НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення інформаційної системи управління особистими фінансами»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми к.т.н., доцент	_____ (підпис)	**Ганна ВАЙГАНГ**
Керівник бакалаврської кваліфікаційної роботи к.т.н., с.н.с.	_____ (підпис)	**Георгій БОРОДКІН**
Виконав	_____ (підпис)	**Богдан ЗАЙЧЕНКО**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

Зайченко Богдану Миколайовичу
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи
«Програмне забезпечення інформаційної системи управління особистими фінансами»
затверджена наказом від «19» грудня 2025 р. №3204 «С»
Термін подання завершеної роботи на кафедру «22 травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):
_____
Перелік питань, що підлягають дослідженню:
проаналізувати предметну область управління особистими фінансами;
дослідити існуючі програмні рішення та визначити їхні переваги й недоліки;
сформувати функціональні, нефункціональні та технічні вимоги до системи;
побудувати UML-моделі основних процесів; розробити логічну та фізичну моделі бази даних;
обґрунтувати вибір технологічних засобів реалізації;
реалізувати основні програмні модулі;
провести тестування та оцінити ефективність розробленого рішення.
Перелік графічних документів (за необхідності).
_____
Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Георгій БОРОДКІН**

Завдання взяв до виконання _____ (підпис) **Богдан ЗАЙЧЕНКО**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

- API - Application Programming Interface, інтерфейс програмної взаємодії між модулями системи.

- БД - база даних, структуроване сховище інформації про користувачів, рахунки та фінансові операції.

- СУБД - система управління базами даних.

- CRUD - набір базових операцій над даними: створення, читання, оновлення та видалення.

- CSV - текстовий формат для експорту фінансових операцій і звітів.

- GUI - графічний інтерфейс користувача.

- IDE - інтегроване середовище розробки програмного забезпечення.

- JSON - формат обміну структурованими даними.

- PDF - формат електронного документа для збереження звітів.

- SQL - мова структурованих запитів до реляційної бази даних.

- SQLite - вбудована реляційна СУБД для локального зберігання даних.

- UML - уніфікована мова моделювання програмних систем.

- UI - користувацький інтерфейс.

- UX - досвід користувача під час взаємодії з програмним продуктом.

- RBAC - role-based access control, рольова модель керування доступом.

- ПЗ - програмне забезпечення.

- ІС - інформаційна система.

- MVP - мінімально життєздатна версія програмного продукту.

- KPI - ключовий показник ефективності, що використовується в аналітичних панелях.

ВСТУП

У сучасних умовах цифровізації повсякденного життя особисті фінанси дедалі частіше потребують системного обліку, аналізу та планування. Зростання кількості безготівкових платежів, використання декількох банківських карток, регулярні підписки, кредити, заощадження та короткострокові фінансові цілі ускладнюють контроль грошових потоків. За відсутності зручного програмного інструмента користувач часто не має цілісного уявлення про структуру власних доходів і витрат, що призводить до нераціонального використання коштів і складності у плануванні бюджету.

Традиційний спосіб ведення особистого фінансового обліку у паперовому вигляді або в окремих електронних таблицях не забезпечує достатньої оперативності, наочності та точності. Такі підходи вимагають значних ручних дій, не завжди підтримують категоризацію витрат, автоматичні підсумки, контроль бюджетних лімітів і формування аналітичних звітів. Тому актуальним є створення інформаційної системи, яка дозволяє користувачу централізовано вести облік фінансових операцій, планувати витрати, аналізувати динаміку доходів і контролювати виконання фінансових цілей.

Інформаційна система управління особистими фінансами має забезпечувати автоматизацію ключових процесів: реєстрацію користувача, створення рахунків, введення доходів і витрат, налаштування категорій, формування бюджетів, облік заощаджень, побудову звітів і експорт даних. Особливо важливо, щоб така система мала зрозумілий інтерфейс, підтримувала локальне збереження персональних даних, забезпечувала цілісність фінансової інформації та давала можливість швидко отримувати узагальнену картину поточного фінансового стану.

Метою бакалаврської кваліфікаційної роботи є розроблення програмного забезпечення інформаційної системи управління особистими фінансами, що забезпечує автоматизований облік доходів і витрат, планування бюджету,

контроль фінансових цілей, формування аналітичних звітів та зручну взаємодію користувача з фінансовими даними.

Для досягнення поставленої мети необхідно виконати такі **завдання:**

1. проаналізувати предметну область управління особистими фінансами;
2. дослідити існуючі програмні рішення та визначити їхні переваги й недоліки;
3. сформувати функціональні, нефункціональні та технічні вимоги до системи;
4. побудувати UML-моделі основних процесів; розробити логічну та фізичну моделі бази даних;
5. обґрунтувати вибір технологічних засобів реалізації;
6. реалізувати основні програмні модулі;
7. провести тестування та оцінити ефективність розробленого рішення.

Об’єктом дослідження є процеси обліку, планування та аналізу особистих фінансів користувача в умовах використання інформаційних технологій.

Предметом дослідження є методи, моделі та програмні засоби створення інформаційної системи для управління особистими фінансами, включаючи структуру даних, алгоритми оброблення фінансових операцій, модулі бюджетування та аналітичної звітності.

Методи дослідження включають системний аналіз предметної області, об’єктно-орієнтоване моделювання, UML-моделювання, ER-проєктування бази даних, реляційне моделювання, алгоритмізацію програмних процесів, функціональне тестування та оцінювання якості програмного забезпечення.

Практична цінність роботи полягає у створенні програмного продукту, який може використовуватися для ведення особистого фінансового обліку, контролю витрат, аналізу структури доходів і підтримки прийняття щоденних фінансових рішень. Розроблена система може бути розширена шляхом додавання імпорту банківських виписок, синхронізації між пристроями, прогнозування витрат та інтеграції з хмарним сховищем.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область управління особистими фінансами охоплює сукупність процесів, пов’язаних із реєстрацією доходів, обліком витрат, плануванням бюджету, контролем фінансових цілей, аналізом залишків коштів і підготовкою звітної інформації для користувача. Особисті фінанси є динамічною сферою, оскільки фінансові операції виконуються щоденно, мають різні джерела, категорії, періодичність та вплив на загальний баланс.

У типовій ситуації користувач отримує доходи з декількох джерел: заробітна плата, стипендія, підробіток, перекази, відсотки за депозитами або інші надходження. Водночас витрати поділяються на обов’язкові та змінні: харчування, транспорт, комунальні платежі, навчання, підписки, медичні послуги, розваги та непередбачені покупки. Без систематизації такі дані швидко втрачають прозорість, а користувачеві складно визначити, які категорії формують найбільше навантаження на бюджет.

Інформаційна система управління особистими фінансами розглядається як програмне середовище, яке надає користувачу засоби для введення, збереження, редагування, видалення та аналізу фінансових операцій. Головним призначенням такої системи є переведення хаотичних фінансових записів у структуровану модель даних, де кожна операція має дату, суму, тип, категорію, рахунок, опис і за потреби додатковий коментар.

Узагальнену структуру предметної області інформаційної системи управління особистими фінансами наведено на рис. 1.1. Вона відображає основні об’єкти, з якими працює користувач: рахунки, операції, категорії, бюджети, фінансові цілі та звіти.

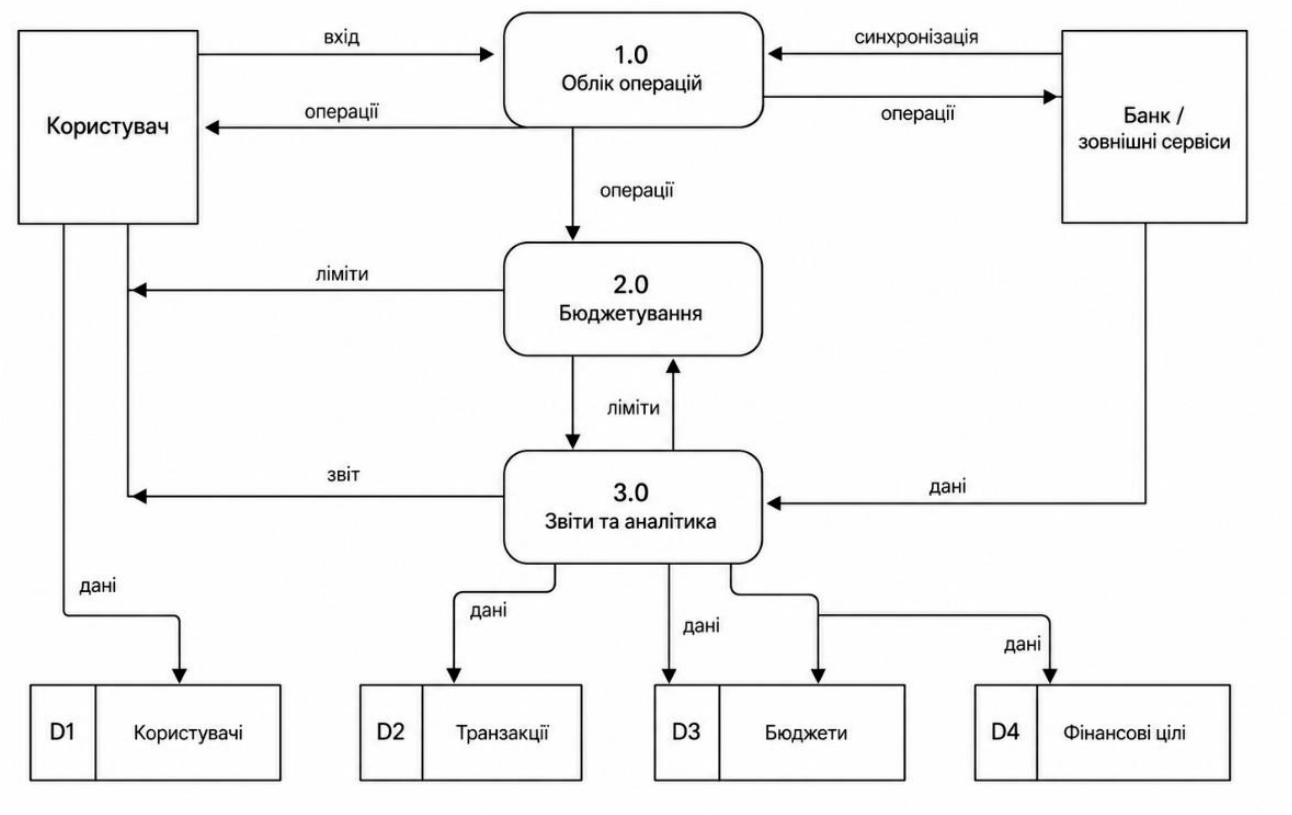

Рис. 1.1 Узагальнена структура предметної області управління особистими фінансами

Як видно з рис. 1.1, центральним елементом предметної області є користувач, який створює фінансові записи та отримує з них аналітичну інформацію. Рахунки відображають джерела зберігання коштів, категорії дають змогу класифікувати операції, бюджети встановлюють обмеження витрат, а звіти узагальнюють результати за обраний період.

Для формального опису предметної області доцільно виділити основні інформаційні об’єкти, наведені у табл. 1.1. Вони визначають мінімальний набір сутностей, необхідних для побудови повноцінної системи персонального фінансового обліку.

Таблиця 1.1

Основні об’єкти предметної області управління особистими фінансами

Об’єкт	**Призначення**	**Основні дані**
Користувач	Власник персонального фінансового профілю	ПІБ, логін, пароль, електронна пошта

Продовження таблиці 1.1

Рахунок	Джерело зберігання або руху коштів	Назва, тип, валюта, початковий баланс
Категорія	Класифікація доходів і витрат	Назва, тип, іконка, опис
Фінансова операція	Факт надходження або витрати коштів	Дата, сума, тип, рахунок, категорія, опис
Бюджет	Обмеження витрат за періодом або категорією	Період, ліміт, фактичні витрати, залишок
Фінансова ціль	План накопичення коштів	Назва, цільова сума, поточна сума, дата завершення
Звіт	Аналітичне узагальнення фінансових даних	Період, показники, графіки, експорт

Систематизація об’єктів у табл. 1.1 показує, що інформаційна система повинна не лише зберігати окремі операції, а й забезпечувати зв’язок між ними. Наприклад, кожна витрата пов’язана з певним рахунком і категорією, а бюджет формується на основі сукупності витрат за визначений період. Завдяки такій структурі користувач отримує можливість аналізувати власні фінанси не ізольовано, а як взаємопов’язану систему показників.

Отже, предметна область управління особистими фінансами потребує програмної підтримки, яка забезпечує точність обліку, швидкий доступ до інформації, зрозумілу аналітику та можливість планування майбутніх витрат. Це визначає необхідність розроблення інформаційної системи з надійною базою даних, логічною структурою модулів і зручним користувацьким інтерфейсом.

1.2 Аналіз вимог до програмної системи

Аналіз вимог є одним з основних етапів розроблення інформаційної системи, оскільки саме на цьому етапі визначаються функції програмного продукту, обмеження його використання, якісні характеристики та очікувані результати взаємодії користувача із системою. Для інформаційної системи

управління особистими фінансами вимоги формуються з урахуванням потреб користувача у швидкому введенні даних, наочному контролі бюджету, захищеному зберіганні фінансової інформації та отриманні зрозумілих звітів.

Вимоги до системи поділено на функціональні, нефункціональні та технічні. Функціональні вимоги описують конкретні дії, які виконує програмне забезпечення. Нефункціональні вимоги визначають якість роботи системи, її зручність, продуктивність, безпеку та надійність. Технічні вимоги задають умови реалізації, середовище виконання та технологічні обмеження.

Таблиця 1.2

Функціональні вимоги до інформаційної системи

№	Вимога	Опис
1	Реєстрація та авторизація	Створення облікового запису користувача, вхід до системи, перевірка пароля та обмеження доступу до персональних даних.
2	Керування рахунками	Додавання, редагування та видалення готівкових, карткових або накопичувальних рахунків.
3	Облік доходів і витрат	Створення фінансових операцій із зазначенням дати, суми, категорії, рахунку та опису.
4	Керування категоріями	Формування власних категорій доходів і витрат, використання типових категорій, редагування назв.
5	Бюджетування	Створення бюджетів за періодами та категоріями, контроль фактичного використання ліміту.
6	Фінансові цілі	Ведення цілей накопичення, контроль поточної суми та відсотка виконання.
7	Аналітичні звіти	Побудова звітів за доходами, витратами, категоріями, рахунками та періодами.
8	Експорт даних	Збереження результатів у CSV, HTML або PDF для подальшого аналізу.

Функціональні вимоги, наведені у табл. 1.2, визначають основний набір можливостей, без яких система не може виконувати своє призначення. Особливу

увагу приділено обліку операцій і бюджетуванню, оскільки саме ці процеси забезпечують користувачу контроль над поточним фінансовим станом.

Таблиця 1.3

Нефункціональні вимоги до інформаційної системи

№	Категорія	Вимога
1	Зручність використання	Інтерфейс має бути зрозумілим без додаткового навчання, а основні дії повинні виконуватися за мінімальну кількість кроків.
2	Продуктивність	Типові операції додавання, пошуку та фільтрації мають виконуватися без помітної затримки для користувача.
3	Надійність	Система має зберігати цілісність даних навіть у разі некоректного завершення роботи.
4	Безпека	Паролі зберігаються у хешованому вигляді, доступ до даних обмежується обліковим записом користувача.
5	Масштабованість	Структура програмного забезпечення повинна дозволяти додавання нових модулів без повної переробки системи.
6	Супроводжуваність	Код і структура бази даних мають бути логічно організовані та зрозумілі для подальшої підтримки.

Нефункціональні вимоги, подані у табл. 1.3, забезпечують якість програмного продукту. Навіть за наявності повного набору функцій система не буде ефективною, якщо користувачу складно вводити дані, звіти формуються повільно або виникає ризик втрати фінансової інформації.

Таблиця 1.4

Технічні вимоги до програмної системи

Компонент	Вимога	Призначення
Мова програмування	Python або інша сучасна мова високого рівня	Реалізація бізнес-логіки, оброблення даних і взаємодії з БД
Інтерфейс	Графічний або вебінтерфейс	Зручна робота користувача з операціями, бюджетами та звітами
База даних	SQLite або інша реляційна СУБД	Локальне збереження структурованих фінансових даних
Звіти	CSV, HTML, PDF	Експорт аналітичної інформації у зручних форматах
Безпека	Хешування паролів, валідація даних	Захист персональної фінансової інформації
Середовище	Windows, macOS або Linux	Можливість запуску системи на типових персональних комп’ютерах

Технічні вимоги, наведені у табл. 1.4, визначають базові умови реалізації системи. Для навчального програмного прототипу доцільним є використання локальної реляційної бази даних, оскільки вона спрощує розгортання, не потребує окремого серверного середовища та забезпечує достатню продуктивність для персонального фінансового обліку.

Отже, сформовані вимоги підтверджують, що розроблювана інформаційна система повинна поєднувати функції щоденного фінансового обліку, аналітики, бюджетного контролю та безпечного зберігання даних. Визначені вимоги використовуються як основа для подальшого моделювання предметної області та проєктування програмного забезпечення.

1.3 Моделювання предметної області

Моделювання предметної області виконується з метою формального опису структури та поведінки інформаційної системи управління особистими фінансами. Для цього використовуються UML-діаграми, які дозволяють

відобразити взаємодію користувача із системою, послідовність виконання основних сценаріїв і логіку оброблення фінансових даних.

Першим етапом моделювання є побудова діаграми прецедентів. Вона показує, які функції доступні користувачу та адміністратору системи. Основним актором є користувач, який веде облік особистих фінансів, створює рахунки, додає операції, переглядає звіти та контролює бюджети. Адміністратор відповідає за службове налаштування системи, резервне копіювання та контроль довідників.

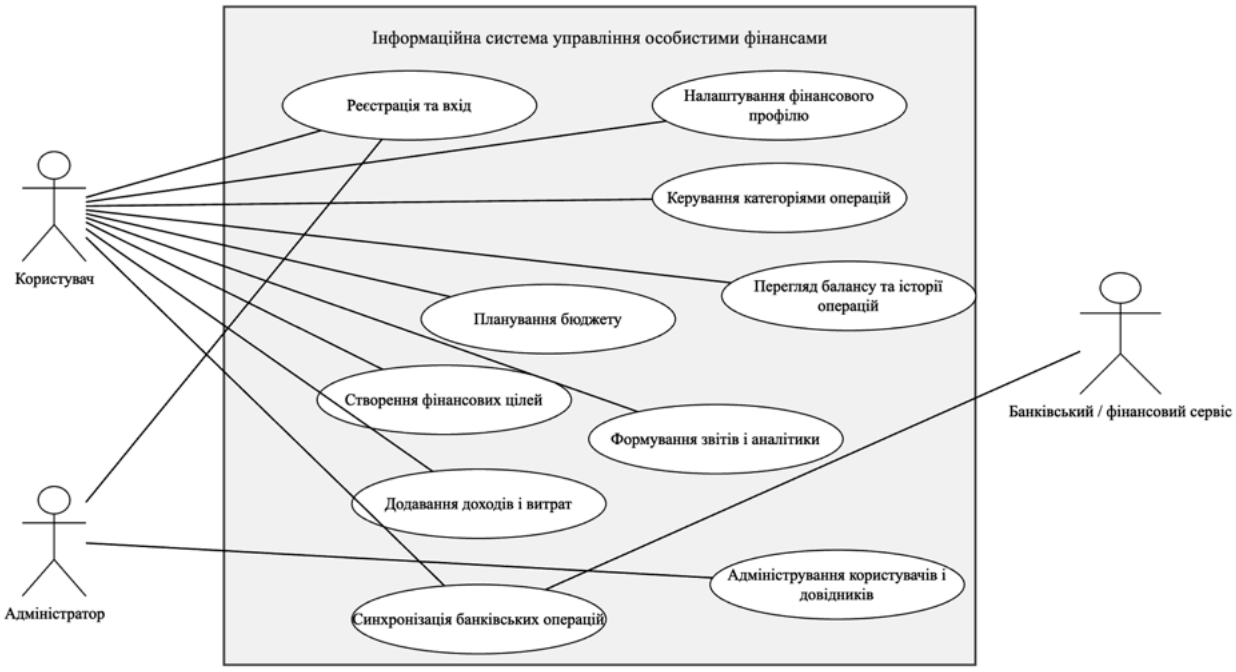

Рис. 1.2 Діаграма прецедентів інформаційної системи управління особистими фінансами

На рис. 1.2 відображено основні варіанти використання системи: авторизація, керування рахунками, введення доходів і витрат, налаштування категорій, формування бюджету, створення фінансової цілі та перегляд аналітичного звіту. Така модель дає змогу визначити межі функціональності майбутнього програмного продукту.

Для деталізації сценарію додавання фінансової операції побудовано діаграму послідовності, наведену на рис. 1.3. Вона показує обмін

повідомленнями між користувачем, інтерфейсом, сервісом оброблення операцій, модулем валідації та базою даних.

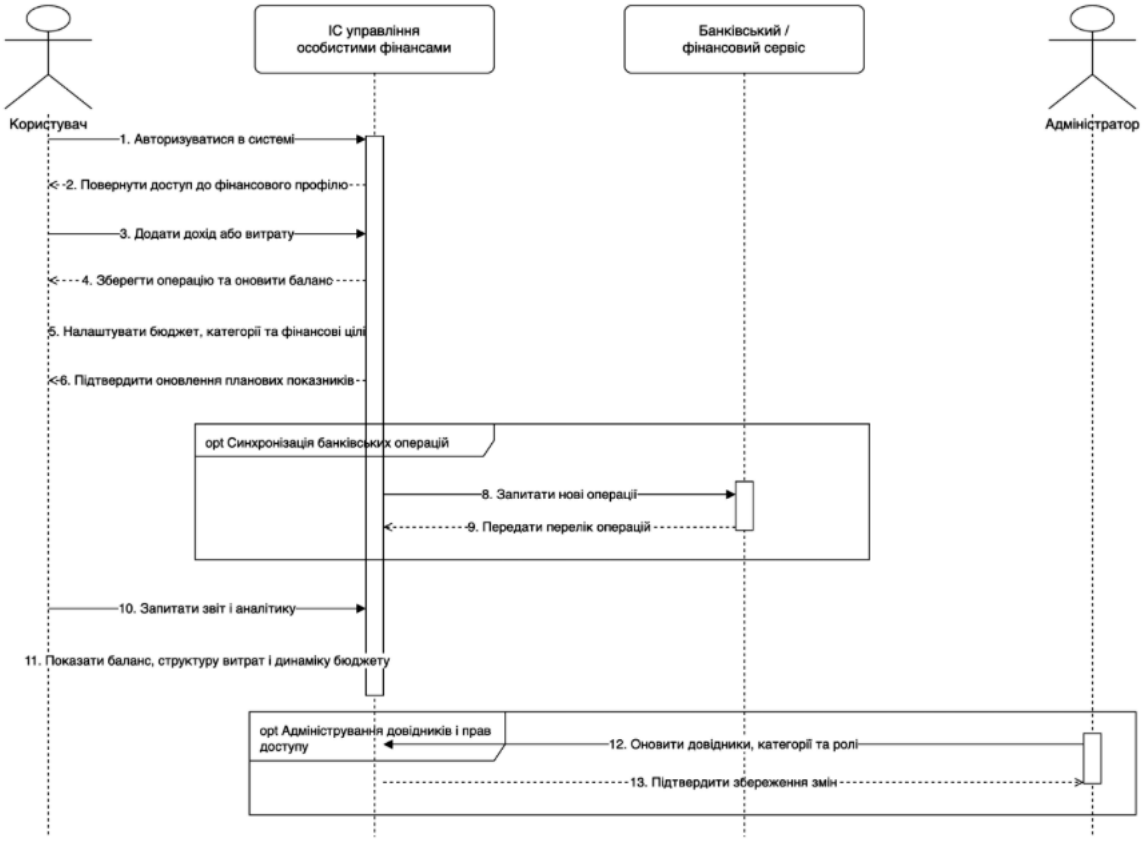

Рис. 1.3 Діаграма послідовності додавання фінансової операції

Як видно з рис. 1.3, користувач вводить дані про операцію, після чого інтерфейс передає їх до сервісу оброблення. Сервіс перевіряє коректність суми, дати, рахунку та категорії. Якщо дані є правильними, запис зберігається в базі даних, баланс рахунку оновлюється, а користувачу повертається повідомлення про успішне виконання операції.

Логіка роботи користувача із системою також може бути представлена у вигляді діаграми активності. Вона описує послідовність кроків від входу в систему до формування фінансового звіту.

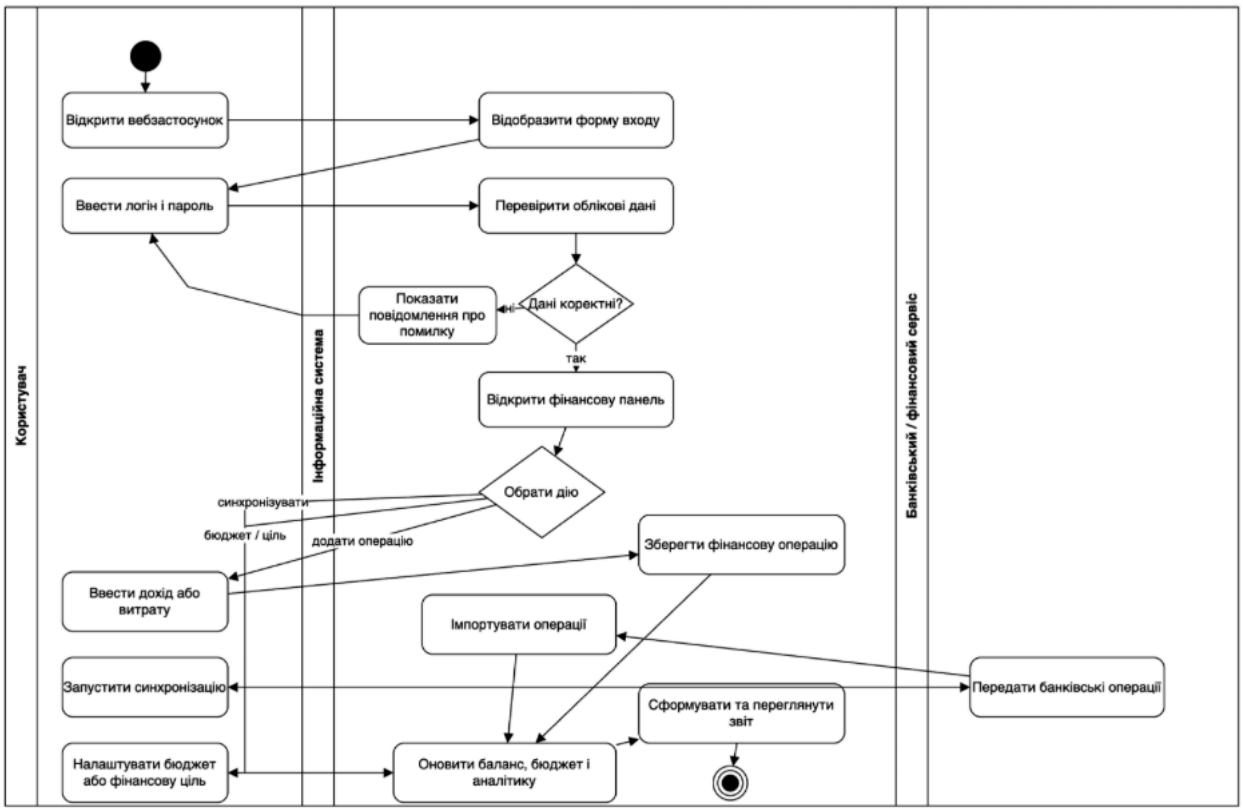

Рис. 1.4 Діаграма активності процесу управління особистими фінансами

Діаграма активності на рис. 1.4 демонструє, що користувач після авторизації може обрати дію: додати операцію, переглянути бюджет, сформувати звіт або змінити фінансову ціль. Якщо дані введено некоректно, система повертає користувача до редагування. У разі успішного виконання операції інформація зберігається, а аналітичні показники оновлюються.

Побудовані UML-моделі дають можливість уточнити функціональну структуру системи до початку програмної реалізації. Вони формують основу для розроблення логічної моделі даних, визначення класів програмної системи та побудови архітектури прикладного програмного забезпечення.

1.4 Огляд інформаційних джерел та існуючих рішень

Для визначення функціональних можливостей розроблюваної інформаційної системи доцільно проаналізувати існуючі рішення у сфері управління особистими фінансами. Такі програмні продукти дозволяють оцінити

типові підходи до організації інтерфейсу, структури фінансових операцій, бюджетування, аналітики та захисту персональних даних.

У межах аналізу розглянуто популярні рішення: Money Manager Expense & Budget, Wallet, Monefy, Spendee та YNAB. Ці системи мають різний рівень складності, проте всі вони орієнтовані на автоматизацію персонального фінансового обліку.

Money Manager Expense & Budget - мобільний застосунок для відстеження витрат, доходів, рахунків і бюджетів. Рішення орієнтоване на користувачів, які прагнуть контролювати власні доходи, витрати та бюджет. Основними перевагами є зручна категоризація, графіки, робота з рахунками, резервне копіювання. Водночас серед обмежень можна виділити такі особливості: частина функцій доступна лише у розширеній версії, складність для початківців.

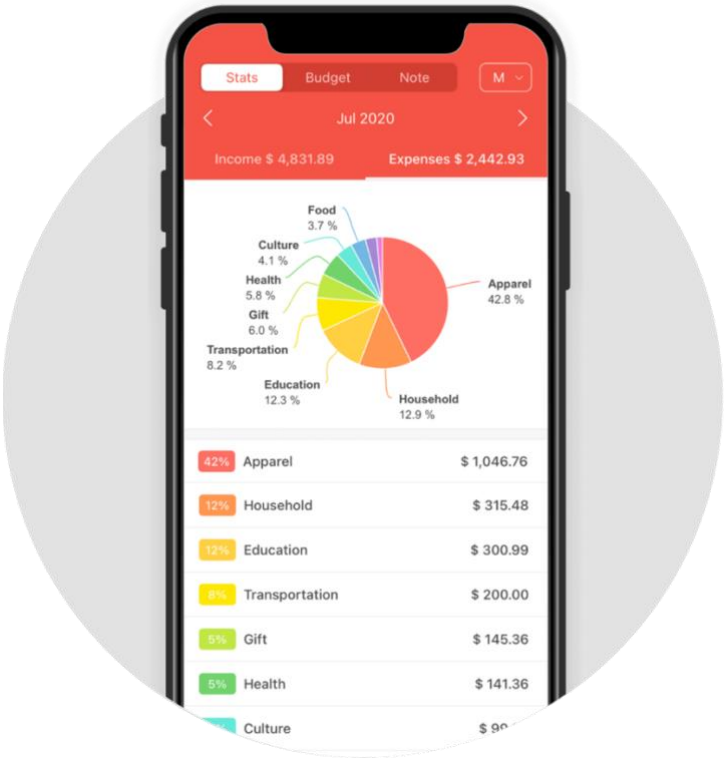

Рис. 1.5 Інтерфейс програмного рішення Money Manager Expense & Budget

Wallet - система персонального бюджетування та синхронізації фінансових даних. Рішення орієнтоване на користувачів, які прагнуть контролювати власні доходи, витрати та бюджет. Основними перевагами є підтримка планування бюджету, банківська синхронізація, аналітика. Водночас серед обмежень можна

виділити такі особливості: залежність від хмарного сервісу, обмеження безплатної версії.

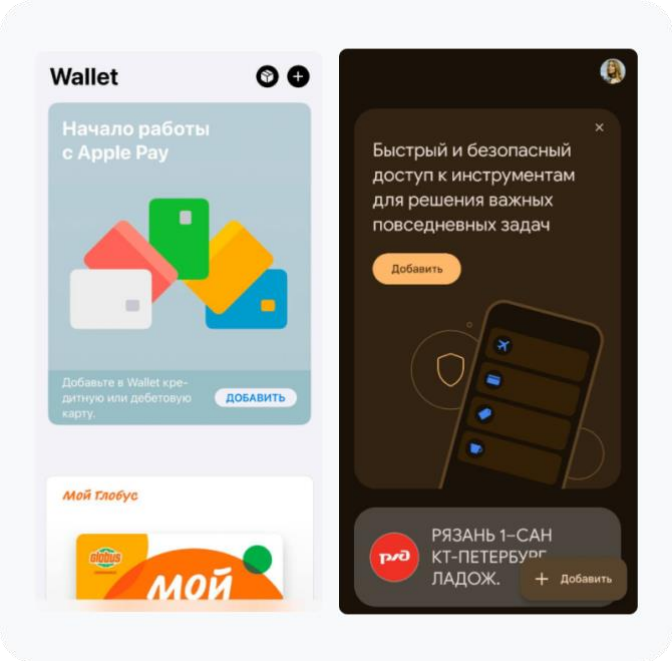

Рис. 1.6 Інтерфейс програмного рішення Wallet

Monefy - простий застосунок для швидкого введення витрат і доходів. Рішення орієнтоване на користувачів, які прагнуть контролювати власні доходи, витрати та бюджет. Основними перевагами є мінімалістичний інтерфейс, швидке додавання операцій. Водночас серед обмежень можна виділити такі особливості: обмежені аналітичні можливості.

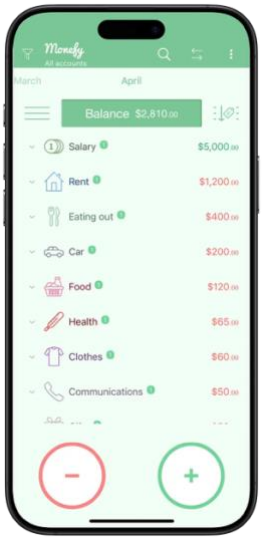

Рис. 1.7 Інтерфейс програмного рішення Monefy

Spendee - фінансовий менеджер для сімейного або групового бюджету. Рішення орієнтоване на користувачів, які прагнуть контролювати власні доходи, витрати та бюджет. Основними перевагами є спільні гаманці, категорії, графіки, планування. Водночас серед обмежень можна виділити такі особливості: частина можливостей потребує платної підписки.

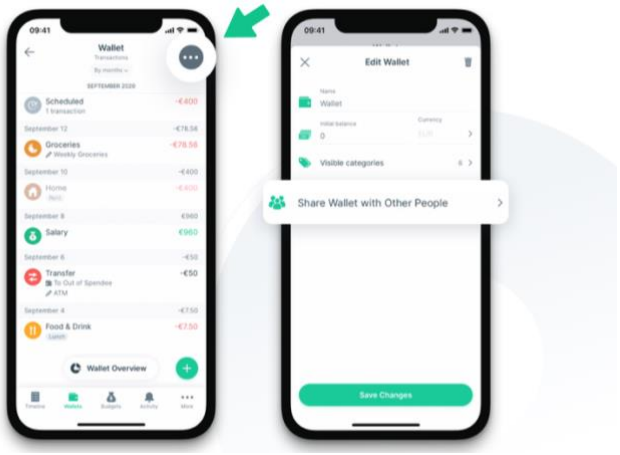

Рис. 1.8 Інтерфейс програмного рішення Spendee

YNAB - система бюджетування за принципом розподілу кожної грошової одиниці. Рішення орієнтоване на користувачів, які прагнуть контролювати власні доходи, витрати та бюджет. Основними перевагами є сильна методика планування, контроль цілей, навчальні матеріали. Водночас серед обмежень можна виділити такі особливості: складніша логіка використання, платна модель.

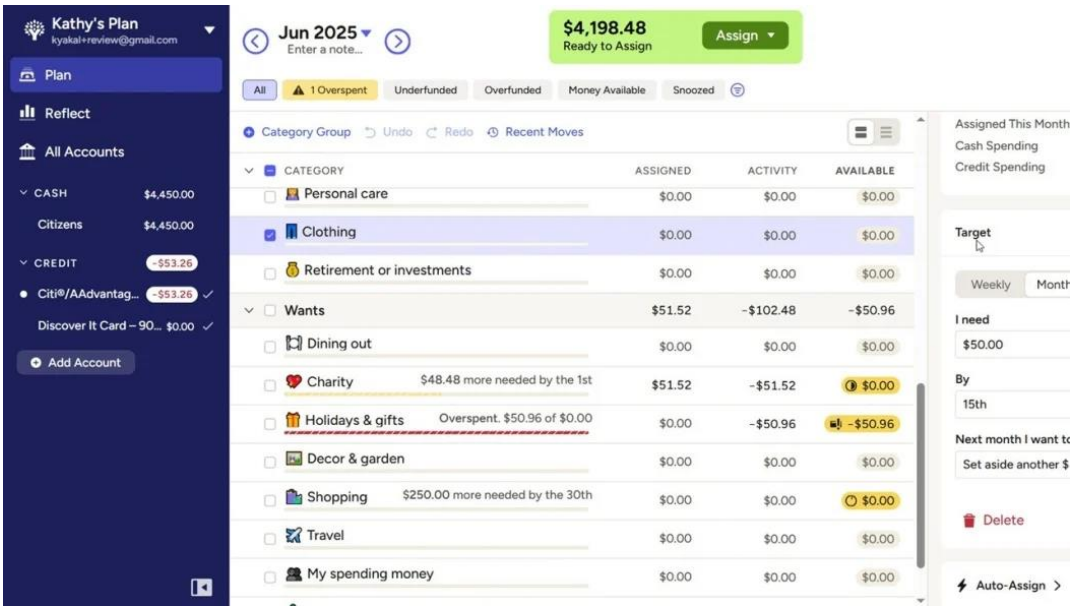

Рис. 1.9 Інтерфейс програмного рішення YNAB

Таблиця 1.5

Порівняльна характеристика існуючих рішень

Рішення	**Облік операцій**	**Бюджет**	**Цілі**	**Звіти**	**Обмеження**
Money Manager Expense & Budget	Так	Так	Частково	Так	частина функцій доступна лише у розширеній версії, складність для початківців
Wallet	Так	Так	Так	Так	залежність від хмарного сервісу, обмеження безплатної версії
Monefy	Так	Частково	Частково	Так	обмежені аналітичні можливості
Spendee	Так	Так	Так	Так	частина можливостей потребує платної підписки
YNAB	Так	Так	Так	Так	складніша логіка використання, платна модель

Як видно з табл. 1.5, усі розглянуті рішення забезпечують базовий облік доходів і витрат, проте відрізняються рівнем аналітики, доступністю бюджетування та гнучкістю налаштувань. Для навчального програмного продукту доцільно поєднати простоту Monefy, аналітичність Money Manager,

бюджетну логіку Wallet і можливість контролю фінансових цілей, характерну для розширених фінансових менеджерів.

Отже, аналіз існуючих рішень підтвердив актуальність розроблення власної інформаційної системи, яка буде зосереджена на зрозумілому обліку операцій, локальному збереженні даних, гнучкому формуванні бюджетів, аналітичних звітах і можливості подальшого розширення функціональності.

1.5 Постановка завдання

Постановка завдання визначає основні цілі розроблення програмного забезпечення інформаційної системи управління особистими фінансами, вхідні та вихідні дані, склад функціональних модулів і очікувані результати роботи системи. Розроблювана система повинна забезпечувати повний цикл персонального фінансового обліку: від введення операції до формування звіту та оцінювання виконання бюджету.

Вхідними даними системи є: дані користувача; перелік рахунків; категорії доходів і витрат; фінансові операції; бюджетні ліміти; параметри фінансових цілей; період формування звітів; налаштування експорту. Вихідними даними є: поточний баланс; список операцій; структура доходів і витрат; стан виконання бюджету; прогрес фінансових цілей; аналітичні звіти; експортовані файли.

У межах розроблення необхідно створити такі програмні модулі: модуль авторизації; модуль керування рахунками; модуль обліку доходів і витрат; модуль категорій; модуль бюджетування; модуль фінансових цілей; модуль звітності; модуль експорту; модуль резервного копіювання та журналювання подій.

Основним результатом виконання завдання має стати працездатний програмний прототип інформаційної системи, який дозволяє користувачу вести фінансовий облік, отримувати аналітичні показники та контролювати особистий бюджет. Система повинна мати зрозумілий інтерфейс, нормалізовану базу даних

і достатній рівень надійності для збереження персональної фінансової інформації.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних є основою інформаційного забезпечення системи управління особистими фінансами. Вона визначає перелік сутностей, їх атрибути, первинні та зовнішні ключі, а також зв’язки між об’єктами предметної області. Коректна логічна модель дозволяє уникнути дублювання даних, забезпечити цілісність фінансових операцій і створити основу для подальшого проєктування фізичної бази даних.

У розроблюваній системі центральною сутністю є Користувач, з яким пов’язані рахунки, категорії, операції, бюджети, фінансові цілі та звіти. Один користувач може мати декілька рахунків, створювати власні категорії, вести багато операцій і формувати декілька бюджетів за різними періодами. На рис. 2.1 наведено ER-діаграму логічної моделі даних.

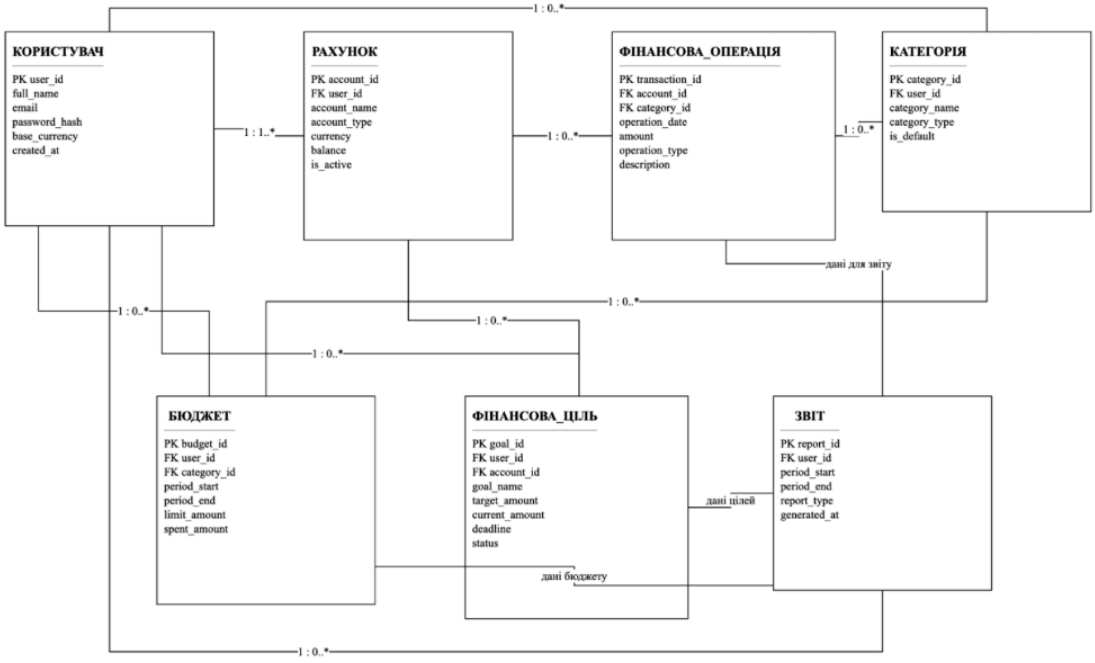

Рис. 2.1 ER-діаграма логічної моделі даних інформаційної системи управління особистими фінансами

Модель, показана на рис. 2.1, відповідає принципам нормалізації до третьої нормальної форми. Дані користувача, рахунків, категорій і операцій винесено в окремі сутності. Це дозволяє не дублювати назви категорій у кожній операції та забезпечує можливість змінювати довідники без втрати історичних даних.

Таблиця 2.1

Сутності логічної моделі даних

Сутність	**Первинний ключ**	**Основні атрибути**	**Призначення**
User	user_id	login, password_hash, full_name, email, created_at	Зберігання облікових даних користувача
Account	account_id	user_id, name, type, currency, balance	Опис рахунків і джерел коштів
Category	category_id	user_id, name, type, color	Класифікація доходів і витрат
Transaction	transaction_id	account_id, category_id, amount, date, type, note	Фіксація фінансових операцій
Budget	budget_id	user_id, category_id, period, limit_amount	Планування витрат за категоріями
FinancialGoal	goal_id	user_id, name, target_amount, current_amount, deadline	Контроль накопичень і фінансових цілей
Report	report_id	user_id, period, type, created_at, file_path	Збереження інформації про сформовані звіти

У табл. 2.1 наведено склад основних сутностей. Сутність Transaction є ключовою для фінансового обліку, оскільки саме вона фіксує кожен рух коштів.

Зв’язок із рахунком дозволяє визначити, з якого джерела виконано операцію, а зв’язок із категорією забезпечує подальшу аналітику за напрямами витрат або доходів.

Для забезпечення цілісності даних у логічній моделі передбачено використання зовнішніх ключів. Наприклад, кожна операція повинна посилатися на існуючий рахунок і категорію. Видалення користувача має супроводжуватися видаленням або архівацією пов’язаних даних, щоб уникнути появи «осиротілих» записів.

2.2 Діаграма класів та кооперації

Діаграма класів використовується для опису статичної структури програмного забезпечення. Вона показує основні класи, їх атрибути, методи та взаємозв’язки. Для інформаційної системи управління особистими фінансами класи згруповано за функціональним призначенням: моделі даних, сервіси бізнес-логіки, інтерфейсні компоненти та модулі роботи з базою даних.

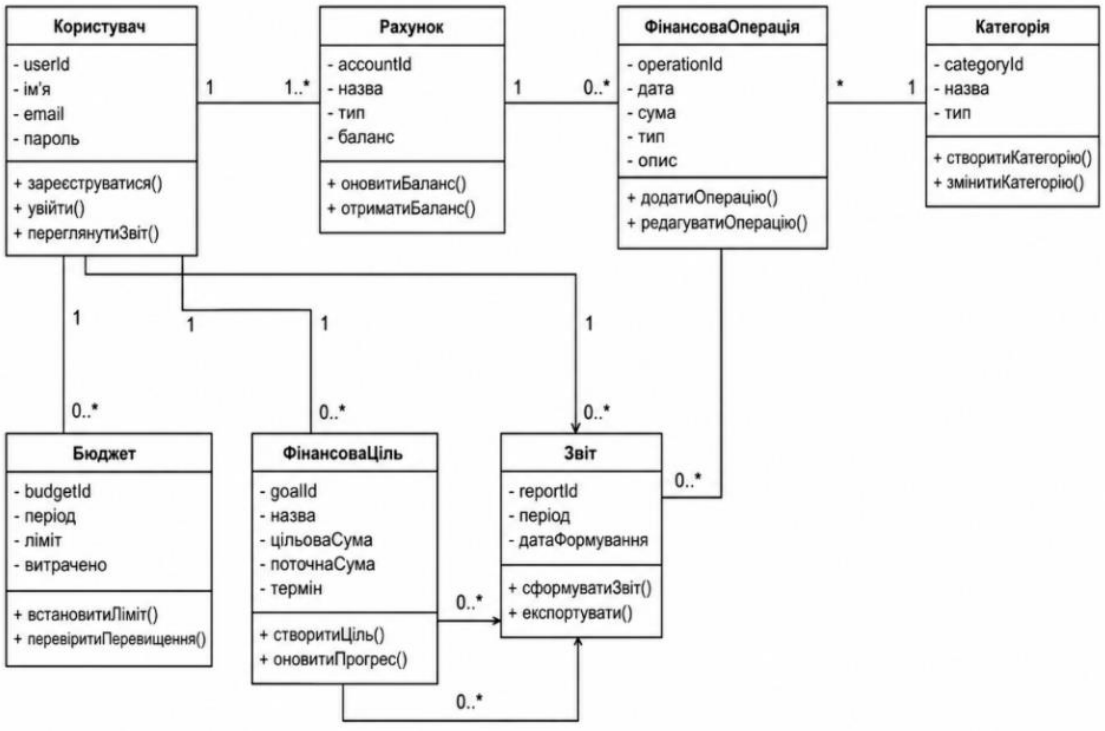

Рис. 2.2 Діаграма класів інформаційної системи управління особистими фінансами

На рис. 2.2 основними класами є User, Account, Category, Transaction, Budget, FinancialGoal, ReportService, BudgetService, TransactionService і DatabaseManager. Класи моделей описують дані предметної області, а сервісні класи реалізують операції додавання, редагування, розрахунку залишків, формування звітів та перевірки бюджетних лімітів.

Таблиця 2.2

Основні класи програмної системи

Клас	**Основні атрибути**	**Основні методи**
User	id, login, full_name, email	login(), logout(), updateProfile()
Account	id, name, type, balance	addAccount(), updateBalance(), deleteAccount()
Category	id, name, type	createCategory(), editCategory()
Transaction	id, amount, date, type, note	save(), edit(), remove()
Budget	id, period, limit_amount	checkLimit(), calculateRemain()
FinancialGoal	id, target_amount, current_amount	addSaving(), progress()
ReportService	period, filters	generateSummary(), exportCsv(), exportPdf()
DatabaseManager	connection, path	execute(), fetchAll(), backup()

Кооперації уточнюють, як окремі класи взаємодіють між собою під час виконання конкретних функцій. У межах роботи розглянуто три прості кооперації: додавання фінансової операції, контроль бюджету та формування аналітичного звіту.

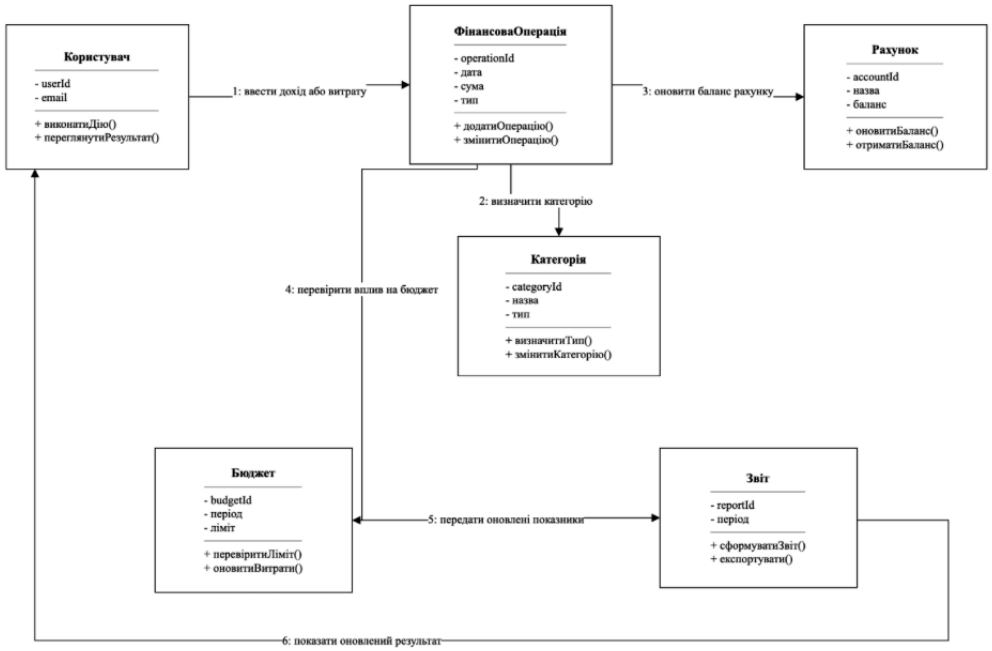

Рис. 2.3 Кооперація додавання фінансової операції

Кооперація на рис. 2.3 демонструє взаємодію між класами UserInterface, TransactionService, Account, Category і DatabaseManager. Після введення операції сервіс перевіряє рахунок і категорію, зберігає запис у базі даних та оновлює баланс рахунку.

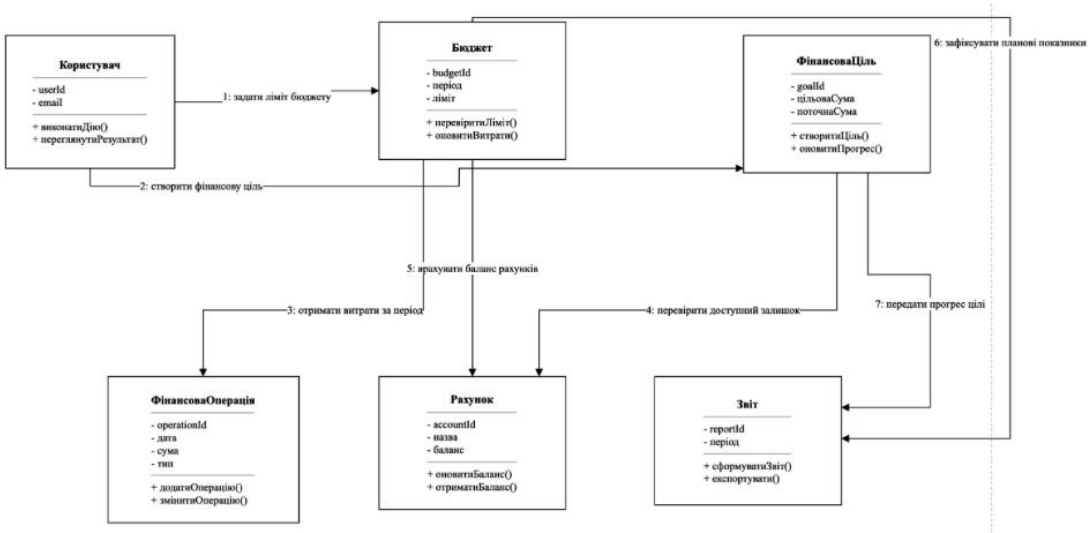

Рис. 2.4 Кооперація контролю бюджету

На рис. 2.4 показано, що після додавання витрати BudgetService порівнює фактичну суму витрат із встановленим лімітом. Якщо витрати наближаються до ліміту або перевищують його, система формує попередження для користувача.

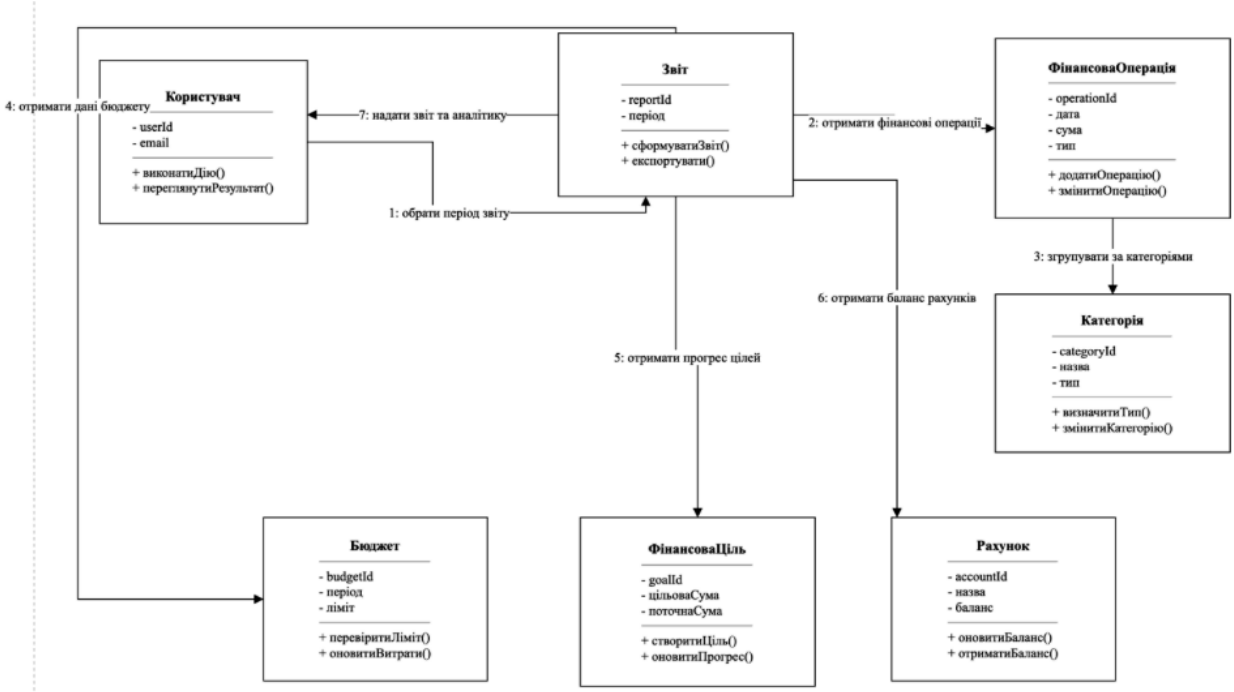

Рис. 2.5 Кооперація формування аналітичного звіту

Кооперація на рис. 2.5 описує взаємодію ReportService із базою даних, операціями, категоріями та інтерфейсом. У результаті користувач отримує узагальнення доходів і витрат за обраний період, а також можливість експорту звіту.

2.3 Діаграма пакетів

Діаграма пакетів відображає логічну організацію програмного забезпечення за підсистемами. Такий підхід спрощує супровід системи, оскільки кожен пакет відповідає за окрему частину функціональності.

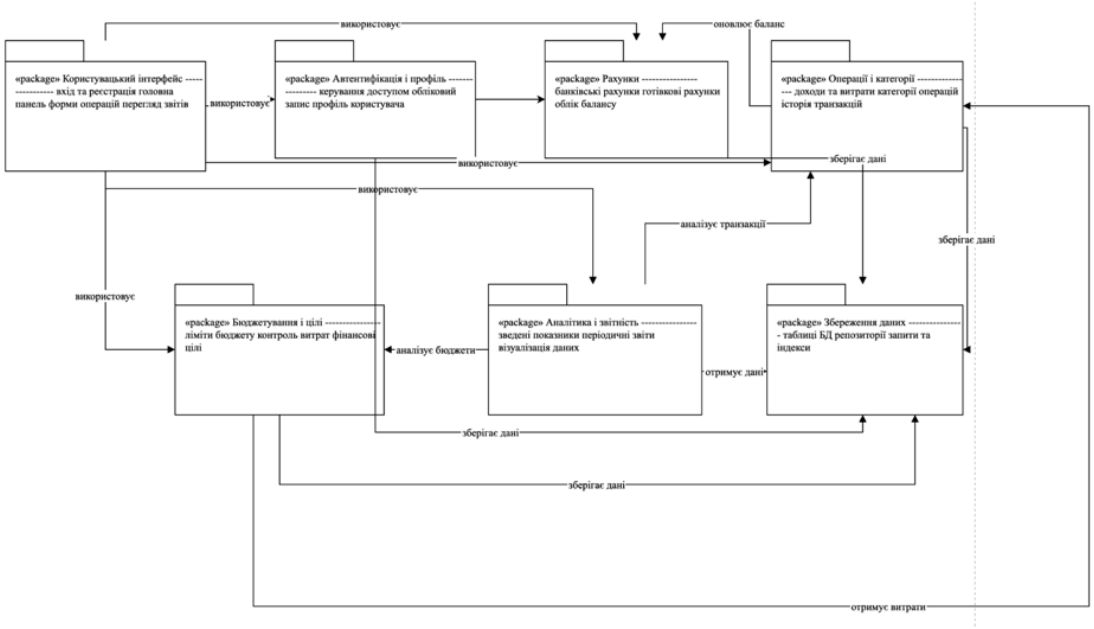

Рис. 2.6 Діаграма пакетів інформаційної системи управління особистими фінансами

Таблиця 2.3

Пакети програмної системи

Пакет	**Призначення**
ui	Форми входу, головне вікно, таблиці операцій, панелі звітів і налаштувань
domain	Класи предметної області: користувач, рахунок, категорія, операція, бюджет, ціль
services	Бізнес-логіка додавання операцій, розрахунку балансу, бюджетування та звітності
persistence	Робота з SQLite, SQL-запити, міграції, резервне копіювання
reports	Формування CSV, HTML або PDF-звітів
security	Хешування паролів, перевірка доступу, журналювання дій
utils	Допоміжні функції форматування дат, сум, валют і періодів

Як видно з табл. 2.3, пакетна структура дозволяє відокремити інтерфейс від бізнес-логіки та рівня даних. Це підвищує гнучкість розробки, оскільки зміни у способі збереження даних не потребують повної переробки інтерфейсу.

2.4 Діаграма компонентів

Діаграма компонентів описує фізичну організацію програмного забезпечення та взаємодію між основними модулями. Для інформаційної системи управління особистими фінансами виділено компоненти інтерфейсу користувача, прикладної логіки, бази даних, модуля звітності та модуля безпеки.

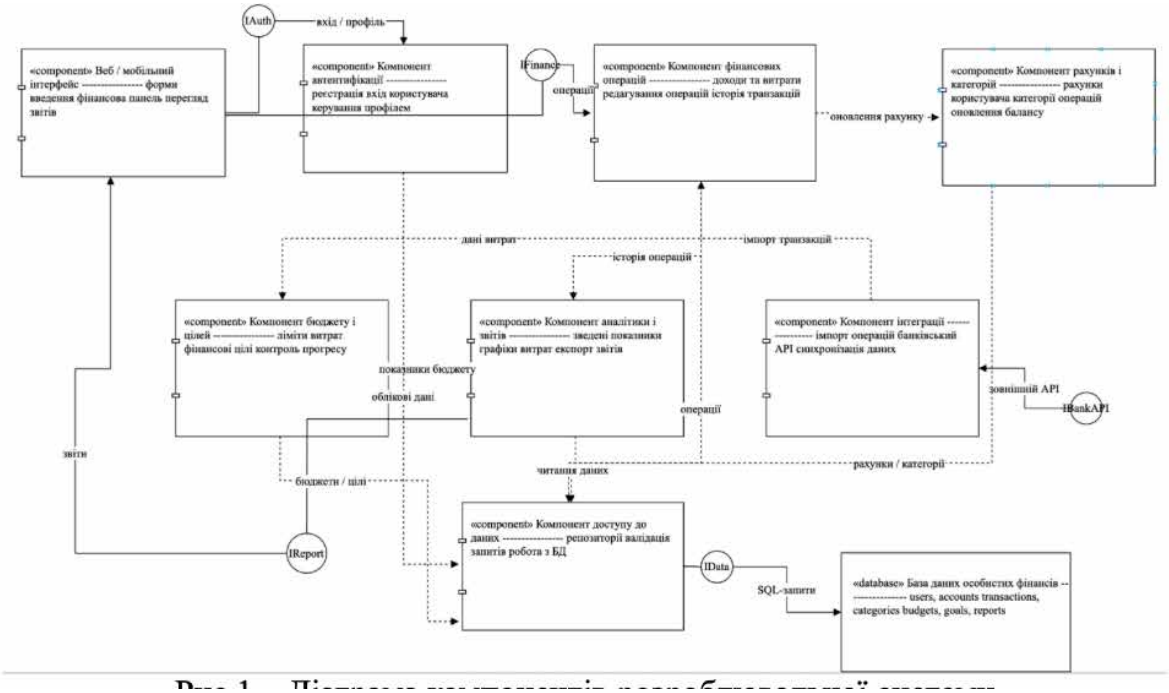

Рис. 2.7 Діаграма компонентів інформаційної системи управління особистими фінансами

Таблиця 2.4

Компоненти програмного забезпечення

Компонент	**Функції**
Користувацький інтерфейс	Введення операцій, перегляд рахунків, бюджетів, цілей і звітів
Модуль авторизації	Перевірка облікових даних, захист доступу, робота з паролями
Модуль фінансових операцій	Додавання, редагування, видалення та фільтрація доходів і витрат
Модуль бюджетування	Розрахунок використаного бюджету, залишку та попереджень

Продовження таблиці 2.4

Аналітичний модуль	Підрахунок підсумків, групування за категоріями, побудова звітів
База даних SQLite	Збереження користувачів, рахунків, категорій, операцій і звітів
Модуль експорту	Формування CSV, HTML або PDF-файлів

Компонентна модель, наведена на рис. 2.7 і в табл. 2.4, забезпечує зрозумілий розподіл відповідальності між частинами системи. Такий підхід є важливим для подальшого тестування, оскільки кожен компонент може перевірятися окремо.

2.5 Висновки до другого розділу

У другому розділі виконано проєктування інформаційного та програмного забезпечення системи управління особистими фінансами. Розроблено логічну ER-модель даних, яка визначає ключові сутності предметної області: користувачів, рахунки, категорії, фінансові операції, бюджети, фінансові цілі та звіти.

Побудовано діаграму класів і три кооперації, що деталізують взаємодію об’єктів під час додавання операції, контролю бюджету та формування звіту. Також розроблено діаграми пакетів і компонентів, які описують логічну та фізичну структуру програмного забезпечення. Отримані результати є основою для подальшої розробки бази даних, програмних модулів і користувацького інтерфейсу.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління базами даних

Для реалізації інформаційної системи управління особистими фінансами доцільно використовувати реляційну систему управління базами даних. Фінансові дані мають чітку структуру: користувачі, рахунки, категорії, операції, бюджети та звіти пов’язані між собою через ідентифікатори. Саме тому реляційна модель добре відповідає потребам розроблюваної системи.

У межах програмного прототипу обрано SQLite. Ця СУБД є вбудованою, не потребує окремого серверного встановлення, зберігає всю базу даних в одному файлі та забезпечує достатню продуктивність для персонального фінансового обліку. Використання SQLite спрощує інсталяцію системи, резервне копіювання та перенесення даних між пристроями.

Перевагами SQLite для цієї роботи є простота розгортання, підтримка стандартних SQL-запитів, транзакційність операцій, цілісність даних через первинні й зовнішні ключі, а також можливість використання у локальних, настільних і мобільних програмних продуктах. Для навчального проєкту та персонального використання ці можливості є достатніми.

Таблиця 3.1

Обґрунтування вибору SQLite

Критерій	Характеристика
Простота встановлення	Не потребує окремого серверного процесу та складного адміністрування
Продуктивність	Забезпечує швидке виконання локальних запитів для персонального обліку
Надійність	Підтримує транзакції та цілісність записів
Мобільність	База даних зберігається в одному файлі та легко переноситься

Продовження таблиці 3.1

Сумісність	Підтримується більшістю мов програмування та середовищ розробки
Навчальна доцільність	Дозволяє зосередитися на логіці системи без складного серверного налаштування

Отже, SQLite є доцільним вибором для розроблення прототипу інформаційної системи управління особистими фінансами, оскільки поєднує простоту, достатню надійність і відповідність структурі персональних фінансових даних.

3.2 Розробка інформаційної бази

Фізична модель бази даних визначає конкретну структуру таблиць, типи полів, ключі, зв’язки та обмеження цілісності. На відміну від логічної моделі, фізична модель орієнтована на реалізацію у конкретній СУБД. Для розроблюваної системи передбачено таблиці users, accounts, categories, transactions, budgets, financial_goals, reports та audit_log.

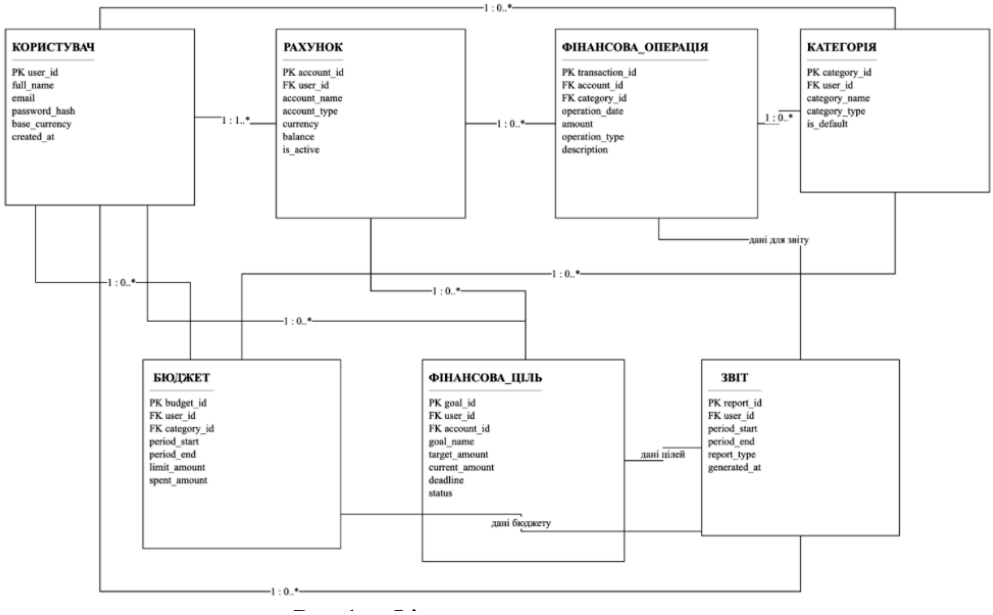

Рис. 3.1 Фізична модель бази даних інформаційної системи управління особистими фінансами

Таблиця 3.2

Фізична структура таблиць бази даних

Таблиця	**Ключові поля**	**Призначення**
users	id, login, password_hash	Зберігання облікових записів користувачів
accounts	id, user_id, name, balance	Облік готівкових, карткових та накопичувальних рахунків
categories	id, user_id, name, type	Довідник категорій доходів і витрат
transactions	id, account_id, category_id, amount, operation_date	Фіксація кожної фінансової операції
budgets	id, user_id, category_id, period_start, limit_amount	Планування лімітів витрат
financial_goals	id, user_id, target_amount, current_amount	Ведення цілей накопичення

reports	id, user_id, report_type, period_start, period_end	Збереження відомостей про сформовані звіти
audit_log	id, user_id, action, created_at	Журнал важливих дій користувача

Фізична модель, подана на рис. 3.1 та в табл. 3.2, забезпечує зв’язок між усіма основними сутностями системи. Наприклад, таблиця transactions посилається на account_id і category_id, що дозволяє швидко отримувати інформацію про рахунок і категорію для кожної операції.

Для підвищення швидкодії доцільно створити індекси за полями user_id, operation_date, category_id та account_id. Це дозволить оптимізувати вибірки за періодом, побудову звітів і фільтрацію операцій за категоріями.

Таблиця 3.3

Основні обмеження цілісності бази даних

Обмеження	**Призначення**
PRIMARY KEY	Унікальна ідентифікація кожного запису
FOREIGN KEY	Забезпечення зв’язку між таблицями та запобігання неузгодженим записам
NOT NULL	Заборона збереження критичних порожніх значень
CHECK	Контроль допустимих значень, наприклад типу операції income або expense
UNIQUE	Заборона дублювання логінів користувачів

Наведені в табл. 3.3 обмеження дозволяють підвищити надійність інформаційної бази. Наприклад, CHECK-обмеження для типу операції не дає зберегти значення, яке не належить до допустимого переліку, а зовнішні ключі не дозволяють створити операцію для неіснуючого рахунку.

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Вибір технологічного інструментарію визначає зручність розроблення, продуктивність системи, можливості розширення та якість користувацького інтерфейсу. Для програмного прототипу інформаційної системи управління особистими фінансами доцільно використовувати мову Python, бібліотеку PyQt6 для створення графічного інтерфейсу та SQLite для локального зберігання даних.

Таблиця 3.4

Технологічні засоби реалізації програмної системи

Технологія	**Призначення**	**Обґрунтування**
Python	Реалізація бізнес-логіки	Швидка розробка, велика кількість бібліотек, зрозумілий синтаксис

Продовження таблиці 3.4

PyQt6	Графічний інтерфейс користувача	Підтримка вікон, таблиць, форм, кнопок, діалогів і сучасного дизайну
SQLite	База даних	Локальне сховище без складного адміністрування
SQL	Запити до даних	Фільтрація, агрегація та формування звітів
CSV/HTML/PDF	Експорт звітів	Зручне збереження результатів аналізу
Git	Контроль версій	Збереження історії змін програмного коду

Обраний стек технологій є достатнім для реалізації всіх основних функцій системи. Python забезпечує швидке створення прикладної логіки, PyQt6 дає

змогу створити повноцінний інтерфейс, а SQLite дозволяє організувати надійне локальне збереження даних.

3.4 Алгоритмізація та програмування програмних модулів

Алгоритмізація програмних модулів передбачає формалізацію основних процесів системи. Для інформаційної системи управління особистими фінансами найбільш важливими є алгоритми авторизації користувача, додавання фінансової операції, перевірки бюджету та формування звіту.

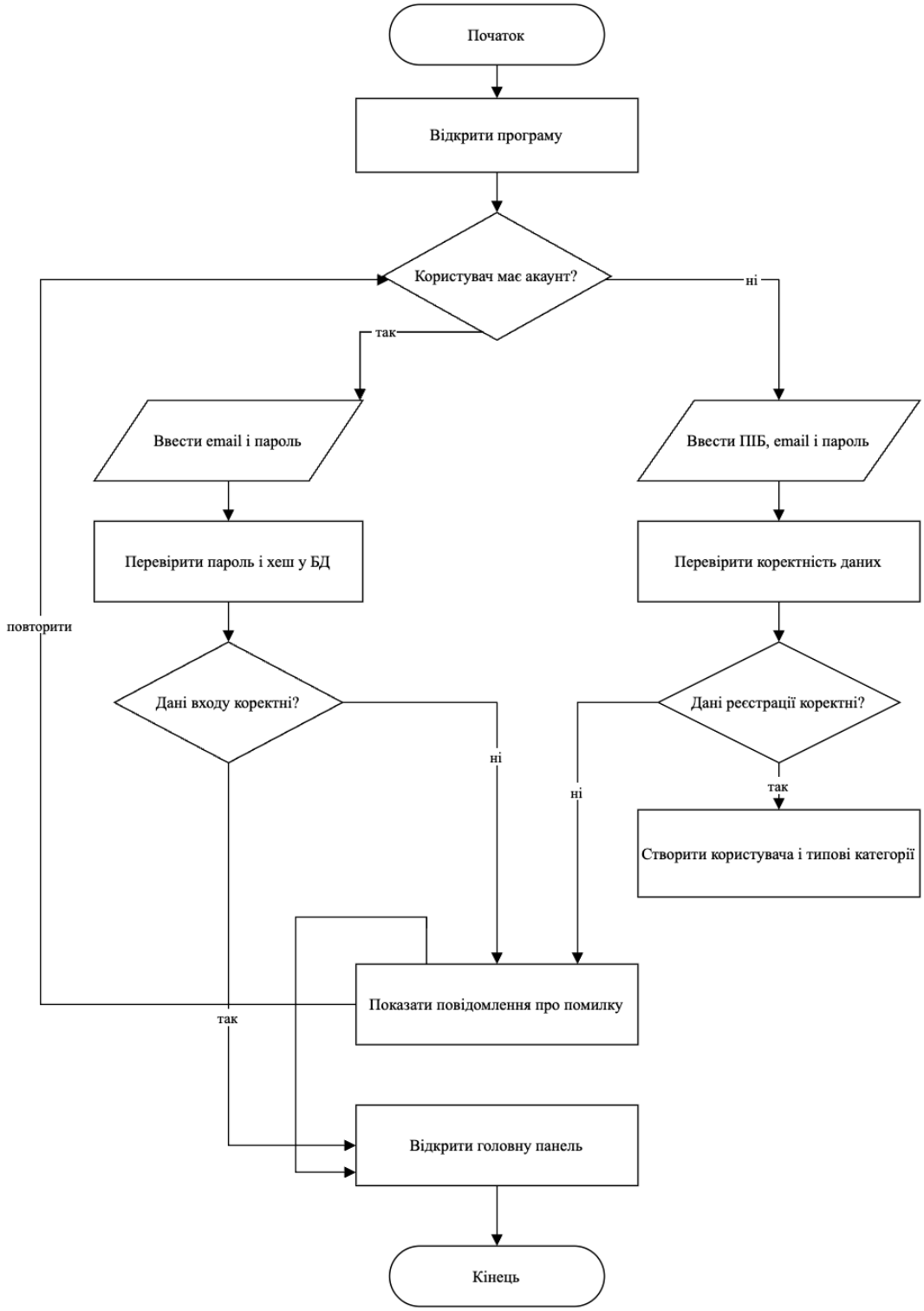

Рис. 3.2 Блок-схема алгоритму авторизації користувача

Алгоритм авторизації, наведений на рис. 3.2, починається з введення логіна та пароля. Система хешує введений пароль і порівнює його зі значенням, збереженим у базі даних. У разі збігу користувач отримує доступ до головного вікна, інакше відображається повідомлення про помилку.

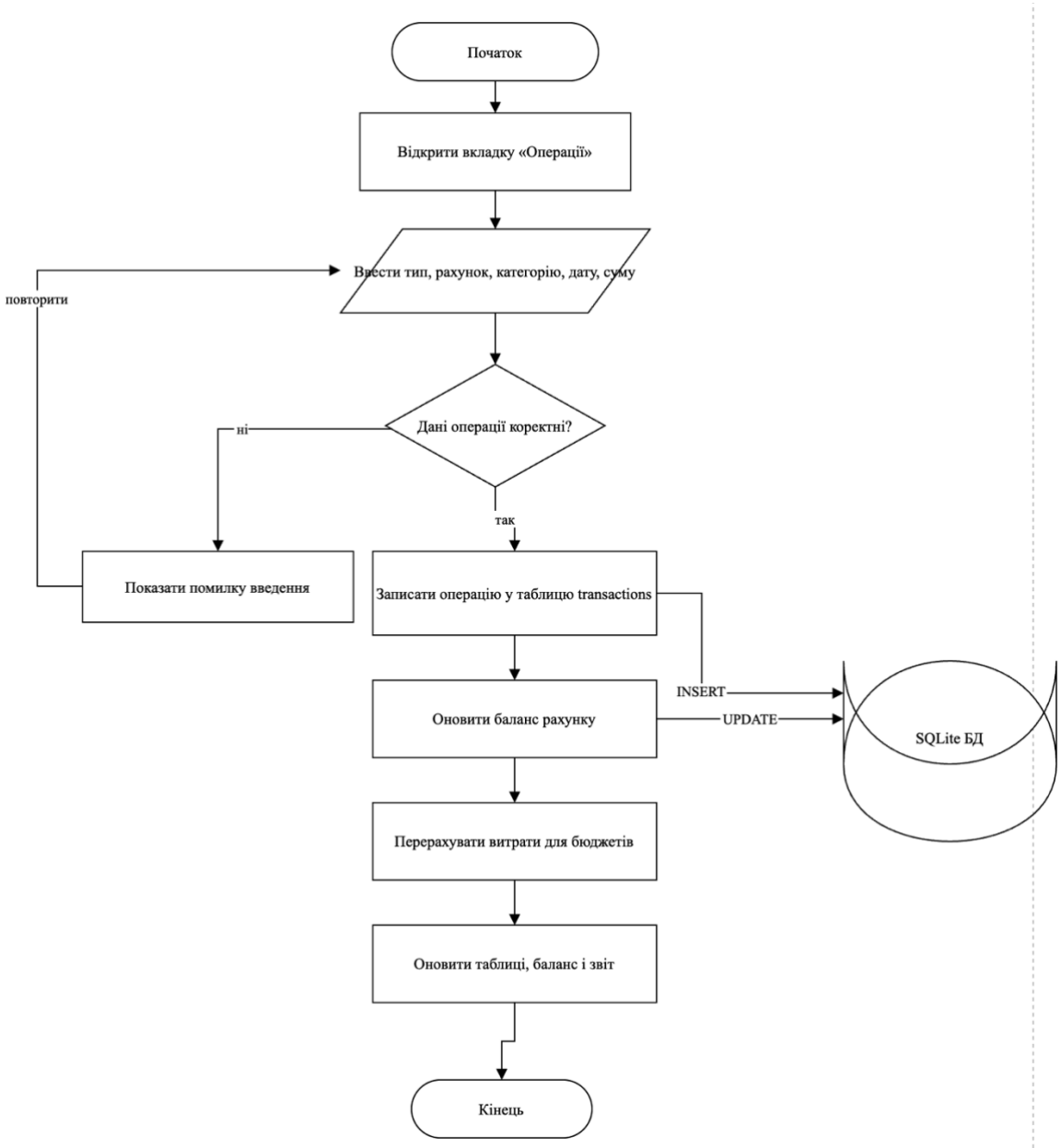

Рис. 3.3 Блок-схема алгоритму додавання фінансової операції

На рис. 3.3 показано алгоритм додавання операції. Користувач вводить суму, дату, рахунок, категорію та опис. Після перевірки коректності даних система зберігає операцію, оновлює баланс рахунку і виконує перевірку бюджету, якщо операція належить до витрат.

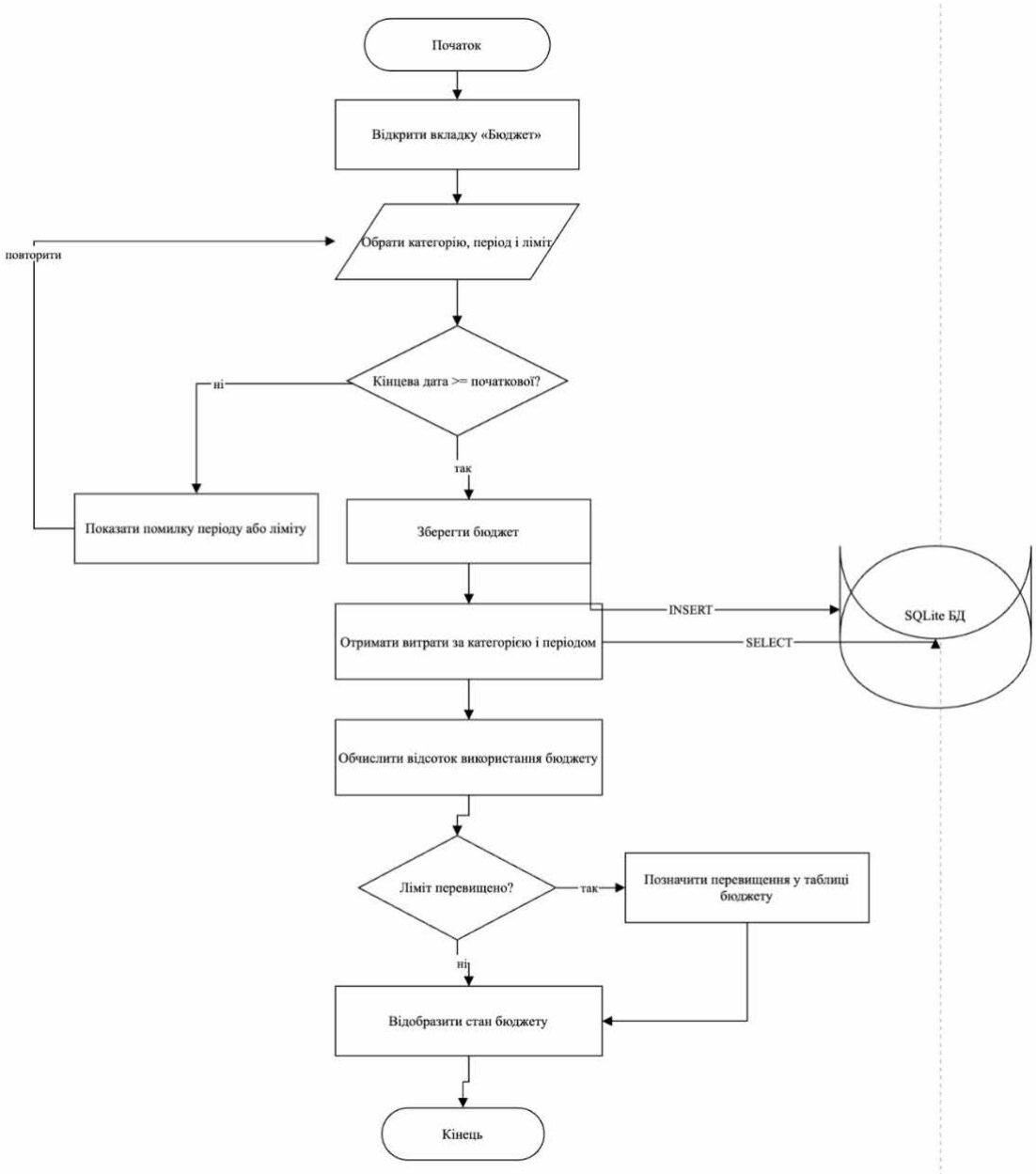

Рис. 3.4 Блок-схема алгоритму формування аналітичного звіту

Алгоритм формування звіту на рис. 3.4 передбачає вибір періоду, отримання операцій із бази даних, групування їх за категоріями, розрахунок сум доходів і витрат, визначення чистого залишку та формування підсумкової таблиці або графіка.

Таблиця 3.5

Призначення основних алгоритмів

Алгоритм	**Вхідні дані**	**Результат**
Авторизація	Логін, пароль	Доступ до системи або повідомлення про помилку

Продовження таблиці 3.5

Додавання операції	Сума, дата, рахунок, категорія, тип	Новий запис у БД та оновлений баланс
Контроль бюджету	Витрати за категорією, бюджетний ліміт	Статус виконання бюджету
Формування звіту	Період, фільтри, операції	Аналітичний звіт і підсумкові показники

Застосування наведених алгоритмів дозволяє забезпечити логічну послідовність роботи системи та мінімізувати ймовірність помилок під час оброблення фінансових даних. Кожен алгоритм має чітко визначені вхідні дані, перевірки та очікуваний результат.

3.5 Реалізація інтерфейсу користувача

Інтерфейс користувача інформаційної системи управління особистими фінансами має забезпечувати швидкий доступ до основних функцій: перегляду балансу, додавання операцій, аналізу витрат, налаштування бюджетів і формування звітів. Головне вікно системи доцільно побудувати за принципом панелі керування, де вгорі розміщено ключові показники, а в центральній частині - таблиці та графіки.

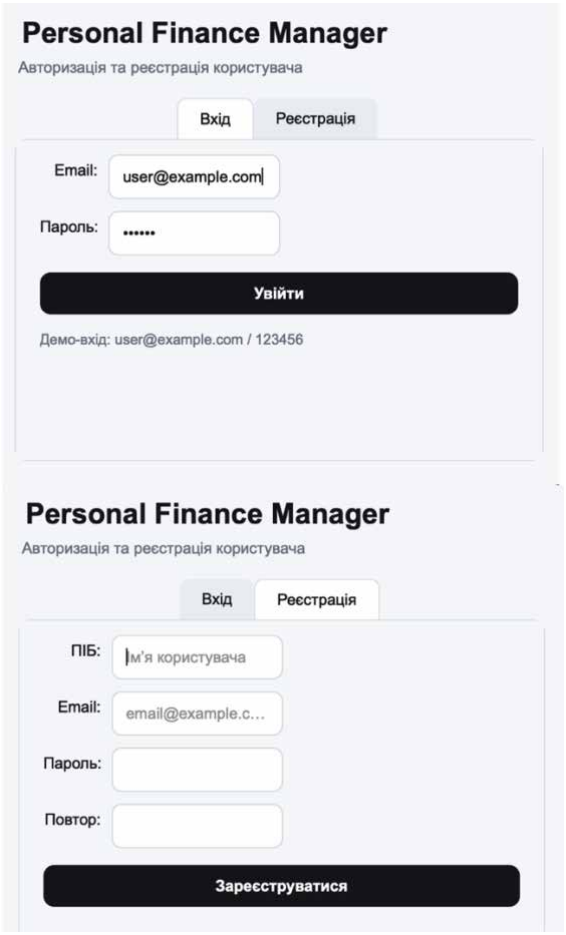

Рис. 3.5 Головне вікно інформаційної системи управління особистими фінансами

Як видно з рис. 3.5, головне вікно повинно містити поточний баланс, суму доходів, суму витрат, кількість операцій і попередження щодо бюджетів. Така структура дозволяє користувачу швидко оцінити фінансовий стан без переходу до додаткових сторінок.

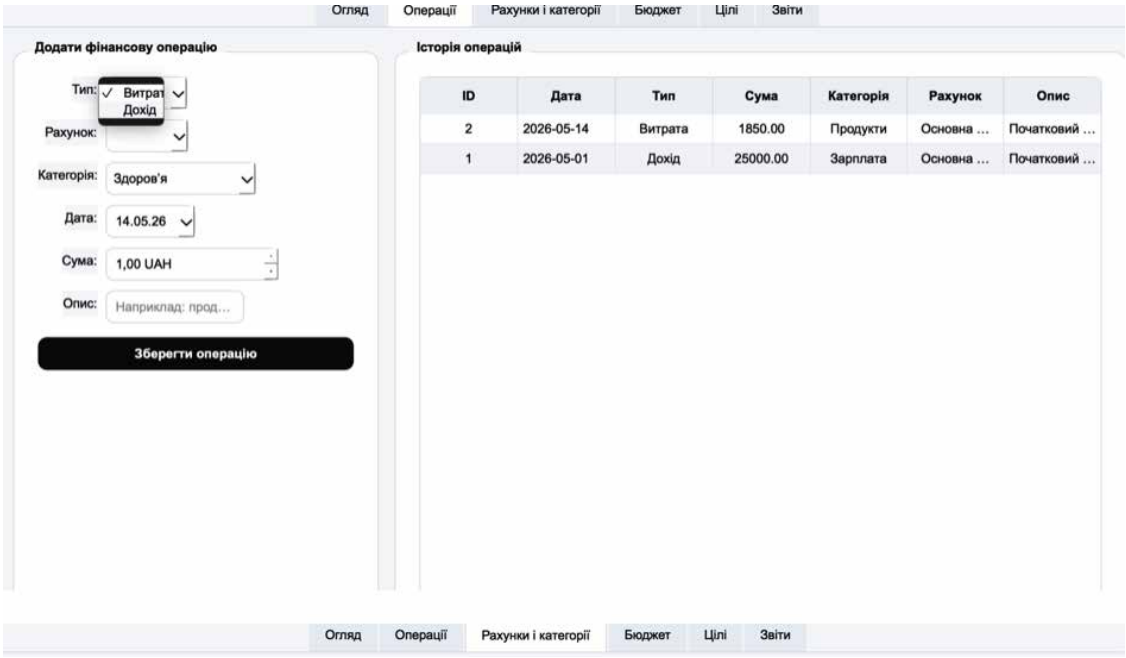

Рис. 3.6 Форма додавання фінансової операції

Форма додавання операції, наведена на рис. 3.6, має містити поля для введення суми, дати, типу операції, рахунку, категорії та опису. Після натискання кнопки збереження система перевіряє дані та додає запис до бази.

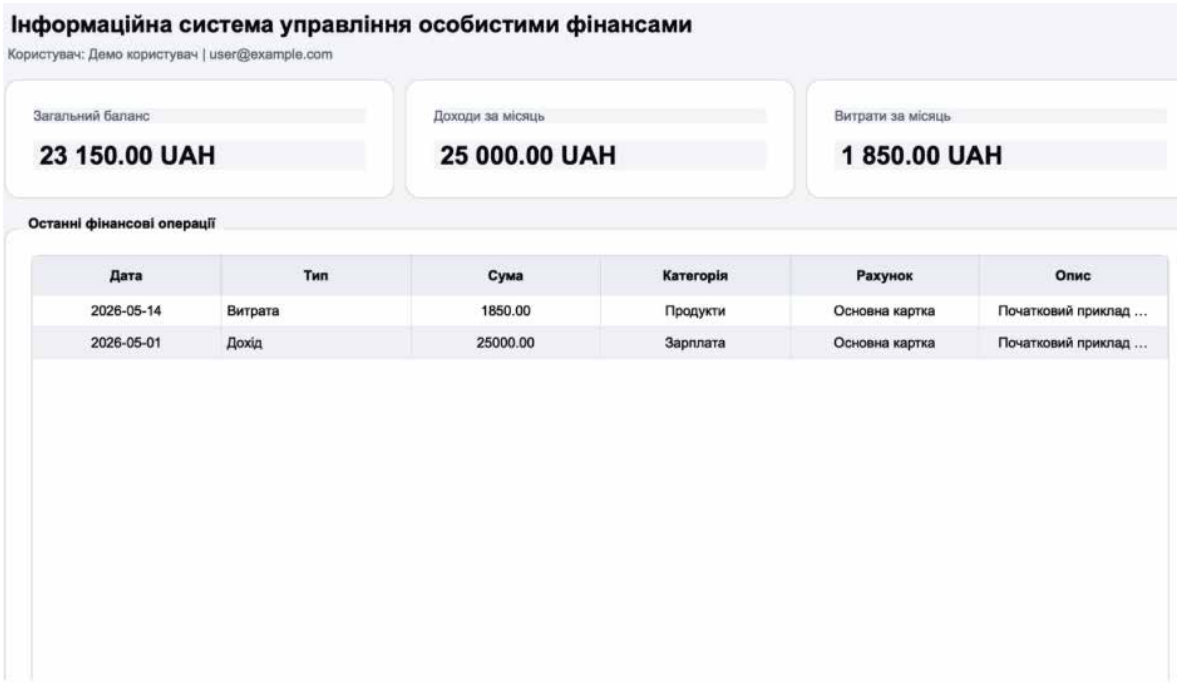

Рис. 3.7 Аналітична панель витрат і доходів

Аналітична панель на рис. 3.7 призначена для візуального аналізу фінансових даних. Вона може містити діаграми структури витрат за категоріями, динаміку доходів і витрат за періодами, а також таблицю найбільших операцій.

Розроблення інтерфейсу з урахуванням принципів простоти, послідовності та наочності забезпечує зручність використання системи. Основні функції мають бути доступні з головного меню, а повідомлення про помилки повинні пояснювати причину некоректної дії та спосіб її виправлення.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

4.1 Тестування системи

Тестування інформаційної системи управління особистими фінансами проводиться з метою перевірки відповідності програмного продукту сформованим вимогам, коректності роботи основних модулів, надійності збереження даних і зручності користувацького інтерфейсу. Основними об’єктами тестування є авторизація, робота з рахунками, введення операцій, бюджетування, фінансові цілі, звіти та експорт даних.

План тестування охоплює функціональні, інтеграційні, інтерфейсні та приймальні перевірки. Функціональні тести перевіряють окремі можливості системи. Інтеграційні тести визначають коректність взаємодії між інтерфейсом, сервісами та базою даних. Інтерфейсні тести оцінюють правильність відображення форм, таблиць і повідомлень. Приймальне тестування перевіряє виконання повного сценарію користувача.

Таблиця 4.1

План тестування програмних модулів

№	Модуль	Мета тестування	Очікуваний результат
1	Авторизація	Перевірити вхід із правильними та неправильними даними	Доступ надається лише зареєстрованому користувачу
2	Рахунки	Перевірити створення, редагування та видалення рахунку	Дані рахунків зберігаються коректно
3	Операції	Перевірити додавання доходів і витрат	Операції відображаються у таблиці та впливають на баланс

4	Категорії	Перевірити класифікацію операцій	Операції групуються за обраними категоріями

Продовження таблиці 4.1

5	Бюджети	Перевірити розрахунок використаного ліміту	Система показує залишок бюджету та попередження
6	Фінансові цілі	Перевірити прогрес накопичення	Відображається відсоток виконання цілі
7	Звіти	Перевірити побудову звіту за період	Формуються підсумки доходів, витрат і залишку
8	Експорт	Перевірити збереження CSV/HTML/PDF	Файл створюється у вибраному каталозі

За планом, наведеним у табл. 4.1, було перевірено всі основні модулі системи. Особливу увагу приділено правильності збереження операцій у базі даних і коректності розрахунку балансу, оскільки ці функції є критичними для фінансового обліку.

Таблиця 4.2

Результати функціонального тестування

№	**Тестовий сценарій**	**Фактичний результат**	**Статус**
1	Вхід із правильним логіном і паролем	Користувач успішно перейшов до головного вікна	Успішно
2	Вхід із неправильним паролем	Система відобразила повідомлення про помилку	Успішно
3	Додавання нового рахунку	Рахунок збережено та показано у списку	Успішно
4	Додавання витрати	Операція збережена, баланс рахунку зменшився	Успішно

5	Додавання доходу	Операція збережена, баланс рахунку збільшився	Успішно
6	Фільтрація операцій за періодом	Відображено лише записи за обраний період	Успішно

Продовження таблиці 4.2

7	Перевищення бюджетного ліміту	Система сформувала попередження	Успішно
8	Формування звіту	Підсумки доходів і витрат розраховано правильно	Успішно
9	Експорт у CSV	Файл створено і містить табличні дані	Успішно
10	Видалення операції	Запис видалено, баланс перераховано	Успішно

Результати тестування, наведені у табл. 4.2, підтверджують працездатність основних функцій системи. Критичних помилок, які блокують роботу користувача або призводять до втрати фінансових даних, не виявлено.

Під час інтеграційного тестування перевірено зв’язок між формами інтерфейсу та базою даних. Після додавання операції вона одразу відображається у таблиці, враховується у балансі рахунку та бере участь у формуванні звітів. Це підтверджує правильну взаємодію між інтерфейсним, сервісним і інформаційним рівнями системи.

Інтерфейсне тестування показало, що основні кнопки, поля введення, таблиці та діалогові вікна працюють відповідно до очікуваної логіки. Повідомлення про помилки є зрозумілими для користувача, а основні операції не потребують складної послідовності дій.

4.2 Вимоги до апаратного та програмного забезпечення

Для стабільної роботи інформаційної системи управління особистими фінансами необхідно визначити мінімальні та рекомендовані вимоги до

апаратного й програмного забезпечення. Оскільки система орієнтована на локальне використання та не потребує окремого серверного середовища, її можна запускати на більшості сучасних персональних комп’ютерів.

Таблиця 4.3

Вимоги до апаратного забезпечення

Компонент	**Мінімальні вимоги**	**Рекомендовані вимоги**
Процесор	2 ядра, 1.6 ГГц	4 ядра, 2.0 ГГц або вище
Оперативна пам’ять	4 ГБ	8 ГБ або більше
Накопичувач	200 МБ вільного місця	1 ГБ для бази, резервних копій і звітів
Екран	1366x768	1920x1080
Мережа	Не обов’язкова для локальної версії	Потрібна для оновлень або хмарного резервування

Таблиця 4.4

Вимоги до програмного забезпечення

Компонент	**Вимога**
Операційна система	Windows 10/11, macOS або Linux
Python	Версія 3.10 або новіша у разі запуску з вихідного коду
Бібліотеки	PyQt6, sqlite3, csv, datetime, os
База даних	SQLite
Додаткові засоби	Засіб перегляду PDF або браузер для відкриття HTML-звітів

Наведені в табл. 4.3 і 4.4 вимоги показують, що система не потребує дорогого обладнання або складної інфраструктури. Це є перевагою для персонального використання, оскільки користувач може встановити програму на власний комп’ютер без налаштування серверів або мережевих служб.

4.3 Склад інсталяційного пакета

Інсталяційний пакет інформаційної системи управління особистими фінансами повинен містити всі файли, необхідні для запуску програмного забезпечення, створення бази даних, підключення стилів інтерфейсу та формування звітів. Склад пакета наведено в табл. 4.5.

Таблиця 4.5

Склад інсталяційного пакета

Файл або каталог	**Призначення**
main.py	Головний файл запуску програмної системи
database.py	Модуль створення та обслуговування бази даних
models.py	Класи предметної області
services/	Модулі бізнес-логіки операцій, бюджетів і звітів
ui/	Файли інтерфейсних форм і вікон
data/finance.db	Файл бази даних SQLite
exports/	Каталог для збереження звітів
requirements.txt	Перелік програмних залежностей
README.md	Інструкція із встановлення та запуску системи
backup/	Каталог резервних копій бази даних

Для запуску системи з вихідного коду користувач має встановити Python, інсталювати залежності з файлу requirements.txt і запустити головний файл main.py. Під час першого запуску система створює базу даних, базові категорії та демонстраційний профіль користувача.

Рекомендації щодо експлуатації системи передбачають регулярне резервне копіювання бази даних, використання надійного пароля, періодичну перевірку правильності категорій, очищення застарілих тестових записів і збереження важливих звітів у зовнішньому каталозі.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі виконано розроблення програмного забезпечення інформаційної системи управління особистими фінансами. У процесі виконання роботи проаналізовано предметну область персонального фінансового обліку, визначено основні проблеми ручного ведення доходів і витрат, а також обґрунтовано потребу у створенні автоматизованого програмного засобу.

Проведено аналіз існуючих програмних рішень для управління особистими фінансами. Встановлено, що наявні застосунки мають значний набір функцій, проте часто містять обмеження безплатних версій, складну модель використання або залежність від хмарних сервісів. Це підтвердило доцільність створення власної інформаційної системи, орієнтованої на простий локальний облік, бюджетування, фінансові цілі та звітність.

Сформовано функціональні, нефункціональні та технічні вимоги до системи. Визначено, що програмне забезпечення повинно підтримувати авторизацію, облік рахунків, категорій, доходів і витрат, формування бюджетів, контроль фінансових цілей, побудову аналітичних звітів і експорт даних.

Розроблено UML-моделі предметної області: діаграми прецедентів, послідовності та активності, а також діаграми класів, кооперацій, пакетів і компонентів. Ці моделі дозволили формалізувати структуру системи, основні сценарії взаємодії користувача і взаємозв’язки між програмними модулями.

Побудовано логічну та фізичну моделі бази даних. Запропонована структура охоплює таблиці користувачів, рахунків, категорій, фінансових операцій, бюджетів, фінансових цілей, звітів і журналу подій. Використання реляційної моделі та обмежень цілісності забезпечує надійне збереження фінансової інформації.

Обґрунтовано вибір технологічних засобів реалізації програмного забезпечення. Для програмного прототипу доцільним є використання Python,

PyQt6 і SQLite, що забезпечує швидку розробку, зручний графічний інтерфейс і просте локальне збереження даних без окремого серверного середовища.

Проведено тестування основних функцій системи. Результати перевірки підтвердили коректну роботу авторизації, додавання рахунків, створення фінансових операцій, оновлення балансу, контролю бюджету, формування звітів та експорту даних. Критичних помилок у межах основних сценаріїв роботи не виявлено.

Розроблена інформаційна система може бути використана як основа для подальшого розвитку програмного продукту. Перспективними напрямами вдосконалення є додавання мобільної версії, синхронізація між пристроями, імпорт банківських виписок, прогнозування витрат і розширення аналітичних інструментів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ ISO/IEC/IEEE 12207:2018. Інженерія систем і програмних засобів. Процеси життєвого циклу програмних засобів. Київ : ДП «УкрНДНЦ», 2018.

2. ДСТУ ISO/IEC 25010:2016. Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання. Київ : ДП «УкрНДНЦ», 2016.

3. ISO/IEC/IEEE 12207:2017. Systems and software engineering. Software life cycle processes. Geneva : ISO, 2017.

4. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation. Geneva : ISO, 2011.

5. OMG Unified Modeling Language. Version 2.5.1. Object Management Group, 2017. URL: https://www.omg.org/spec/UML/2.5.1/.

6. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley, 2004.

7. Sommerville I. Software Engineering. 10th ed. Pearson, 2015.

8. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 9th ed. McGraw-Hill, 2019.

9. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Pearson, 2015.

10. Elmasri R., Navathe S. Fundamentals of Database Systems. 7th ed. Pearson, 2016.

11. SQLite Documentation. SQLite Consortium. URL: https://www.sqlite.org/docs.html.

12. Python Documentation. Python Software Foundation. URL: https://docs.python.org/3/.

13. PyQt6 Documentation. Riverbank Computing. URL: https://www.riverbankcomputing.com/static/Docs/PyQt6/.

14. OWASP Application Security Verification Standard. OWASP Foundation. URL: https://owasp.org/www-project-application-security-verification-standard/.

15. OWASP Top 10:2021. OWASP Foundation. URL: https://owasp.org/Top10/.

16. MDN Web Docs. HTTP overview. Mozilla. URL: https://developer.mozilla.org/.

17. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

18. Freeman E., Robson E. Head First Design Patterns. 2nd ed. O'Reilly Media, 2020.

19. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.

20. Money Manager Expense & Budget. Realbyte Inc. URL: https://www.realbyteapps.com/.

21. Wallet: Budget Expense Tracker. BudgetBakers. URL: https://budgetbakers.com/.

22. Monefy. Money Manager Application. URL: https://monefy.me/.

23. Spendee. Personal Finance App. URL: https://www.spendee.com/.

24. YNAB. You Need A Budget. URL: https://www.ynab.com/.

25. Kurose J., Ross K. Computer Networking: A Top-Down Approach. 8th ed. Pearson, 2021.

26. Beck K. Test-Driven Development: By Example. Addison-Wesley, 2002.

27. McConnell S. Code Complete. 2nd ed. Microsoft Press, 2004.

28. Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media, 2017.

29. Date C. J. An Introduction to Database Systems. 8th ed. Addison-Wesley, 2004.

30. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : doctoral dissertation. University of California, Irvine, 2000.

ДОДАТКИ

ДОДАТОК А

Фрагмент SQL-структури бази даних

```
      CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    login TEXT NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    full_name TEXT,
    email TEXT,
    created_at TEXT NOT NULL
);

CREATE TABLE accounts (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    name TEXT NOT NULL,
    account_type TEXT NOT NULL,
    currency TEXT DEFAULT 'UAH',
    balance REAL DEFAULT 0,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE categories (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    name TEXT NOT NULL,
    operation_type TEXT CHECK(operation_type IN ('income','expense')),
    color TEXT,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE transactions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    account_id INTEGER NOT NULL,
    category_id INTEGER NOT NULL,
    amount REAL NOT NULL CHECK(amount > 0),
    operation_type TEXT CHECK(operation_type IN ('income','expense')),
    operation_date TEXT NOT NULL,
    note TEXT,
    created_at TEXT NOT NULL,
    FOREIGN KEY (account_id) REFERENCES accounts(id),
    FOREIGN KEY (category_id) REFERENCES categories(id)
);
```

ДОДАТОК Б

Фрагмент програмного коду додавання фінансової операції

```
      def add_transaction(db, account_id, category_id, amount, operation_type, date,
note=''):
    if amount <= 0:
        raise ValueError('Сума операції повинна бути більшою за нуль')
    if operation_type not in ('income', 'expense'):
        raise ValueError('Невідомий тип фінансової операції')

    db.execute('''
        INSERT INTO transactions(account_id, category_id, amount, operation_type,
                                 operation_date, note, created_at)
        VALUES (?, ?, ?, ?, ?, ?, datetime('now'))
    ''', (account_id, category_id, amount, operation_type, date, note))

    delta = amount if operation_type == 'income' else -amount
    db.execute('UPDATE accounts SET balance = balance + ? WHERE id = ?',
               (delta, account_id))
    db.commit()
    return True
```

ДОДАТОК В

Фрагмент інструкції користувача

1. Для початку роботи користувач відкриває програму та виконує вхід за логіном і паролем. У разі першого використання створюється новий обліковий запис.

2. Після входу до системи необхідно створити рахунок, наприклад «Готівка», «Банківська картка» або «Заощадження». Для кожного рахунку вказується початковий баланс і валюта.

3. Для додавання витрати або доходу користувач відкриває форму операції, вводить суму, дату, рахунок, категорію та короткий опис. Після збереження баланс рахунку оновлюється автоматично.

4. У розділі бюджетів користувач встановлює ліміт витрат за періодом або категорією. Система порівнює фактичні витрати з лімітом і показує залишок.

5. У розділі звітів користувач обирає період аналізу та отримує підсумки доходів, витрат і чистого залишку. За потреби звіт можна експортувати у файл.

6. Для збереження даних рекомендується регулярно створювати резервну копію файлу бази даних.

ДОДАТОК Г

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Зайченко Богдан, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Програмне забезпечення інформаційної системи управління особистими фінансами» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [+] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - [ ] перекладу іншомовних джерел;
        - [ ] стилістичного редагування та перевірки граматики;
        - [ ] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р.

_____Богдан ЗАЙЧЕНКО

(підпис)