати нові облікові записи з унікальними іменами користувачів та безпечними паролями.
    - Реалізувати механізм автентифікації користувачів, який забезпечить захищений вхід до системи та збереження сеансу користувача.
    - Передбачити можливість відновлення забутого пароля через електронну пошту.
2. Управління транзакціями:
    - Створити функціонал для додавання, редагування та видалення транзакцій користувачів.
    - Забезпечити можливість категоризації транзакцій за типами (доходи, витрати) та користувацькими категоріями.
    - Реалізувати пошук та фільтрацію транзакцій за різними критеріями (дата, категорія, сума тощо).

- Передбачити можливість імпорту транзакцій з зовнішніх файлів (наприклад, CSV) для зручності користувачів.

3. Категоризація та управління категоріями:
   - Розробити систему категоризації транзакцій, яка дозволить користувачам створювати власні категорії та підкатегорії.
   - Забезпечити можливість редагування та видалення користувацьких категорій.
   - Реалізувати функціонал для встановлення бюджетів для кожної категорії та відстеження їх виконання.
4. Генерація звітів та візуалізація даних:
   - Створити функціонал для генерації звітів за певний період (день, тиждень, місяць, рік) з можливістю вибору користувачем.
   - Забезпечити відображення звітів у вигляді таблиць та графіків для кращого сприйняття фінансової інформації.
   - Реалізувати можливість експорту звітів у різних форматах (PDF, CSV, Excel) для подальшого використання.
   - Розробити інтерактивні візуалізації даних, такі як діаграми розподілу витрат за категоріями, графіки динаміки доходів і витрат тощо.
5. Планування бюджету та фінансових цілей:
   - Створити інструменти для планування бюджету на основі доходів і витрат користувача.
   - Забезпечити можливість встановлення фінансових цілей (наприклад, накопичення певної суми) та відстеження прогресу їх досягнення.
   - Реалізувати сповіщення та нагадування про наближення до встановлених цілей або перевищення бюджетних лімітів.
6. Безпека та конфіденційність даних:
   - Забезпечити високий рівень безпеки веб-сайту, використовуючи шифрування даних, захищені протоколи передачі даних (HTTPS) та надійні методи зберігання паролів.

- Реалізувати механізми захисту від поширених веб-вразливостей, таких як SQL-ін'єкції та міжсайтовий скриптинг (XSS).
- Забезпечити конфіденційність фінансових даних користувачів та дотримуватися вимог законодавства щодо захисту персональних даних.

7. Користувацький інтерфейс та досвід користувача (UI/UX):
   - Розробити інтуїтивно зрозумілий та привабливий користувацький інтерфейс, який забезпечить зручну навігацію по веб-сайту.
   - Забезпечити адаптивність дизайну для коректного відображення на різних пристроях (настільні комп'ютери, планшети, смартфони).
   - Оптимізувати швидкість завантаження сторінок та забезпечити плавну роботу веб-сайту.
   - Провести юзабіліті-тестування для виявлення та усунення потенційних проблем зручності використання.
8. Інтеграція з іншими сервісами:
   - Передбачити можливість інтеграції веб-сайту з популярними платіжними системами для автоматичного імпорту транзакцій.
   - Реалізувати інтеграцію з банківськими сервісами для синхронізації даних про рахунки та транзакції користувача.
   - Розглянути можливість інтеграції з іншими фінансовими інструментами та додатками для розширення функціональності веб-сайту.
9. Підтримка та супровід:
   - Забезпечити постійну підтримку та оновлення веб-сайту після його запуску.
   - Здійснювати моніторинг роботи веб-сайту, виявляти та оперативно усувати помилки та неполадки.
   - Збирати відгуки користувачів та враховувати їх побажання для подальшого вдосконалення функціональності та зручності використання веб-сайту.

Для успішної реалізації поставлених завдань необхідно провести ретельне планування та розробку веб-сайту з використанням сучасних веб-технологій та інструментів. Варто обрати відповідний стек технологій, який забезпечить надійність, масштабованість та безпеку веб-сайту. Для розробки серверної частини можна використовувати мову програмування PHP у поєднанні з фреймворком Laravel або інші популярні рішення, такі як Node.js з Express.js. Для клієнтської частини доцільно використовувати HTML, CSS та JavaScript з можливим залученням фреймворків, таких як React, Angular або Vue.js, для створення інтерактивних та динамічних компонентів.

Важливо також приділити увагу проектуванню бази даних, яка буде зберігати фінансові дані користувачів. Необхідно розробити ефективну схему бази даних з урахуванням нормалізації та оптимізації запитів. Для роботи з базою даних можна використовувати реляційну СУБД, таку як MySQL або PostgreSQL.

При розробці веб-сайту необхідно дотримуватися принципів безпеки та конфіденційності даних. Це включає використання надійних методів шифрування, захист від поширених веб-вразливостей, регулярне оновлення програмного забезпечення та дотримання стандартів безпеки при розробці.

Для забезпечення якості та надійності веб-сайту необхідно провести ретельне тестування на різних етапах розробки. Це включає модульне тестування окремих компонентів, інтеграційне тестування для перевірки взаємодії між компонентами, а також користувацьке тестування для оцінки зручності використання та виявлення потенційних проблем.

Окрім функціональних вимог, важливо також враховувати нефункціональні вимоги, такі як продуктивність, масштабованість та доступність веб-сайту. Необхідно забезпечити оптимальну швидкість завантаження сторінок, ефективне використання ресурсів сервера та можливість обробки зростаючого навантаження зі збільшенням кількості користувачів.

Для успішного виконання завдання на розробку веб-сайту необхідно також забезпечити ефективну комунікацію та співпрацю між замовником та командою розробників. Регулярні зустрічі та обговорення вимог, дизайну та функціональності

допоможуть досягти розуміння потреб користувачів та забезпечити відповідність веб-сайту очікуванням замовника.

Таким чином, формування завдання на розробку веб-сайту для управління особистими фінансами на основі сформованих вимог є важливим етапом, який визначає основні цілі, функціональні можливості та технічні аспекти реалізації проекту. Чітке розуміння потреб користувачів, врахування принципів безпеки та конфіденційності даних, а також вибір відповідних технологій та інструментів дозволять створити ефективний та зручний у використанні веб-сайт, який допоможе користувачам контролювати та управляти своїми фінансами.

1.4. Висновки до розділу

У першому розділі було проведено ґрунтовний аналіз предметної області розробки веб-сайту для управління особистими фінансами. Цей аналіз включав огляд існуючих рішень на ринку, визначення функціональних та нефункціональних вимог до веб-сайту, а також формування завдання на розробку.

Під час огляду існуючих рішень для управління особистими фінансами було розглянуто кілька популярних додатків та веб-сайтів, таких як Mint, YNAB, Goodbudget, Personal Capital та Spendee. Кожне з цих рішень має свої особливості та переваги, які задовольняють потреби різних груп користувачів. Було виявлено, що більшість додатків мають базовий набір функцій, таких як відстеження транзакцій, категоризація витрат, складання бюджетів та генерація звітів. Деякі рішення також пропонують додаткові можливості, такі як інвестиційні інструменти або підтримка різних валют.

Аналіз функціональних вимог до веб-сайту для управління фінансами дозволив визначити основні функції та можливості, які необхідні для ефективного управління особистими фінансами. Ці вимоги включають реєстрацію та авторизацію користувачів, додавання та редагування транзакцій, автоматичний імпорт даних, категоризацію транзакцій, складання бюджетів, генерацію звітів та аналітику, сповіщення та нагадування, а також інтеграцію з іншими фінансовими сервісами.

Окрім функціональних вимог, було також визначено нефункціональні вимоги,

які впливають на якість та ефективність веб-сайту. Ці вимоги включають безпеку, продуктивність, масштабованість, зручність використання, адаптивний дизайн, доступність та локалізацію. Врахування цих вимог є важливим для створення надійного та зручного інструменту для управління фінансами.

На основі проведеного аналізу було сформовано завдання на розробку веб-сайту для управління особистими фінансами. Це завдання включає реалізацію основних функціональних можливостей, таких як реєстрація користувачів, відстеження доходів і витрат, складання бюджетів та генерація звітів. Також важливо забезпечити високий рівень безпеки, продуктивності та зручності використання веб-сайту.

Результати аналізу предметної області, проведеного в першому розділі, закладають міцний фундамент для подальшої розробки веб-сайту для управління особистими фінансами. Визначені функціональні та нефункціональні вимоги будуть використані як основа для проектування архітектури, бази даних та користувацького інтерфейсу веб-сайту. Врахування досвіду існуючих рішень на ринку дозволить створити продукт, який буде виділятися своїми перевагами та задовольняти потреби цільової аудиторії.

У наступних розділах кваліфікаційної роботи буде детально розглянуто процес проектування та розробки веб-сайту з використанням обраних технологій, таких як PHP, MySQL, HTML та CSS. Буде приділено увагу реалізації визначених функціональних можливостей, забезпеченню безпеки та оптимізації продуктивності веб-сайту. Також буде розроблено зручний та інтуїтивно зрозумілий користувацький інтерфейс, який дозволить користувачам ефективно керувати своїми фінансами.

Таким чином, перший розділ кваліфікаційної роботи заклав міцне підґрунтя для успішної розробки веб-сайту для управління особистими фінансами. Проведений аналіз предметної області дозволив визначити ключові вимоги та завдання, які будуть реалізовані в процесі розробки. Це забезпечить створення якісного та конкурентоспроможного продукту, який буде корисним інструментом для ефективного управління особистими фінансами.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ВЕБ-САЙТУ З ВИКОРИСТАННЯМ HTML, PHP, CSS ТА MYSQL

2.1. Проектування архітектури веб-сайту на базі PHP

Проектування архітектури веб-сайту є важливим етапом розробки, який дозволяє визначити структуру та компоненти системи, а також забезпечити її масштабованість, гнучкість та зручність подальшої підтримки. В даному розділі буде розглянуто проектування архітектури веб-сайту для управління особистими фінансами на базі мови програмування PHP.

PHP (Hypertext Preprocessor) - це широко використовувана серверна мова програмування, яка спеціально розроблена для веб-розробки. PHP дозволяє створювати динамічні веб-сторінки, обробляти дані форм, взаємодіяти з базами даних та виконувати різноманітні операції на сервері перед відправкою результатів клієнту.

Для проектування архітектури веб-сайту на базі PHP було обрано популярний шаблон проектування Model-View-Controller (MVC). MVC - це архітектурний шаблон, який розділяє систему на три основні компоненти: модель (Model), представлення (View) та контролер (Controller). Кожен з цих компонентів має свою чітко визначену роль та відповідальність.

Модель (Model) відповідає за роботу з даними та бізнес-логікою системи. Вона включає в себе класи та функції, які взаємодіють з базою даних, виконують операції з даними та реалізують основні функціональні можливості системи. В контексті веб-сайту для управління особистими фінансами, модель може містити класи для роботи з транзакціями, категоріями, рахунками та іншими фінансовими об'єктами.

Представлення (View) відповідає за відображення даних та взаємодію з користувачем. Воно містить HTML-шаблони, які визначають структуру та зовнішній вигляд веб-сторінок. Представлення отримує дані від моделі та відображає їх у зрозумілому для користувача вигляді. В нашому випадку, представлення може включати сторінки для відображення списку транзакцій, форми для додавання нових транзакцій, графіки та діаграми для візуалізації фінансових даних тощо.

Контролер (Controller) виступає посередником між моделлю та представленням. Він отримує запити від користувача, обробляє їх, викликає відповідні методи моделі для виконання необхідних операцій та повертає результати представленню для відображення. Контролер також відповідає за обробку вхідних даних, валідацію, автентифікацію та авторизацію користувачів.

Для реалізації шаблону MVC в PHP можна використовувати різні фреймворки, такі як Laravel, Symfony, CodeIgniter тощо. Ці фреймворки надають готову структуру та інструменти для швидкої та ефективної розробки веб-додатків на основі шаблону MVC. Однак, в рамках даного проекту було вирішено використовувати чистий PHP без застосування фреймворків, щоб краще зрозуміти основні принципи та концепції веб-розробки.

Структура файлів та директорій веб-сайту на базі PHP з використанням шаблону MVC може виглядати наступним чином:

Рис. 2.1. Структура файлів системи.

Директорія app/ містить основні компоненти MVC:

- controllers/ - містить класи контролерів, які обробляють запити та викликають методи моделей;
- models/ - містить класи моделей, які відповідають за роботу з даними та бізнес-логіку;
- views/ - містить файли представлень (HTML-шаблони) для відображення даних.

Директорія config/ призначена для зберігання файлів конфігурації, таких як налаштування підключення до бази даних.

Директорія public/ є кореневою директорією веб-сервера і містить загальнодоступні файли, такі як CSS-стилі, JavaScript-скрипти, зображення та головний файл index.php, який виступає точкою входу в додаток.

Ось приклад того, як може виглядати частина коду в контролері

TransactionController.php:

```
<?php
namespace App\Controllers;

use App\Models\Transaction;

class TransactionController
{
    public function index()
    {
        $transactions = Transaction::getAll();
        require_once __DIR__ . '/../views/transactions/index.php';
    }

    public function create()
    {
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
            $amount = $_POST['amount'];
            $description = $_POST['description'];
            $category = $_POST['category'];

            Transaction::create($amount, $description, $category);
            header('Location: /transactions');
            exit;
        }

        require_once __DIR__ . '/../views/transactions/create.php';
    }

    // ...
```

```
    }
```

У цьому прикладі контролер TransactionController має два методи: index() та create(). Метод index() отримує список всіх транзакцій з моделі Transaction та підключає відповідне представлення для їх відображення. Метод create() обробляє створення нової транзакції: якщо надійшов POST-запит з даними форми, він створює нову транзакцію за допомогою моделі Transaction, а потім перенаправляє користувача на сторінку зі списком транзакцій. Якщо запит не є POST-запитом, то відображається форма для створення нової транзакції.

Приклад моделі Transaction.php:

```
<?php
namespace App\Models;

use PDO;

class Transaction
{
    private $db;

    public function __construct()
    {
        $this->db = new PDO('mysql:host=localhost;dbname=finance_db', 'username', 'password');
    }

    public static function getAll()
    {
        $stmt = $this->db->query('SELECT * FROM transactions');
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }
```

```
    public static function create($amount, $description, $category)
    {
        $stmt = $this->db->prepare('INSERT INTO transactions (amount, description, category) VALUES (?, ?, ?)');
        $stmt->execute([$amount, $description, $category]);
    }

    // ...
}
```

Модель Transaction містить методи для роботи з транзакціями в базі даних. Метод getAll() повертає список всіх транзакцій, а метод create() додає нову транзакцію до бази даних.

Приклад представлення transactions/index.php:

```
<!DOCTYPE html>
<html>
<head>
    <title>Транзакції</title>
</head>
<body>
    <h1>Список транзакцій</h1>
    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Сума</th>
                <th>Опис</th>
                <th>Категорія</th>
            </tr>
        </thead>
        <tbody>
```

```
            <?php foreach ($transactions as $transaction): ?>
            <tr>
                <td><?php echo $transaction['id']; ?></td>
                <td><?php echo $transaction['amount']; ?></td>
                <td><?php echo $transaction['description']; ?></td>
                <td><?php echo $transaction['category']; ?></td>
            </tr>
            <?php endforeach; ?>
        </tbody>
    </table>
    <a href="/transactions/create">Додати транзакцію</a>
</body>
</html>
```

Представлення transactions/index.php відображає список транзакцій у вигляді HTML-таблиці. Дані про транзакції передаються з контролера через змінну $transactions.

Таким чином, використання шаблону MVC та мови програмування PHP дозволяє створити чітку та структуровану архітектуру веб-сайту для управління особистими фінансами. Розділення системи на модель, представлення та контролер забезпечує розподіл відповідальності, покращує читабельність та підтримуваність коду, а також спрощує подальший розвиток та масштабування системи.

Крім того, важливо звернути увагу на безпеку веб-сайту. PHP надає різноманітні інструменти та практики для забезпечення безпеки, такі як фільтрація вхідних даних, екранування виводу, використання підготовлених запитів до бази даних для запобігання SQL-ін'єкцій, захист від міжсайтового скриптингу (XSS) та багато іншого. Необхідно ретельно дотримуватися рекомендацій щодо безпеки при розробці веб-сайту, щоб захистити дані користувачів та запобігти потенційним атакам.

Для забезпечення високої якості коду та зручності розробки також рекомендується використовувати додаткові інструменти та практики, такі як:

- Система контролю версій (наприклад, Git) для відстеження змін у коді та співпраці з іншими розробниками;
- Пакетний менеджер Composer для управління залежностями та зовнішніми бібліотеками;
- Автоматизовані тести (модульні, інтеграційні, приймальні) для перевірки коректності роботи системи;
- Налагоджувальні інструменти та засоби логування для виявлення та усунення помилок;
- Дотримання стандартів кодування (наприклад, PSR) для забезпечення узгодженості та читабельності коду.

Таблиця 2.1

Опис компонентів шаблону MVC

Компонент MVC	**Опис**
Модель (Model)	Відповідає за роботу з даними та бізнес-логіку системи. Включає класи та функції для взаємодії з базою даних, виконання операцій з даними та реалізації основних функціональних можливостей системи.
Представлення (View)	Відповідає за відображення даних та взаємодію з користувачем. Містить HTML-шаблони, які визначають структуру та зовнішній вигляд веб-сторінок. Отримує дані від моделі та відображає їх у зрозумілому для користувача вигляді.
Контролер (Controller)	Виступає посередником між моделлю та представленням. Отримує запити від користувача, обробляє їх, викликає відповідні методи моделі для виконання необхідних операцій та повертає результати представленню для відображення. Також відповідає за обробку вхідних даних, валідацію, автентифікацію та авторизацію користувачів.

Отже, проектування архітектури веб-сайту на базі PHP з використанням шаблону MVC дозволяє створити структуровану, масштабовану та зручну в підтримці систему для управління особистими фінансами. Розділення відповідальності між моделлю, представленням та контролером забезпечує гнучкість, читабельність та можливість подальшого розширення функціональності веб-сайту. Дотримання принципів безпеки та використання додаткових інструментів та практик розробки дозволяє створити надійний та якісний продукт, який відповідає потребам користувачів та бізнесу.

2.2. Проектування структури бази даних з використанням MySQL

Проектування структури бази даних є важливим етапом у розробці веб-сайту для управління особистими фінансами. Правильно спроектована база даних забезпечує ефективне зберігання, організацію та доступ до даних, що є ключовим фактором для функціонування системи. У цьому розділі ми розглянемо проектування структури бази даних з використанням MySQL для нашого веб-сайту.

MySQL - це популярна реляційна система управління базами даних (РСУБД) з відкритим кодом, яка широко використовується для розробки веб-додатків. MySQL надає потужні можливості для зберігання, обробки та отримання даних, а також забезпечує високу продуктивність, надійність та масштабованість.

Перш ніж розпочати проектування бази даних, необхідно визначити основні сутності та зв'язки між ними, виходячи з функціональних вимог до системи. Для нашого веб-сайту управління особистими фінансами ми визначили такі основні сутності:

1. Користувачі (users): Зберігає інформацію про зареєстрованих користувачів системи, таку як ім'я користувача, пароль та баланс;
2. Транзакції (transactions): Представляє фінансові транзакції користувачів, включаючи дату, тип (дохід або витрата), суму, опис та пов'язану категорію;

3. Категорії транзакцій (transaction_categories): Зберігає інформацію про категорії, до яких можуть належати транзакції, наприклад, "Їжа", "Транспорт", "Розваги" тощо.
4. Заплановані транзакції (planned_transactions): Представляє майбутні заплановані транзакції користувачів, такі як регулярні платежі або очікувані надходження;
5. Повідомлення чату (chat_messages): Зберігає повідомлення, надіслані користувачами через вбудований чат на веб-сайті.

Визначивши основні сутності, можна приступати до проектування структури бази даних. Ось детальний опис таблиць та зв'язків між ними:

1. Таблиця "users":
   - id (INT, PRIMARY KEY, AUTO_INCREMENT): Унікальний ідентифікатор користувача.
   - username (VARCHAR(255), UNIQUE): Ім'я користувача для входу в систему.
   - password (VARCHAR(255)): Захешований пароль користувача.
   - balance (DECIMAL(10,2), DEFAULT 0.00): Поточний баланс користувача.
2. Таблиця "transactions":
   - id (INT, PRIMARY KEY, AUTO_INCREMENT): Унікальний ідентифікатор транзакції.
   - user_id (INT, FOREIGN KEY (users.id)): Зовнішній ключ, що зв'язує транзакцію з користувачем.
   - timestamp (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP): Мітка часу створення транзакції.
   - type (ENUM('income', 'expense')): Тип транзакції (дохід або витрата).
   - amount (DECIMAL(10,2)): Сума транзакції.
   - description (TEXT): Опис транзакції.
   - credit (TINYINT(1), DEFAULT 0): Прапорець, що вказує, чи є транзакція кредитною.

- category_id (INT, FOREIGN KEY (transaction_categories.id)): Зовнішній ключ, що зв'язує транзакцію з категорією.

3. Таблиця "transaction_categories":
   - id (INT, PRIMARY KEY, AUTO_INCREMENT): Унікальний ідентифікатор категорії.
   - user_id (INT, FOREIGN KEY (users.id)): Зовнішній ключ, що зв'язує категорію з користувачем.
   - name (VARCHAR(255)): Назва категорії.
4. Таблиця "planned_transactions":
   - id (INT, PRIMARY KEY, AUTO_INCREMENT): Унікальний ідентифікатор запланованої транзакції.
   - user_id (INT, FOREIGN KEY (users.id)): Зовнішній ключ, що зв'язує заплановану транзакцію з користувачем.
   - type (ENUM('income', 'expense')): Тип запланованої транзакції (дохід або витрата).
   - amount (DECIMAL(10,2)): Сума запланованої транзакції.
   - description (TEXT): Опис запланованої транзакції.
   - date (DATE): Дата запланованої транзакції.
5. Таблиця "chat_messages":
   - id (INT, PRIMARY KEY, AUTO_INCREMENT): Унікальний ідентифікатор повідомлення чату.
   - user_id (INT, FOREIGN KEY (users.id)): Зовнішній ключ, що зв'язує повідомлення з користувачем.
   - message (TEXT): Текст повідомлення.
   - task_time (INT): Час, витрачений на завдання, пов'язане з повідомленням.
   - timestamp (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP): Мітка часу створення повідомлення.

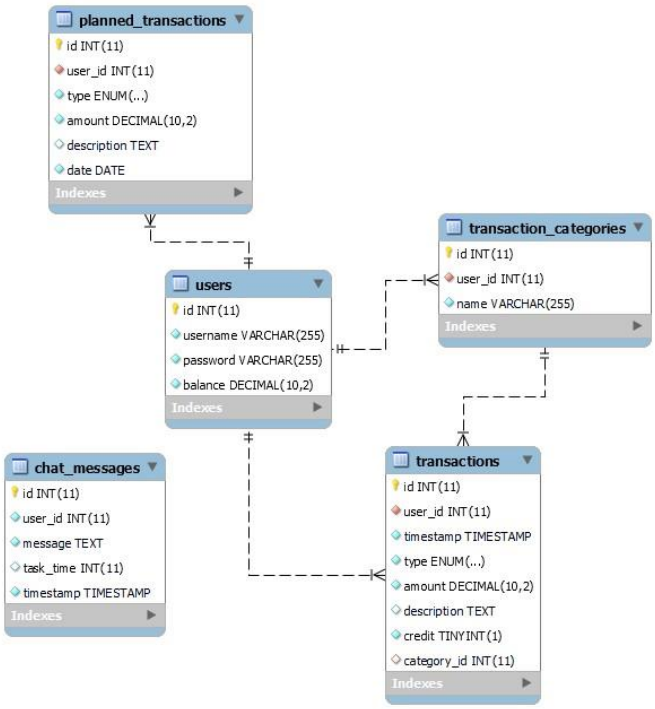

Рис. 2.2. Діаграма бази даних.

На діаграмі бази даних можна побачити зв'язки між таблицями. Таблиця "users" має зв'язок "один до багатьох" з таблицями "transactions", "transaction_categories", "planned_transactions" та "chat_messages". Це означає, що один користувач може мати багато транзакцій, категорій, запланованих транзакцій та повідомлень чату, але кожна транзакція, категорія, запланована транзакція та повідомлення належить лише одному користувачу.

Таблиця "transactions" має зв'язок "багато до одного" з таблицею "transaction_categories". Це означає, що кожна транзакція може належати до однієї категорії, але одна категорія може містити багато транзакцій.

Для створення таблиць бази даних можна використовувати SQL-запит наведений в додатку Б.

Ці SQL-запити створюють таблиці users, transactions, transaction_categories, planned_transactions та chat_messages з відповідними полями та зв'язками між ними.

При проектуванні бази даних важливо враховувати вимоги до продуктивності, масштабованості та цілісності даних. Для забезпечення цілісності даних використовуються обмеження (constraints), такі як первинні ключі (PRIMARY KEY), зовнішні ключі (FOREIGN KEY) та унікальні індекси (UNIQUE).

Первинні ключі гарантують унікальність кожного запису в таблиці та дозволяють однозначно ідентифікувати кожен рядок. Зовнішні ключі забезпечують зв'язок між таблицями та підтримують цілісність даних, гарантуючи, що зв'язані записи завжди існують у відповідних таблицях. Унікальні індекси забезпечують унікальність значень у певних стовпцях, наприклад, для уникнення дублювання імен користувачів.

Також важливо враховувати вимоги до безпеки бази даних. Для захисту даних користувачів необхідно реалізувати механізми автентифікації та авторизації, щоб обмежити доступ до бази даних лише авторизованим користувачам. Крім того, слід використовувати безпечні методи зберігання паролів, такі як хешування з сіллю, щоб запобігти несанкціонованому доступу до облікових записів користувачів.

При розробці веб-сайту на базі PHP для взаємодії з базою даних MySQL можна використовувати розширення PHP, такі як mysqli або PDO (PHP Data Objects). Ці розширення надають зручні методи для встановлення з'єднання з базою даних, виконання запитів та отримання результатів.

Наприклад, для встановлення з'єднання з базою даних за допомогою PDO можна використати наступний код:

```
$host = 'localhost';
$dbname = 'finance_db';
$username = 'your_username';
$password = 'your_password';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Помилка підключення до бази даних: " . $e->getMessage());
}
```

Цей код створює новий об'єкт PDO, використовуючи відповідні параметри підключення (хост, ім'я бази даних, ім'я користувача та пароль), і встановлює режим обробки помилок для викидання винятків у разі виникнення проблем з підключенням.

Після встановлення з'єднання з базою даних можна виконувати SQL-запити для вставки, оновлення, видалення та вибірки даних. PDO надає методи для безпечного виконання запитів з параметрами, що допомагає запобігти SQL-ін'єкціям.

Ось приклад виконання запиту з параметрами за допомогою PDO:

```
$username = 'john_doe';
$password = 'password123';

$stmt = $pdo->prepare('SELECT * FROM users WHERE username = ? AND password = ?');
$stmt->execute([$username, $password]);
$user = $stmt->fetch(PDO::FETCH_ASSOC);
```

Цей код готує запит з параметрами, використовуючи знаки питання (?) як заповнювачі для значень, а потім виконує запит, передаючи масив зі значеннями параметрів. Результат запиту отримується за допомогою методу fetch() з параметром PDO::FETCH_ASSOC, який повертає асоціативний масив з даними користувача.

Отже, правильне проектування структури бази даних з використанням MySQL є важливим етапом у розробці веб-сайту для управління особистими фінансами. Визначення основних сутностей, створення таблиць з відповідними полями та зв'язками, забезпечення цілісності даних за допомогою обмежень та врахування вимог до продуктивності, масштабованості та безпеки - все це сприяє створенню надійного та ефективного сховища даних для системи. Використання мови SQL та розширень PHP, таких як PDO, дозволяє зручно взаємодіяти з базою даних та виконувати необхідні операції з даними для забезпечення функціональності веб-сайту.

2.3. Розробка дизайну користувацького інтерфейсу з використанням HTML та CSS

Розробка дизайну користувацького інтерфейсу є важливим етапом у створенні веб-сайту для управління особистими фінансами. Привабливий, інтуїтивно зрозумілий та зручний у використанні інтерфейс значно покращує взаємодію користувачів із системою та підвищує їхнє задоволення від користування нею. У цьому розділі ми розглянемо розробку дизайну користувацького інтерфейсу з використанням HTML та CSS для нашого веб-сайту.

HTML (HyperText Markup Language) - це стандартна мова розмітки для створення структури та вмісту веб-сторінок. За допомогою HTML ми визначаємо основні елементи сторінки, такі як заголовки, абзаци, списки, таблиці, форми та інші. HTML дозволяє організувати вміст сторінки у логічній та семантичній формі, що полегшує його сприйняття та доступність.

CSS (Cascading Style Sheets) - це мова стилів, яка використовується для опису зовнішнього вигляду та форматування HTML-елементів. За допомогою CSS ми можемо визначити кольори, шрифти, розміри, відступи, позиціонування та інші візуальні властивості елементів. CSS дозволяє відокремити дизайн від структури та вмісту, що робить код більш читабельним, модульним та легким в обслуговуванні.

Для нашого веб-сайту управління особистими фінансами ми розробили привабливий та інтуїтивно зрозумілий дизайн користувацького інтерфейсу з використанням HTML та CSS. Ось детальний опис основних елементів та компонентів інтерфейсу:

1. Загальна структура сторінки:
   - Використано семантичні теги HTML5, такі як <header>, <nav>, <main>, <section>, <footer>, для визначення основних розділів сторінки.
   - Застосовано техніку "контейнера" для центрування та обмеження ширини вмісту на сторінці.

- Розроблено адаптивний дизайн з використанням медіа-запитів CSS для забезпечення коректного відображення на різних пристроях та розмірах екрану.

2. Шапка сайту (header):
   - Створено привабливу шапку сайту з логотипом та навігаційним меню.
   - Використано CSS для стилізації фону, кольорів, шрифтів та відступів шапки.
   - Реалізовано випадаюче меню для зручної навігації по сайту.
3. Форми для введення даних:
   - Розроблено інтуїтивно зрозумілі форми для введення даних, такі як форма реєстрації, форма входу, форма додавання транзакцій тощо.
   - Використано відповідні типи полів введення (текстові поля, прапорці, перемикачі, випадаючі списки) для зручності введення даних користувачем.
   - Застосовано CSS для стилізації полів введення, міток, кнопок та повідомлень про помилки.
4. Таблиці для відображення даних:
   - Створено таблиці для відображення списків транзакцій, категорій, запланованих транзакцій тощо.
   - Використано CSS для стилізації таблиць, включаючи межі, відступи, кольори фону та тексту.
   - Застосовано техніки для покращення читабельності даних у таблицях, такі як чергування кольорів рядків та виділення рядків при наведенні.
5. Графіки та діаграми:
   - Інтегровано бібліотеки JavaScript, такі як Chart.js або D3.js, для створення візуально привабливих графіків та діаграм.
   - Відображено статистичні дані, такі як розподіл витрат за категоріями, динаміка доходів та витрат тощо.
   - Застосовано CSS для стилізації графіків та діаграм, включаючи кольори, розміри, підписи та легенду.

6. Кнопки та елементи керування:
   - Розроблено привабливі та інтуїтивно зрозумілі кнопки та елементи керування для виконання дій, таких як додавання транзакцій, редагування категорій, генерування звітів тощо.
   - Використано CSS для стилізації кнопок, включаючи кольори, розміри, відступи та ефекти при наведенні та натисканні.
7. Повідомлення та сповіщення:
   - Реалізовано відображення повідомлень про успішні дії, помилки та попередження користувачеві.
   - Використано CSS для стилізації повідомлень, включаючи кольори, іконки та анімації.
8. Адаптивність та мобільна оптимізація:
   - Застосовано техніки адаптивного веб-дизайну (responsive web design) для забезпечення коректного відображення сайту на різних пристроях, таких як настільні комп'ютери, планшети та смартфони.
   - Використано медіа-запити CSS для визначення стилів для різних роздільних здатностей екрану.
   - Оптимізовано розмір та розташування елементів інтерфейсу для зручності використання на мобільних пристроях.

Для розробки HTML-структури сторінок використано семантичні теги та дотримано принципів доступності та валідності коду. Наприклад:

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Управління особистими фінансами</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
```

```
<header>
    <nav>
        <ul>
            <li><a href="index.php">Головна</a></li>
            <li><a href="transactions.php">Транзакції</a></li>
            <li><a href="categories.php">Категорії</a></li>
            <li><a href="reports.php">Звіти</a></li>
        </ul>
    </nav>
</header>

<main>
    <section>
        <h1>Список транзакцій</h1>
        <table>
            <thead>
                <tr>
                    <th>Дата</th>
                    <th>Категорія</th>
                    <th>Сума</th>
                    <th>Опис</th>
                </tr>
            </thead>
            <tbody>
                <!-- Динамічно сформований список транзакцій -->
            </tbody>
        </table>
    </section>
</main>
```

```
    <footer>
        <p>&copy; 2023 Управління особистими фінансами. Всі права захищені.</p>
    </footer>
</body>
</html>
```

Для стилізації елементів інтерфейсу використано CSS. Ось приклад деяких стилів:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

header {
    background-color: #333;
    color: #fff;
    padding: 20px;
}

nav ul {
    list-style: none;
    margin: 0;
    padding: 0;
}

nav ul li {
    display: inline-block;
    margin-right: 10px;
}
```

```
nav ul li a {
    color: #fff;
    text-decoration: none;
}

main {
    padding: 20px;
}

table {
    width: 100%;
    border-collapse: collapse;
}
```

Ці стилі визначають шрифти, кольори, відступи та інші візуальні властивості елементів інтерфейсу, такі як заголовки, навігаційне меню, таблиці та підвал сторінки.

Для забезпечення адаптивності інтерфейсу використано медіа-запити CSS. Наприклад:

```
@media screen and (max-width: 600px) {
    nav ul li {
        display: block;
        margin-bottom: 10px;
    }

    table th, table td {
        font-size: 14px;
    }
}
```

Цей медіа-запит застосовує специфічні стилі для екранів з шириною до 600px, змінюючи відображення навігаційного меню та розмір шрифту в таблиці для кращої читабельності на мобільних пристроях.

Окрім HTML та CSS, для розробки інтерфейсу використано також препроцесор CSS, такий як Sass або Less, для спрощення та оптимізації написання стилів. Препроцесори дозволяють використовувати змінні, вкладеність селекторів, міксини та інші корисні функції, що робить код більш модульним та легким в обслуговуванні.

Для інтерактивності та динамічності інтерфейсу використано JavaScript та бібліотеки, такі як jQuery або Vue.js. Наприклад, для відображення модальних вікон, обробки подій натискання кнопок, валідації форм та інших інтерактивних елементів.

Також для покращення користувацького досвіду застосовано техніки, такі як анімації та переходи CSS, що надають інтерфейсу більшої плавності та візуальної привабливості.

Важливо зазначити, що при розробці інтерфейсу враховано принципи юзабіліті (usability) та доступності (accessibility). Це означає, що інтерфейс розроблено з урахуванням зручності використання для різних груп користувачів, включаючи людей з обмеженими можливостями. Дотримано рекомендацій щодо контрастності кольорів, розмірів шрифтів, альтернативних текстів для зображень та інших аспектів доступності.

Для тестування та налагодження інтерфейсу використано інструменти веб-розробки, такі як Chrome Developer Tools або Firefox Developer Tools. Ці інструменти дозволяють перевіряти та модифікувати HTML-структуру та CSS-стилі в реальному часі, що спрощує процес розробки та усунення помилок.

Отже, розробка дизайну користувацького інтерфейсу з використанням HTML та CSS є важливим етапом у створенні веб-сайту для управління особистими фінансами. Застосування семантичної розмітки HTML, модульних та адаптивних стилів CSS, а також врахування принципів юзабіліті та доступності дозволяє створити привабливий, зручний у використанні та доступний інтерфейс для користувачів. Використання препроцесорів CSS, JavaScript та бібліотек для інтерактивності та

динамічності, а також тестування та налагодження за допомогою веб-інструментів забезпечують високу якість та надійність інтерфейсу. Розроблений інтерфейс повинен відповідати потребам та очікуванням користувачів, спрощувати взаємодію з системою та надавати приємний досвід користування веб-сайтом для управління особистими фінансами.

2.4. Висновки до розділу

У другому розділі кваліфікаційної роботи було розглянуто проектування веб-сайту для управління особистими фінансами з використанням HTML, PHP, CSS та MySQL. Було проведено детальний аналіз архітектури веб-сайту, структури бази даних та дизайну користувацького інтерфейсу.

При проектуванні архітектури веб-сайту на базі PHP було обрано шаблон проектування Model-View-Controller (MVC), який забезпечує чітке розділення логіки на три компоненти: модель, представлення та контролер. Такий підхід дозволяє досягти модульності, масштабованості та зручності підтримки коду. Було описано структуру файлів та директорій веб-сайту, а також надано приклади коду для контролерів, моделей та представлень. Також було розглянуто питання безпеки веб-сайту та рекомендовано використання додаткових інструментів та практик для забезпечення високої якості коду.

Проектування структури бази даних з використанням MySQL було важливим етапом розробки веб-сайту. Було визначено основні сутності, такі як користувачі, транзакції, категорії, заплановані транзакції та повідомлення чату, та створено відповідні таблиці з необхідними полями та зв'язками між ними. Для забезпечення цілісності даних використано обмеження, такі як первинні ключі, зовнішні ключі та унікальні індекси. Також було розглянуто питання безпеки бази даних та надано приклади коду для встановлення з'єднання та виконання запитів з використанням PDO.

Розробка дизайну користувацького інтерфейсу з використанням HTML та CSS дозволила створити привабливий та зручний у використанні інтерфейс для веб-сайту. Було описано основні елементи та компоненти інтерфейсу, такі як загальна структура

сторінки, шапка сайту, форми для введення даних, таблиці для відображення даних, графіки та діаграми, кнопки та елементи керування, повідомлення та сповіщення. Для забезпечення адаптивності інтерфейсу використано медіа-запити CSS, а для інтерактивності та динамічності - JavaScript та бібліотеки. Також було враховано принципи юзабіліті та доступності при розробці інтерфейсу.

Результатом проектування веб-сайту став детальний план його архітектури, структури бази даних та дизайну користувацького інтерфейсу. Цей план є основою для подальшої реалізації веб-сайту та забезпечує чітке розуміння всіх аспектів його розробки.

Таким чином, у другому розділі кваліфікаційної роботи було успішно спроектовано веб-сайт для управління особистими фінансами з використанням сучасних веб-технологій та підходів до проектування. Обрані технології та архітектурні рішення дозволяють створити надійний, масштабований та зручний у використанні веб-сайт, який відповідає вимогам та потребам користувачів. Проведена робота з проектування закладає міцний фундамент для подальшої реалізації та впровадження веб-сайту в рамках кваліфікаційної роботи.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБ-САЙТУ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

3.1. Розробка дизайну веб-сайту

Розробка дизайну веб-сайту є важливим етапом у процесі створення веб-сайту для управління особистими фінансами. Привабливий та інтуїтивно зрозумілий дизайн не тільки покращує естетичний вигляд сайту, але й сприяє зручності використання та залученню користувачів. У цьому підрозділі ми детально розглянемо процес розробки дизайну веб-сайту відповідно до розробленої нами системи.

Перш за все, важливо визначити загальну концепцію дизайну та стилістичне оформлення веб-сайту. Для нашого проекту ми обрали сучасний, мінімалістичний дизайн з акцентом на функціональність та зручність використання. Основними кольорами було обрано приємні пастельні відтінки, які створюють дружню та заспокійливу атмосферу для користувачів.

Далі ми розробили структуру та макет сторінок веб-сайту. Головна сторінка (рис. 3.1.) слугує відправною точкою для користувачів і містить основну інформацію про функціональність сайту та його переваги.

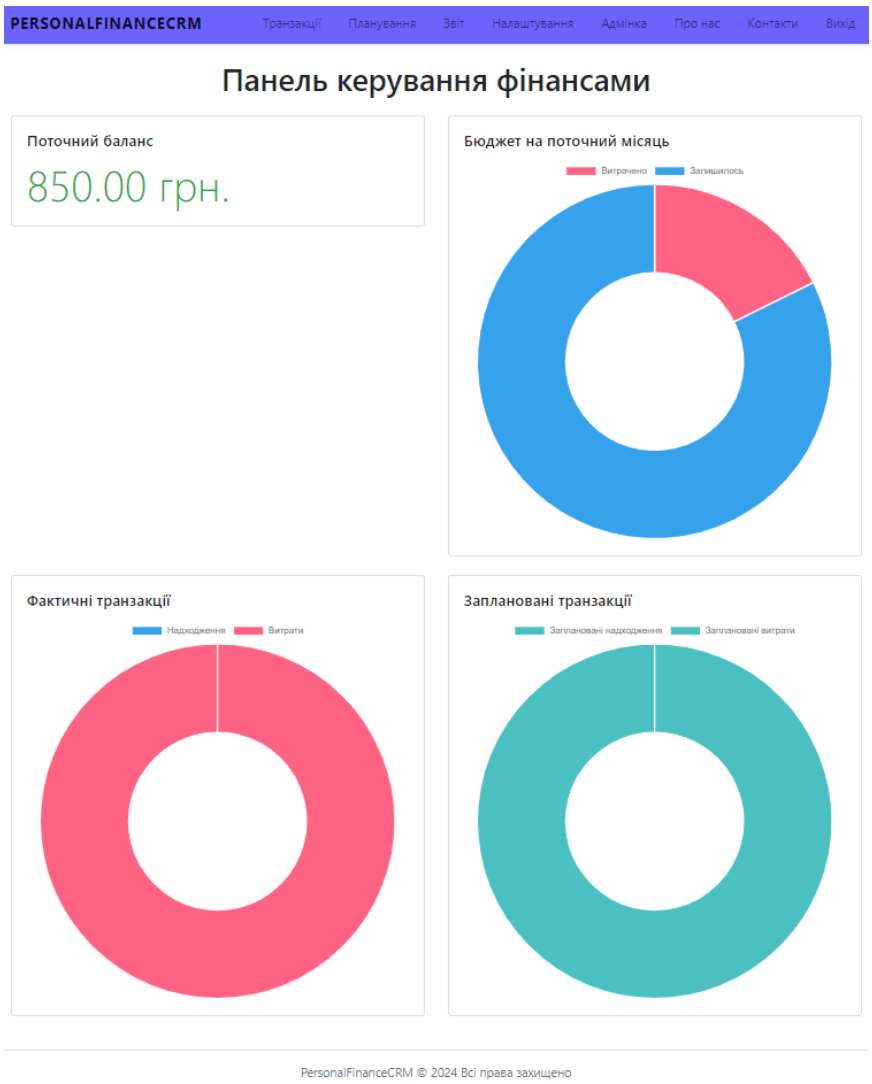

Рис. 3.1. Головна сторінка веб-сайту

На головній сторінці розміщено заголовок з назвою сайту та привітальним повідомленням. Нижче знаходиться блок з основними функціями сайту, такими як додавання транзакцій, перегляд статистики, планування бюджету тощо. Кожна функція представлена у вигляді картки з відповідною іконкою та коротким описом. Це дозволяє користувачам швидко зорієнтуватися та зрозуміти, що пропонує сайт.

У верхній частині сторінки розташовано навігаційне меню (рис. 3.2.), яке забезпечує зручний доступ до різних розділів сайту. Меню містить посилання на головну сторінку, сторінку транзакцій, сторінку статистики, сторінку планування та сторінку профілю користувача. Активна сторінка виділяється відповідним кольором для зручності орієнтації.

Рис. 3.2. Навігаційне меню веб-сайту

Однією з ключових сторінок веб-сайту є сторінка транзакцій (рис. 3.3.). На цій сторінці користувачі можуть переглядати список своїх транзакцій, додавати нові транзакції та редагувати існуючі. Транзакції відображаються у вигляді таблиці з такими полями, як дата, категорія, сума, опис та тип (дохід або витрата).

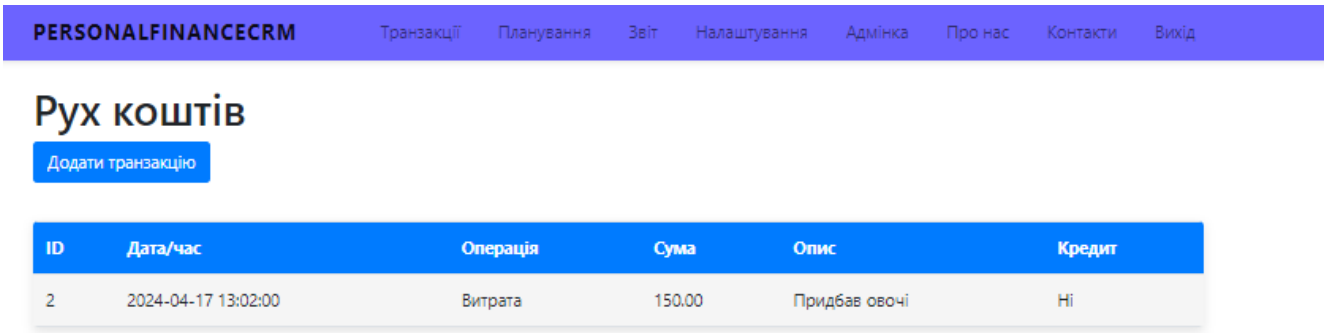

Рис. 3.3. Сторінка транзакцій

Для додавання нової транзакції передбачено спеціальну форму (рис. 3.4.), яка з'являється у вигляді модального вікна при натисканні кнопки "Додати транзакцію". Форма містить поля для введення даних про транзакцію, такі як дата, категорія, сума, опис та тип. Після заповнення форми користувач може зберегти транзакцію, натиснувши кнопку "Зберегти".

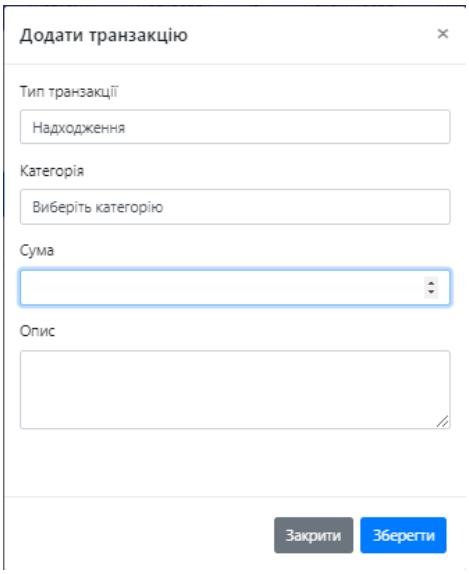

Рис. 3.4. Форма додавання нової транзакції

Сторінка статистики (рис. 3.5.) призначена для візуалізації фінансових даних користувача. На цій сторінці відображаються графіки та діаграми, які дозволяють користувачу аналізувати свої доходи та витрати за різними категоріями та періодами часу. Графіки можуть включати лінійні графіки для відображення динаміки доходів і витрат, кругові діаграми для показу розподілу витрат за категоріями тощо.

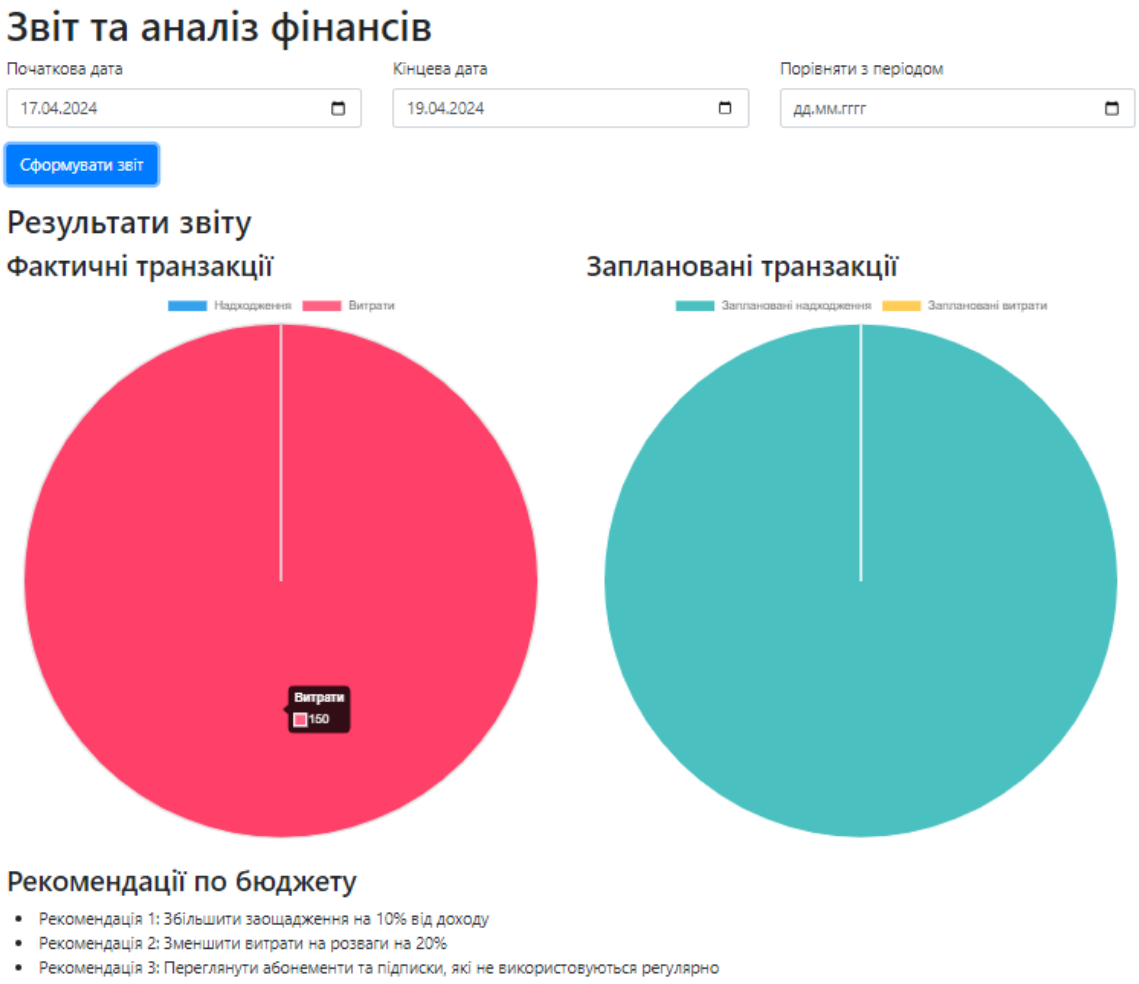

Рис. 3.5. Сторінка статистики

Для зручності планування бюджету передбачено сторінку планування (рис. 3.6.). На цій сторінці користувачі можуть встановлювати фінансові цілі, створювати бюджети на певний період та відстежувати їх виконання. Бюджети можуть бути представлені у вигляді таблиці з категоріями витрат та запланованими сумами. Користувачі можуть вносити корективи до своїх бюджетів та отримувати сповіщення про досягнення цілей або перевищення запланованих сум.

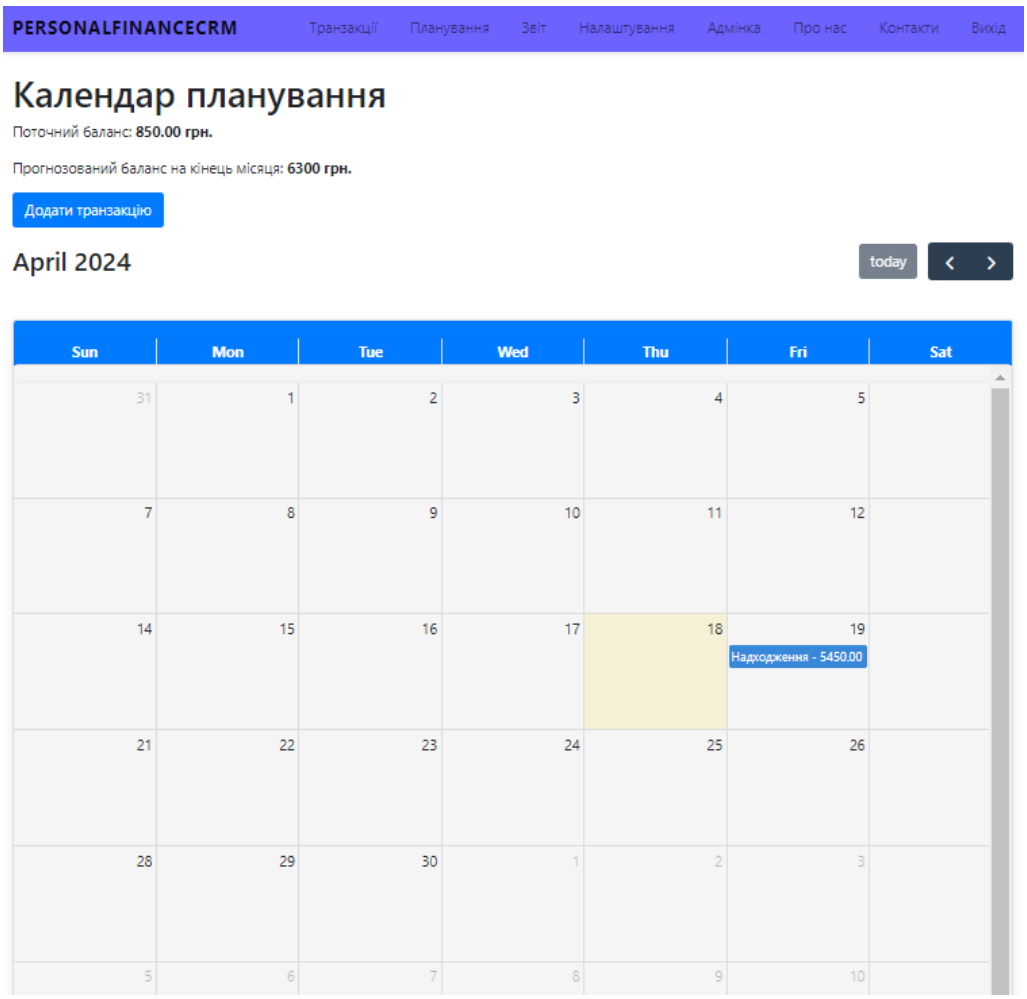

Рис. 3.6. Сторінка планування

Сторінка персональних налаштувань (рис. 3.7.) дозволяє користувачам переглядати та редагувати свою особисту інформацію, таку як ім'я, електронна пошта, пароль тощо. Користувачі також можуть налаштовувати категорії для своїх транзакцій.

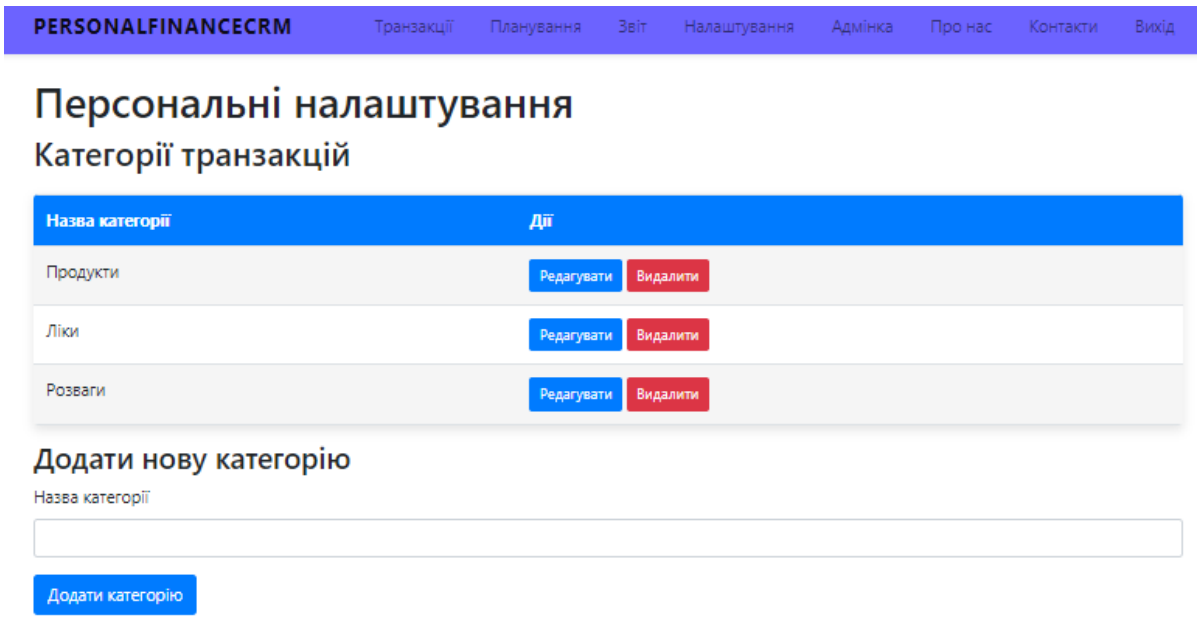

Рис. 3.7. Сторінка профілю користувача

Для забезпечення цілісності та узгодженості дизайну веб-сайту було розроблено єдину систему стилів та компонентів інтерфейсу. Це включає вибір шрифтів, кольорів, розмірів елементів, іконок тощо. Усі сторінки веб-сайту слідують цій системі стилів, що створює візуальну єдність та покращує користувацький досвід.

Особливу увагу було приділено адаптивності дизайну для різних пристроїв та розмірів екрану. Веб-сайт розроблено з використанням принципів адаптивного веб-дизайну (responsive web design), що дозволяє коректно відображати вміст на настільних комп'ютерах, планшетах та смартфонах. Елементи інтерфейсу автоматично перебудовуються та змінюють розмір залежно від доступного простору екрану, забезпечуючи зручність використання на будь-якому пристрої.

Окрім функціональних сторінок, на веб-сайті передбачено також інформаційні сторінки, такі як "Про нас" (рис. 3.8.) та "Контакти" (рис. 3.9.). Ці сторінки надають користувачам додаткову інформацію про розробників веб-сайту та способи зв'язку з ними у разі потреби.

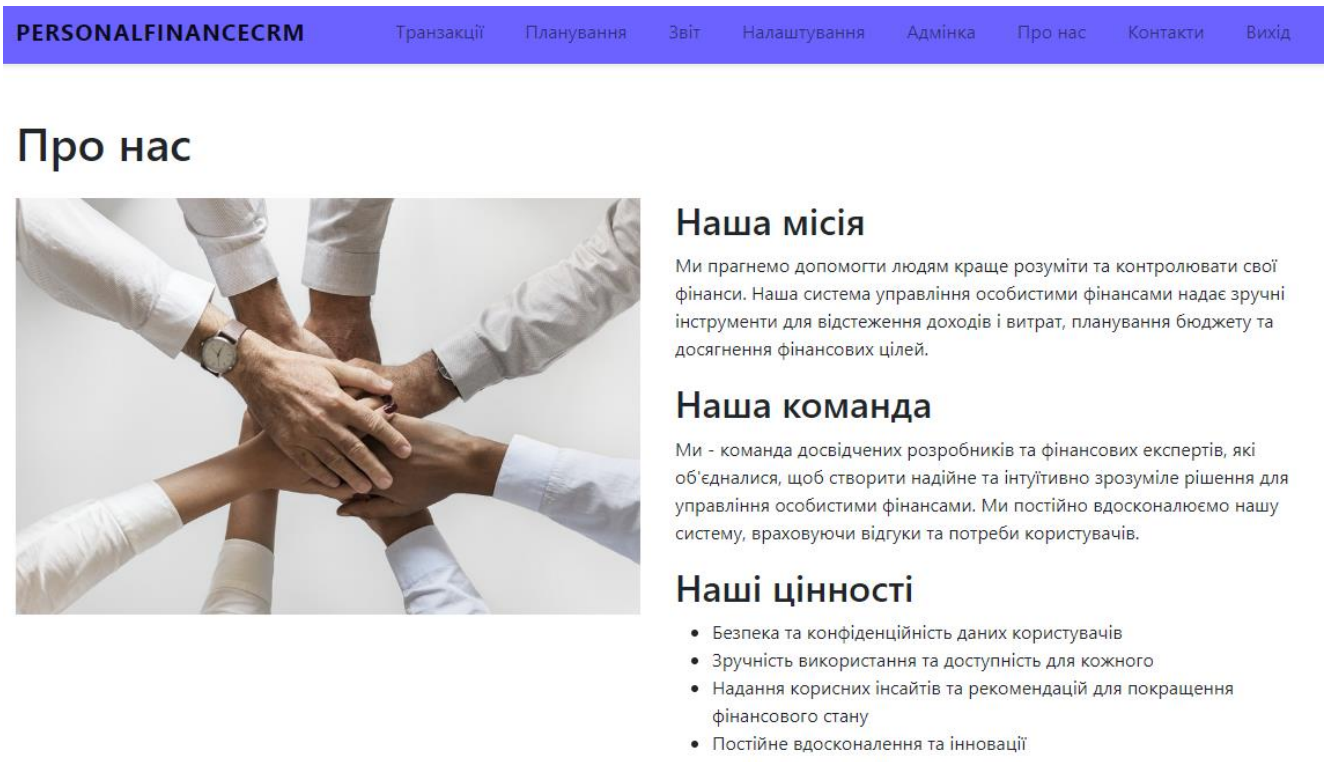

Рис. 3.8. Сторінка "Про нас"

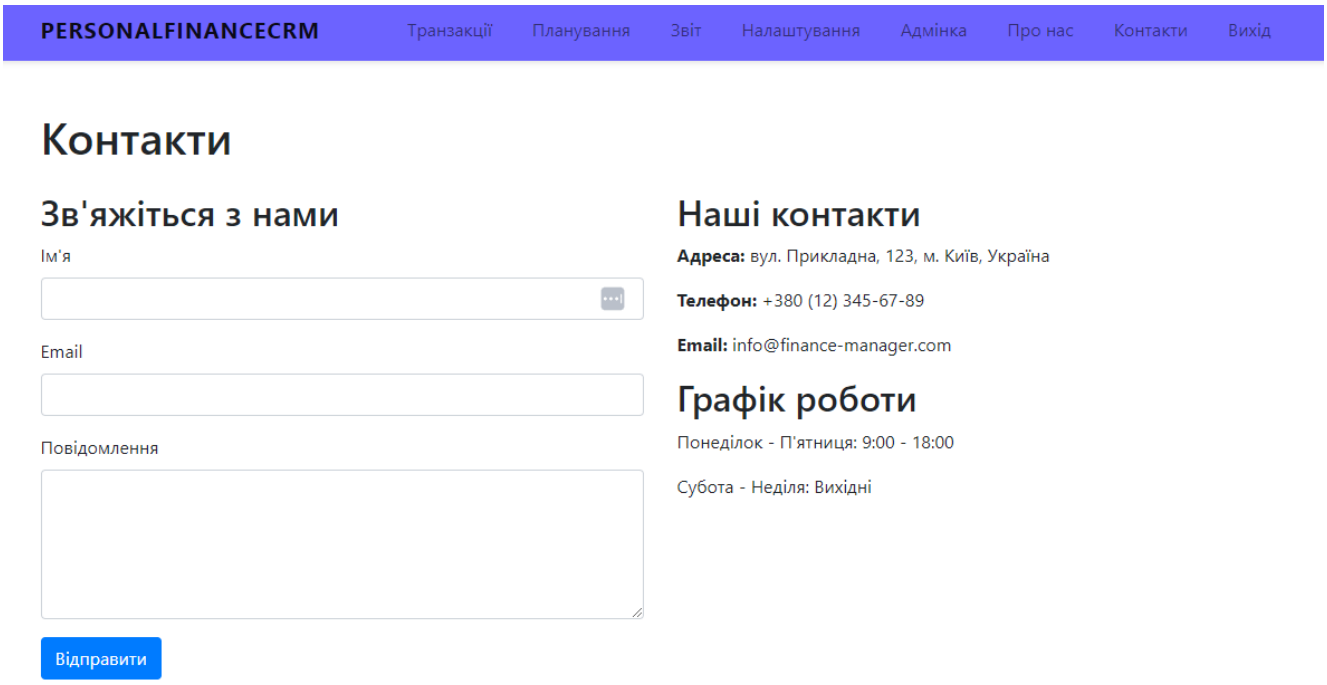

Рис. 3.9. Сторінка "Контакти"

Для покращення візуальної привабливості та зручності використання веб-сайту було застосовано різноманітні графічні елементи та ефекти. Наприклад, кнопки та посилання мають візуальний ефект при наведенні курсора миші, що підкреслює інтерактивність елементів. Також використано іконки та зображення для ілюстрації

різних функцій та категорій, що робить інтерфейс більш інтуїтивно зрозумілим та привабливим.

Процес розробки дизайну веб-сайту також включав тестування юзабіліті (usability testing) та збір відгуків від потенційних користувачів. Це дозволило виявити можливі проблеми з навігацією, розміщенням елементів та зрозумілістю інтерфейсу. На основі отриманих відгуків було внесено необхідні корективи та вдосконалення в дизайн веб-сайту.

Таким чином, розробка дизайну веб-сайту для управління особистими фінансами була виконана з урахуванням сучасних тенденцій веб-дизайну, принципів юзабіліті та потреб користувачів. Результатом є привабливий, інтуїтивно зрозумілий та функціональний веб-сайт, який забезпечує зручний доступ до всіх необхідних інструментів для ефективного управління фінансами. Розроблений дизайн створює приємний користувацький досвід і сприяє регулярному використанню веб-сайту для контролю та оптимізації особистих фінансів.

3.2. Розробка функціональної частини системи

Розробка функціональної частини системи є ключовим етапом у процесі створення веб-сайту для управління особистими фінансами. Функціональність системи визначає, які дії та операції можуть виконувати користувачі, і як веб-сайт обробляє та відображає дані. У цьому підрозділі ми детально розглянемо процес розробки функціональної частини системи відповідно до визначених вимог та архітектури.

Перш за все, необхідно визначити основні функціональні модулі системи. Для нашого веб-сайту ми виділили такі ключові модулі:

1. Модуль автентифікації та авторизації користувачів;
2. Модуль управління транзакціями;
3. Модуль генерації звітів та статистики;
4. Модуль планування бюджету;
5. Модуль сповіщень та нагадувань;
6. Модуль налаштувань користувача.

Розглянемо кожен з цих модулів детальніше.

Модуль автентифікації та авторизації користувачів відповідає за реєстрацію нових користувачів, вхід в систему та управління сесіями. Для реалізації цього модуля використовуються такі технології, як PHP та MySQL. При реєстрації користувач надає свої персональні дані, такі як ім'я, електронна пошта та пароль. Ці дані зберігаються в базі даних MySQL з використанням надійних методів хешування паролів. Під час входу в систему, користувач вводить свої облікові дані, які перевіряються на відповідність записам у базі даних. У разі успішної автентифікації, створюється сесія користувача, яка дозволяє зберігати інформацію про поточний стан користувача протягом його взаємодії з веб-сайтом.

Модуль управління транзакціями є центральним компонентом системи і відповідає за додавання, редагування та видалення фінансових транзакцій користувача. Транзакції можуть бути доходами або витратами і належати до певних категорій. Для реалізації цього модуля використовується комбінація PHP, MySQL та JavaScript. Користувач може додавати нові транзакції через веб-інтерфейс, заповнюючи форму з полями для введення даних, такими як дата, категорія, сума, опис тощо. Введені дані відправляються на сервер за допомогою AJAX-запиту і зберігаються в базі даних. Для відображення списку транзакцій використовуються SQL-запити до бази даних з фільтрацією та сортуванням. Користувач також має можливість редагувати або видаляти існуючі транзакції за допомогою відповідних функцій.

Модуль генерації звітів та статистики дозволяє користувачам переглядати аналітичну інформацію про свої фінансові дані. Цей модуль використовує дані з бази даних для формування різноманітних звітів та графіків. Наприклад, користувач може переглянути звіт про доходи та витрати за певний період часу, розподіл витрат за категоріями, динаміку зміни балансу тощо. Для реалізації цього модуля використовуються такі технології, як PHP, MySQL та бібліотеки для створення графіків, такі як Chart.js або Google Charts. SQL-запити до бази даних дозволяють отримати необхідні дані, які потім обробляються та візуалізуються за допомогою відповідних бібліотек.

Модуль планування бюджету надає користувачам інструменти для створення та управління фінансовими планами. Користувач може встановлювати бюджети для різних категорій витрат на певний період часу (наприклад, місяць або рік). Система автоматично відстежує фактичні витрати та порівнює їх з запланованими, надаючи користувачу інформацію про відхилення та прогрес виконання бюджету. Для реалізації цього модуля використовуються PHP, MySQL та JavaScript. Користувач може створювати, редагувати та видаляти бюджети через веб-інтерфейс, а система зберігає ці дані в базі даних. За допомогою SQL-запитів та обчислень, система формує звіти та сповіщення про стан виконання бюджету.

Модуль сповіщень та нагадувань відповідає за своєчасне інформування користувача про важливі події та дати. Наприклад, система може надсилати користувачу нагадування про необхідність внесення регулярних платежів, сповіщати про досягнення фінансових цілей або попереджати про перевищення бюджетних лімітів. Для реалізації цього модуля використовуються PHP, MySQL та додаткові технології, такі як відправка електронних листів або Push-сповіщення. Система періодично перевіряє відповідні умови в базі даних і генерує сповіщення за потреби.

Модуль налаштувань користувача дозволяє користувачам персоналізувати свій досвід використання веб-сайту. Користувачі можуть змінювати свої особисті дані, налаштовувати категорії витрат, встановлювати валюту за замовчуванням, управляти сповіщеннями тощо. Для реалізації цього модуля використовуються PHP, MySQL та веб-форми. Користувач може змінювати налаштування через веб-інтерфейс, а система зберігає ці зміни в базі даних.

Для забезпечення ефективної взаємодії між клієнтською та серверною частинами веб-сайту використовується архітектура AJAX (Asynchronous JavaScript and XML). AJAX дозволяє здійснювати асинхронні запити до сервера без перезавантаження сторінки, що забезпечує плавність та інтерактивність користувацького інтерфейсу. Наприклад, при додаванні нової транзакції, дані відправляються на сервер за допомогою AJAX-запиту, а сервер повертає оновлений список транзакцій, який динамічно відображається на сторінці без її перезавантаження.

Для реалізації функціональної частини системи також використовуються різноманітні бібліотеки та фреймворки. Наприклад, для спрощення роботи з DOM-деревом та обробки подій використовується бібліотека jQuery. Для стилізації веб-сторінок та забезпечення адаптивності дизайну використовується фреймворк Bootstrap. Ці інструменти значно прискорюють процес розробки та покращують якість коду.

Важливим аспектом розробки функціональної частини системи є забезпечення безпеки даних користувачів. Для захисту від поширених веб-вразливостей, таких як SQL-ін'єкції та міжсайтовий скриптинг (XSS), використовуються відповідні методи фільтрації та екранування вхідних даних. Також застосовуються техніки шифрування даних при передачі між клієнтом та сервером за допомогою протоколу HTTPS.

Процес розробки функціональної частини системи також включає ретельне тестування та налагодження. Кожен модуль та функція проходять модульне тестування (unit testing) для перевірки коректності роботи окремих компонентів. Інтеграційне тестування дозволяє переконатися, що різні модулі правильно взаємодіють між собою. Крім того, проводиться тестування користувацького інтерфейсу та юзабіліті для забезпечення зручності використання веб-сайту.

Для управління версіями коду та командної роботи над проектом використовується система контролю версій Git. Git дозволяє зберігати історію змін коду, створювати гілки для паралельної розробки різних функцій та злиття змін в основну версію проекту. Це забезпечує ефективну співпрацю між розробниками та дозволяє відстежувати та вирішувати конфлікти в коді.

Таким чином, розробка функціональної частини системи для веб-сайту управління особистими фінансами є комплексним процесом, який охоплює різні аспекти веб-розробки. Від визначення основних функціональних модулів до реалізації конкретних функцій з використанням сучасних технологій та інструментів, кожен етап вимагає ретельного планування, проектування та тестування. Результатом є потужна та надійна система, яка надає користувачам зручні інструменти для ефективного управління своїми фінансами та досягнення фінансових цілей.

3.3. Написання інструкції користувача

Інструкція користувача є невід'ємною частиною будь-якого програмного продукту, і веб-сайт для управління особистими фінансами не є винятком. Вона слугує путівником для користувачів, надаючи чіткі вказівки та пояснення щодо використання системи та її функціональних можливостей. У цьому підрозділі ми розглянемо процес написання інструкції користувача для нашого веб-сайту та наведемо її приклад.

Інструкція користувача повинна бути написана простою та зрозумілою мовою, враховуючи різний рівень технічної підготовки користувачів. Вона повинна мати логічну структуру та бути розділена на окремі розділи або глави, кожен з яких присвячений певній функціональній області веб-сайту.

Нижче наведена інструкція користувача для нашого веб-сайту управління особистими фінансами:

Інструкція користувача: Веб-сайт для управління особистими фінансами

1. **Вступ.** Ласкаво просимо до веб-сайту для управління особистими фінансами! Наша система надає зручні інструменти для відстеження, аналізу та планування ваших фінансів. Ця інструкція допоможе вам ознайомитися з основними функціями та можливостями веб-сайту.
2. **Реєстрація та вхід.** Для початку роботи з веб-сайтом необхідно зареєструватися та створити обліковий запис. Для цього перейдіть на сторінку реєстрації та заповніть необхідні поля, такі як ім'я користувача, електронна пошта та пароль. Після успішної реєстрації ви зможете увійти в систему, використовуючи свої облікові дані.
3. **Головна сторінка та навігація.** Після входу в систему ви потрапите на головну сторінку веб-сайту. На ній відображається загальна інформація про ваші фінанси, такі як поточний баланс, останні транзакції та зведена статистика. Для навігації по веб-сайту використовуйте головне меню, розташоване у верхній частині сторінки. Меню містить посилання на основні розділи, такі як "Транзакції", "Категорії", "Звіти" тощо.

4. **Управління транзакціями.** Розділ "Транзакції" дозволяє вам відстежувати ваші доходи та витрати. Для додавання нової транзакції натисніть кнопку "Додати транзакцію" та заповніть необхідні поля, такі як тип транзакції (дохід або витрата), сума, категорія та опис. Ви також можете редагувати або видаляти існуючі транзакції за потреби.
5. **Категорії та бюджети.** У розділі "Категорії" ви можете створювати та управляти категоріями для ваших транзакцій. Це допоможе вам краще організувати ваші фінанси та відстежувати витрати за різними категоріями. Для створення нової категорії натисніть кнопку "Додати категорію" та введіть назву категорії. Ви також можете встановлювати бюджети для кожної категорії та відстежувати їх виконання.
6. **Звіти та статистика.** Розділ "Звіти" надає вам детальну інформацію про ваші фінанси у вигляді звітів та статистичних даних. Ви можете генерувати звіти за певний період, такі як місячний або річний звіт, та переглядати розподіл ваших доходів і витрат за категоріями. Також доступні графіки та діаграми для візуалізації ваших фінансових даних.
7. **Планування та цілі.** У розділі "Планування" ви можете встановлювати фінансові цілі та планувати ваші майбутні доходи та витрати. Ви можете створювати плани на короткострокову та довгострокову перспективу, встановлювати цільові суми та відстежувати прогрес досягнення ваших цілей. Система надає інструменти для візуалізації вашого фінансового плану та допомагає вам залишатися на шляху до фінансової стабільності.
8. **Налаштування користувача.** У розділі "Налаштування" ви можете персоналізувати ваш обліковий запис та налаштувати параметри системи відповідно до ваших потреб. Ви можете змінювати особисті дані, такі як ім'я та електронна пошта, оновлювати пароль, налаштовувати сповіщення та нагадування, а також вибирати мову інтерфейсу.
9. **Вирішення проблем та підтримка.** Якщо у вас виникли питання або проблеми під час користування веб-сайтом, перегляньте розділ "Поширені запитання" (FAQ) для отримання швидких відповідей. Якщо

ви не знайшли потрібну інформацію, ви можете звернутися до нашої служби підтримки, використовуючи контактну форму або електронну пошту, вказану в розділі "Контакти". Наша команда підтримки завжди готова допомогти вам у вирішенні будь-яких питань, пов'язаних з використанням системи.

Ця інструкція користувача надає загальний огляд основних функцій та можливостей веб-сайту для управління особистими фінансами. Вона охоплює ключові аспекти, такі як реєстрація та вхід, навігація по сайту, управління транзакціями, категоріями та бюджетами, генерація звітів, планування фінансових цілей та налаштування користувача.

При написанні інструкції було враховано принципи ясності, стислості та послідовності. Кожен розділ містить чіткі вказівки та пояснення, доповнені візуальними елементами, такими як скріншоти або ілюстрації, для кращого розуміння.

Інструкція користувача буде доступна безпосередньо на веб-сайті у форматі онлайн-довідки, а також у вигляді окремого документа, який можна завантажити у форматі PDF для зручності друку або перегляду в автономному режимі.

Для полегшення навігації по інструкції користувача передбачено використання змісту, гіперпосилань між розділами та функції пошуку. Це дозволить користувачам швидко знаходити потрібну інформацію та отримувати відповіді на свої запитання.

Інструкція користувача буде регулярно оновлюватися та доповнюватися відповідно до змін та оновлень функціональності веб-сайту. Відгуки та запитання користувачів будуть враховуватися для подальшого вдосконалення інструкції та забезпечення її актуальності та корисності.

Окрім текстової інструкції, планується створення відеоуроків та інтерактивних туторіалів, які наочно продемонструють використання різних функцій веб-сайту. Це дозволить користувачам швидко освоїти систему та ефективно використовувати її можливості для управління особистими фінансами.

Таким чином, інструкція користувача є важливим компонентом веб-сайту для

управління особистими фінансами. Вона слугує надійним путівником для користувачів, надаючи чіткі вказівки та пояснення щодо використання системи. Розроблена інструкція враховує потреби та очікування користувачів, забезпечуючи їх необхідною інформацією для ефективного управління фінансами за допомогою нашого веб-сайту.

3.4. Тестування та налагодження веб-сайту

Тестування та налагодження веб-сайту є невід'ємною частиною процесу розробки, яка забезпечує якість, надійність та безпеку системи. У цьому підрозділі ми розглянемо підходи та методи, які були використані для тестування та налагодження нашого веб-сайту для управління особистими фінансами.

Процес тестування веб-сайту включав кілька етапів і типів тестування, кожен з яких мав свою специфіку та цілі. Нижче наведено основні типи тестування, які були застосовані:

1. Функціональне тестування:
   - Перевірка коректності роботи основних функцій веб-сайту, таких як реєстрація, авторизація, додавання та редагування транзакцій, генерація звітів тощо.
   - Тестування граничних значень та перевірка обробки некоректних даних.
   - Перевірка навігації та переходів між сторінками веб-сайту.
2. Тестування зручності використання (Usability Testing):
   - Оцінка зручності інтерфейсу користувача та простоти використання веб-сайту.
   - Перевірка інтуїтивності та послідовності виконання основних задач користувачами.
   - Збір відгуків та пропозицій від реальних користувачів для покращення досвіду користування.
3. Тестування безпеки:

- Перевірка захищеності веб-сайту від поширених вразливостей, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та підробка міжсайтових запитів (CSRF).
- Тестування автентифікації та авторизації користувачів, перевірка безпеки паролів та сеансів.
- Перевірка конфіденційності та цілісності даних користувачів.

4. Тестування сумісності:
   - Перевірка коректності відображення та функціонування веб-сайту на різних веб-браузерах (Chrome, Firefox, Safari, Edge тощо).
   - Тестування веб-сайту на різних пристроях (настільні комп'ютери, ноутбуки, планшети, смартфони) та операційних системах.
   - Перевірка адаптивності дизайну та коректності відображення елементів інтерфейсу на різних розмірах екрану.
5. Тестування продуктивності:
   - Оцінка швидкості завантаження сторінок веб-сайту та часу відгуку сервера.
   - Перевірка стабільності роботи веб-сайту під навантаженням та при великій кількості одночасних користувачів.
   - Виявлення потенційних вузьких місць та оптимізація продуктивності системи.

Для проведення тестування використовувалися як ручні, так і автоматизовані методи. Ручне тестування виконувалося командою тестувальників, які перевіряли функціональність веб-сайту відповідно до розроблених тестових сценаріїв. Автоматизоване тестування здійснювалося за допомогою спеціальних інструментів, таких як Selenium, PHPUnit та JMeter, які дозволяють створювати та виконувати автоматизовані тести.

Під час тестування були виявлені та задокументовані різноманітні помилки та недоліки веб-сайту. Кожна знайдена проблема була класифікована за рівнем критичності (низький, середній, високий) та пріоритетом виправлення. Розробники активно працювали над усуненням виявлених помилок та вдосконаленням системи

на основі результатів тестування.

Таблиця 3.2

Результати тестування

№	Опис помилки	Тип тестування	Рівень критичності	Статус
1	Неможливість додати транзакцію з негативною сумою	Функціональне тестування	Високий	Виправлено
2	Некоректне відображення дати транзакції у звітах	Функціональне тестування	Середній	Виправлено
3	Відсутність обмеження на довжину поля "Опис транзакції"	Функціональне тестування	Низький	Виправлено
4	Повільне завантаження сторінки зі списком транзакцій при великій кількості записів	Тестування продуктивності	Високий	Виправлено
5	Некоректне відображення веб-сайту на мобільних пристроях з маленьким екраном	Тестування сумісності	Середній	Виправлено

Процес налагодження веб-сайту тісно пов'язаний з тестуванням і спрямований на пошук та усунення помилок у коді. Для налагодження використовувалися інструменти розробки веб-браузерів, такі як Chrome DevTools та Firefox Developer Tools, які дозволяють відстежувати виконання коду, переглядати консольні повідомлення та змінювати код в режимі реального часу.

Крім того, використовувалися спеціалізовані інструменти для налагодження PHP-коду, такі як Xdebug та PHPStorm. Ці інструменти надають можливість встановлювати точки зупинки, покроково виконувати код та перевіряти значення змінних під час виконання.

Налагодження веб-сайту також включало аналіз журналів помилок сервера та

бази даних. Журнали помилок дозволяють виявити винятки, попередження та інші проблеми, які виникають під час роботи веб-сайту. Аналіз журналів допомагає зрозуміти причини виникнення помилок та знайти способи їх усунення.

Окрім функціональних помилок, особлива увага приділялася питанням безпеки веб-сайту. Були проведені тести на проникнення (penetration testing) для виявлення потенційних вразливостей та недоліків безпеки. Виявлені проблеми, такі як можливість SQL-ін'єкцій або міжсайтового скриптингу, були ретельно досліджені та виправлені для забезпечення захисту даних користувачів та цілісності системи.

Процес тестування та налагодження веб-сайту був ітеративним та тривав протягом усього циклу розробки. Кожна нова версія веб-сайту проходила ретельне тестування перед розгортанням на робочому середовищі. Це дозволило забезпечити високу якість та надійність кінцевого продукту.

Важливо зазначити, що тестування та налагодження веб-сайту не обмежуються лише етапом розробки. Після запуску веб-сайту в експлуатацію продовжується моніторинг його роботи, збір відгуків користувачів та виявлення потенційних проблем. Регулярні оновлення та виправлення помилок забезпечують стабільність роботи веб-сайту та задоволеність користувачів.

Таким чином, тестування та налагодження веб-сайту для управління особистими фінансами є критично важливими етапами розробки, які забезпечують якість, надійність та безпеку системи. Застосування різних типів тестування, використання автоматизованих інструментів та ретельний аналіз результатів дозволили виявити та усунути більшість помилок та недоліків веб-сайту. Завдяки цьому процесу ми змогли створити високоякісний та надійний продукт, який відповідає потребам користувачів та забезпечує ефективне управління особистими фінансами.

3.5. Висновки до розділу

У третьому розділі кваліфікаційної роботи було детально розглянуто процес реалізації веб-сайту для управління особистими фінансами. Розділ охопив ключові аспекти розробки, включаючи дизайн веб-сайту, розробку функціональної частини

системи, написання інструкції користувача, а також тестування та налагодження веб-сайту.

При розробці дизайну веб-сайту було враховано сучасні тенденції веб-дизайну, принципи юзабіліті та потреби користувачів. Було створено привабливий, інтуїтивно зрозумілий та функціональний інтерфейс користувача, який забезпечує зручну навігацію та доступ до всіх необхідних інструментів для ефективного управління фінансами. Особливу увагу було приділено адаптивності дизайну для коректного відображення на різних пристроях та розмірах екрану.

Розробка функціональної частини системи включала реалізацію основних модулів та функцій веб-сайту. Було створено систему реєстрації та автентифікації користувачів, яка забезпечує безпечний доступ до особистих фінансових даних. Реалізовано функціонал для управління транзакціями, включаючи додавання, редагування та видалення записів, а також категоризацію транзакцій за типами та користувацькими категоріями. Розроблено інструменти для генерації звітів та візуалізації фінансових даних у вигляді графіків та діаграм. Також було реалізовано функції планування бюджету та встановлення фінансових цілей.

Для забезпечення ефективного використання веб-сайту було створено детальну інструкцію користувача. Інструкція надає покрокові вказівки щодо основних функцій та можливостей системи, охоплюючи процеси реєстрації, авторизації, управління транзакціями, генерації звітів та налаштування користувача. Інструкція була розроблена з урахуванням принципів ясності, стислості та послідовності, щоб користувачі могли легко знаходити потрібну інформацію та розуміти, як працювати з веб-сайтом.

Тестування та налагодження веб-сайту були невід'ємною частиною процесу розробки. Було проведено ретельне функціональне тестування для перевірки коректності роботи всіх модулів та функцій системи. Також було виконано тестування зручності використання, безпеки, сумісності та продуктивності веб-сайту. Виявлені помилки та недоліки були задокументовані та успішно виправлені. Процес налагодження дозволив оптимізувати роботу веб-сайту та забезпечити його стабільність.

Результатом реалізації веб-сайту для управління особистими фінансами став повнофункціональний та зручний у використанні інструмент, який відповідає потребам користувачів та забезпечує ефективне управління фінансовими даними. Веб-сайт надає можливості для відстеження доходів і витрат, категоризації транзакцій, генерації звітів, планування бюджету та досягнення фінансових цілей. Завдяки продуманому дизайну, інтуїтивно зрозумілому інтерфейсу та детальній інструкції користувача, веб-сайт є доступним та зрозумілим для широкого кола користувачів.

Під час розробки веб-сайту було дотримано принципів безпеки та конфіденційності даних. Використано надійні методи шифрування, захищені протоколи передачі даних та реалізовано механізми захисту від поширених веб-вразливостей. Це забезпечує високий рівень захисту фінансової інформації користувачів та дотримання вимог законодавства щодо захисту персональних даних.

Варто зазначити, що розроблений веб-сайт для управління особистими фінансами має потенціал для подальшого розвитку та вдосконалення. Можливі напрямки розвитку включають інтеграцію з додатковими фінансовими сервісами, розширення функціональності для інвестиційного планування, додавання можливості співпраці та обміну даними з іншими користувачами, а також розробку мобільних додатків для зручного доступу до фінансової інформації з будь-якого пристрою.

Таким чином, у третьому розділі кваліфікаційної роботи було успішно реалізовано веб-сайт для управління особистими фінансами з використанням сучасних веб-технологій та підходів до розробки. Розроблений веб-сайт відповідає поставленим вимогам, забезпечує зручність використання, безпеку даних та надає користувачам ефективні інструменти для контролю та оптимізації їхніх фінансів. Реалізація веб-сайту демонструє практичне застосування набутих знань та навичок у галузі веб-розробки та закладає основу для подальшого професійного зростання та вдосконалення.

ВИСНОВКИ

У даній кваліфікаційній роботі було успішно розроблено веб-сайт для управління особистими фінансами з використанням сучасних веб-технологій, таких як HTML, PHP, CSS та MySQL. Метою роботи було створення функціонального та зручного у використанні інструменту, який допоможе користувачам ефективно контролювати свої доходи та витрати, планувати бюджет та аналізувати фінансові показники.

В ході виконання роботи було проведено ґрунтовний аналіз предметної області, який включав огляд існуючих рішень для управління особистими фінансами, визначення функціональних та нефункціональних вимог до веб-сайту, а також формування завдання на розробку. Цей етап дозволив чітко окреслити цілі та завдання проекту та закласти міцний фундамент для подальшої розробки.

Проектування веб-сайту було виконано з урахуванням принципів модульності, масштабованості та безпеки. Було обрано архітектуру MVC (Model-View-Controller) на базі PHP, яка забезпечує чіткий розподіл відповідальності між компонентами системи. Для зберігання даних було спроектовано структуру бази даних з використанням MySQL, яка відповідає вимогам цілісності та ефективності доступу до даних. Розроблено привабливий та інтуїтивно зрозумілий дизайн користувацького інтерфейсу з використанням HTML та CSS, який забезпечує зручну навігацію та взаємодію з веб-сайтом.

Реалізація веб-сайту включала розробку основних функціональних модулів, таких як реєстрація та автентифікація користувачів, управління транзакціями, генерація звітів та статистики, планування бюджету, сповіщення та нагадування. Кожен модуль було ретельно протестовано та налагоджено для забезпечення коректної роботи та відповідності вимогам. Особливу увагу було приділено питанням безпеки, таким як захист від поширених веб-вразливостей та конфіденційність даних користувачів.

Для ефективного використання веб-сайту було розроблено детальну

інструкцію користувача, яка надає чіткі вказівки та пояснення щодо всіх функцій та можливостей системи. Інструкція доступна безпосередньо на веб-сайті та у форматі PDF для зручності користувачів.

Результатом виконаної роботи є повнофункціональний веб-сайт для управління особистими фінансами, який відповідає поставленим вимогам та очікуванням користувачів. Веб-сайт надає зручні інструменти для відстеження доходів і витрат, категоризації транзакцій, генерації звітів, планування бюджету та досягнення фінансових цілей. Завдяки інтуїтивно зрозумілому інтерфейсу та детальній інструкції, веб-сайт є доступним для широкого кола користувачів, незалежно від їх технічної підготовки.

Розроблений веб-сайт має значний потенціал для подальшого розвитку та вдосконалення. Можливі напрямки включають інтеграцію з додатковими фінансовими сервісами, розширення функціональності для інвестиційного планування, додавання можливості співпраці та обміну даними з іншими користувачами, а також розробку мобільних додатків.

Таким чином, кваліфікаційна робота демонструє успішне застосування набутих знань та навичок у галузі веб-розробки для створення практичного та корисного продукту. Розроблений веб-сайт для управління особистими фінансами є ефективним інструментом, який допоможе користувачам покращити контроль над своїми фінансами, приймати обґрунтовані фінансові рішення та досягати поставлених фінансових цілей.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Duckett J. HTML and CSS: Design and Build Websites. John Wiley & Sons, 2011. 490 с.
2. Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. O'Reilly Media, 2018. 832 с.
3. Флэнаган Д. JavaScript: The Definitive Guide. O'Reilly Media, 2020. 706 с.
4. Роббинс Дж. Н. HTML5 Pocket Reference. O'Reilly Media, 2013. 238 с.
5. Бибо Б., Кац И. jQuery. Докладний посібник із просунутого JavaScript. Символ-Плюс, 2017. 624 с.
6. Зандстра М. PHP. Об'єкти, шаблони та методики програмування. Вильямс, 2019. 576 с.
7. Що таке HTML і для чого він потрібен. URL: https://goit.global/ua/articles/html-i-css-shcho-tse-komu-ta-dlia-choho-potribno/ (дата звернення 24.04.2024)
8. Основні переваги субд MySql. URL: https://studfile.net/preview/5607354/page:3/ (дата звернення 25.04.2024)
9. Що таке MySQL. URL: https://romul.name/mysql/#:~:text=MySQL (дата звернення 25.04.2024)
10. Нормальні форми бази даних. URL: https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture02 (дата звернення 25.04.2024)
11. АУТЕНТИФІКАЦІЯ, АВТОРИЗАЦІЯ ТА ІДЕНТИФІКАЦІЯ: ЯК НЕ СПЛУТАТИ. URL: https://training.qatestlab.com/blog/technical-articles/authentication-authorization-and-identification/ (дата звернення 25.04.2024)
12. ЩО ТАКЕ ТЕСТУВАННЯ ПЗ, ЙОГО ЕТАПИ, ВИДИ, ІНСТРУМЕНТИ. URL: https://university.sigma.software/what-is-software-testing/ (дата звернення 26.04.2024)
13. Як за допомогою тестів пришвидшити реліз. URL: https://dou.ua/lenta/articles/how-testing-speed-up-release/ (дата звернення 26.04.2024)

14. Тестова піраміда. URL: https://www.linkedin.com/pulse/%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D0%B0-%D0%BF%D1%96%D1%80%D0%B0%D0%BC%D1%96%D0%B4%D0%B0-alexander-borodin/ (дата звернення 26.04.2024)
15. Модульне тестування. URL: https://qalight.ua/baza-znaniy/modulne-testuvannya/ (дата звернення 26.04.2024)
16. Інтеграційне тестування. URL: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F (дата звернення 26.04.2024)
17. Що таке End-to-End тестування. URL: https://www.it-notes.wiki/software-testing/what-is-end-to-end-testing/ (дата звернення 26.04.2024)

ДОДАТКИ

Додаток А

ПРОГРАМНИЙ КОД

Програмний код з файлу functions.php.

```
//Планування транзакцій
function getPlannedTransactions($userId) {
    global $db;
    $query = "SELECT * FROM planned_transactions WHERE user_id = :user_id ORDER BY date";
    $stmt = $db->prepare($query);
    $stmt->execute(['user_id' => $userId]);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}

function addPlannedTransaction($userId, $type, $amount, $description, $date) {
    global $db;
    $query = "INSERT INTO planned_transactions (user_id, type, amount, description, date) VALUES (:user_id, :type, :amount, :description, :date)";
    $stmt = $db->prepare($query);
    $stmt->execute([
        'user_id' => $userId,
        'type' => $type,
        'amount' => $amount,
        'description' => $description,
        'date' => $date
    ]);
}
```

```
function updatePlannedTransaction($id, $type, $amount, $description, $date) {
    global $db;
    $query = "UPDATE planned_transactions SET type = :type, amount = :amount,
description = :description, date = :date WHERE id = :id";
    $stmt = $db->prepare($query);
    $stmt->execute([
        'id' => $id,
        'type' => $type,
        'amount' => $amount,
        'description' => $description,
        'date' => $date
    ]);
}

function deletePlannedTransaction($id) {
    global $db;
    $query = "DELETE FROM planned_transactions WHERE id = :id";
    $stmt = $db->prepare($query);
    $stmt->execute(['id' => $id]);
}

function calculateProjectedBalance($userId) {
    global $db;
    $currentBalance = getUserBalance($userId);
    $currentMonth = date('Y-m');

    $query = "SELECT SUM(CASE WHEN type = 'income' THEN amount ELSE -amount
END) AS balance_change
            FROM planned_transactions
```

```
        WHERE user_id = :user_id AND DATE_FORMAT(date, '%Y-%m') =
:current_month";
    $stmt = $db->prepare($query);
    $stmt->execute(['user_id' => $userId, 'current_month' => $currentMonth]);
    $result = $stmt->fetch(PDO::FETCH_ASSOC);

    $balanceChange = $result['balance_change'] ?? 0;
    $projectedBalance = $currentBalance + $balanceChange;

    return $projectedBalance;
}

//Звіт

function getTransactionsBetweenDates($userId, $startDate, $endDate) {
    global $db;
    $query = "SELECT * FROM transactions WHERE user_id = :user_id AND
DATE(timestamp) BETWEEN :start_date AND :end_date";
    $stmt = $db->prepare($query);
    $stmt->execute(['user_id' => $userId, 'start_date' => $startDate, 'end_date' =>
$endDate]);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}

function getPlannedTransactionsBetweenDates($userId, $startDate, $endDate) {
    global $db;
    $query = "SELECT * FROM planned_transactions WHERE user_id = :user_id AND
date BETWEEN :start_date AND :end_date";
    $stmt = $db->prepare($query);
```

```
    $stmt->execute(['user_id' => $userId, 'start_date' => $startDate, 'end_date' =>
$endDate]);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}

function generateBudgetRecommendations($userId, $startDate, $endDate,
$compareStartDate = null) {
    // Тут ви можете реалізувати логіку для генерації рекомендацій по бюджету
    // на основі фактичних та запланованих транзакцій, а також порівняння з іншим
періодом, якщо вказано

    $recommendations = [
        'Рекомендація 1: Збільшити заощадження на 10% від доходу',
        'Рекомендація 2: Зменшити витрати на розваги на 20%',
        'Рекомендація 3: Переглянути абонементи та підписки, які не використовуються
регулярно'
    ];

    return $recommendations;
}

function generateReport($userId, $startDate, $endDate) {
    global $db;
    $query = "SELECT c.name AS category, t.type, SUM(t.amount) AS total
            FROM transactions t
            JOIN transaction_categories c ON t.category_id = c.id
            WHERE t.user_id = :user_id AND t.date BETWEEN :start_date AND :end_date
            GROUP BY c.name, t.type";
    $stmt = $db->prepare($query);
```

```
    $stmt->execute(['user_id' => $userId, 'start_date' => $startDate, 'end_date' =>
$endDate]);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}

//віджети на головній
function getTransactionsSumByType($userId, $type) {
    global $db;
    $query = "SELECT SUM(amount) AS total FROM transactions WHERE user_id =
:user_id AND type = :type";
    $stmt = $db->prepare($query);
    $stmt->execute(['user_id' => $userId, 'type' => $type]);
    return $stmt->fetchColumn();
}

function getPlannedTransactionsSumByType($userId, $type) {
    global $db;
    $query = "SELECT SUM(amount) AS total FROM planned_transactions WHERE
user_id = :user_id AND type = :type";
    $stmt = $db->prepare($query);
    $stmt->execute(['user_id' => $userId, 'type' => $type]);
    return $stmt->fetchColumn();
}

//категорії
function createTransactionCategory($userId, $categoryName) {
    global $db;
    $query = "INSERT INTO transaction_categories (user_id, name) VALUES (:user_id,
:name)";
```

```
    $stmt = $db->prepare($query);
    $stmt->execute(['user_id' => $userId, 'name' => $categoryName]);
}

function updateTransactionCategory($categoryId, $categoryName) {
    global $db;
    $query = "UPDATE transaction_categories SET name = :name WHERE id = :id";
    $stmt = $db->prepare($query);
    $stmt->execute(['id' => $categoryId, 'name' => $categoryName]);
}

function deleteTransactionCategory($categoryId) {
    global $db;
    $query = "DELETE FROM transaction_categories WHERE id = :id";
    $stmt = $db->prepare($query);
    $stmt->execute(['id' => $categoryId]);
}

function getTransactionCategories($userId) {
    global $db;
    $query = "SELECT * FROM transaction_categories WHERE user_id = :user_id";
    $stmt = $db->prepare($query);
    $stmt->execute(['user_id' => $userId]);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

Програмний код з файлу index.php.

```
<?php
session_start();
```

```
require_once 'php/functions.php';

if (!isLoggedIn()) {
    header('Location: login.php');
    exit;
}

$userId = $_SESSION['user_id'];
$actualTransactions = getUserTransactions($userId);
$plannedTransactions = getPlannedTransactions($userId);
$currentBalance = getUserBalance($userId);

include 'templates/header.php';
?>

<div class="container">
    <h1 class="text-center mb-4">Панель керування фінансами</h1>

    <div class="row">
        <div class="col-md-6">
            <div class="card mb-4">
                <div class="card-body">
                    <h5 class="card-title">Поточний баланс</h5>
                    <p class="card-text display-4 text-success" id="currentBalance"><?=
$currentBalance ?> грн.</p>
                </div>
            </div>
        </div>
        <div class="col-md-6">
```

```
            <div class="card mb-4">
                <div class="card-body">
                    <h5 class="card-title">Бюджет на поточний місяць</h5>
                    <canvas id="budgetChart"></canvas>
                </div>
            </div>
        </div>

        <div class="row">
            <div class="col-md-6">
                <div class="card mb-4">
                    <div class="card-body">
                        <h5 class="card-title">Фактичні транзакції</h5>
                        <canvas id="actualTransactionsChart"></canvas>
                    </div>
                </div>
            </div>
            <div class="col-md-6">
                <div class="card mb-4">
                    <div class="card-body">
                        <h5 class="card-title">Заплановані транзакції</h5>
                        <canvas id="plannedTransactionsChart"></canvas>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

```
<?php include 'templates/footer.php'; ?>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
    var actualTransactionsCtx =
document.getElementById('actualTransactionsChart').getContext('2d');
    var plannedTransactionsCtx =
document.getElementById('plannedTransactionsChart').getContext('2d');
    var budgetChartCtx = document.getElementById('budgetChart').getContext('2d');

    var actualTransactionsData = {
        labels: ['Надходження', 'Витрати'],
        datasets: [{
            data: [<?= getTransactionsSumByType($userId, 'income') ?>, <?=
getTransactionsSumByType($userId, 'expense') ?>],
            backgroundColor: ['#36a2eb', '#ff6384']
        }]
    };

    var plannedTransactionsData = {
        labels: ['Заплановані надходження', 'Заплановані витрати'],
        datasets: [{
            data: [<?= getPlannedTransactionsSumByType($userId, 'income') ?>, <?=
getPlannedTransactionsSumByType($userId, 'expense') ?>],
            backgroundColor: ['#4bc0c0', '#ffcd56']
        }]
    };

    var budgetData = {
```

```
        labels: ['Витрачено', 'Залишилось'],
        datasets: [{
            data: [<?= getTransactionsSumByType($userId, 'expense') ?>, <?=
$currentBalance - getTransactionsSumByType($userId, 'expense') ?>],
            backgroundColor: ['#ff6384', '#36a2eb']
        }]
    };
    new Chart(actualTransactionsCtx, {
        type: 'doughnut',
        data: actualTransactionsData,
        options: {
            responsive: true,
            title: {
                display: true,
                text: 'Фактичні транзакції'
            }
        }
    });

    new Chart(plannedTransactionsCtx, {
        type: 'doughnut',
        data: plannedTransactionsData,
        options: {
            responsive: true,
            title: {
                display: true,
                text: 'Заплановані транзакції'
            }
        }
```

```
    });

    new Chart(budgetChartCtx, {
        type: 'doughnut',
        data: budgetData,
        options: {
            responsive: true,
            title: {
                display: true,
                text: 'Бюджет на поточний місяць'
            }
        }
    });

    function animateBalance() {
        var balance = document.getElementById('currentBalance');
        var currentValue = parseFloat(balance.textContent);
        var targetValue = <?= $currentBalance ?>;
        var increment = (targetValue - currentValue) / 100;
        if (currentValue < targetValue) {
            currentValue += increment;
            balance.textContent = currentValue.toFixed(2) + ' грн.';
            requestAnimationFrame(animateBalance);
        } else {
            balance.textContent = targetValue.toFixed(2) + ' грн.';
        }
    }
    animateBalance();
</script>
```

Додаток Б

ЗАПИТ ДЛЯ СТВОРЕННЯ БАЗИ ДАННИХ

```
DROP TABLE IF EXISTS `chat_messages`;
CREATE TABLE `chat_messages` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `message` text NOT NULL,
  `task_time` int(11) DEFAULT NULL,
  `timestamp` timestamp NOT NULL DEFAULT current_timestamp(),
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_general_ci;

DROP TABLE IF EXISTS `planned_transactions`;
CREATE TABLE `planned_transactions` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `type` enum('income','expense') NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `description` text DEFAULT NULL,
  `date` date NOT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `planned_transactions_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_general_ci;

DROP TABLE IF EXISTS `transactions`;
CREATE TABLE `transactions` (
```

```
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `timestamp` timestamp NOT NULL DEFAULT current_timestamp(),
  `type` enum('income','expense') NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `description` text DEFAULT NULL,
  `credit` tinyint(1) NOT NULL DEFAULT 0,
  `category_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  KEY `category_id` (`category_id`),
  CONSTRAINT `transactions_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`),
  CONSTRAINT `transactions_ibfk_2` FOREIGN KEY (`category_id`) REFERENCES `transaction_categories` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_general_ci;

SET NAMES utf8mb4;

DROP TABLE IF EXISTS `transaction_categories`;
CREATE TABLE `transaction_categories` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `transaction_categories_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;

DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `balance` decimal(10,2) NOT NULL DEFAULT 0.00,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_general_ci;
```