ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Цифровий сервіс для аналізу та управління особистими фінансами з інтеграцією великої мовної моделі Gemini»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Технології цифрового розвитку»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Вадим ДОВБИУС

(підпис)

Виконав: здобувач вищої освіти групи ТЦР-41

Вадим ДОВБИУС

Керівник: Микола ГЕРЦЮК
д-р філософії

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Технологій цифрового розвитку

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Технології цифрового розвитку»

ЗАТВЕРДЖУЮ

Завідувач кафедри
Технологій цифрового розвитку
_____Вікторія ЖЕБКА
« ____» _____2026 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Довбиусу Вадиму Сергійовичу

1. Тема кваліфікаційної роботи: «Цифровий сервіс для аналізу та управління особистими фінансами з інтеграцією великої мовної моделі Gemini» керівник кваліфікаційної роботи д-р філософії, Микола ГЕРЦЮК, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 57.

2. Строк подання кваліфікаційної роботи «___» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні основи управління особистими фінансами, принципи побудови клієнт-серверних веб-застосунків на базі фреймворку Django, документація Google Gemini API для інтеграції великих мовних моделей, методи візуалізації статистичних даних.

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

1. Теоретичні основи розробки систем управління особистими фінансами та аналіз сучасних підходів.

2. Проєктування архітектури веб-застосунку, структури бази даних та алгоритмів інтелектуального аналізу.

3. Програмна реалізація системи, розробка користувацького інтерфейсу та імплементація модуля інтеграції з Gemini AI.

4. Тестування функціональних можливостей та адміністрування системи.

5. Перелік графічного матеріалу: *презентація*

1. Актуальність та мета дослідження.
2. Порівняльний аналіз існуючих рішень.
3. Обґрунтування технологічного стека (Django, Python, Gemini API).
4. Діаграма варіантів використання (Use Case).
5. Архітектура системи та схема інтеграції з ШІ.
6. Алгоритм формування фінансових рекомендацій.
7. Візуалізація фінансових даних та аналітична звітність.
8. Результати тестування та висновки.

6. Дата видачі завдання «20» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та існуючих систем управління фінансами (Mint, YNAB тощо)	21.02.2026	
2	Обґрунтування вибору технологічного стеку: Python, Django та СУБД SQLite	04.03 – 20.03.2026	
3	Проєктування архітектури бази даних та розробка логічної структури (ER-діаграми)	21.03 – 05.04.2026	
4	Розробка серверної логіки на Django та реалізація моделей даних (Транзакції, Бюджети, Категорії)	06.04 – 25.04.2026	
5	Проєктування та програмна реалізація модуля інтеграції з Gemini API для аналізу витрат	26.04 – 03.05.2026	
6	Розробка інтерфейсу користувача (Dashboard) з використанням шаблонізатора Django та CSS	04.05 – 09.05.2026	
7	Тестування функціональності сервісу, валідація ШІ-порад та налагодження коду	04.05 – 09.05.2026	
8	Оформлення пояснювальної записки та підготовка демонстраційних матеріалів	06.05 – 09.05.2026	
9	Попередній захист роботи	11.05-22.05.2026	

Здобувач вищої освіти _____ Вадим ДОВБИУС
(підпис)

Керівник
кваліфікаційної роботи _____ Микола ГЕРЦЮК
(підпис)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 58 сторінок, 27 рисунків, 3 таблиці, 27 джерел.

Об'єкт дослідження – процеси автоматизованого збору, обробки та аналізу даних особистих фінансів із застосуванням сучасних веб-технологій та інструментів штучного інтелекту.

Предмет дослідження – архітектурні рішення, алгоритми та інструментальні засоби розробки веб-додатку для фінансового обліку, а також методи інтеграції великої мовної моделі Gemini для семантичного аналізу транзакцій та генерації персоналізованих рекомендацій.

Мета роботи – створення повнофункціонального веб-додатку для управління особистими фінансами, який поєднує класичний інструментарій обліку та візуалізації з можливостями великої мовної моделі Gemini для автоматичного пошуку фінансових патернів та надання порад щодо оптимізації бюджету.

Методи дослідження:

Аналіз існуючих архітектурних рішень та технологічних стеків для побудови фінансових веб-додатків.

Алгоритмічне моделювання процесів обробки, категоризації та агрегації фінансових даних.

Проєктування та реалізація клієнт-серверної архітектури з використанням сучасних веб-фреймворків.

Програмна інтеграція (через API) великої мовної моделі Gemini для реалізації модуля інтелектуальної аналітики.

Функціональне тестування розробленого програмного продукту та оцінка якості згенерованих аналітичних звітів.

ВЕБ-ЗАСТОСУНОК, ПЕРСОНАЛЬНІ ФІНАНСИ, БЮДЖЕТУВАННЯ, ТРАНЗАКЦІЇ, АНАЛІТИКА ДАНИХ, ШТУЧНИЙ ІНТЕЛЕКТ, GEMINI AI, DJANGO, PYTHON.

ЗМІСТ

ВСТУП

Актуальність дослідження. На сьогоднішній день більшість фінансових операцій користувачів розподілена між різними банківськими картками, онлайн-сервісами та регулярними підписками. Зазначене зумовлює проблему фрагментації фінансових даних: відстеження реальної структури витрат і планування бюджету ускладнюється пропорційно до кількості джерел транзакцій. Наявні програмні рішення для обліку особистих фінансів здебільшого забезпечують лише базовий функціонал – ручну або напівавтоматичну категоризацію транзакцій та побудову стандартних графіків – без можливості глибокого аналізу фінансової поведінки користувача.

Одним із перспективних напрямів вирішення зазначеної проблеми є інтеграція інструментів штучного інтелекту, зокрема великих мовних моделей (LLM), у системи фінансового обліку. Застосування моделі Gemini дозволяє реалізувати перехід від простої агрегації числових даних до семантичного аналізу фінансової поведінки. На відміну від традиційних підходів, за яких користувач самостійно інтерпретує статистичні дані, запропонована система забезпечує автоматичне виявлення патернів надмірних витрат, оцінювання стану бюджету та формування аргументованих рекомендацій щодо його оптимізації. Отримані результати підтверджують доцільність розробки відповідного веб-додатку, що поєднує класичний облік фінансових операцій з інтелектуальним аналізом даних.

Об’єкт дослідження – процеси автоматизованого збору, обробки та аналізу даних особистих фінансів із застосуванням сучасних веб-технологій та інструментів штучного інтелекту.

Предмет дослідження – програмні методи, алгоритми та інструменти розробки веб-додатків для аналізу фінансових операцій, формування статистики витрат і доходів, а також інтеграція великої мовної моделі Gemini для інтелектуальної обробки фінансових даних та надання рекомендацій користувачам.

Мета роботи – розробка програмного забезпечення у вигляді веб-додатку, який забезпечує ведення обліку фінансових операцій, автоматичний підрахунок

балансу, візуалізацію статистики витрат і доходів, а також використовує API великої мовної моделі Gemini для генерації аналітичних рекомендацій щодо оптимізації фінансових витрат користувача.

Завдання роботи:

1. Проаналізувати недоліки існуючих підходів до розробки систем обліку особистих фінансів та обґрунтувати доцільність впровадження інтелектуальних модулів аналізу даних.
2. Обрати та обґрунтувати технологічний стек для реалізації клієнтської та серверної частин веб-додатку.
3. Спроєктувати реляційну базу даних та архітектуру системи, що забезпечують надійне збереження й ефективну обробку фінансових транзакцій.
4. Розробити алгоритми агрегації фінансових даних та їх підготовки для передачі до зовнішнього API.
5. Здійснити інтеграцію великої мовної моделі Gemini із налаштуванням механізму формування запитів для отримання релевантних аналітичних рекомендацій.
6. Розробити адаптивний інтерфейс користувача для управління бюджетами, перегляду транзакцій та відображення аналітичних звітів.
7. Провести тестування програмного забезпечення на коректність обробки даних та стабільність взаємодії з інтелектуальним модулем.

Методи дослідження:

1. Порівняльний аналіз сучасних веб-фреймворків та архітектурних патернів побудови інформаційних систем.
2. Алгоритмічне моделювання логіки класифікації та групування фінансових операцій за категоріями доходів і витрат.
3. Програмна реалізація клієнтського інтерфейсу з використанням технологій HTML, CSS
4. Практичне тестування працездатності системи на типових сценаріях використання: додавання транзакцій, формування звітів, контроль бюджетних лімітів.

Очікуваним результатом роботи є проєктування та програмна реалізація вебдодатка для обліку й аналізу особистих фінансів. Система забезпечує автоматизований моніторинг фінансових потоків у реальному часі, формування звітності та візуалізацію показників через інтуїтивно зрозумілий інтерфейс.

Центральне місце у дослідженні займає розробка та впровадження інтелектуального модуля на основі великої мовної моделі (LLM) Gemini. Використання методів обробки природної мови уможливлює динамічний семантичний аналіз фінансової поведінки користувача. Впроваджений алгоритм дозволяє системі інтерпретувати контекст операцій, виявляти закономірності у витратах та формувати персоналізовані рекомендації для оптимізації бюджету.

За результатами роботи створено повнофункціональний програмний продукт, архітектура якого побудована на принципах модульності. Це дозволило ефективно інтегрувати класичний вебстек (фреймворк Django та СУБД SQLite) із передовими хмарними інструментами штучного інтелекту від компанії Google.

Апробація результатів дослідження. Основні положення та результати кваліфікаційної роботи були представлені, обговорені та отримали позитивну оцінку на:

1. XI Міжнародній науковій конференції «Здобутки та досягнення прикладних та фундаментальних наук XXI століття» (01 травня 2026 р., м. Одеса);
2. VI Міжнародній науковій конференції «Інноваційна наука: пошук відповідей на виклики сучасності» (15 травня 2026 р., м. Чернівці).

Публікації**.** За темою дослідження опубліковано:

1. Довбиус В. С. Архітектурні рішення та алгоритми обробки даних у системі управління особистими фінансами.
2. Довбиус В. С. Проєктування веб-застосунку для управління особистими фінансами: архітектурний підхід та інтеграція LLM.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

1.1 Аналіз сучасних підходів до управління особистими фінансами

В умовах цифровізації економіки та інтенсивного розвитку фінансових технологій питання ефективного управління особистими фінансами набуває особливої актуальності. Збільшення кількості фінансових операцій, поширення банківських карток, електронних платежів, онлайн-сервісів та підписок зумовлює необхідність систематичного обліку доходів і витрат, а також їх подальшого аналізу для прийняття обґрунтованих фінансових рішень.

Управління особистими фінансами розглядається як процес планування, обліку, контролю та аналізу фінансових ресурсів фізичної особи з метою досягнення фінансової стабільності та підвищення рівня добробуту. Основними складовими цього процесу є облік доходів, контроль витрат, формування бюджету, аналіз фінансових потоків та прогнозування майбутніх витрат (рис. 1.1).

Рис. 1.1 Сучасні підходи до управління особистими фінансами

На поточному етапі розвитку інформаційних технологій виокремлюють кілька основних підходів до управління особистими фінансами, що відрізняються рівнем автоматизації, зручністю застосування та аналітичними можливостями.

Перший підхід – ручний облік фінансів – передбачає ведення записів у паперовому вигляді або засобами електронних таблиць. Перевагами є простота

реалізації та відсутність потреби у спеціалізованому програмному забезпеченні. Разом з тим даний підхід характеризується суттєвими недоліками: значною трудомісткістю, ризиком виникнення помилок при введенні даних та відсутністю засобів автоматизованого аналізу.

Другий підхід передбачає використання електронних таблиць, зокрема Microsoft Excel та Google Sheets. Застосування формул, діаграм і зведених таблиць дозволяє частково автоматизувати процеси обліку та аналізу фінансових даних. Проте даний підхід потребує відповідних технічних навичок користувача, а також не забезпечує повноцінної інтеграції з фінансовими сервісами та автоматичного формування аналітичних рекомендацій.

Третій підхід полягає у застосуванні спеціалізованих мобільних і веб-додатків для управління особистими фінансами. Такі системи забезпечують автоматизований облік доходів і витрат, категоризацію фінансових операцій, формування статистики та візуалізацію даних у вигляді графіків і діаграм. Більшість сучасних рішень підтримують синхронізацію між пристроями та інтеграцію з банківськими сервісами. Водночас значна частина таких додатків має обмежений аналітичний функціонал і не забезпечує глибокого інтелектуального опрацювання фінансових даних.

Протягом останніх років спостерігається активне впровадження методів штучного інтелекту у сферу фінансового аналізу. Інтелектуальні системи здатні опрацьовувати значні обсяги фінансових даних, виявляти закономірності у структурі витрат, прогнозувати фінансові показники та формувати рекомендації щодо оптимізації бюджету. Зокрема, застосування великих мовних моделей розширює можливості інтелектуального аналізу, оскільки такі моделі здатні інтерпретувати дані та генерувати зрозумілі для користувача текстові пояснення і рекомендації [1].

Аналіз даних, наведених у табл. 1.1, свідчить про те, що найбільш перспективним є підхід, що поєднує автоматизований облік фінансових операцій з аналітичними можливостями та формуванням рекомендацій на основі штучного інтелекту. Саме цей підхід покладено в основу розробки досліджуваного веб-

додатку.

Таблиця 1.1

Порівняння сучасних підходів до управління особистими фінансами

Підхід	Інструменти	Переваги	Недоліки
Ручний облік	Паперові записи, блокнот	Простота, не потребує технічних знань	Висока трудомісткість, відсутність автоматизації, ризик помилок
Електронні таблиці	Microsoft Excel, Google Sheets	Часткова автоматизація, можливість побудови графіків	Потребує навичок, обмежена аналітика, відсутність інтелектуальних підказок
Мобільні та веб-додатки	Mint, YNAB, Spendee	Автоматизація обліку, зручний інтерфейс, візуалізація даних	Обмежена персоналізація, відсутність глибокого аналізу
Інтелектуальні системи (AI)	Системи з використанням ШІ, LLM (Gemini)	Аналіз великих обсягів даних, персональні рекомендації, прогнозування	Залежність від API, складність реалізації, потреба в інтернеті

Незважаючи на наявність значної кількості інструментів управління особистими фінансами, більшість існуючих рішень характеризуються недостатнім рівнем персоналізації, обмеженими аналітичними можливостями та складністю використання. Зазначене обумовлює необхідність розробки веб-додатків, що поєднують зручний інтерфейс, автоматизований облік фінансів і засоби

інтелектуального аналізу даних.

Таким чином, аналіз сучасних підходів до управління особистими фінансами підтверджує доцільність створення програмних систем, що використовують сучасні веб-технології та інтегрують інструменти штучного інтелекту з метою підвищення ефективності фінансового управління.

1.2 Огляд існуючих програмних рішень для обліку та аналізу фінансових даних

На сьогоднішній день ринок програмного забезпечення для управління особистими фінансами представлений широким спектром рішень – від мобільних додатків до комплексних веб-платформ з інтеграцією банківських систем. Попри різноманіття наявних інструментів, їх аналітичні можливості та рівень інтелектуального опрацювання даних залишаються суттєво обмеженими.

Основним призначенням таких програмних продуктів є спрощення контролю фінансових потоків, підвищення фінансової дисципліни користувача та забезпечення інструментарію для бюджетного планування. До найбільш відомих систем управління особистими фінансами належать Mint, YNAB (You Need A Budget), Spendee, а також банківські мобільні додатки – Privat24 та Monobank [2].

Сервіс Mint забезпечує автоматичну синхронізацію з банківськими рахунками користувача, категоризацію витрат та формування фінансових звітів. Перевагою є високий рівень автоматизації обліку та розвинений інтерфейс. До характерних обмежень належать географічна недоступність у ряді країн та відсутність механізмів глибокої персоналізації рекомендацій.

Система YNAB побудована на концепції проактивного бюджетного планування, за якою кожна одиниця отриманого доходу заздалегідь розподіляється між статтями витрат [3]. Така методика сприяє формуванню фінансової дисципліни, однак передбачає постійну участь користувача у введенні та актуалізації даних, що знижує практичну зручність для певної категорії користувачів.

Додаток Spendee реалізований у форматі мобільного та веб-рішення і поєднує функції обліку фінансових операцій з їх візуалізацією засобами графіків і діаграм. Серед переваг – зручний інтерфейс та підтримка спільного доступу до фінансових даних кількох користувачів. Разом з тим аналітичний функціонал додатку є обмеженим, а формування інтелектуальних рекомендацій фактично не реалізовано.

Окрему групу складають банківські додатки – Privat24 та Monobank, що надають користувачам доступ до історії транзакцій, платіжних операцій і базової статистики витрат. Їх ключовою перевагою є безпосередня інтеграція з банківською інфраструктурою та автоматичне отримання фінансових даних без ручного введення. Водночас зазначені системи не орієнтовані на розширений аналіз даних і не формують персоналізованих рекомендацій, що обмежує їх застосування як повноцінного інструменту фінансового планування.

Порівняльний аналіз розглянутих рішень дозволяє виокремити низку спільних недоліків: недостатній рівень аналітики, обмежені можливості інтелектуального опрацювання фінансових даних, слабка персоналізація та невисокий ступінь адаптації до індивідуальних потреб користувача.

Актуальним напрямом розвитку програмних рішень у фінансовій сфері є інтеграція технологій штучного інтелекту, зокрема великих мовних моделей. Їх застосування уможливлює виявлення прихованих закономірностей у структурі витрат, прогнозування фінансових показників та формування персоналізованих рекомендацій у текстовому форматі. Проте широке впровадження таких технологій у наявні комерційні системи на сьогодні залишається недостатнім.

Таблиця 1.2

Порівняння існуючих програмних рішень для управління особистими фінансами

Сервіс	Тип	Основні функції	Переваги	Недоліки
Mint	Веб/мобільний	Автоматична синхронізація, категоризація, звіти	Висока автоматизація, зручність	Обмежена доступність, слабка персоналізація
YNAB	Веб/мобільний	Планування бюджету, контроль витрат	Ефективне планування	Потребує ручного введення
Spendee	Веб/мобільний	Облік витрат, графіки, бюджети	Зручний інтерфейс, візуалізація	Обмежена аналітика
Privat24	Мобільний	Історія транзакцій, платежі	Інтеграція з банком	Мінімальна аналітика
Monobank	Мобільний	Категорії витрат, статистика	Автоматичний облік	Відсутність AI-аналітики

Проведений аналіз існуючих програмних рішень засвідчує, що переважна більшість наявних інструментів управління особистими фінансами не забезпечує належного рівня інтелектуального аналізу даних та формування персоналізованих рекомендацій. Зазначене обумовлює актуальність розроблення нових програмних систем, що поєднують автоматизований облік фінансових операцій із застосуванням сучасних технологій штучного інтелекту.

З огляду на викладене, доцільним є розроблення веб-додатку для аналізу та управління особистими фінансами з інтеграцією великої мовної моделі Gemini, що забезпечить підвищення ефективності фінансового аналізу та надання користувачам персоналізованих рекомендацій.

1.3 Використання веб-технологій у створенні фінансових інформаційних систем

Веб-додатки забезпечують доступ до функціоналу системи незалежно від місця перебування користувача та типу пристрою, що є визначальним чинником для сучасних цифрових сервісів. Підставою для вибору веб-платформи як основи реалізації системи управління особистими фінансами слугує її універсальна доступність: користувач має змогу працювати із системою через браузер з будь-якого підключеного до мережі пристрою – персонального комп'ютера, ноутбука, планшета або смартфона. Це скорочує технічні бар'єри взаємодії із системою та розширює аудиторію її потенційних користувачів.

Суттєвою перевагою веб-додатків є відсутність необхідності локального встановлення програмного забезпечення. Обчислення та опрацювання даних виконуються на серверній стороні, тоді як користувач взаємодіє із системою через браузерний інтерфейс. Така архітектура централізує процес оновлення програмного продукту: усі зміни впроваджуються на рівні сервера без будь-якого втручання з боку кінцевого користувача.

Архітектура сучасного веб-додатку ґрунтується на клієнт-серверній моделі, що передбачає розмежування функціональності між клієнтською та серверною частинами. Клієнтська частина (frontend) відповідає за відображення інформації та організацію взаємодії з користувачем, серверна частина (backend) – за обробку даних, реалізацію бізнес-логіки та забезпечення взаємодії з базою даних.

Frontend веб-додатку реалізується з використанням технологій HTML, CSS та JavaScript: HTML визначає структуру сторінки, CSS – її візуальне оформлення, JavaScript – інтерактивність інтерфейсу. Застосування сучасних бібліотек і фреймворків, зокрема React, Vue або Angular, дозволяє розробляти динамічні та зручні у використанні інтерфейси [4].

Backend відповідає за обробку користувацьких запитів, виконання обчислень та взаємодію з базою даних. У межах даної роботи для реалізації серверної частини

обрано мову програмування Python, що є однією з провідних у галузі веб-розробки та аналізу даних. Як серверний фреймворк може застосовуватись Django, він забезпечує високий рівень абстракції та прискорену розробку складних систем.

Для збереження фінансових даних застосовується база даних, що забезпечує надійне зберігання відомостей про доходи, витрати, категорії та інші параметри фінансової діяльності користувача – SQLite.

Застосування веб-технологій у розробці фінансових інформаційних систем зумовлює низку практичних переваг. По-перше, кросплатформеність забезпечує функціонування системи на різних пристроях без необхідності адаптації під конкретну операційну систему (рис. 1.2). По-друге, централізоване зберігання даних гарантує їх цілісність та захист. По-третє, веб-додатки характеризуються високим потенціалом масштабування, що створює передумови для подальшого розширення їх функціональних можливостей.

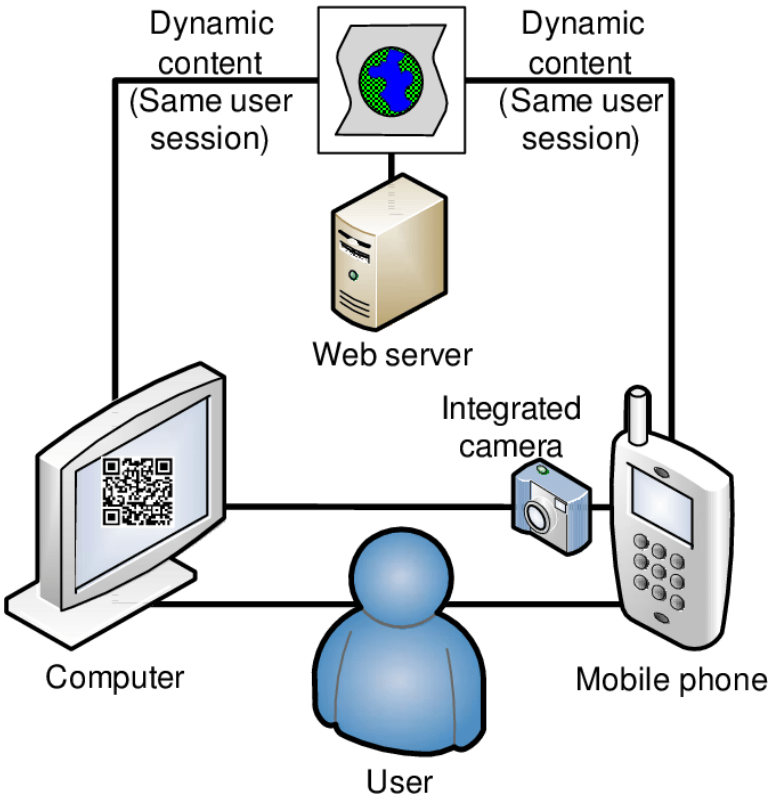


Рис. 1.2 Взаємодія користувача з веб-додатком через різні пристрої

Веб-технології також забезпечують інтеграцію із зовнішніми сервісами через

API, що є особливо значущим у контексті даної роботи. Зокрема, це створює технічну основу для підключення великих мовних моделей, таких як Gemini, з метою реалізації функцій інтелектуального аналізу фінансових даних та формування рекомендацій.

Для розуміння принципу функціонування веб-додатку доцільно розглянути його архітектурну організацію. У переважній більшості сучасних інформаційних систем застосовується клієнт-серверна модель, що забезпечує розподіл обчислювальних процесів між клієнтською та серверною частинами.

У межах даної роботи веб-додаток реалізовано за трирівневою архітектурою, що охоплює клієнтський рівень (frontend), серверний рівень (backend) та рівень збереження даних (база даних). Обмін даними між компонентами системи здійснюється за протоколом HTTP.

Загальну структуру взаємодії основних компонентів системи представлено на рис. 1.3.

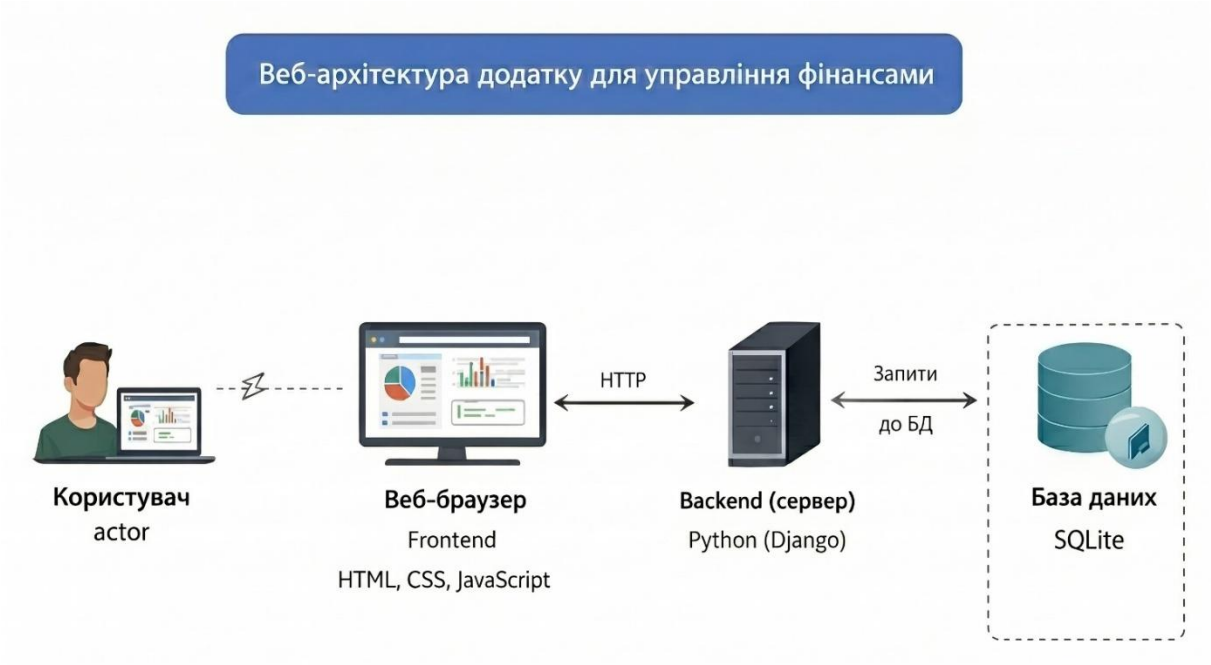


Рис. 1.3 Архітектура веб-додатку для управління особистими фінансами

Взаємодія користувача із системою відбувається через веб-браузер, який виконує функції клієнтської частини. На цьому рівні здійснюється введення даних

про фінансові операції, перегляд статистики та використання функціонального інструментарію веб-додатку.

Запити від клієнтської частини передаються до серверної за протоколом HTTP. Серверна частина, реалізована засобами мови програмування Python із застосуванням фреймворку Django, виконує опрацювання вхідних даних, реалізує бізнес-логіку системи та формує відповіді для клієнта [6].

За результатами обробки запиту сервер звертається до бази даних для збереження або вибірки необхідних відомостей. База даних забезпечує зберігання інформації про фінансові операції користувача – доходи, витрати, категорії та інші параметри, що використовуються у подальшому аналізі.

Отримані з бази даних дані передаються до серверної частини, де формуються у вигляді структурованих об'єктів, придатних для відображення у веб-інтерфейсі. Сформована відповідь повертається клієнту та представляється користувачеві у табличному вигляді, у форматі графіків або діаграм залежно від контексту запиту.

Описана архітектура забезпечує ефективну взаємодію між компонентами системи, уможливлює гнучке опрацювання даних та формує технічну основу для інтеграції інтелектуальних модулів, зокрема великої мовної моделі Gemini, що розширює аналітичні можливості веб-додатку.

1.4 Можливості застосування великих мовних моделей у фінансовому аналізі

З інтенсивним розвитком технологій штучного інтелекту дедалі більшої уваги набуває застосування великих мовних моделей (Large Language Models, LLM) у різних галузях, зокрема у сфері фінансового аналізу. Великі мовні моделі є складними нейронними мережами, навченими на значних масивах текстових даних, що забезпечує їх здатність до обробки природної мови, аналізу інформації та генерації змістовних відповідей.

Характерною перевагою великих мовних моделей є здатність опрацьовувати

неструктуровані дані та перетворювати їх у форму, доступну для сприйняття користувачем. У контексті управління особистими фінансами це створює передумови для автоматизації аналізу фінансової інформації та підтримки прийняття рішень.

Зокрема, у межах відповідних систем можуть бути реалізовані такі функції:

– аналіз структури витрат користувача, визначення основних категорій та виявлення закономірностей фінансової поведінки, що забезпечує глибше розуміння власних фінансових звичок;

– формування персоналізованих рекомендацій щодо оптимізації витрат на основі аналізу фінансових даних – зокрема, пропозиції щодо скорочення витрат у окремих категоріях або коригування бюджету для досягнення визначених фінансових цілей; рекомендації надаються у текстовому форматі, що підвищує їх практичну цінність;

– реалізація інтелектуального діалогового інтерфейсу, що дозволяє користувачеві звертатись із запитами щодо власних фінансів, отримувати пояснення та аналітичні відповіді в інтерактивному режимі.

Також є можливість прогнозування фінансових показників. На основі історичних даних великі мовні моделі можуть оцінювати тенденції у витратах та прогнозувати майбутні фінансові ситуації, що є важливим інструментом для планування бюджету.

У межах даної роботи передбачається застосування великої мовної моделі Gemini – сучасного інструменту штучного інтелекту, здатного опрацьовувати значні обсяги даних та генерувати аналітичні висновки. Інтеграція Gemini у веб-додаток розширює функціональні можливості системи, зокрема забезпечує інтелектуальний аналіз фінансових операцій та формування рекомендацій у режимі реального часу.

Разом з тим застосування великих мовних моделей у фінансових системах пов'язане з певними обмеженнями: залежністю від якості вхідних даних, можливістю формування неточних або надмірно узагальнених рекомендацій, а також необхідністю забезпечення захисту персональних даних користувача.

Відповідно, під час розробки системи питанням інформаційної безпеки та конфіденційності даних приділяється окрема увага.

У структурі запропонованої системи велика мовна модель функціонує як автономний аналітичний модуль, що взаємодіє із серверною частиною веб-додатку через API. Фінансові дані, введені користувачем, або сформовані ним запити опрацьовуються системою та передаються до моделі для подальшого аналізу.

Загальну схему інтеграції великої мовної моделі Gemini у веб-додаток представлено на рис. 1.4.

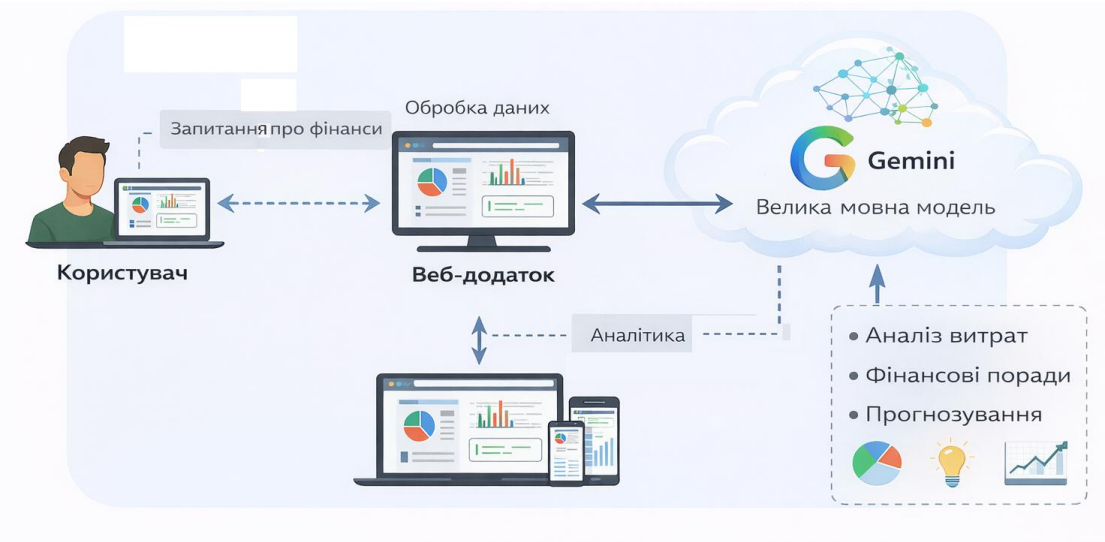


Рис. 1.4 Інтеграція великої мовної моделі Gemini у веб-додаток

Як показано на рис. 1.4, користувач взаємодіє з веб-додатком через інтерфейс, вводячи фінансові дані або надсилаючи запити щодо аналізу витрат. Отримана інформація передається до серверної частини, де виконується її первинне опрацювання та підготовка до аналізу.

Далі підготовлені дані передаються до моделі Gemini, яка здійснює інтелектуальний аналіз, виявляє закономірності у структурі витрат, формує аналітичні висновки та генерує рекомендації щодо оптимізації фінансової діяльності. Результати повертаються до веб-додатку та відображаються у зручному для користувача форматі.

2 ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ АНАЛІЗУ ТА УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

2.1 Формування вимог до веб-додатку та визначення його функціональних можливостей

Процес взаємодії користувача з інтелектуальним модулем системи базується на алгоритмі семантичного аналізу фінансових потоків. Основна логіка роботи сервісу полягає не лише у простій агрегації числових даних, а й у перетворенні сирих транзакційних записів на структурований контекст для великої мовної моделі.

Алгоритм функціонування модуля аналізу можна розділити на кілька ключових етапів:

1. Ініціалізація та збір даних: Система здійснює вибірку актуальних транзакцій користувача за визначений часовий період (місяць або тиждень) із реляційної бази даних SQLite.
2. Формування контекстного запиту (Prompt Engineering): На основі отриманих даних формується спеціалізований запит, що містить деталізовану інформацію про категорії витрат, перевищення бюджетних лімітів та динаміку балансу.
3. Взаємодія з Gemini API: Запит передається до моделі Gemini 1.5 Flash за допомогою захищеного API-з’єднання. На цьому етапі відбувається когнітивна обробка інформації нейромережею.
4. Інтерпретація та збереження: Отримана відповідь проходить через блок валідації, після чого зберігається в локальній базі даних для забезпечення миттєвого доступу без повторних звернень до API.

Деталізовану логіку проходження даних та прийняття рішень системою відображено на блок-схемі алгоритму (Рис. 2.1), що наведена нижче:

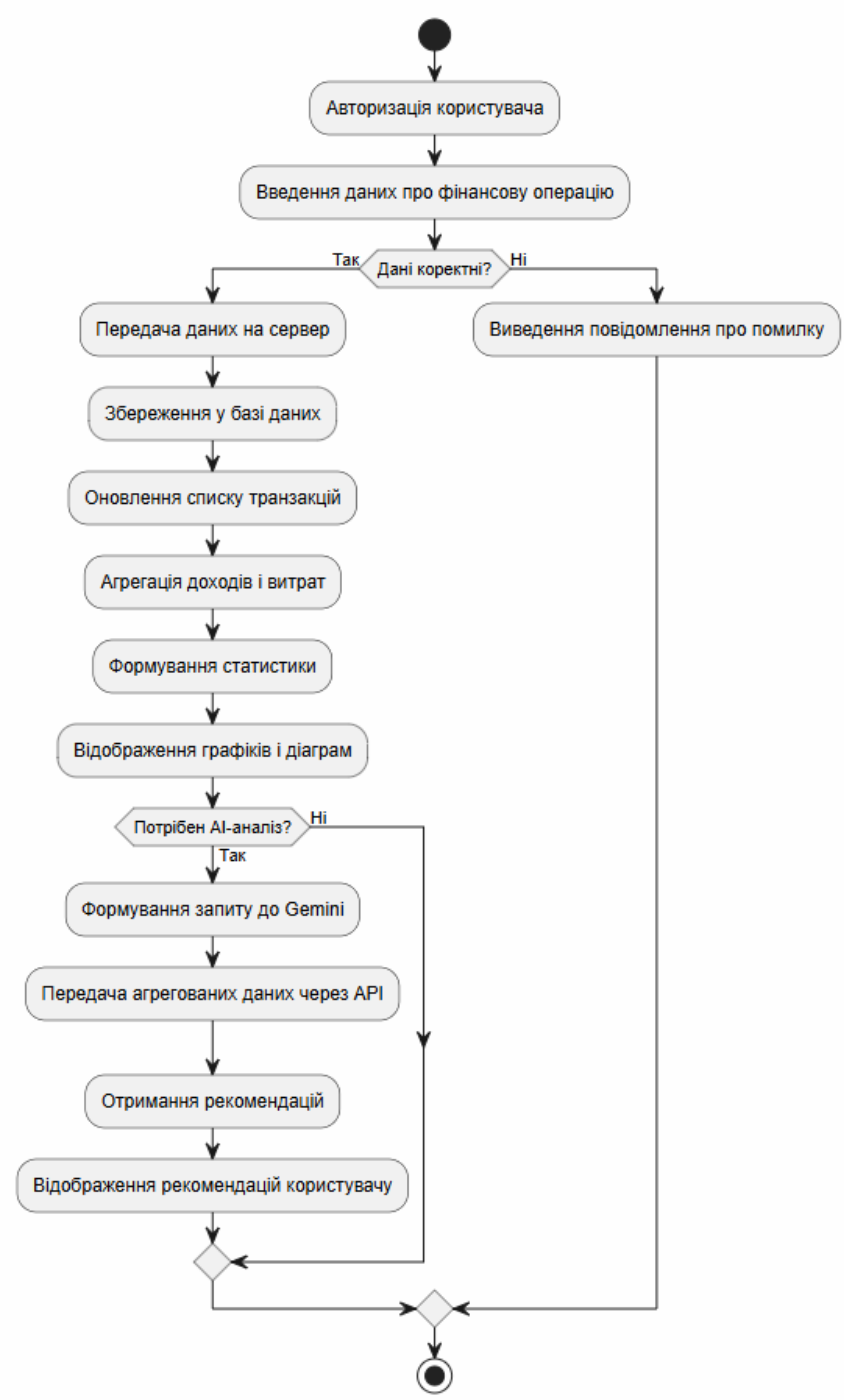


Рис. 2.1 UML блок-схема алгоритму обробки фінансових операцій у системі

Функціональні вимоги веб-додатку:

– введення даних про фінансові операції (доходи та витрати) – система повинна забезпечувати можливість введення даних про доходи та витрати із зазначенням суми, категорії, дати та додаткового опису;

- збереження інформації у базі даних – усі введені дані повинні зберігатися у базі даних із можливістю подальшої обробки та аналізу;
- відображення списку фінансових операцій – система повинна надавати користувачу доступ до списку фінансових операцій у вигляді структурованої таблиці;
- класифікація витрат за категоріями – передбачено розподіл доходів і витрат за категоріями для впорядкування даних та спрощення аналізу;
- побудова статистики витрат та доходів система повинна забезпечувати розрахунок основних фінансових показників за обраний період;
- візуалізація даних у вигляді графіків і діаграм – результати обробки повинні відображатися у вигляді графіків і діаграм для наочного представлення інформації;
- аналіз фінансових даних користувача – система повинна виконувати аналіз фінансової активності з метою виявлення тенденцій та закономірностей;
- інтеграція з великою мовною моделлю Gemini – передбачено застосування LLM для інтелектуального аналізу фінансових даних користувача та формування рекомендацій;
- представлення результатів аналізу – аналітичні висновки подаються у структурованому та зрозумілому для користувача форматі.

На UML-діаграмі варіантів використання відображено основні сценарії взаємодії користувача з веб-додатком для управління особистими фінансами (рис. 2.2). Головним актором системи є користувач, який виконує такі дії: додає фінансові операції, переглядає перелік транзакцій, аналізує витрати та доходи, працює з бюджетами, а також отримує рекомендації, сформовані засобами моделі Gemini. Окремим зовнішнім компонентом системи є сервіс Gemini API, що забезпечує інтелектуальний аналіз фінансових даних і генерацію персоналізованих рекомендацій.

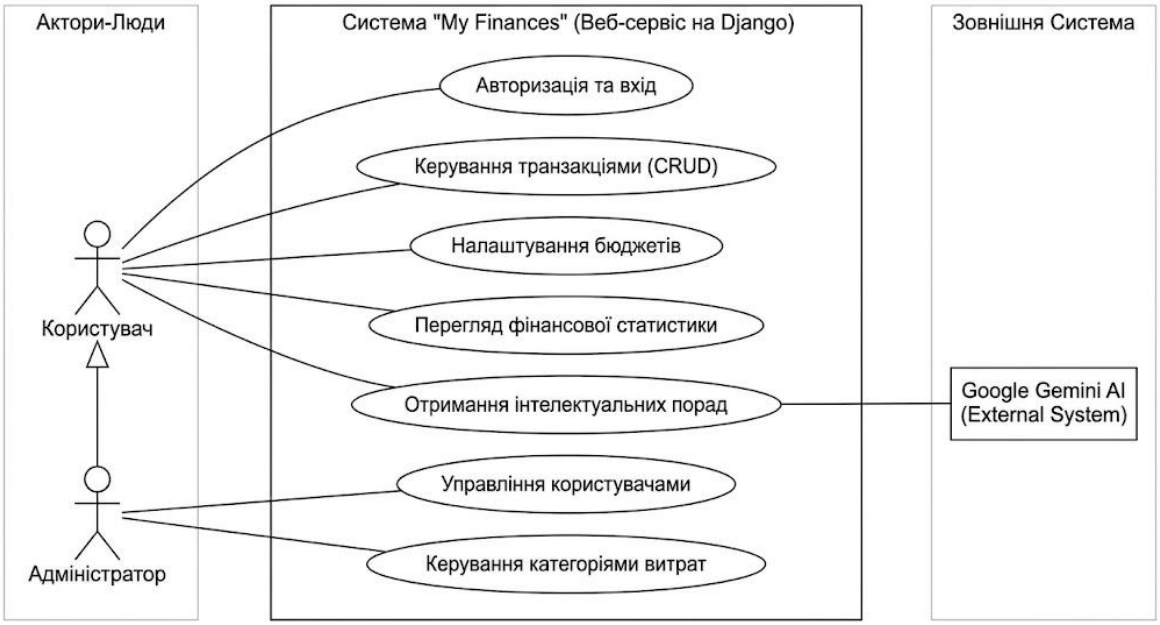


Рис. 2.2 UML-діаграма варіантів використання веб-додатку

До основних нефункціональних вимог належать:

- зручність та інтуїтивність інтерфейсу – система має забезпечувати інтуїтивно зрозумілий і простий у використанні користувацький інтерфейс;
- швидкодія при обробці запитів – веб-додаток має забезпечувати оперативну обробку запитів користувача та своєчасне відображення результатів;
- надійність збереження даних – система має гарантувати цілісність фінансових даних користувача та їх збереження без втрат;
- масштабованість – архітектура додатку має допускати розширення функціональності та збільшення кількості користувачів;
- доступність з різних пристроїв – веб-додаток має бути доступним через браузер на пристроях різних типів: персональних комп'ютерах, планшетах та смартфонах;
- захист персональних даних – система має забезпечувати захист фінансових даних користувача від несанкціонованого доступу.

Для формалізації внутрішньої структури веб-додатку та відображення взаємозв'язків між його компонентами застосовано UML-діаграму класів. Вона дозволяє визначити ключові сутності системи, їх атрибути й методи, а також

описати логіку взаємодії між окремими елементами програмного забезпечення. На рис. 2.3 представлено UML-діаграму класів веб-додатку для управління особистими фінансами.

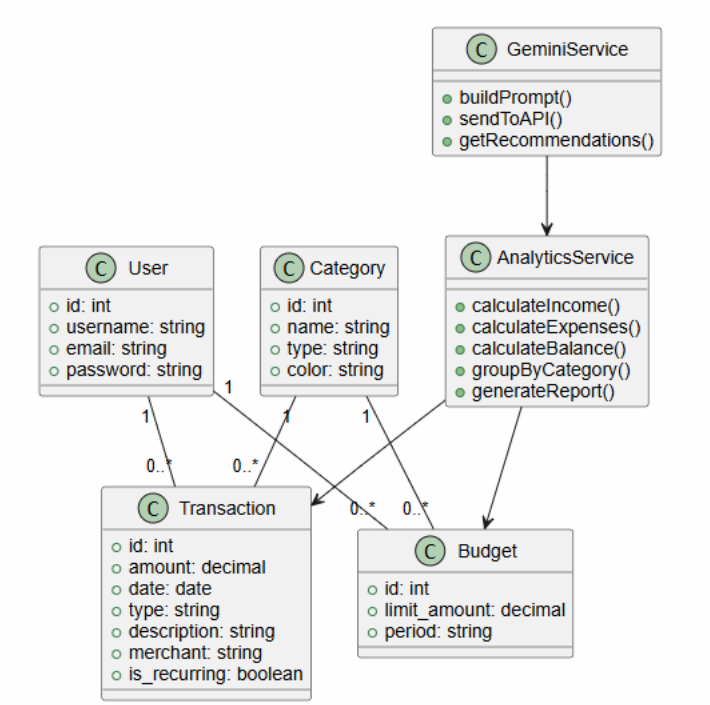


Рис. 2.3 UML-діаграма класів веб-додатку для управління особистими фінансами

Система забезпечує введення фінансових операцій через веб-інтерфейс, що надає користувачу можливість оперативно фіксувати відомості про доходи та витрати. Введені дані зберігаються у базі даних і використовуються для подальшого аналізу.

Відображення фінансових даних реалізовано у табличному вигляді та у форматі графічних елементів, що забезпечує наочне представлення структури витрат. Застосування діаграм спрощує процес аналізу та сприяє виявленню основних фінансових тенденцій [6].

Аналітичний модуль призначений для оцінювання фінансової активності користувача та формування рекомендацій щодо оптимізації витрат. Інтеграція

великої мовної моделі Gemini підвищує якість аналізу та забезпечує генерацію текстових рекомендацій у формі, доступній для сприйняття користувачем.

Сформовані вимоги визначають структуру та функціональні можливості розроблюваного веб-додатку й слугують основою для подальшого проєктування системи.

2.2 Проєктування архітектури системи та структури бази даних

Проєктування архітектури веб-додатку є одним із визначальних етапів розробки програмної системи, оскільки саме на цьому етапі встановлюється структура взаємодії між основними компонентами, визначаються принципи обробки даних та механізми їх збереження.

У межах даної роботи веб-додаток для аналізу та управління особистими фінансами реалізовано на основі клієнт-серверної архітектури, що передбачає розмежування функціональності між клієнтською частиною (frontend), серверною частиною (backend) та базою даних.

Архітектура веб-додатку охоплює такі основні компоненти:

– клієнтська частина (Frontend) – відповідає за організацію взаємодії з користувачем та відображення інформації; реалізована засобами HTML, CSS та JavaScript; через інтерфейс користувач вводить фінансові дані, переглядає статистику та отримує результати аналізу;

– серверна частина (Backend) - реалізована з використанням мови програмування Python та відповідного веб-фреймворку; виконує обробку запитів, реалізацію бізнес-логіки, обчислення та формування відповідей для клієнта;

– база даних – забезпечує збереження відомостей про фінансові операції користувача, зокрема даних про доходи, витрати, категорії та пов'язані параметри;

– інтеграційний модуль Gemini – функціонує як окремий аналітичний компонент, що здійснює інтелектуальний аналіз фінансових даних та генерацію рекомендацій; взаємодія з моделлю реалізована через API.

Взаємодія між компонентами системи відбувається за такою схемою: користувач через веб-інтерфейс формує запит, який передається до серверної частини. Сервер опрацьовує запит, звертається до бази даних для отримання або збереження інформації та, за необхідності, передає підготовлені дані до моделі Gemini. Сформовані результати повертаються до клієнта та відображаються в інтерфейсі веб-додатку (рис. 2.4).

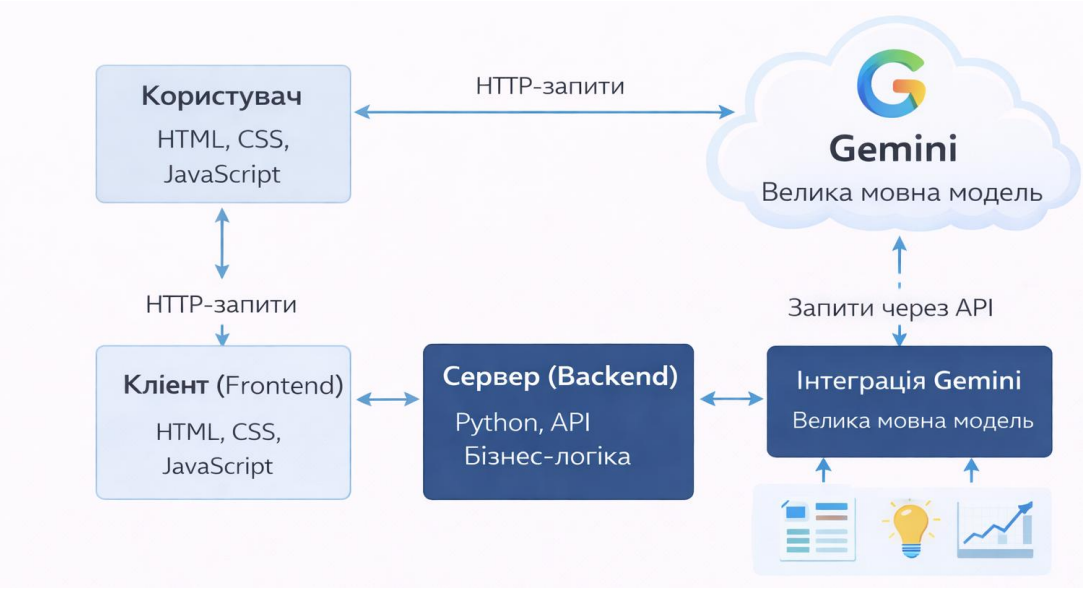


Рис. 2.4 Архітектура веб-додатку для аналізу та управління особистими фінансами

Для збереження та опрацювання фінансових даних у системі застосовується реляційна база даних. Основними сутностями бази даних є:

– користувач (User) – містить відомості про користувача системи, зокрема ідентифікатор, логін та інші облікові параметри;

– фінансова операція (Transaction) – центральна сутність, що акумулює інформацію про фінансові операції: суму, тип (дохід або витрата), дату та категорію;

– категорія (Category) – центральна сутність, що акумулює інформацію про фінансові операції: суму, тип (дохід або витрата), дату та категорію;.

Зв'язки між сутностями реалізовано з урахуванням того, що кожна фінансова операція асоційована з конкретним користувачем та категорією. Така організація даних забезпечує їх ефективну структуризацію та сприяє подальшому аналізу [12].

Процес опрацювання даних у веб-додатку охоплює такі основні етапи:

1. Користувач вводить дані про фінансову операцію через веб-інтерфейс.
2. Дані передаються до серверної частини за допомогою HTTP-запиту.
3. Сервер опрацьовує отримані дані та зберігає їх у базі даних.
4. За запитом користувача сервер формує аналітичну інформацію.
5. За необхідності дані передаються до моделі Gemini для інтелектуального аналізу.
6. Результати аналізу повертаються користувачу у вигляді текстових рекомендацій або графічних представлень.

Інтеграцію з Gemini реалізовано через API, що забезпечує передачу фінансових даних для аналізу та отримання відповіді у текстовому форматі. Такий підхід дозволяє розширювати функціонал системи без ускладнення її архітектури, а також виносити складні аналітичні обчислення за межі основного застосунку, що позитивно позначається на загальній продуктивності системи.

Розроблена архітектура забезпечує ефективну взаємодію між компонентами системи, уможливлює опрацювання та аналіз фінансових даних, а також формує підґрунтя для інтеграції сучасних інтелектуальних технологій. Запропонована структура є гнучкою, масштабованою та придатною для подальшого нарощування функціональних можливостей.

2.3 Розробка алгоритмів аналізу фінансових операцій

Реалізація відповідних алгоритмів забезпечує автоматизацію аналізу витрат і доходів та підвищує обґрунтованість фінансових рішень користувача.

Для формалізації логіки функціонування модуля обробки фінансових операцій застосовано UML-діаграму діяльності. Вона відображає послідовність основних дій, що виконуються в процесі введення, перевірки, передавання та

збереження фінансових даних у системі, і дозволяє унаочнити взаємозв'язок між окремими етапами алгоритму та його роль у забезпеченні цілісності даних і підготовці інформації до подальшого аналізу.

Як показано на рис. 2.5, процес ініціюється введенням користувачем даних про фінансову операцію – суми, типу, категорії та дати. Система виконує перевірку коректності введених значень: у разі успішної валідації дані передаються на сервер через HTTP-запит, де відбувається їх опрацювання та збереження у базі даних. Завершальним етапом є оновлення переліку транзакцій і відображення актуальних даних в інтерфейсі користувача. У разі виявлення помилок система повідомляє користувача про необхідність їх виправлення. Описана послідовність дій забезпечує коректність опрацювання фінансових операцій і формує основу для подальшого аналітичного опрацювання інформації.

Рис. 2.5 UML-діаграма діяльності алгоритму обробки фінансових операцій у веб-додатку

Для аналізу фінансового стану користувача використовується алгоритм агрегування та статистичної обробки даних.

Основні етапи алгоритму:

1. Отримання всіх фінансових операцій користувача з бази даних.
2. Розподіл операцій за типами (доходи та витрати).
3. Групування витрат за категоріями.
4. Обчислення сумарних значень для кожної категорії.
5. Побудова статистичних показників:

- загальна сума витрат;
- загальна сума доходів;
- баланс (різниця між доходами та витратами);
- частка кожної категорії у загальних витратах.

Отримані результати використовуються для побудови графіків та діаграм, що дозволяє користувачу наочно оцінити свою фінансову активність.

Однією з особливостей розроблюваного додатку є використання великої мовної моделі Gemini для формування інтелектуальних рекомендацій.

Алгоритм формування рекомендацій включає:

1. Збір агрегованих фінансових даних користувача.
2. Формування текстового запиту (prompt) до моделі Gemini.
3. Передача запиту через API.
4. Обробка запиту моделлю.
5. Отримання текстової відповіді у вигляді рекомендацій.
6. Відображення рекомендацій у веб-додатку.

Приклад логіки рекомендацій:

- якщо витрати перевищують доходи → рекомендація скоротити витрати;
- якщо певна категорія займає значну частку → рекомендація оптимізації;
- якщо є стабільний залишок → рекомендація заощаджень або інвестування.

Інтеграція з мовною моделлю реалізується через API та включає наступні кроки:

1. Формування запиту у вигляді структурованого тексту.
2. Відправлення HTTP-запиту до API Gemini.

3. Очікування відповіді від моделі.
4. Обробка отриманого результату.
5. Передача результату у frontend для відображення.

Дана стратегія дозволяє гнучко інтегрувати інтелектуальні можливості без ускладнення основної логіки додатку.

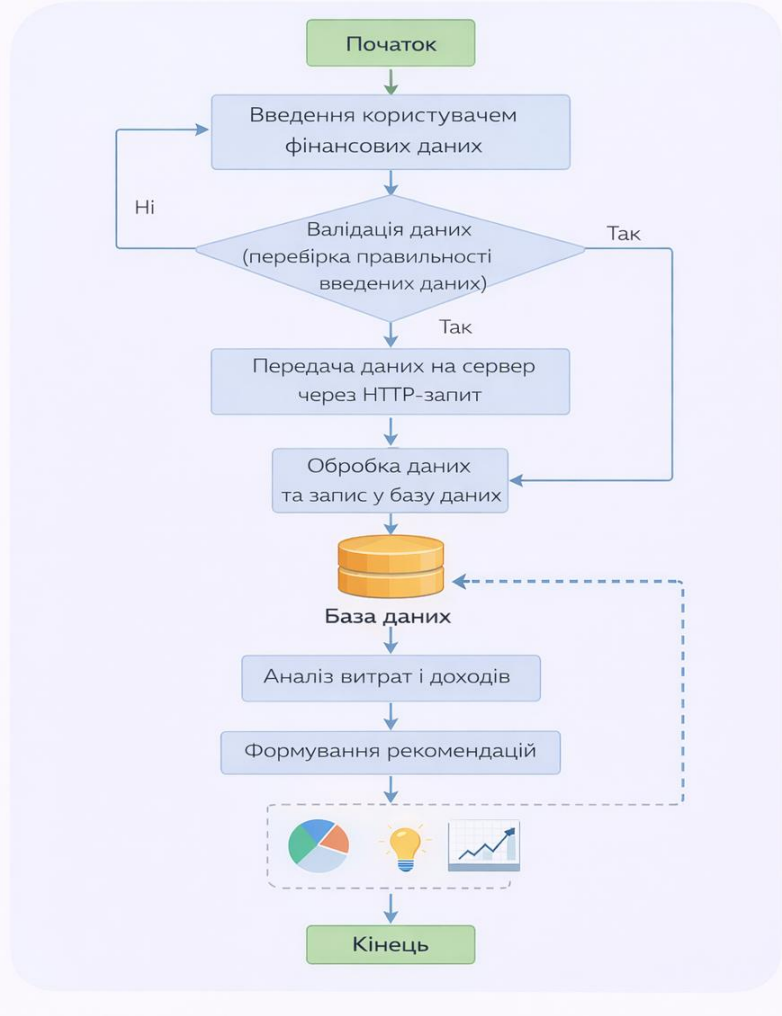


Рис. 2.6 Блок-схема алгоритму обробки та аналізу фінансових даних

Розроблені алгоритми мають ряд переваг:

– автоматизація фінансового аналізу;
– швидка обробка даних;
– наочне представлення результатів;
– можливість масштабування;
– інтеграція інтелектуальних технологій.

Запропоновані алгоритми забезпечують ефективну обробку інформації та підвищують функціональність веб-додатку, що дозволяє користувачу отримувати корисні аналітичні висновки та рекомендації для оптимізації власних фінансів.

2.4 Інтеграція великої мовної моделі Gemini для аналізу фінансових даних

Розроблений веб-додаток реалізовано з використанням сучасних веб-технологій, що забезпечують зручність використання, швидкодію та можливість масштабування системи. Архітектура додатку побудована за клієнт-серверною моделлю, що дозволяє розділити обробку даних та інтерфейс користувача.

Frontend частина реалізована з використанням HTML, CSS та JavaScript, що забезпечує створення інтерактивного та адаптивного інтерфейсу. Backend частина розроблена мовою Python із використанням відповідного веб-фреймворку, що забезпечує обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних [15].

Користувацький інтерфейс веб-додатку розроблений з урахуванням принципів зручності та простоти використання. Основною метою інтерфейсу є забезпечення швидкого доступу до основних функцій системи та наочного відображення фінансових даних.

Інтерфейс включає наступні основні елементи:

- форма додавання фінансових операцій;
- список транзакцій;
- блок аналітики;
- графічне відображення статистики;
- модуль рекомендацій.

Форма введення даних дозволяє користувачу додавати інформацію про доходи та витрати, вказуючи суму, категорію та дату операції. Це забезпечує швидке внесення даних у систему.

Список транзакцій відображає всі фінансові операції користувача у зручному вигляді, що дозволяє контролювати витрати та доходи.

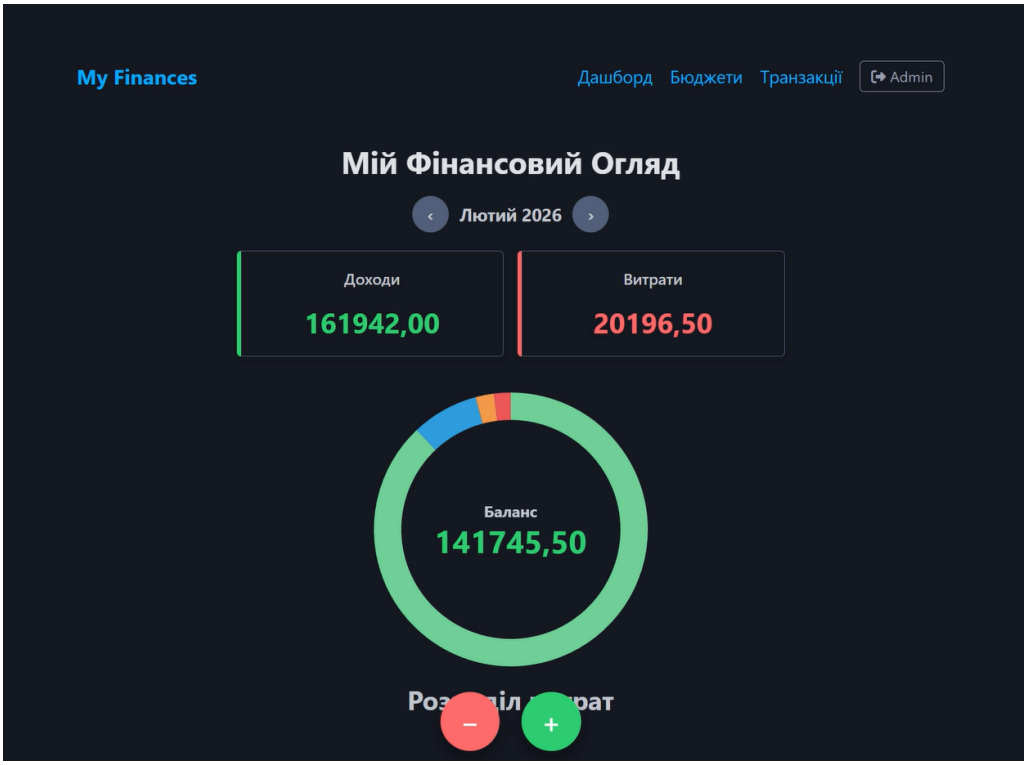


Рис. 2.7 Інтерфейс додавання фінансових операцій

Аналітичний модуль забезпечує обробку фінансових даних та їх представлення у вигляді графіків і діаграм. завдяки чому користувач отримує наочне уявлення про структуру витрат та доходів.

У системі реалізовано:

– побудову кругових діаграм витрат;
– відображення динаміки доходів і витрат;
– аналіз категорій витрат.

Графічне представлення даних значно підвищує зручність використання системи та дозволяє швидко виявляти фінансові тенденції.

Рис. 2.8 Візуалізація фінансових даних користувача

Однією з особливостей розробленого веб-додатку є інтеграція з мовною моделлю Gemini. Використання цього інструменту уможливлює реалізацію функцій інтелектуального аналізу фінансових даних [16].

Інтеграція реалізована через API, що дозволяє передавати фінансові дані на обробку та отримувати текстові рекомендації.

Основні можливості інтеграції:

– аналіз витрат користувача;
– формування персоналізованих рекомендацій;
– генерація фінансових порад;
– прогнозування фінансового стану.

Отримані результати відображаються у вигляді текстових повідомлень у веб-додатку, що робить їх зрозумілими для користувача.

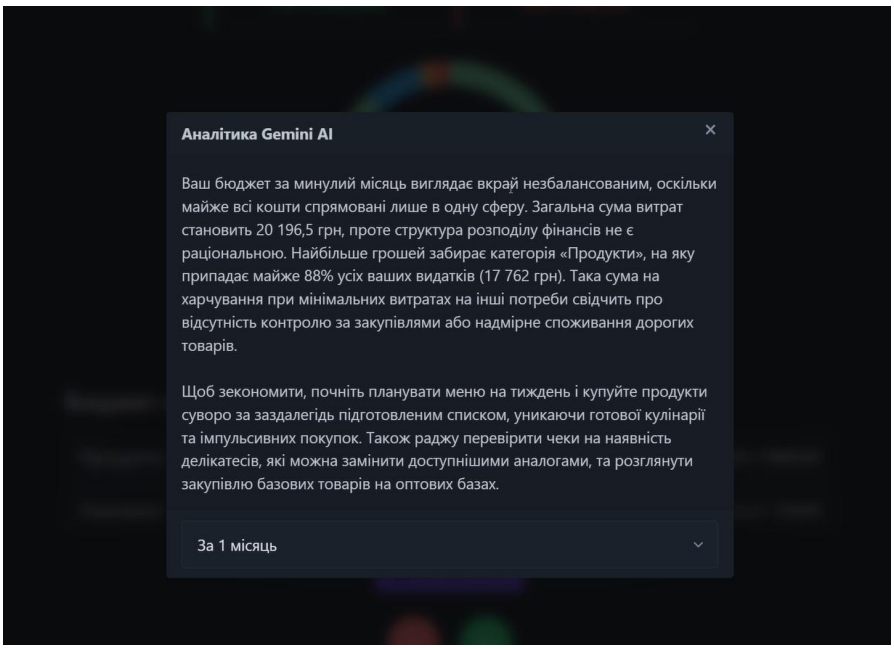


Рис. 2.9 – Відображення рекомендацій, згенерованих Gemini

У процесі розробки було використано наступні технології:

– мова програмування Python;
– веб-фреймворк (Django);
– HTML, CSS, JavaScript для frontend;
– база даних (SQLite);
– REST API для взаємодії з Gemini.

Такий стек технологій забезпечує гнучкість системи, її масштабованість та можливість подальшого розвитку.

Реалізована система забезпечує зручний облік фінансових операцій, ефективний аналіз даних та інтеграцію з сучасними інтелектуальними технологіями. Це сприяє підвищенню ефективності управління особистими фінансами та забезпечує користувача ґрунтовними аналітичними рекомендаціями.

3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ВЕБ-ДОДАТКУ

3.1 Розробка програмної частини веб-додатку

У даному підрозділі розглянуто структуру програмного забезпечення веб-додатку для аналізу та управління особистими фінансами, а також описано призначення основних модулів і компонентів системи.

Розроблений веб-додаток реалізовано з використанням мови програмування Python та веб-фреймворку Django, що дозволяє ефективно організувати структуру проєкту відповідно до принципів модульності та масштабованості [17].

Програмний застосунок має багаторівневу структуру, що включає основні директорії та файли, які відповідають за різні аспекти функціонування системи.

До основних компонентів структури належать:

– директорія core – основний модуль додатку;

– директорія templates – HTML-шаблони інтерфейсу;

– директорія static – статичні ресурси (CSS, JavaScript, зображення);

– файл manage.py – службовий файл для керування додатком;

– база даних db.sqlite3 – збереження даних користувача.

Директорія core є центральним компонентом додатку, який містить основну бізнес-логіку системи.

До її складу входять наступні файли:

– models.py – опис структури бази даних, у цьому файлі визначено моделі, що відповідають сутностям системи, зокрема фінансовим операціям та категоріям;

– views.py – реалізація логіки обробки запитів, відповідає за отримання даних від користувача, їх обробку та передачу у шаблони;

– urls.py – маршрутизація запитів – визначає, які функції обробляють конкретні URL-адреси;

– forms.py – форми для введення даних, використовуються для створення інтерфейсу введення фінансових операцій;

– admin.py – адміністративна панель, дозволяє керувати даними через стандартний інтерфейс Django;

– apps.py – конфігурація додатку;

– tests.py – тестування функціоналу системи.

Директорія templates містить HTML-шаблони, які використовуються для відображення інтерфейсу веб-додатку.

Основні шаблони:

– base.html – базовий шаблон, що містить загальну структуру сторінок;

– dashboard.html – головна сторінка з аналітикою;

– transactions_list.html – список фінансових операцій;

– transaction_form.html – форма додавання транзакцій;

– budget_form.html / budget_list.html – робота з бюджетами;

– login.html – сторінка авторизації користувача [18].

Шаблони побудовані за принципом наслідування, що дозволяє уникнути дублювання коду та спростити підтримку інтерфейсу.

Директорія static містить файли стилів та скриптів, які забезпечують зовнішній вигляд та інтерактивність веб-додатку.

До них належать:

– CSS-файли – оформлення інтерфейсу;

– JavaScript – обробка подій та взаємодія з користувачем;

– зображення та іконки.

У якості системи управління базами даних використовується SQLite, що забезпечує простоту розгортання та використання на етапі розробки.

База даних містить таблиці для:

– користувачів;

– фінансових операцій;

– категорій;

– бюджетів.

Що забезпечує структуроване зберігання та ефективну обробку інформації.

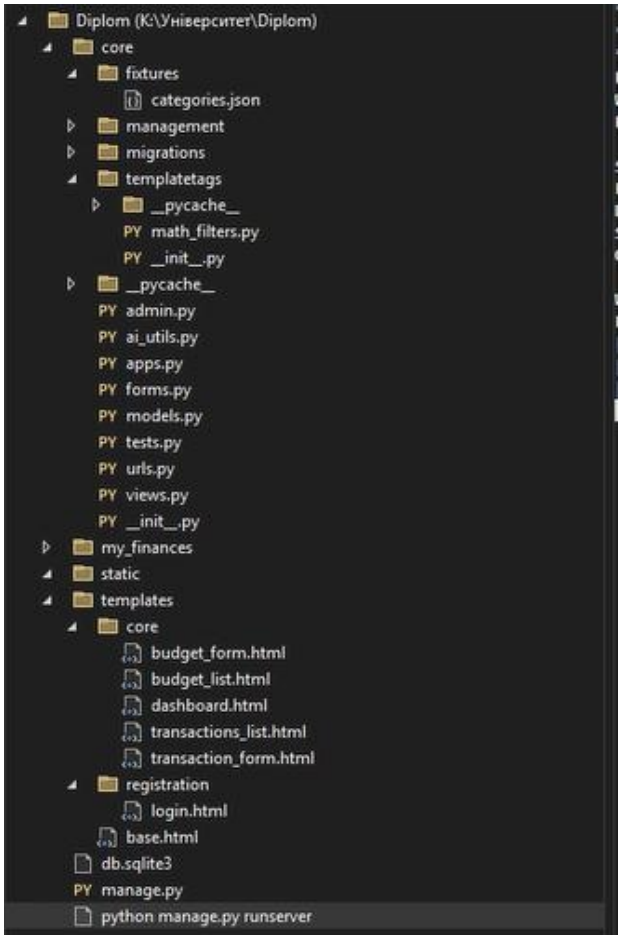


Рис. 3.1 Структура програмного забезпечення веб-додатку

Структура програмного забезпечення веб-додатку організована відповідно до сучасних підходів розробки веб-систем та забезпечує ефективну реалізацію функціональних можливостей. Використання фреймворку Django дозволило створити гнучку, масштабовану та зручну у підтримці систему.

3.2 Реалізація користувацького інтерфейсу та візуалізації фінансових даних

У даному підрозділі розглянуто реалізацію основних функціональних можливостей веб-додатку для аналізу та управління особистими фінансами, а також описано логіку роботи ключових компонентів системи.

Розробка функціоналу здійснювалась із використанням фреймворку Django, що дозволяє ефективно організувати обробку запитів, роботу з базою даних та формування інтерфейсу користувача [19].

Однією з основних функцій веб-додатку є можливість додавання, перегляду та керування фінансовими операціями.

Для реалізації цієї функціональності використовується модель, яка описує структуру фінансової транзакції (див. Додаток Б).

Дана модель забезпечує збереження ключових параметрів фінансової операції та дозволяє організувати їх подальший аналіз.

Обробка запитів користувача здійснюється за допомогою функцій у файлі views.py. Саме тут реалізується логіка взаємодії між frontend та базою даних (див. Додаток В).

Ця функція отримує всі транзакції з бази даних та передає їх у шаблон для відображення.

Для введення фінансових операцій використовуються Django-форми, які дозволяють забезпечити валідацію даних (див. Додаток Е).

Форма автоматично генерує HTML-інтерфейс та перевіряє правильність введених даних (рис. 3.2).

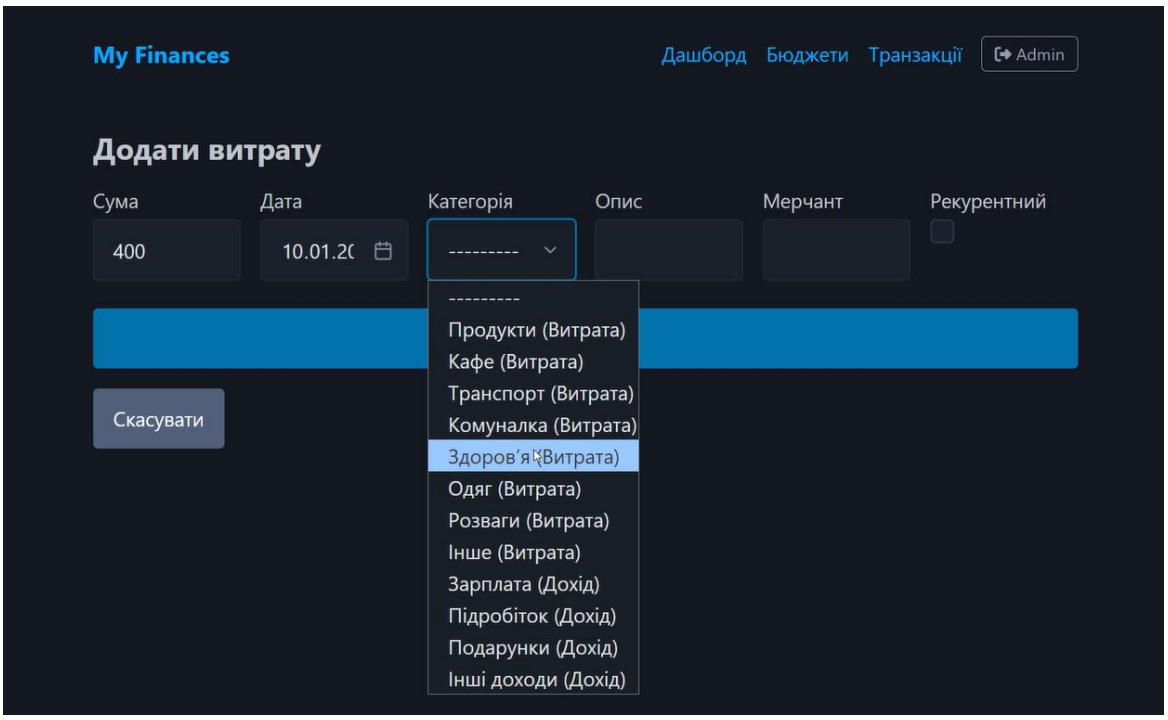


Рис. 3.2 Форма додавання фінансової операції

Для формування аналітичної звітності у системі реалізовано функцію dashboard у модулі views.py (див. Додаток В). Алгоритм роботи функції базується на використанні засобів агрегації Django ORM, зокрема методу Sum для підрахунку загальних витрат та annotate для групування даних за категоріями. Результатом обробки є словник із сумарними показниками, який передається до frontend-частини для побудови інтерактивних графіків за допомогою бібліотеки Chart.js

Для реалізації інтелектуального аналізу фінансових показників розроблено функцію get_recommendation, програмна логіка якої зосереджена у модулі ai_utils.py (див. Додаток Г). Функція приймає агреговані дані про доходи та витрати користувача, на основі яких формується структурований текстовий запит (prompt) до великої мовної моделі Gemini.

Взаємодія із зовнішнім сервісом відбувається через офіційну бібліотеку google-generativeai, що забезпечує стабільне передавання даних та отримання персоналізованих фінансових порад у текстовому форматі для подальшої візуалізації в інтерфейсі застосунку

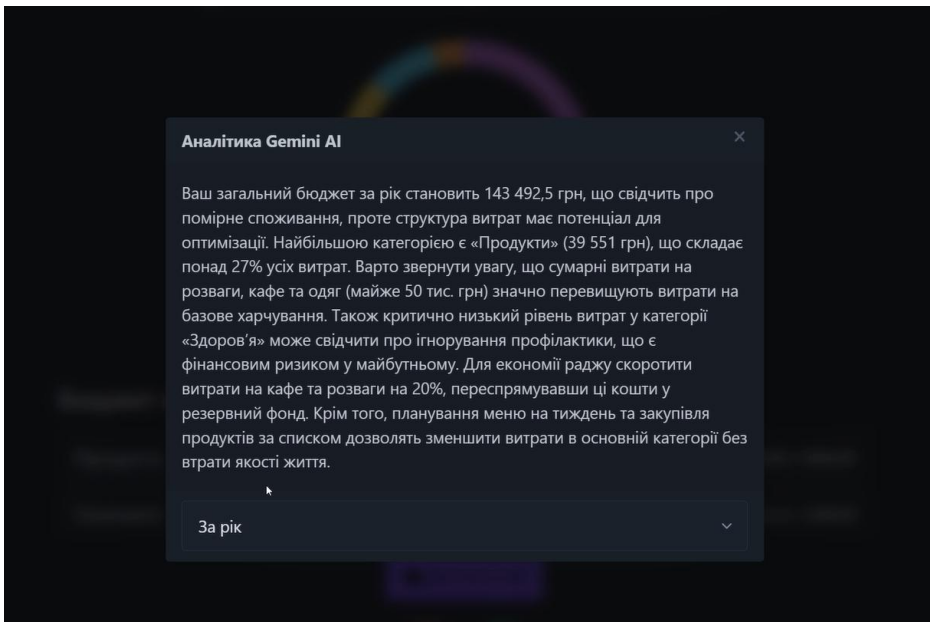


Рис. 3.3 Відображення рекомендацій Gemini

Система маршрутизації запитів веб-застосунку базується на списку urlpatterns, визначеному у модулі urls.py (див. Додаток Г). Даний компонент забезпечує відображення URL-адрес на відповідні функції-контролери з модуля views.py.

Конфігурація охоплює маршрути для формування аналітичного дашборда (dashboard), отримання структурованого переліку фінансових операцій (transaction_list) та доступу до інтерфейсу додавання нових транзакцій (add_transaction). Така організація дозволяє забезпечити чітку логічну структуру веб-застосунку та гнучкість при подальшому розширенні функціональних можливостей системи.

Реалізований веб-додаток має наступні переваги:

– чітка структура коду;
– використання MVC-підходу (Django);
– зручний інтерфейс;
– можливість масштабування;
– інтеграція сучасних AI-технологій.

Запропоновані рішення забезпечують ефективну роботу системи та створюють основу для подальшого розвитку функціоналу.

3.3 Тестування функціональних можливостей системи

У процесі розробки веб-додатку для аналізу та управління особистими фінансами важливим етапом є перевірка його працездатності та відповідності поставленим вимогам. Тестування функціональних можливостей системи дозволяє оцінити коректність реалізації основних функцій, виявити можливі помилки та забезпечити стабільну роботу програмного застосунку.

На рис. 3.4 представлено діаграму прецедентів системи управління персональними фінансами, яка відображає взаємодію користувачів та зовнішніх компонентів із функціональними можливостями веб-застосунку.

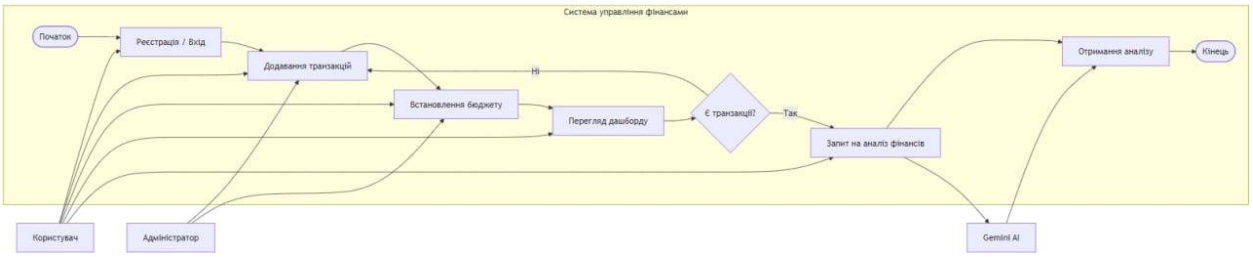


Рис. 3.4 Діаграма прецедентів системи управління фінансами

Основним актором системи є користувач, який здійснює ключові дії, зокрема реєстрацію та вхід до системи, додавання фінансових транзакцій, встановлення бюджету та перегляд аналітичної інформації на дашборді. У процесі роботи система перевіряє наявність транзакцій, що є умовою для виконання подальшого аналізу фінансових даних. У разі наявності даних формується запит до модуля інтелектуального аналізу, реалізованого із використанням технології Gemini AI, який виступає як зовнішній сервіс. Результатом обробки є отримання аналітичних висновків та рекомендацій, що дозволяють користувачу оптимізувати власні витрати. Також у діаграмі враховано роль адміністратора, який має доступ до управління даними системи та забезпечує її коректне функціонування. Представлена діаграма наочно демонструє основні сценарії використання системи та взаємозв’язки між її компонентами, що дозволяє краще зрозуміти логіку роботи розробленого програмного продукту.

У межах даного підрозділу проведено комплексне тестування ключових функціональних компонентів веб-додатку, зокрема модулів обробки фінансових операцій, аналітики даних та інтеграції з великою мовною моделлю Gemini. Особливу увагу приділено перевірці правильності обробки вхідних даних, швидкодії системи та якості сформованих рекомендацій.

Для формалізації процесу тестування було розроблено набір тест-кейсів, що охоплюють основні функціональні можливості системи. Кожен тест-кейс містить вхідні дані, очікуваний результат та фактичний результат виконання.

Тестування проводилося за такими категоріями:

- тестування введення даних;
- тестування обробки транзакцій;
- тестування аналітичного модуля;
- тестування інтеграції з API Gemini.

Таблиця 3.1

Результати функціонального тестування компонентів веб-застосунку

№	Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Додавання транзакції	Сума=100, Категорія="Їжа"	Запис збережено в БД	Відповідає очікуваному	Успішно
2	Валідація некоректних даних	Сума=-100	Виведення повідомлення про помилку	Відповідає очікуваному	Успішно
3	Розрахунок фінансового балансу	Доходи=1000, Витрати=500	Баланс = 500	Відповідає очікуваному	Успішно
4	Генерація рекомендацій Gemini	Агреговані фінансові дані	Отримання аналітичної поради	Відповідає очікуваному	Успішно

На рис. 3.5 представлено головну сторінку веб-додатку, яка реалізує функцію фінансового огляду користувача. Даний інтерфейс є центральним елементом системи та забезпечує відображення ключових фінансових показників у зручному та наочному вигляді. На дашборді відображаються основні агреговані дані за обраний період, зокрема загальна сума доходів, витрат та розрахований баланс. Це надає користувачеві можливість оперативно оцінити свій фінансовий стан без необхідності детального аналізу кожної транзакції.

Використання графічної візуалізації у вигляді кругової діаграми забезпечує наочне представлення співвідношення доходів і витрат. Даний спосіб візуалізації значно спрощує сприйняття інформації та дозволяє швидко виявляти фінансові тенденції.

Також інтерфейс передбачає можливість зміни періоду аналізу, що дозволяє користувачу переглядати фінансові дані за різні часові проміжки. Наявність інтерактивних елементів керування забезпечує зручність роботи з додатком.

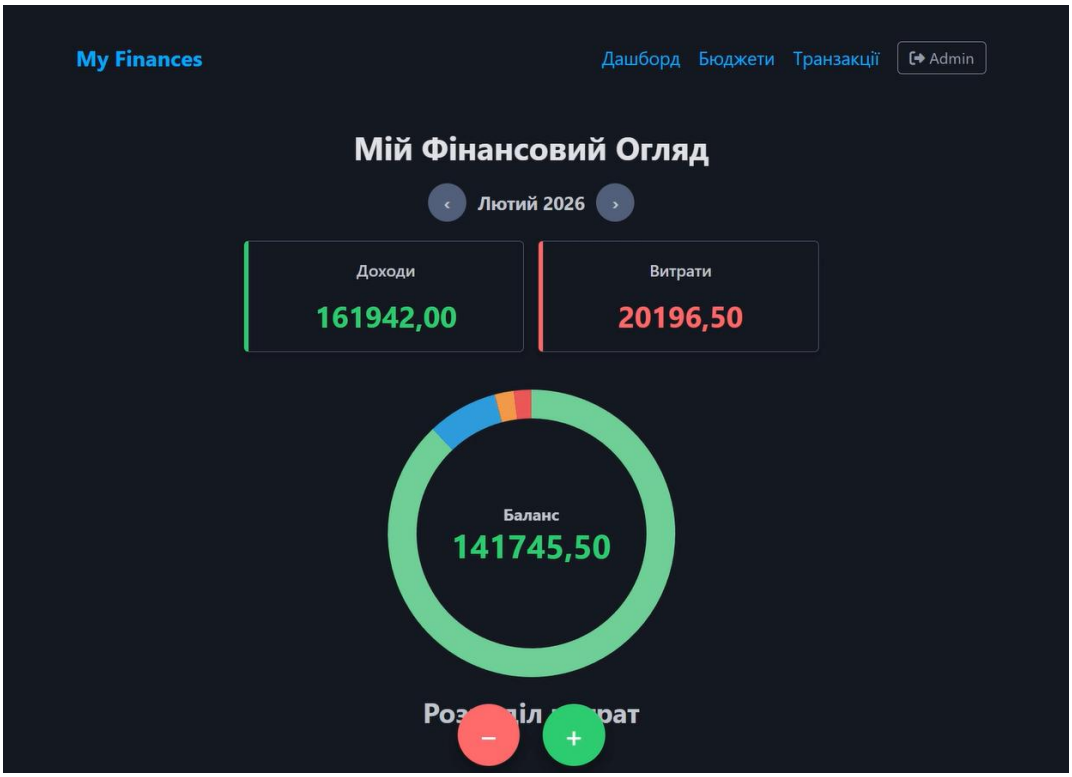


Рис. 3.5 Головна сторінка веб-додатку з фінансовою аналітикою

На рис. 3.6 представлено результат роботи модуля інтеграції з великою мовною моделлю Gemini, який забезпечує інтелектуальний аналіз фінансових даних користувача та формування рекомендацій.

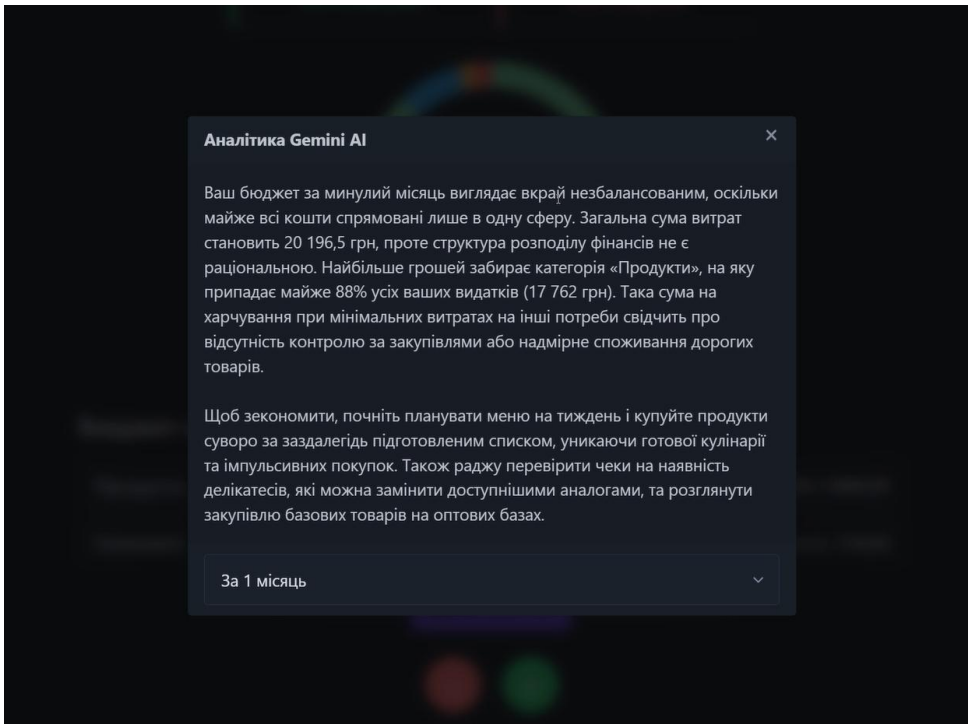


Рис. 3.6 Відображення рекомендацій, згенерованих моделлю Gemini у веб-додатку

У даному вікні відображається текстовий аналіз фінансового стану за обраний період, зокрема оцінка структури витрат, визначення найбільш витратних категорій та виявлення потенційних проблем у розподілі бюджету. Модель Gemini здійснює обробку агрегованих даних та генерує зрозумілі для користувача рекомендації у текстовому форматі.

Зокрема, у представленому прикладі система визначає дисбаланс у витратах, звертаючи увагу на надмірну частку витрат у певній категорії, а також пропонує практичні поради щодо оптимізації фінансової поведінки, такі як планування витрат, контроль покупок та використання альтернативних рішень.

Інтерфейс подання рекомендацій реалізований у вигляді модального вікна, що забезпечує зручність перегляду результатів без переходу на інші сторінки. Це

підвищує зручність користування системою та ефективність взаємодії з аналітичним модулем.

Отримані результати підтверджують коректність роботи інтеграції з великою мовною моделлю та демонструють ефективність використання штучного інтелекту для аналізу фінансових даних і підтримки прийняття рішень.

На рис. 3.7 представлено інтерфейс модуля управління бюджетами, який реалізує функцію контролю витрат користувача за окремими категоріями. Даний функціонал дозволяє встановлювати обмеження витрат та відслідковувати їх виконання у межах заданого періоду.

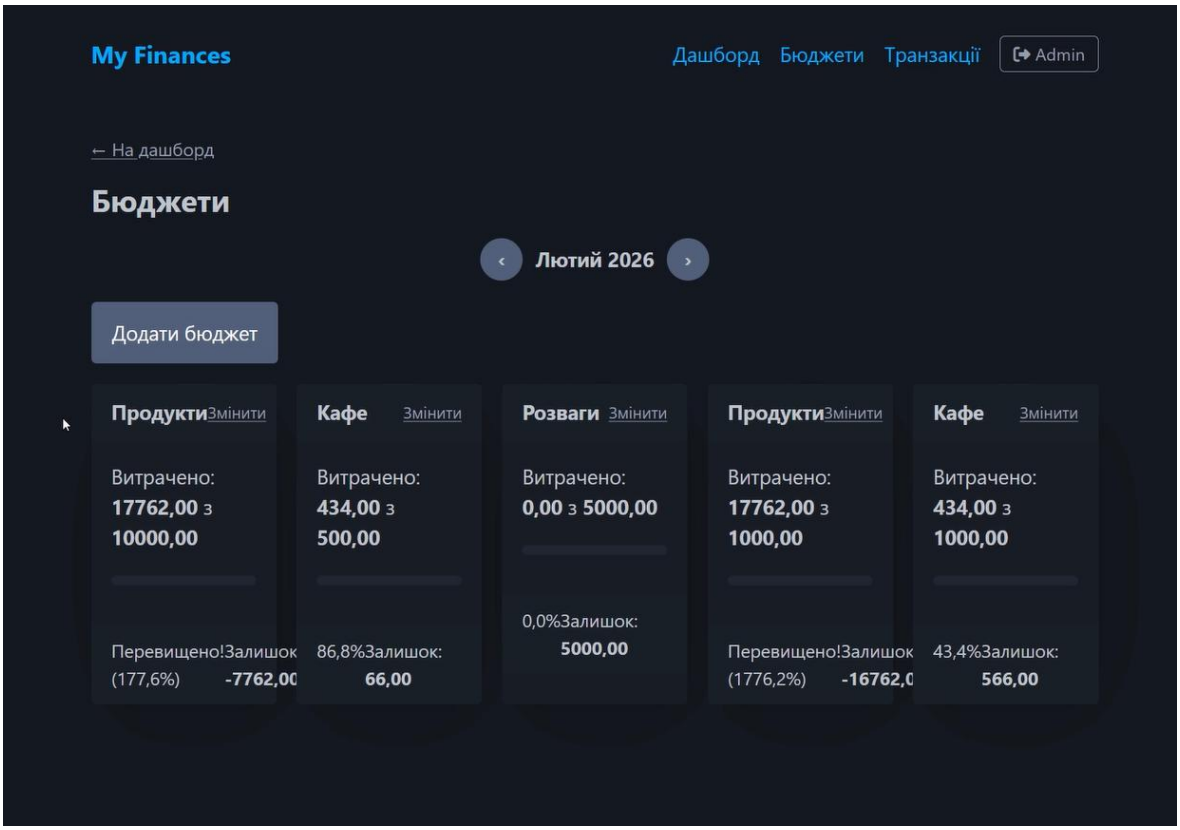


Рис. 3.7 Інтерфейс управління бюджетами у веб-додатку

Інтерфейс відображає перелік бюджетів, розподілених за категоріями – «Продукти», «Кафе», «Розваги» та іншими. Для кожної категорії представлено відомості про загальну суму бюджету, фактичні витрати, а також залишок коштів або величину перевищення встановленого ліміту.

Характерною особливістю реалізації є наочне представлення стану бюджету за допомогою текстових показників та індикаторів перевищення, що дозволяє користувачу оперативно оцінити фінансову ситуацію та ідентифікувати проблемні категорії витрат.

Користувач має змогу додавати нові бюджети, а також вносити зміни до наявних, що забезпечує гнучкість налаштування фінансового планування. Функція контролю перевищення бюджетного ліміту сприяє формуванню фінансової дисципліни користувача.

Результати перевірки підтверджують коректність реалізації функціоналу бюджетування та свідчать про придатність системи для контролю особистих фінансів.

На рис. 3.8 представлено інтерфейс форми додавання фінансової операції, що забезпечує введення даних про доходи та витрати користувача. Зазначений модуль є одним із ключових елементів системи, оскільки саме через нього здійснюється наповнення бази даних фінансовою інформацією. Форма надає користувачу можливість визначити основні параметри операції: суму, дату, категорію, опис та найменування мерчанта. Для вибору категорії реалізовано випадаючий список, що забезпечує стандартизацію введених даних і унеможливлює некоректне заповнення поля. Передбачено також можливість позначення операції як регулярної, що дозволяє враховувати повторювані фінансові надходження або витрати та підвищує точність подальшого аналізу. Інтерфейс форми розроблено відповідно до принципів зручності використання: наявність кнопок підтвердження та скасування забезпечує керованість процесу введення даних. Результати тестування підтверджують коректність функціонування форми – правильність опрацювання введених даних, їх збереження у базі даних та подальше використання в аналітичних модулях системи.

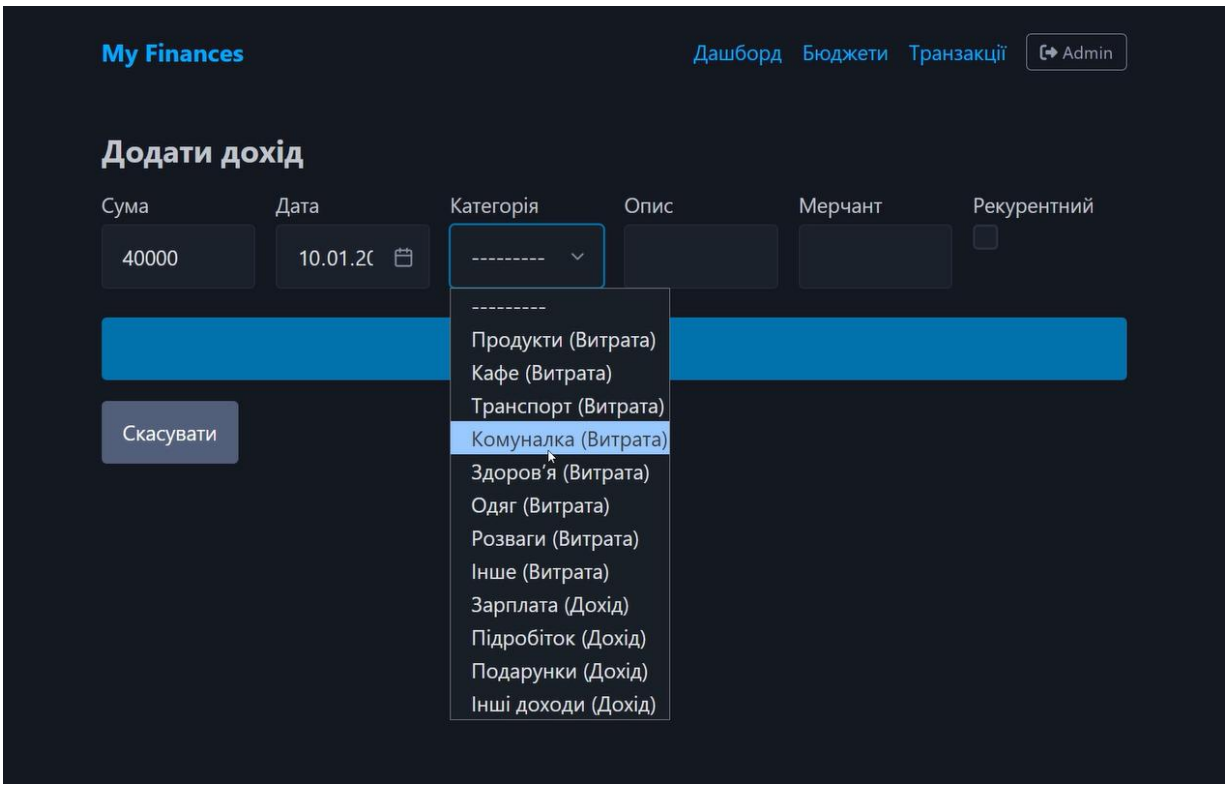


Рис. 3.8 Форма додавання фінансової операції у веб-додатку

На рис. 3.9 представлено результат функціонування аналітичного модуля із застосуванням великої мовної моделі Gemini для аналізу фінансових даних за тривалий період, зокрема за рік. У відповідному вікні відображається узагальнений аналіз бюджету користувача, що охоплює оцінювання структури витрат, визначення найбільш витратних категорій та виявлення потенційних ризиків у фінансовій поведінці. Система автоматично опрацьовує накопичені дані, встановлює ключові закономірності та формує персоналізовані рекомендації щодо оптимізації витрат – зокрема щодо скорочення видатків у певних категоріях і перерозподілу коштів.

Відмінною характеристикою зазначеного функціоналу є можливість вибору періоду аналізу: місяць, квартал, півроку або рік. Така варіативність забезпечує гнучкість системи та уможливлює проведення як короткострокового, так і довгострокового фінансового аналізу. Інтерфейс представлення результатів побудовано з дотриманням принципів зручності використання, що сприяє ефективній взаємодії користувача із системою.

Одержані результати підтверджують коректність функціонування аналітичного модуля та обґрунтовують доцільність застосування засобів штучного інтелекту для поглибленого аналізу фінансових даних і формування аргументованих рекомендацій.

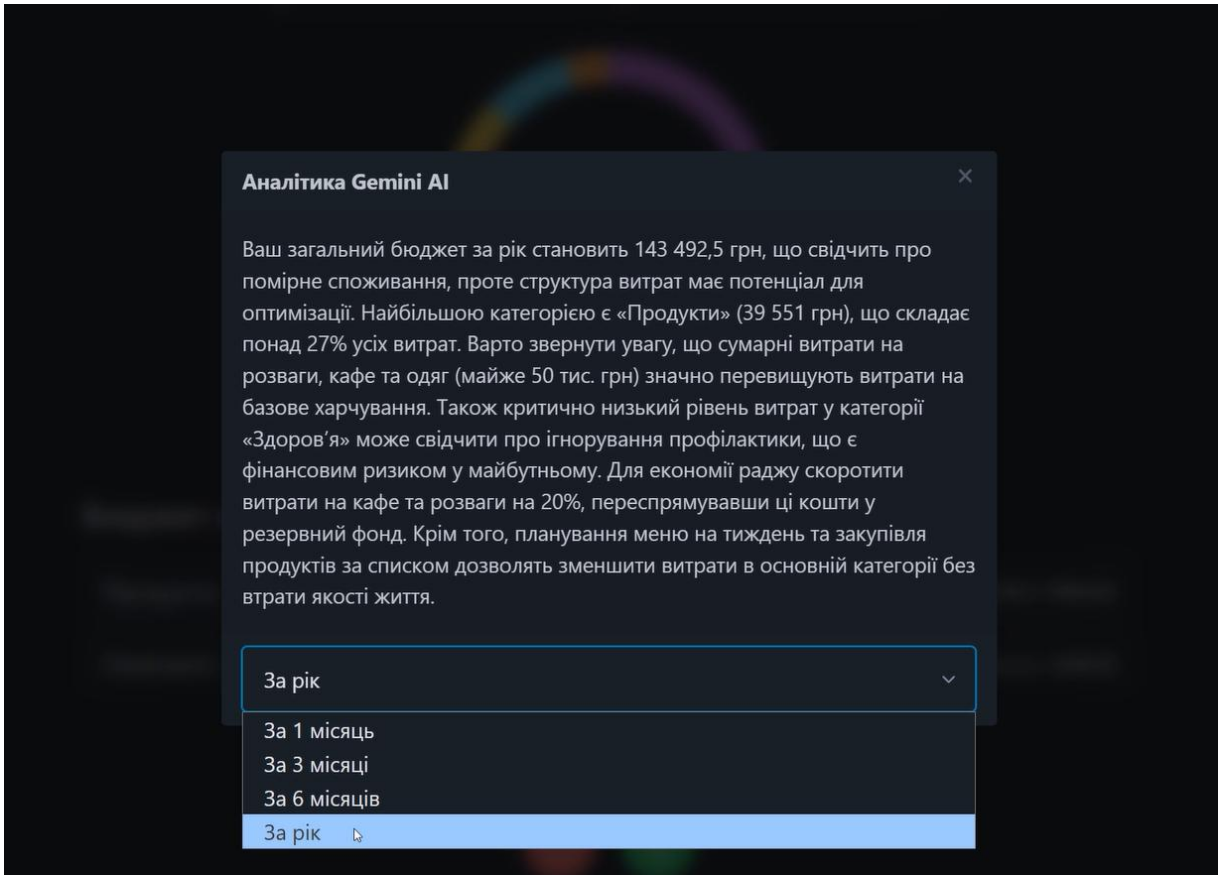


Рис. 3.9 Аналіз фінансових даних за рік із використанням моделі Gemini

На рис. 3.10 представлено інтерфейс сторінки перегляду транзакцій, що забезпечує відображення фінансових операцій користувача та керування ними. Модуль реалізує фільтрацію транзакцій за типом (доходи або витрати) і категорією, а також пошук за описом або найменуванням мерчанта, що спрощує навігацію у значних масивах даних. Результати відображаються у табличному форматі з основними атрибутами: датою, типом операції, категорією, сумою, описом та мерчантом. Для кожного запису передбачено можливість редагування або видалення, що забезпечує гнучке керування даними. Інтерфейс також містить

елементи швидкого додавання нових доходів або витрат, що скорочує кількість дій, необхідних для внесення інформації.

Реалізований механізм фільтрації та пошуку функціонує коректно і забезпечує оперативне отримання необхідних відомостей, що підтверджено за результатами тестування зазначеного модуля.

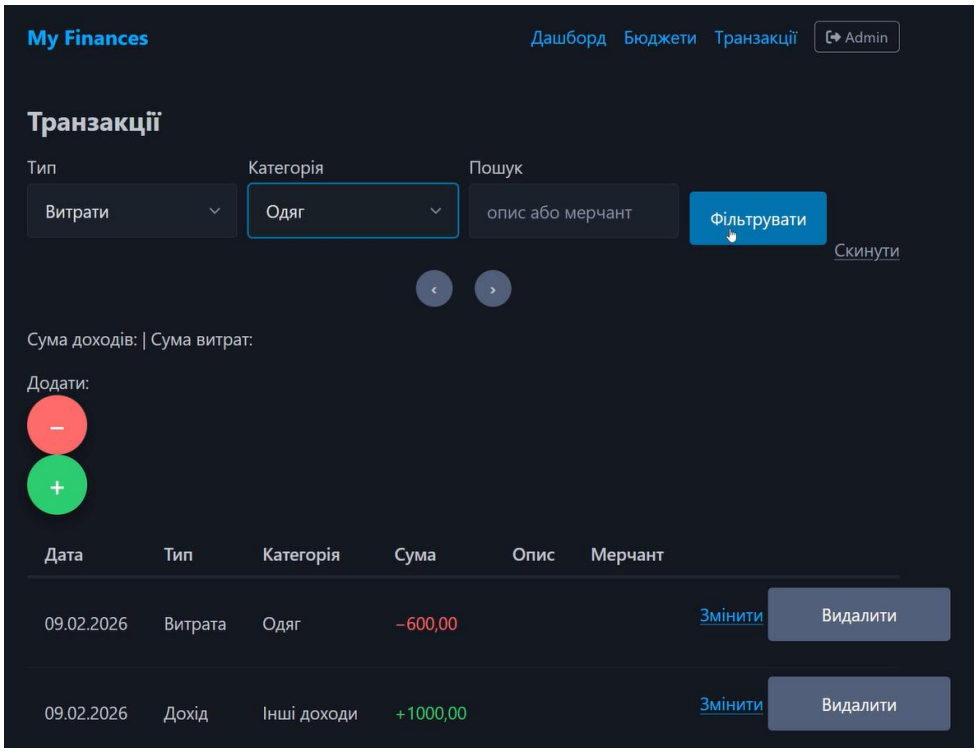


Рис. 3.10 Інтерфейс перегляду та фільтрації транзакцій

Для оцінки ефективності роботи системи було проаналізовано основні показники її функціонування. Зокрема, оцінювалися швидкодія обробки запитів, коректність виконання операцій та стабільність взаємодії із зовнішнім API. Середній час обробки запиту на додавання транзакції становить менше 200 мс, що відповідає вимогам до швидкодії веб-додатків.

Час отримання відповіді від API Gemini залежить від складності запиту та в середньому становить 1-2 секунди. Результати тестування показали, що система коректно обробляє вхідні дані, забезпечує стабільну роботу та відповідає поставленим функціональним вимогам

3.4 Адміністрування та управління даними системи

У інформаційних системах особливу роль відіграє можливість централізованого контролю за даними, користувачами та бізнес-логікою додатку, що дозволяє своєчасно оновлювати інформацію, виправляти помилки та забезпечувати цілісність бази даних. У межах даного підрозділу розглядається використання вбудованого інструментарію фреймворку Django – адміністративної панелі (Django Admin), яка забезпечує зручний інтерфейс для управління сутностями системи, зокрема транзакціями, категоріями, бюджетами та користувачами. Описано основні можливості адміністративного інтерфейсу, механізми автентифікації та авторизації, а також принципи організації доступу до функцій управління. Реалізація даного модуля дозволяє підвищити ефективність супроводження системи, забезпечити безпечну роботу з даними та спростити процес адміністрування.

На рис. 3.11 представлено інтерфейс сторінки авторизації адміністративної панелі, реалізованої на базі фреймворку Django.

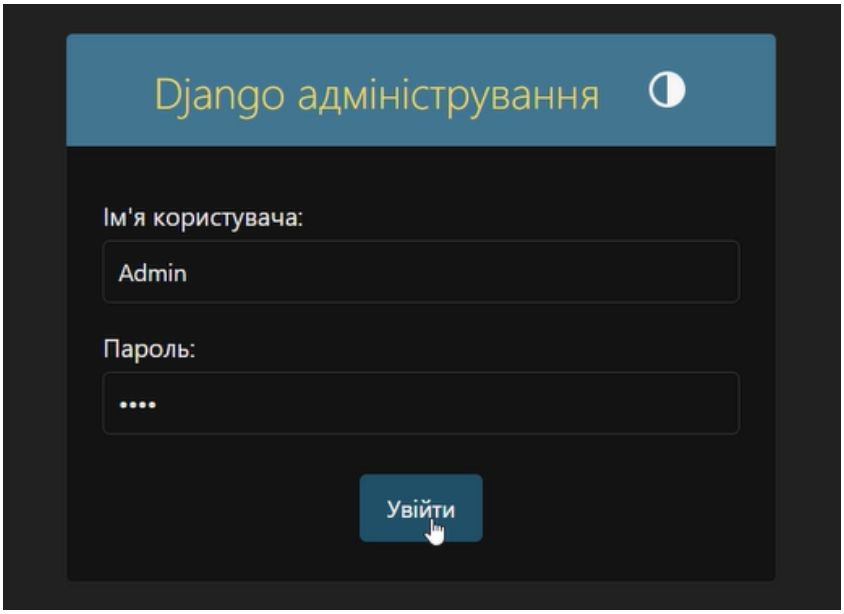


Рис. 3.11 Сторінка авторизації адміністративної панелі системи

Даний модуль забезпечує контроль доступу до функцій адміністрування

системи та дозволяє лише авторизованим користувачам здійснювати управління даними. Для входу необхідно ввести ім’я користувача та пароль, після чого система виконує перевірку облікових даних та надає доступ до адміністративного інтерфейсу. Реалізація механізму автентифікації базується на вбудованих засобах Django, що гарантує високий рівень безпеки та надійність обробки даних. Інтерфейс сторінки є мінімалістичним та інтуїтивно зрозумілим, що спрощує процес входу до системи. Результати тестування підтверджують коректність роботи механізму авторизації, зокрема правильність перевірки облікових даних та обмеження доступу до адміністративних функцій для неавторизованих користувачів.

На рис. 3.12 представлено головну сторінку адміністративної панелі веб-застосунку, реалізованої за допомогою фреймворку Django, яка забезпечує централізоване управління всіма основними сутностями системи. Інтерфейс містить перелік моделей, зокрема транзакції, категорії, бюджети та аналітичні рекомендації, для яких доступні операції додавання, редагування та перегляду записів.

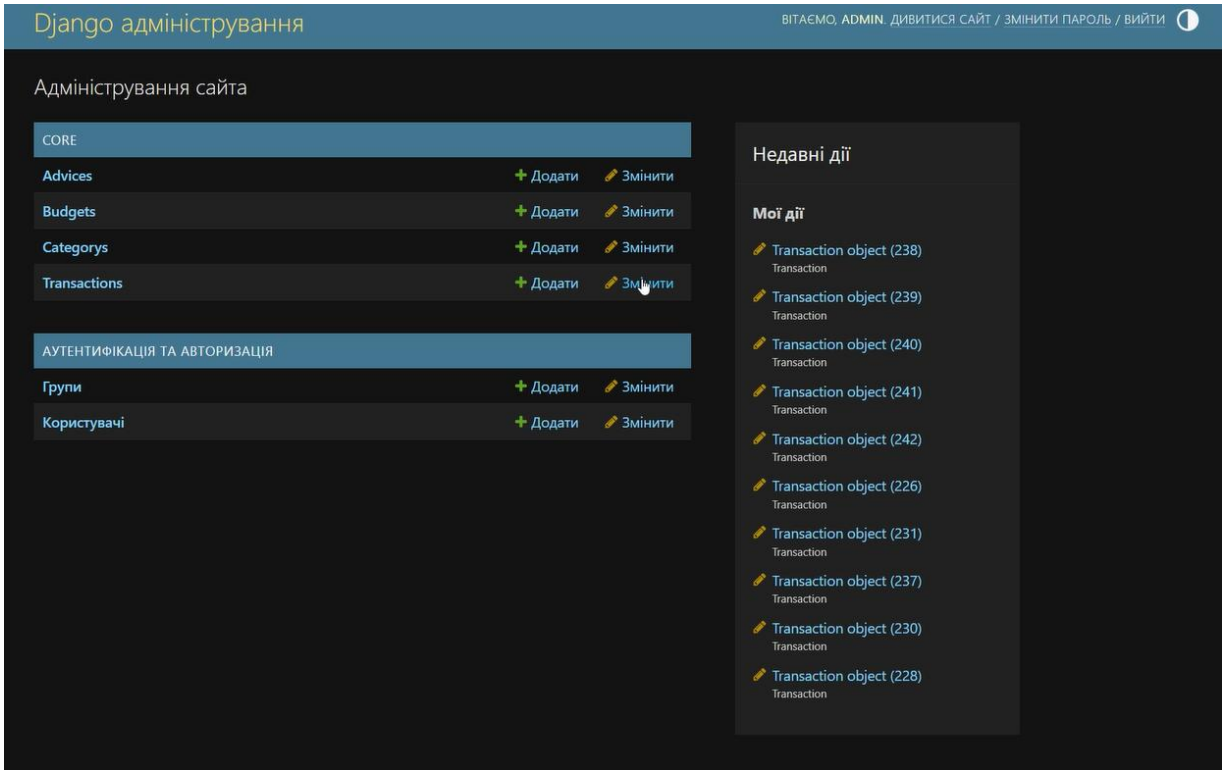


Рис. 3.12 Головна сторінка адміністративної панелі Django

Також передбачено модулі керування користувачами та групами, що дозволяє реалізувати механізми авторизації та розмежування прав доступу. У правій частині інтерфейсу відображається журнал останніх дій адміністратора, що забезпечує контроль змін у системі та підвищує прозорість роботи. Завдяки використанню Django Admin досягається висока швидкість розробки адміністративного функціоналу без необхідності створення окремого інтерфейсу, що значно спрощує процес супроводження системи. Результати тестування підтверджують коректність роботи адміністративної панелі, зручність управління даними та ефективність реалізованих механізмів контролю доступу.

На рис. 3.13 представлено інтерфейс управління транзакціями в адміністративній панелі Django, який забезпечує повний контроль над фінансовими операціями, що зберігаються у системі.

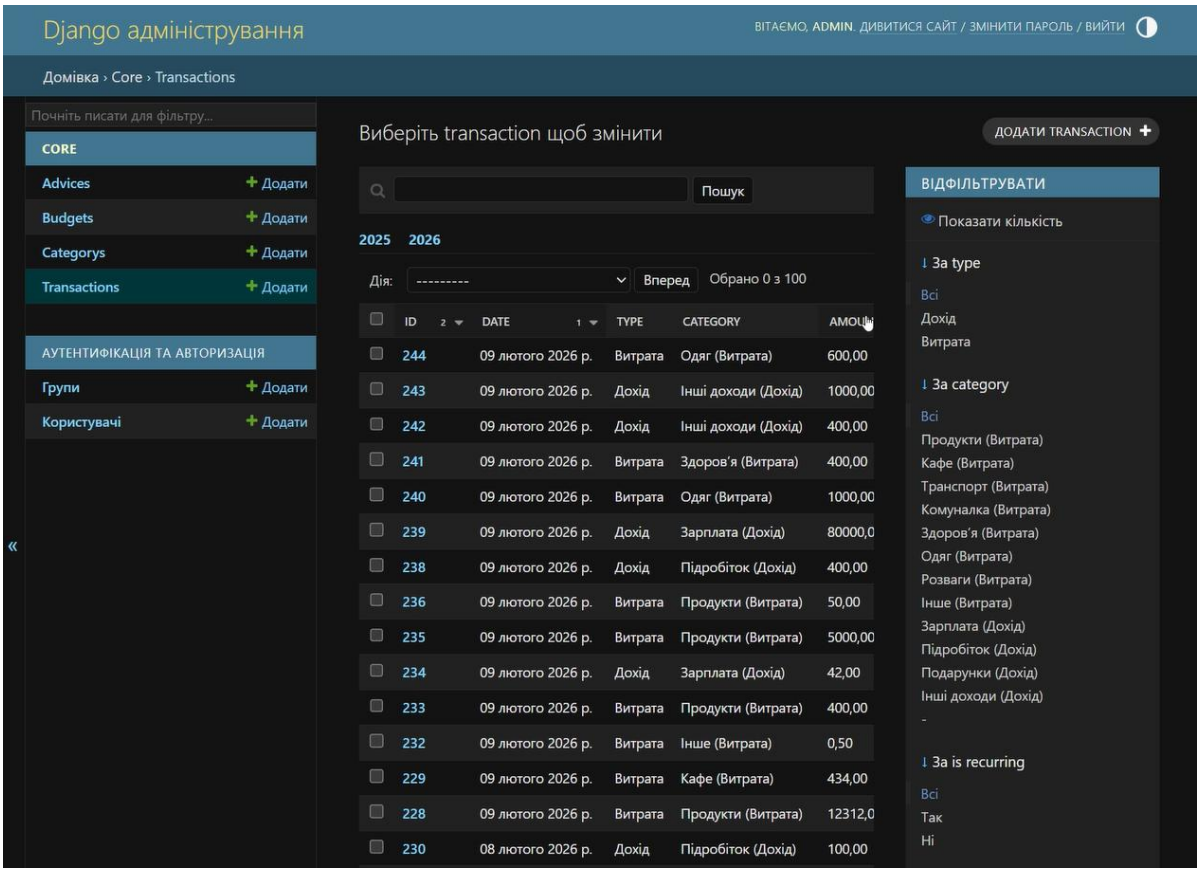


Рис. 3.13 Інтерфейс управління транзакціями в адміністративній панелі

Даний модуль дозволяє переглядати список усіх транзакцій у вигляді таблиці з основними параметрами, такими як ідентифікатор, дата, тип операції, категорія та сума. Реалізовано зручні механізми пошуку та фільтрації за різними критеріями, зокрема за типом, категорією та наявністю регулярності, що значно спрощує роботу з великими обсягами даних. Крім того, система підтримує виконання групових дій над записами, що підвищує ефективність адміністрування. Адміністратор має можливість швидко перейти до редагування конкретної транзакції або створити новий запис через відповідні елементи інтерфейсу. Використання стандартних можливостей Django Admin дозволяє забезпечити надійність, зручність та високу швидкість управління даними без необхідності реалізації окремого інтерфейсу. Результати тестування підтверджують коректність роботи механізмів фільтрації, пошуку та редагування транзакцій.

На рис. 3.14 представлено інтерфейс управління категоріями в адміністративній панелі Django, який забезпечує централізоване редагування довідника категорій доходів та витрат.

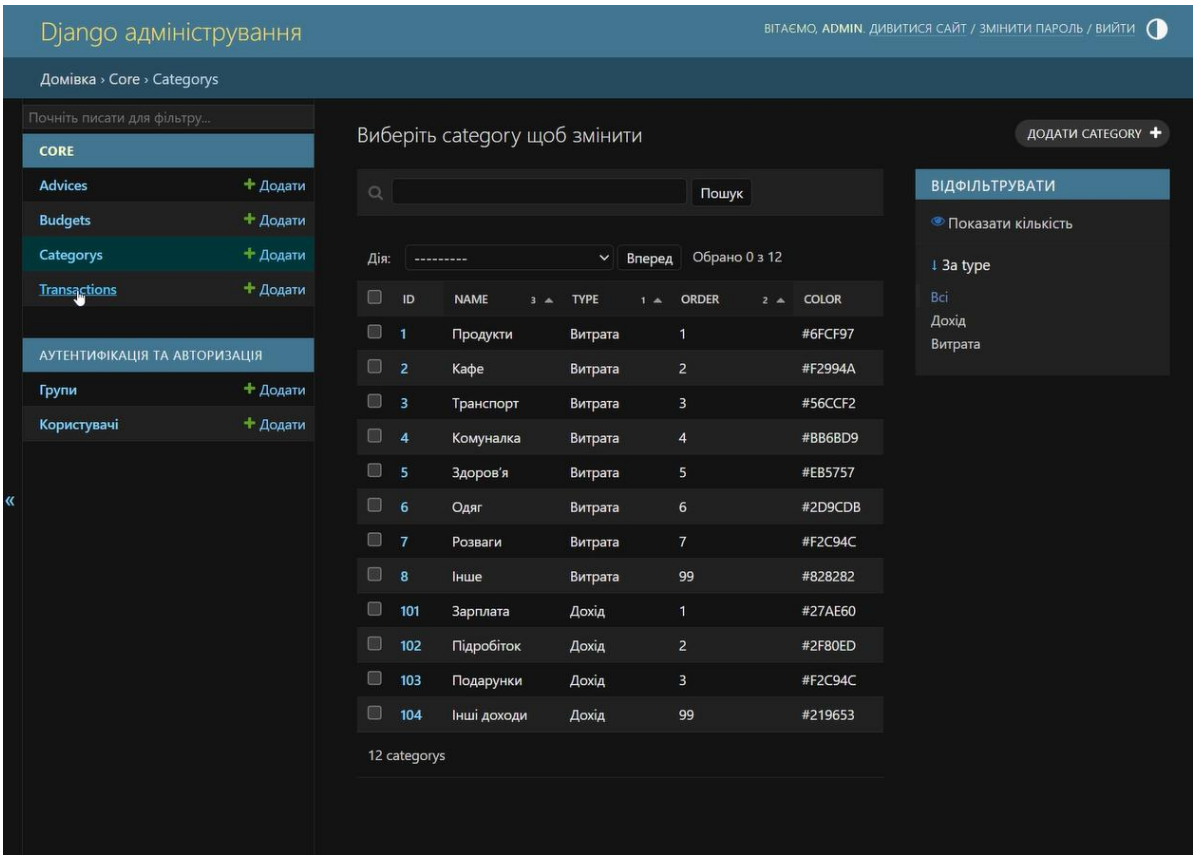


Рис. 3.14 Інтерфейс управління категоріями в адміністративній панелі

Даний модуль дозволяє переглядати список усіх категорій із зазначенням їх назви, типу (дохід або витрата), порядку відображення та кольору, що використовується для візуалізації у графіках та аналітичних елементах системи. Реалізовано функціонал пошуку та фільтрації категорій за типом, що спрощує роботу адміністратора при великій кількості записів. Також передбачено можливість створення нових категорій, редагування існуючих та виконання групових операцій. Використання кольорового кодування дозволяє підвищити наочність відображення фінансових даних у користувацькому інтерфейсі. Результати тестування підтверджують коректність роботи механізмів управління категоріями, зручність їх редагування та інтеграцію з іншими модулями системи.

На рис. 3.15 представлено інтерфейс управління користувачами в адміністративній панелі Django, який забезпечує контроль за обліковими записами системи та їх правами доступу.

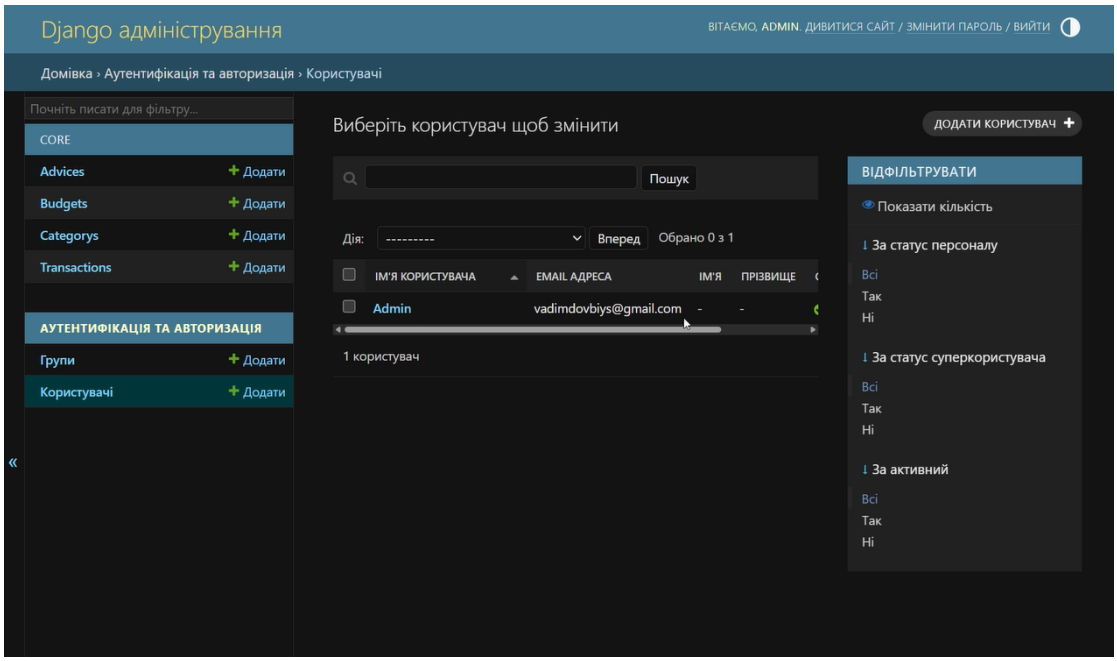


Рис. 3.15 Інтерфейс управління користувачами в адміністративній панелі

Даний модуль забезпечує повний контроль над доступом до системи та управлінням обліковими записами. Функціональні можливості реалізованого модуля дозволяють переглядати повний список зареєстрованих у системі користувачів із деталізацією їхніх основних атрибутів. До переліку параметрів, що підлягають моніторингу, належать: унікальне ім'я користувача (username), верифікована адреса електронної пошти, поточний статус активності профілю, а також ієрархія ролей і прав доступу.

З метою забезпечення ефективного управління при потенційному масштабуванні системи та зростанні кількості користувачів, у модулі реалізовано розширені можливості пошуку, сортування та фільтрації. Адміністратор може миттєво відсіювати записи за такими критеріями:

- статус персоналу (is_staff): визначає рівень доступу користувача до адміністративної частини сервісу;
- статус суперкористувача (is_superuser): надає повний обсяг прав на зміну будь-яких параметрів системи;
- активність (is_active): дозволяє оперативно блокувати або активувати доступ без видалення даних із бази.

Особлива увага при розробці була приділена аспектам безпеки. Адміністратор має повноваження не лише створювати нові облікові записи, а й здійснювати глибоке редагування існуючих профілів, включаючи скидання паролів та зміну параметрів авторизації. Використання стандартних механізмів Django Authentication System гарантує надійність збереження конфіденційної інформації, оскільки система використовує сучасні алгоритми хешування паролів (зокрема PBKDF2 з сіллю), що унеможливлює отримання доступу до них навіть у разі прямого доступу до файлу бази даних SQLite.

ВИСНОВКИ

Розроблено та досліджено веб-застосунок для управління персональними фінансами, що забезпечує облік доходів і витрат, планування бюджету та аналітичне опрацювання фінансових даних. У ході виконання роботи проведено аналіз предметної області, визначено основні функціональні вимоги до системи та обґрунтовано вибір технологій для її реалізації, зокрема застосування фреймворку Django для серверної частини та сучасних підходів до побудови користувацького інтерфейсу.

Сформовано архітектуру системи, визначено структуру бази даних, розроблено основні функціональні модулі: підсистему обліку транзакцій, модуль управління категоріями та бюджетами, аналітичний модуль і підсистему адміністрування. Відмінною характеристикою розробленого застосунку є інтеграція великої мовної моделі Gemini, застосування якої уможливлює інтелектуальний аналіз фінансової інформації та формування персоналізованих рекомендацій щодо оптимізації витрат, що підвищує практичну цінність системи.

Реалізовано користувацький інтерфейс, побудований з дотриманням принципів зручності використання, що забезпечує ефективну взаємодію користувача з системою. Проведене тестування підтвердило коректність функціонування всіх компонентів, стабільність системи та відповідність одержаних результатів визначеним вимогам. Реалізовано адміністративну панель на базі Django Admin, що забезпечує централізоване управління даними, обліковими записами та правами доступу користувачів, сприяючи надійності й захисту системи.

У межах роботи досягнуто поставлену мету, вирішено всі визначені завдання та створено повнофункціональний продукт, придатний для подальшого розвитку, масштабування і практичного впровадження. Перспективами подальших досліджень є розширення функціоналу застосунку шляхом інтеграції з банківськими API, розроблення мобільної версії, а також удосконалення аналітичного модуля із залученням складніших моделей штучного інтелекту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Python Software Foundation. Python documentation [Електронний ресурс]. Режим доступу: https://docs.python.org/3/contents.html (дата звернення: 18.02.2026).
2. Python Software Foundation. The Python Standard Library [Електронний ресурс]. Режим доступу: https://docs.python.org/3/library/index.html (дата звернення: 18.03.2026).
3. Python Software Foundation. Зміст документації Python 3.14 [Електронний ресурс]. Режим доступу: https://docs.python.org/uk/3.14/contents.html (дата звернення: 18.02.2026).
4. Django Software Foundation. Django documentation [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/ (дата звернення: 18.02.2026).
5. Django Software Foundation. Models [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/topics/db/models/ (дата звернення: 05.03.2026).
6. Django Software Foundation. Making queries [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/topics/db/queries/ (дата звернення: 05.03.2026).
7. Django Software Foundation. Working with forms [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/topics/forms/ (дата звернення: 05.03.2026).
8. Django Software Foundation. Using the Django authentication system [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/topics/auth/default/ (дата звернення: 07.03.2026).
9. Django Software Foundation. User authentication in Django [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/topics/auth/ (дата звернення: 07.03.2026).

10. Django Software Foundation. The Django admin site [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/ref/contrib/admin/ (дата звернення: 07.03.2026).

11. Django Software Foundation. URL dispatcher [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/topics/http/urls/ (дата звернення: 10.03.2026).

12. Django Software Foundation. Writing views [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/topics/http/views/ (дата звернення: 10.03.2026).

13. Django Software Foundation. The Django template language [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/ref/templates/language/ (дата звернення: 10.03.2026).

14. Django Software Foundation. django-admin and manage.py [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/6.0/ref/django-admin/ (дата звернення: 10.03.2026).

15. SQLite Consortium. SQLite Home Page [Електронний ресурс]. Режим доступу: https://sqlite.org/ (дата звернення: 12.03.2026).

16. SQLite Consortium. SQLite Documentation [Електронний ресурс]. Режим доступу: https://sqlite.org/docs.html (дата звернення: 14.03.2026).

17. SQLite Consortium. About SQLite [Електронний ресурс]. Режим доступу: https://sqlite.org/about.html (дата звернення: 14.03.2026).

18. MDN Web Docs. HTML: HyperText Markup Language [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 18.03.2026).

19. MDN Web Docs. CSS: Cascading Style Sheets [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 18.03.2026).

20. MDN Web Docs. JavaScript Guide [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 18.03.2026).

21. Chart.js. Documentation [Електронний ресурс]. Режим доступу: https://www.chartjs.org/docs/ (дата звернення: 18.03.2026).

22. Chart.js. Getting Started [Електронний ресурс]. Режим доступу: https://www.chartjs.org/docs/latest/getting-started/ (дата звернення: 18.03.2026).

23. Google AI for Developers. Gemini API docs [Електронний ресурс]. Режим доступу: https://ai.google.dev/gemini-api/docs (дата звернення: 18.03.2026).

24. Google AI for Developers. Gemini API reference [Електронний ресурс]. Режим доступу: https://ai.google.dev/api (дата звернення: 18.03.2026).

25. Google AI for Developers. Gemini API quickstart [Електронний ресурс]. Режим доступу: https://ai.google.dev/gemini-api/docs/quickstart (дата звернення: 18.03.2026).

26. Довбиус В.С. Архітектурні рішення та алгоритми обробки даних у системі управління особистими фінансами. // XI Міжнародна наукова конференція «Здобутки та досягнення прикладних та фундаментальних наук XXI століття». Збірник тез. Одесса, (01 травня 2026 р.), ст. 337-338.

27. Довбиус В.С. Проєктування веб-застосунку для управління особистими фінансами: архітектурний підхід та інтеграція LLM. // VI Міжнародна наукова конференція «Інноваційна наука: пошук відповідей на виклики сучасності». Збірник тез. Чернівці, (15 травня 2026 р.), ст. 360-361.

ДОДАТОК А

Код файлу models.py – моделі бази даних

Файл: core/models.py

```
from django.db import models
from django.contrib.auth.models import User
class Category(models.Model):
    TYPE_CHOICES = [
        ('income', 'Дохід'),
        ('expense', 'Витрата'),
    ]
    name = models.CharField(max_length=100)
    type = models.CharField(max_length=10, choices=TYPE_CHOICES)
    color = models.CharField(max_length=7, default='#808080')
    order = models.IntegerField(default=99)
    class Meta:
        ordering = ['type', 'order', 'name']
    def __str__(self):
        return f'{self.name} ({self.get_type_display()})'
class Transaction(models.Model):
    TYPE_CHOICES = [
        ('income', 'Дохід'),
        ('expense', 'Витрата'),
    ]
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name='transactions'
    )
    amount = models.DecimalField(max_digits=10, decimal_places=2)
    category = models.ForeignKey(
        Category, on_delete=models.CASCADE
    )
    type = models.CharField(max_length=10, choices=TYPE_CHOICES)
    date = models.DateField()
```

```
    description = models.TextField(blank=True)
    merchant = models.CharField(max_length=200, blank=True)
    is_recurring = models.BooleanField(default=False)
    class Meta:
        ordering = ['-date', '-id']
    def __str__(self):
        return f'{self.date} | {self.category} | {self.amount}'
class Budget(models.Model):
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name='budgets'
    )
    category = models.ForeignKey(
        Category, on_delete=models.CASCADE
    )
    month = models.DateField()
    limit_amount = models.DecimalField(max_digits=10, decimal_places=2)
    class Meta:
        unique_together = ('user', 'category', 'month')
    def __str__(self):
        return f'{self.category} | {self.month.strftime("%Y-%m")} | {self.limit_amount}'
class Advice(models.Model):
    SEVERITY_CHOICES = [
        ('info', 'Інформація'),
        ('warning', 'Попередження'),
        ('critical', 'Критичне'),
    ]
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name='advices'
    )
    title = models.CharField(max_length=200, blank=True)
    text = models.TextField()
    period = models.CharField(max_length=20, blank=True)
    severity = models.CharField(
```

```
    max_length=10, choices=SEVERITY_CHOICES, default='info'
)
created_at = models.DateTimeField(auto_now_add=True)
class Meta:
    ordering = ['-created_at']
def __str__(self):
    return f'{self.created_at:%Y-%m-%d} | {self.title}'
```

ДОДАТОК Б

Код файлу views.py – логіка обробки запитів

Файл: core/views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required
from django.db.models import Sum
from django.utils import timezone
from datetime import date
import calendar
from .models import Transaction, Category, Budget, Advice
from .forms import TransactionForm, BudgetForm
from . import ai_utils
@login_required
def dashboard(request):
    """Головна сторінка – фінансовий огляд за місяць."""
    today = date.today()
    year  = int(request.GET.get('year',  today.year))
    month = int(request.GET.get('month', today.month))
    current = date(year, month, 1)
    qs = Transaction.objects.filter(user=request.user, date__year=year,
                                    date__month=month)
    income_total   = qs.filter(type='income').aggregate(
        s=Sum('amount'))['s'] or 0
    expense_total  = qs.filter(type='expense').aggregate(
        s=Sum('amount'))['s'] or 0
    balance = income_total - expense_total
    expenses_by_cat = (
        qs.filter(type='expense')
          .values('category__name', 'category__color')
          .annotate(total=Sum('amount'))
          .order_by('-total')
    )
```

```
    context = {
        'income_total':   income_total,
        'expense_total':  expense_total,
        'balance':        balance,
        'expenses_by_cat': list(expenses_by_cat),
        'current':        current,
        'prev_month':     _prev_month(current),
        'next_month':     _next_month(current),
    }
    return render(request, 'core/dashboard.html', context)

def _prev_month(d):
    if d.month == 1:
        return date(d.year - 1, 12, 1)
    return date(d.year, d.month - 1, 1)

def _next_month(d):
    if d.month == 12:
        return date(d.year + 1, 1, 1)
    return date(d.year, d.month + 1, 1)

@login_required
def transaction_list(request):
    """Перегляд та фільтрація списку транзакцій."""
    qs = Transaction.objects.filter(user=request.user)
    tx_type  = request.GET.get('type', '')
    cat_id   = request.GET.get('category', '')
    search   = request.GET.get('search', '')
    year     = request.GET.get('year', '')
    month    = request.GET.get('month', '')
    if tx_type:   qs = qs.filter(type=tx_type)
    if cat_id:    qs = qs.filter(category_id=cat_id)
    if search:    qs = qs.filter(
        description__icontains=search) | qs.filter(
        merchant__icontains=search)
    if year:      qs = qs.filter(date__year=year)
```

```
    if month:     qs = qs.filter(date__month=month)
    income_total  = qs.filter(type='income').aggregate(
        s=Sum('amount'))['s'] or 0
    expense_total = qs.filter(type='expense').aggregate(
        s=Sum('amount'))['s'] or 0
    categories = Category.objects.all().order_by('type', 'order', 'name')
    context = {
        'transactions':   qs,
        'categories':     categories,
        'income_total':   income_total,
        'expense_total':  expense_total,
        'selected_type':  tx_type,
        'selected_cat':   cat_id,
        'search':         search,
    }
    return render(request, 'core/transactions_list.html', context)
@login_required
def add_transaction(request, tx_type):
    """Додавання нової фінансової операції."""
    if request.method == 'POST':
        form = TransactionForm(request.POST, tx_type=tx_type)
        if form.is_valid():
            tx = form.save(commit=False)
            tx.user = request.user
            tx.type = tx_type
            tx.save()
            return redirect('transactions')
    else:
        form = TransactionForm(tx_type=tx_type)
    return render(request, 'core/transaction_form.html',
                  {'form': form, 'tx_type': tx_type})
@login_required
def edit_transaction(request, pk):
```

```
    """Редагування існуючої транзакції."""
    tx = get_object_or_404(Transaction, pk=pk, user=request.user)
    if request.method == 'POST':
        form = TransactionForm(request.POST, instance=tx, tx_type=tx.type)
        if form.is_valid():
            form.save()
            return redirect('transactions')
    else:
        form = TransactionForm(instance=tx, tx_type=tx.type)
    return render(request, 'core/transaction_form.html',
                  {'form': form, 'tx_type': tx.type})
@login_required
def delete_transaction(request, pk):
    """Видалення транзакції."""
    tx = get_object_or_404(Transaction, pk=pk, user=request.user)
    if request.method == 'POST':
        tx.delete()
    return redirect('transactions')
@login_required
def budget_list(request):
    """Управління бюджетами з розрахунком перевищень."""
    today = date.today()
    year  = int(request.GET.get('year',  today.year))
    month = int(request.GET.get('month', today.month))
    current = date(year, month, 1)
    budgets = Budget.objects.filter(
        user=request.user, month=current
    ).select_related('category')
    budget_data = []
    for b in budgets:
        spent = Transaction.objects.filter(
            user=request.user,
            category=b.category,
```

```
            type='expense',
            date__year=year,
            date__month=month,
        ).aggregate(s=Sum('amount'))['s'] or 0
        budget_data.append({
            'budget':  b,
            'spent':   spent,
            'remain':  b.limit_amount - spent,
            'pct':     round(spent / b.limit_amount * 100, 1)
                       if b.limit_amount else 0,
            'over':    spent > b.limit_amount,
        })
    context = {
        'budget_data': budget_data,
        'current':     current,
        'prev_month':  _prev_month(current),
        'next_month':  _next_month(current),
    }
    return render(request, 'core/budget_list.html', context)
@login_required
def get_ai_analysis(request):
    """Отримання аналітики від Gemini AI."""
    if request.method != 'POST':
        return redirect('dashboard')
    period = request.POST.get('period', '1month')
    today  = date.today()
    if period == '1month':
        start = date(today.year, today.month, 1)
    elif period == '3months':
        m = today.month - 2 if today.month > 2 else today.month + 10
        y = today.year if today.month > 2 else today.year - 1
        start = date(y, m, 1)
    elif period == '6months':
```

```
    m = today.month - 5 if today.month > 5 else today.month + 7
    y = today.year if today.month > 5 else today.year - 1
    start = date(y, m, 1)
else:
    start = date(today.year, 1, 1)
qs = Transaction.objects.filter(
    user=request.user, date__gte=start, date__lte=today
)
income_total  = qs.filter(type='income').aggregate(
    s=Sum('amount'))['s'] or 0
expense_total = qs.filter(type='expense').aggregate(
    s=Sum('amount'))['s'] or 0
by_cat = (
    qs.filter(type='expense')
      .values('category__name')
      .annotate(total=Sum('amount'))
      .order_by('-total')
)
data = {
    'period':        period,
    'income_total':  float(income_total),
    'expense_total': float(expense_total),
    'balance':       float(income_total - expense_total),
    'by_category':   [
        {'name': r['category__name'], 'total': float(r['total'])}
        for r in by_cat
    ],
}
recommendation = ai_utils.get_recommendation(data)
Advice.objects.create(
    user=request.user,
    text=recommendation,
    period=period,
```

```
        )
        return redirect('dashboard')
```

ДОДАТОК В

Код файлу ai_utils.py – інтеграція з Gemini

Файл: core/ai_utils.py

```
import google.generativeai as genai
from django.conf import settings
def get_recommendation(data: dict) -> str:
    """
    Формує текстовий запит (prompt) на основі агрегованих
    фінансових даних та повертає відповідь моделі Gemini.
    """
    genai.configure(api_key=settings.GEMINI_API_KEY)
    model = genai.GenerativeModel('gemini-1.5-flash')
    period_labels = {
        '1month':  'за 1 місяць',
        '3months': 'за 3 місяці',
        '6months': 'за 6 місяців',
        '1year':   'за рік',
    }
    period_label = period_labels.get(data.get('period', ''), 'за вказаний період')
    cat_lines = []
    for item in data.get('by_category', []):
        cat_lines.append(
            f"  - {item['name']}: {item['total']:.2f} грн"
        )
    cat_text = '\n'.join(cat_lines) if cat_lines else '  (дані відсутні)'
    prompt = f"""
Ти фінансовий аналітик. Проаналізуй витрати користувача {period_label}
та надай конкретні рекомендації щодо оптимізації бюджету.
```

```
Фінансові дані:
  Загальний дохід:  {data['income_total']:.2f} грн
  Загальні витрати: {data['expense_total']:.2f} грн
  Баланс:           {data['balance']:.2f} грн
Витрати за категоріями:
{cat_text}
Вимоги до відповіді:
  1. Зазнач найбільш витратні категорії та їх частку.
  2. Виклич конкретні ризики (якщо витрати > доходів тощо).
  3. Надай 3-5 практичних порад щодо скорочення витрат.
  4. Використовуй українську мову. Відповідь до 200 слів.
"""
    response = model.generate_content(prompt)
    return response.text
```

ДОДАТОК Г

Код файлу urls.py – маршрутизація запитів

Файл: core/urls.py

```
from django.urls import path

from . import views

urlpatterns = [

    path('', views.dashboard, name='dashboard'),

    path('transactions/',

         views.transaction_list, name='transactions'),

    path('transactions/add/<str:tx_type>/',

         views.add_transaction, name='add_transaction'),

    path('transactions/<int:pk>/edit/',

         views.edit_transaction, name='edit_transaction'),

    path('transactions/<int:pk>/delete/',

         views.delete_transaction, name='delete_transaction'),

    path('budgets/',

         views.budget_list, name='budgets'),

    path('budgets/add/',

         views.add_budget, name='add_budget'),

    path('budgets/<int:pk>/edit/',

         views.edit_budget, name='edit_budget'),

    path('budgets/<int:pk>/delete/',

         views.delete_budget, name='delete_budget'),

    path('ai-analysis/',

         views.get_ai_analysis, name='ai_analysis'),

]
```

ДОДАТОК Д

Код файлу admin.py – адміністративна панель

Файл: core/admin.py

```
from django.contrib import admin

from .models import Category, Transaction, Budget, Advice

@admin.register(Category)

class CategoryAdmin(admin.ModelAdmin):

    list_display  = ('id', 'name', 'type', 'order', 'color')

    list_filter   = ('type',)

    search_fields = ('name',)

    ordering      = ('type', 'order', 'name')

@admin.register(Transaction)

class TransactionAdmin(admin.ModelAdmin):

    list_display  = ('id', 'date', 'type', 'category',

                     'amount', 'merchant', 'is_recurring')

    list_filter   = ('type', 'category', 'is_recurring', 'date')

    search_fields = ('description', 'merchant')

    date_hierarchy = 'date'

@admin.register(Budget)

class BudgetAdmin(admin.ModelAdmin):

    list_display = ('id', 'category', 'month', 'limit_amount')

    list_filter  = ('category', 'month')

@admin.register(Advice)

class AdviceAdmin(admin.ModelAdmin):

    list_display = ('id', 'created_at', 'severity', 'title')
```

ДОДАТОК Е

Код файлу forms.py – форми введення даних

Файл: core/forms.py

```
from django import forms
from .models import Transaction, Category, Budget
class TransactionForm(forms.ModelForm):
    class Meta:
        model  = Transaction
        fields = ['amount', 'date', 'category',
                  'description', 'merchant', 'is_recurring']
        widgets = {
            'date': forms.DateInput(attrs={'type': 'date'}),
        }
    def __init__(self, *args, **kwargs):
        tx_type = kwargs.pop('tx_type', None)
        super().__init__(*args, **kwargs)
        if tx_type:
            self.fields['category'].queryset = \
                Category.objects.filter(type=tx_type).order_by('order', 'name')
class BudgetForm(forms.ModelForm):
    month = forms.DateField(
        widget=forms.DateInput(attrs={'type': 'month'}),
        input_formats=['%Y-%m']
    )
    class Meta:
        model  = Budget
        fields = ['month', 'category', 'limit_amount']
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['category'].queryset = \
            Category.objects.filter(type='expense').order_by('order', 'name')
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Технологій цифрового розвитку

Цифровий сервіс для аналізу та управління особистими фінансами з інтеграцією великої мовної моделі Gemini

Виконав студент 4 курсу
Групи ТЦР-41
Довбиус Вадим Сергійович
Керівник роботи
Доктор філософії Герцюк Микола Модестович

Київ-2026

МЕТА

Оптимізувати процес управління особистими фінансами шляхом розробки повнофункціонального веб-додатку, який поєднує класичний інструментарій обліку та візуалізації з можливостями великої мовної моделі Gemini для семантичного аналізу та персоналізованого фінансового консультування

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. проаналізувати недоліки існуючих підходів до розробки систем обліку особистих фінансів та обґрунтувати доцільність впровадження інтелектуальних модулів аналізу даних;

2. обрати та обґрунтувати технологічний стек для реалізації клієнтської та серверної частин веб-додатку;

3. спроєктувати реляційну базу даних та архітектуру системи, що забезпечують надійне збереження й ефективну обробку фінансових транзакцій;

4. розробити алгоритми агрегації фінансових даних та їх підготовки для передачі до зовнішнього API;

5. здійснити інтеграцію великої мовної моделі Gemini із налаштуванням механізму формування запитів для отримання аналітичних рекомендацій;

6. розробити адаптивний інтерфейс користувача для управління бюджетами, перегляду транзакцій та відображення аналітичних звітів;

7. провести тестування програмного забезпечення на коректність обробки даних та стабільність взаємодії з інтелектуальним модулем.

ВИМОГИ ДО СИСТЕМИ

Функціональні вимоги:

- введення операцій: сума, категорія, дата, опис;
- збереження операцій у базі даних;
- відображення списку операцій у таблиці;
- класифікація витрат та доходів за категоріями;
- управління бюджетними лімітами з індикацією перевищень;
- візуалізація даних (кругова діаграма);
- інтеграція з Gemini API для генерації рекомендацій.

Нефункціональні вимоги:

- зручність та інтуїтивність інтерфейсу;
- швидкодія при обробці запитів (відгук < 200 мс);
- надійність збереження та цілісність даних;
- масштабованість архітектури;
- захист персональних даних від несанкціонованого доступу.

ОБ'ЄКТ, ПРЕДМЕТ ДОСЛІДЖЕННЯ ТА МЕТОДИ

Об'єкт дослідження: процеси збору, зберігання та аналізу особистих фінансових даних користувача у веб - середовищі.

Предмет дослідження: архітектурні рішення та методи інтеграції великої мовної моделі Gemini у веб - застосунок для управління особистими фінансами.

Методи дослідження: аналіз і синтез (огляд існуючих рішень), структурне проєктування (UML -діаграми, трирівнева архітектура), методи програмної інженерії (MVC -патерн, REST API), тестування програмного забезпечення.

АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

Система	Категоризація	Бюджетування	AI-аналітика
Mint (Intuit)	Автоматична	Так	Відсутня
YNAB	Ручна	Так (Zero-Based)	Відсутня
Spendee	Автоматична	Обмежене	Відсутня
Розроблений застосунок	Автоматична + ручна	Так	Gemini API (LLM)

Висновок: жоден з розглянутих аналогів не реалізує повноцінної інтеграції з великими мовними моделями для семантичного аналізу фінансових даних

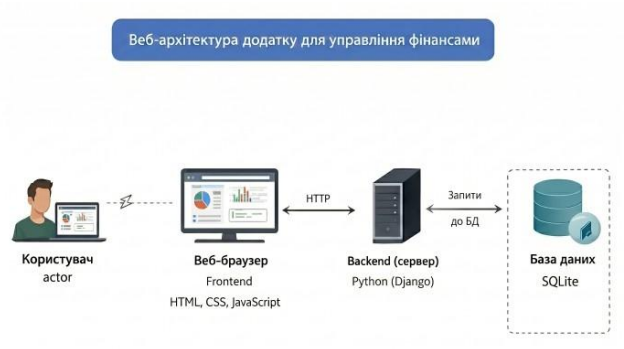
АРХІТЕКТУРА СИСТЕМИ

Архітектура відповідає стандарту MVT:

- рівень даних: Django ORM та СУБД SQLite;

- рівень бізнес-логіки: забезпечує інтеграцію з моделлю Gemini через API-запити;

- рівень представлення: побудований на системі шаблонів Django з використанням HTML5 та CSS-фреймворку Pico.css для забезпечення адаптивності інтерфейсу.



Архітектура веб-застосунку

ПРОЄКТУВАННЯ СИСТЕМИ – UML-ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

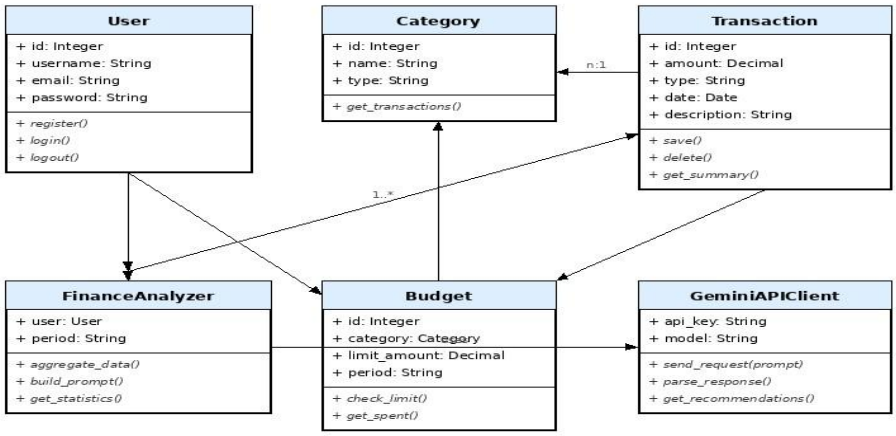
Актори системи:

- Користувач: реєстрація та авторизація, введення фінансових операцій, перегляд списку транзакцій, перегляд статистики доходів і витрат, візуалізація даних, управління бюджетами, формування аналітики, отримання рекомендацій від Gemini;

- Gemini API: отримання агрегованих фінансових даних для генерації аналітичних рекомендацій.

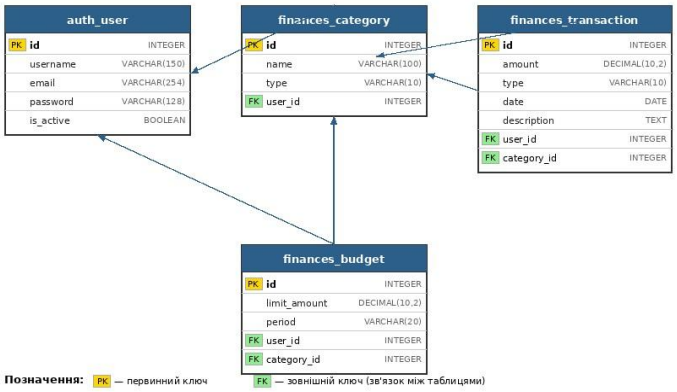
UML - діаграма варіантів використання

ДІАГРАМА КЛАСІВ ВЕБ -ЗАСТОСУНКУ



UML - діаграма класів веб -застосунку

СХЕМА БАЗИ ДАНИХ



Основні сутності:

auth_user — користувач системи (Django)

finances_category — категорія доходу або витрати

finances_transaction — фінансова операція (сума, тип, дата, категорія)

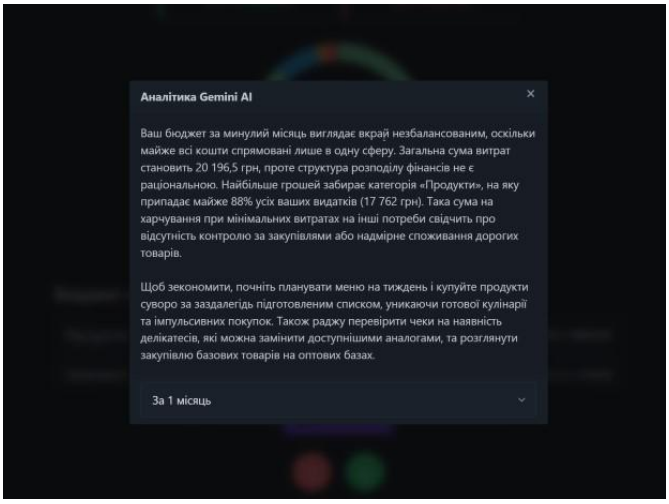
finances_budget — бюджет за категорією та лімітом

Схема бази даних веб -застосунку (SQLite)

МОДУЛЬ ІНТЕГРАЦІЇ З GEMINI API

Алгоритм роботи модуля:
1. вибірка транзакцій користувача за обраний період (місяць / квартал /рік) ;
2. агрегування витрат за категоріями, обчислення часток та відхиленьвід бюджету;
3. формування структурованого текстового промпту з фінансовим профілем;
4. передача запиту до Gemini APIчерез REST-інтерфейс;
5. обробка відповіді та відображення рекомендацій у веб-інтерфейсі.

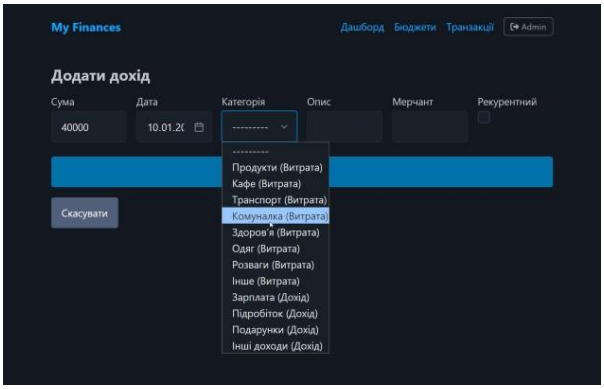
Результат: семантичний аналіз структури витрат, виявлення аномалій та генерація персоналізованих порад щодо оптимізації бюджету.



Результат роботи модуля Gemini API

ІНТЕРФЕЙС ВЕБ-ЗАСТОСУНКУ

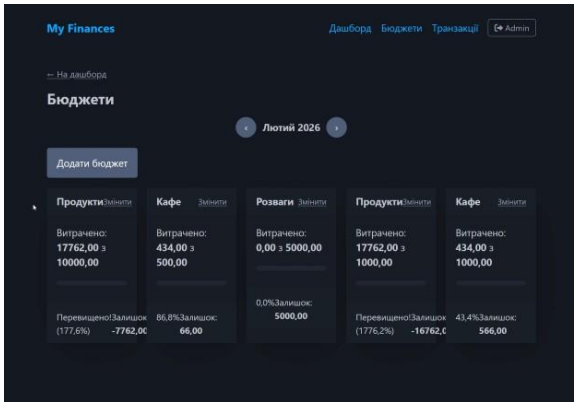
Головна сторінка



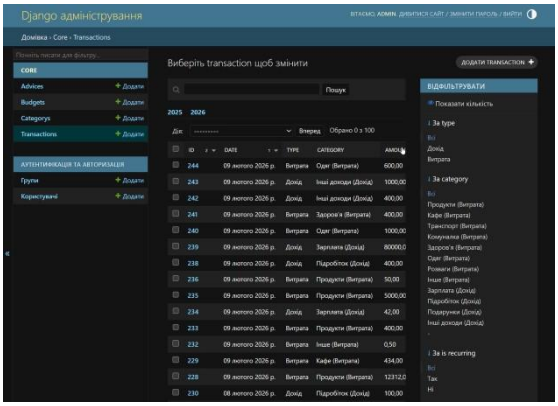
Форма введення операції

Дашборд забезпечує відображення балансу, доходів та витрат у реальному часі, а також кругову діаграму структури бюджету. Форма введення реалізує стандартизоване введення транзакцій із валідацією на рівні Django Forms.

КЕРУВАННЯ БЮДЖЕТАМИ ТА АДМІНІСТРУВАННЯ



Картки бюджетів з індикаторами



Панель адміністратора

Модуль бюджетування відображає ліміти по категоріях та індикує перевищення (наприклад, +177% по категорії «Продукти»). Django Admin забезпечує централізоване управління даними: фільтрацію, пошук і редагування транзакцій.

ТЕСТУВАННЯ СИСТЕМИ

Тестування компонентів системи проводилось відповідно до розробленого плану. Перевірялись: коректність обробки транзакцій, стабільність взаємодії з Gemini API, валідація форм та продуктивність запитів до бази даних.

№	Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Додавання транзакції	Сума=100, Категорія="Їжа"	Запис успішно збережено в БД	Відповідає очікуваному	Успішно
2	Валідація некоректних даних	Сума=-100	Виведення повідомлення про помилку	Відповідає очікуваному	Успішно
3	Розрахунок фінансового балансу	Доходи=1000, Витрати=500	Баланс = 500	Відповідає очікуваному	Успішно
4	Генерація рекомендацій Gemini	Агреговані фінансові дані	Отримання аналітичної поради	Відповідає очікуваному	Успішно

Результати тестування компонентів

Основні показники: час обробки запитів на сервері - менше 200 мс; час відгуку Gemini API- 1-3 секунди; коректність валідації форм - 100%.

НАУКОВА НОВИЗНА

Вперше запропоновано та реалізовано архітектурне рішення інтеграції великої мовної моделі Gemini у веб - застосунок для управління особистими фінансами, що забезпечує перехід від статистичного до семантичного аналізу фінансових даних користувача.

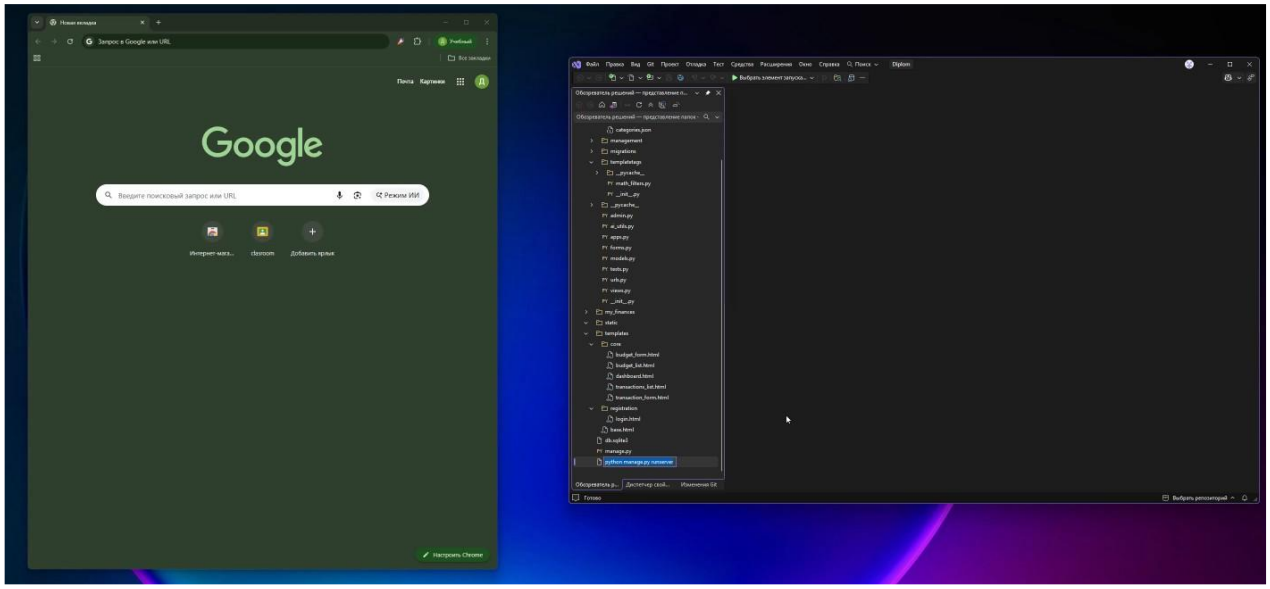
Практичне значення:

- розроблений веб-застосунок може використовуватись як самостійний продукт для управління особистими фінансами;

- модуль інтеграції з Gemini API може бути адаптований для інших застосунків, що потребують аналітики на основі LLM;

- отримані результати можуть бути використані в навчальному процесі в межах курсів з веб -програмування та штучного інтелекту.

Відео-демонстрація роботи додатку



АПРОБАЦІЯ

1. Довбиус В.С. Архітектурні рішення та алгоритми обробки даних у системі управління особистими фінансами. // Матеріали XI Міжнародна наукова конференція «Здобутки та досягнення прикладних та фундаментальних наук XXI століття». Збірник тез. Одесса, 2026, ст. 337-338.

2. Довбиус В.С. Проєктування веб-застосунку для управління особистими фінансами: архітектурний підхід та інтеграція LLM. // Матеріали VI Міжнародної наукової конференції «Інноваційна наука: пошук відповідей на виклики сучасності». Збірник тез. Чернівці, 2026, (подано до друку).

ВИСНОВКИ

1. Проведено аналіз існуючих систем обліку особистих фінансів (Mint, YNAB, Spendee), встановлено відсутність у них механізмів інтелектуального семантичного аналізу на основі LLM.

2. Спроєктовано трирівневу клієнт-серверну архітектуру веб-застосунку на основі фреймворку Django з використанням СУБД SQLite.

3. Розроблено та реалізовано модуль інтеграції з Gemini API, що забезпечує автоматичний семантичний аналіз агрегованих фінансових даних і формування персоналізованих рекомендацій.

4. Реалізовано повний функціонал системи: облік транзакцій, категоризацію витрат, управління бюджетами, інтерактивну візуалізацію та панель адміністратора.

5. Проведено тестування, яке підтвердило коректність і стабільність роботи системи: час обробки запитів - до 200 мс, відгук API - 1–2 с.

Дякую за увагу!