НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ Ігор БОЛБОТ	**_____ Белла ГОЛУБ**
"___" _____ 2026 р.	**"___" _____ 2026 р.**

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему " Програмне забезпечення для обліку побутових витрат з аналізом бюджету"

Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи
асистент _____ **Тетяна БАРАНОВА**
(підпис)

Консультант бакалаврської кваліфікаційної роботи
к.т.н., доцент _____ **Ірина БОРОДКІНА**
(підпис)

Виконав _____ **Артем ГАНОВСЬКИЙ**
(підпис)

КИЇВ – 2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ

"___" _____ 2026 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Гановському Артему Сергійовичу
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи
"Програмне забезпечення для обліку побутових витрат з аналізом бюджету"

затверджена наказом ректора від "19" грудня 2025р. № 3204 "С"

Термін подання завершеної роботи на кафедру " 22 " травня 2026р.

Вихідні дані до бакалаврської кваліфікаційної роботи:
Навчальні матеріали з розробки програмного забезпечення для обліку фінансів, роботи з базами даних, обробки та аналізу даних.
Перелік питань, що підлягають дослідженню:
Аналіз предметної області обліку побутових витрат та управління бюджетом.
Дослідження підходів до збереження, категоризації та аналізу фінансових операцій користувача.
Перелік графічних документів (за необхідності).

Дата видачі завдання " 19 " грудня 2025р.

Керівник бакалаврської кваліфікаційної роботи Асистент	_____ (підпис)	**Тетяна БАРАНОВА**
Консультант бакалаврської кваліфікаційної роботи к.т.н, доцент	_____ (підпис)	**Ірина БОРОДКІНА**
Завдання взяв до виконання	_____ (підпис)	**Артем ГАНОВСЬКИЙ**

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

– API – прикладний програмний інтерфейс (Application Programming Interface)
– CAN – контролерна мережа транспортного засобу (Controller Area Network)
– CSV – формат табличних даних із розділенням значень комами (Comma-Separated Values)
– DTC – діагностичний код несправності транспортного засобу (Diagnostic Trouble Code)
– DTO – об’єкт передавання даних між модулями програми (Data Transfer Object)
– ER-діаграма – діаграма «сутність–зв’язок» (Entity–Relationship Diagram)
– GPS – глобальна система позиціонування (Global Positioning System)
– GUI – графічний інтерфейс користувача (Graphical User Interface)
– IMEI – міжнародний ідентифікатор мобільного обладнання (International Mobile Equipment Identity)
– JSON – текстовий формат обміну структурованими даними (JavaScript Object Notation)
– KPI – ключовий показник ефективності (Key Performance Indicator)
– OBD-II – бортова система діагностики автомобіля другого покоління (On-Board Diagnostics II)
– PDF – формат електронного документа (Portable Document Format)
– PID – ідентифікатор параметра OBD-II (Parameter ID)
– PyQt6 – бібліотека Python для створення графічних інтерфейсів на основі Qt 6
– Qt – кросплатформний фреймворк для розроблення графічних інтерфейсів
– RBAC – керування доступом на основі ролей (Role-Based Access Control)
– RPM – кількість обертів двигуна за хвилину (Revolutions Per Minute)
– SQLite – вбудована файлова система керування базами даних
– SQL – мова структурованих запитів до баз даних (Structured Query Language)

– UML – уніфікована мова моделювання (Unified Modeling Language)

ВСТУП

Відсутність систематичного обліку доходів і витрат може призводити до нераціонального використання коштів, перевищення запланованого бюджету, несвоєчасного виявлення фінансових проблем і зниження ефективності планування особистих фінансів.

Особливо важливим є не лише збереження інформації про окремі фінансові операції, а й можливість її подальшого аналізу. Користувач повинен мати змогу бачити загальну суму доходів, структуру витрат за категоріями, залишок бюджету, рівень використання коштів, а також отримувати попередження у разі наближення до встановлених лімітів. Це дозволяє підвищити фінансову дисципліну, своєчасно коригувати витрати та приймати більш обґрунтовані рішення щодо планування бюджету.

Актуальність теми полягає у необхідності розроблення зручного програмного забезпечення для обліку побутових витрат з аналізом бюджету, яке забезпечує ведення доходів і витрат, створення бюджетів, керування категоріями, контроль лімітів, формування сповіщень та побудову звітів. Таке програмне забезпечення дає змогу автоматизувати процес фінансового обліку, зменшити кількість ручних розрахунків і забезпечити користувача актуальною інформацією про власний фінансовий стан.

Існуючі засоби обліку особистих фінансів не завжди повністю відповідають потребам користувачів. Частина з них має складний інтерфейс, частина орієнтована переважно на мобільні пристрої або потребує постійного підключення до мережі Інтернет. Крім того, не всі рішення забезпечують достатньо гнучке налаштування категорій, лімітів, звітів та локального збереження даних. Тому доцільним є створення програмного застосунку, який поєднує простоту використання, зрозумілу структуру даних, наявність аналітичних функцій і можливість подальшого розширення.

Метою роботи є розроблення програмного забезпечення для обліку побутових витрат з аналізом бюджету, яке забезпечує реєстрацію користувачів, ведення доходів і витрат, створення бюджетів, керування категоріями, контроль лімітів, формування сповіщень та отримання аналітичних звітів.

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

1. Проаналізувати предметну область обліку побутових доходів і витрат.
2. Визначити основні функціональні можливості програмного забезпечення.
3. Сформувати вимоги до системи з точки зору користувача та з точки зору програмної реалізації.
4. Розробити UML-діаграми, що відображають логіку роботи системи, взаємодію користувача з програмою та структуру основних компонентів.
5. Побудувати логічну модель даних для опису основних сутностей системи, зокрема користувачів, бюджетів, транзакцій, категорій, лімітів, звітів і сповіщень.
6. Розробити фізичну модель бази даних та SQL-код для створення таблиць, зв’язків, ключів, обмежень і початкових даних.
7. Реалізувати програмний застосунок з графічним інтерфейсом користувача.
8. Реалізувати модуль авторизації та реєстрації користувачів.
9. Реалізувати функції створення бюджетів, додавання доходів і витрат, керування категоріями та встановлення лімітів.
10. Реалізувати модуль аналізу бюджету, який розраховує загальні доходи, витрати, залишок коштів і відсоток використання бюджету.
11. Реалізувати механізм формування сповіщень у разі наближення до ліміту або його перевищення.
12. Реалізувати формування звітів та експорт фінансових операцій у файл.

13. Виконати тестування основних функцій програмного забезпечення та перевірити коректність роботи з базою даних.

Об’єктом дослідження є процес обліку побутових доходів і витрат користувача та управління особистим бюджетом.

Предметом дослідження є методи, моделі та програмні засоби розроблення інформаційної системи для фіксації фінансових операцій, аналізу бюджету, контролю лімітів і формування звітності.

У роботі використано методи об’єктно-орієнтованого аналізу та проєктування, UML-моделювання, проєктування реляційних баз даних, структурного аналізу вимог, алгоритмізації програмних модулів і тестування програмного забезпечення. Для реалізації програмного застосунку передбачено використання мови програмування Python, бібліотеки PyQt6 для створення графічного інтерфейсу користувача та SQLite для локального збереження даних.

Практичне значення роботи полягає у створенні програмного застосунку, який може використовуватися для щоденного ведення особистого фінансового обліку. Розроблена система дозволяє користувачу швидко вносити доходи й витрати, переглядати стан бюджету, аналізувати структуру витрат, контролювати встановлені ліміти та формувати звіти за вибраний період. Це сприяє підвищенню прозорості особистих фінансів, зменшенню ризику неконтрольованих витрат і покращенню планування бюджету.

Результатом виконання роботи є програмне забезпечення для обліку побутових витрат з аналізом бюджету, яке має модульну структуру та включає підсистеми авторизації, керування бюджетами, обліку транзакцій, роботи з категоріями, контролю лімітів, формування сповіщень, аналітики та звітності. Запропоноване рішення може бути використане як основа для подальшого вдосконалення, зокрема додавання синхронізації даних, розширеної аналітики, багатокористувацького режиму або інтеграції з банківськими сервісами.

Структурно робота складається зі вступу, основних розділів, висновків, списку використаних джерел і додатків. У першому розділі розглядається предметна область та вимоги до системи. У другому розділі подано

моделювання програмного забезпечення, зокрема UML-діаграми, логічну та фізичну модель даних. У третьому розділі описується реалізація програмного застосунку, база даних, інтерфейс користувача та основні програмні модулі. У четвертому розділі наведено тестування системи, аналіз результатів її роботи та оцінювання коректності виконання основних функцій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область програмного забезпечення для обліку побутових витрат з аналізом бюджету охоплює процеси фіксації доходів і витрат користувача, створення особистого бюджету, групування фінансових операцій за категоріями, контролю встановлених лімітів, формування сповіщень та підготовки звітів. Основне призначення такої системи полягає в тому, щоб надати користувачу зручний інструмент для щоденного контролю власних фінансів і швидкого отримання узагальненої інформації про стан бюджету.

У межах предметної області користувач створює бюджет на певний період, наприклад на тиждень, місяць або інший заданий проміжок часу. До бюджету прив’язуються фінансові операції, які поділяються на доходи та витрати. Кожна операція має суму, дату, опис і категорію. Категорії використовуються для групування фінансових даних за економічним змістом, наприклад продукти, транспорт, комунальні послуги, заробітна плата, подарунки або інші надходження.

Важливою частиною системи є контроль лімітів. Для окремих категорій витрат користувач може встановити допустиму суму, після досягнення якої система формує відповідне сповіщення. Це дає змогу своєчасно виявляти перевищення витрат і коригувати фінансову поведінку. Крім того, система повинна забезпечувати формування звітів, у яких узагальнюються доходи, витрати, залишок коштів, структура операцій і рівень використання бюджету.

Узагальнену структуру основних компонентів предметної області наведено в таблиці 1.1.

Таблиця 1.1

Основні компоненти предметної області

Компонент предметної області	Призначення
Користувач	Створює бюджети, додає доходи й витрати, переглядає аналітику, сповіщення та звіти
Адміністратор	Керує службовими налаштуваннями, категоріями, лімітами та контролює коректність роботи системи
Бюджет	Визначає планову суму коштів, період фінансового обліку, залишок і загальний стан виконання бюджету
Операція	Фіксує факт надходження коштів або здійснення витрати із зазначенням суми, дати, опису та категорії
Категорія	Групує доходи й витрати за економічним змістом, наприклад продукти, транспорт, комунальні платежі або заробітна плата
Ліміт	Визначає допустиму суму витрат для певної категорії в межах вибраного бюджету
Сповіщення	Інформує користувача про наближення до встановленого ліміту або його перевищення
Звіт	Узагальнює фінансові показники за вибраний період і відображає доходи, витрати, залишок та структуру операцій

Як видно з таблиці 1.1, предметна область має чітку логічну структуру, у якій кожний компонент виконує окрему функцію. Користувач є головним учасником системи, бюджет виступає центральним об’єктом обліку, операції фіксують рух коштів, а категорії, ліміти, сповіщення та звіти забезпечують аналітичну й контрольну складові програмного забезпечення.

Дані в системі взаємопов’язані між собою. Кожний бюджет належить конкретному користувачу, операції прив’язуються до бюджету та категорії, а звіти формуються на основі збережених транзакцій. Ліміти встановлюються для окремих категорій витрат у межах певного бюджету, а сповіщення створюються тоді, коли фактичні витрати наближаються до заданого обмеження або перевищують його. Така структура забезпечує цілісність інформації та створює основу для подальшого проєктування логічної і фізичної моделей бази даних.

1.2 Аналіз існуючих рішень у сфері обліку побутових витрат

Для визначення функціональних можливостей розроблюваного програмного забезпечення доцільно проаналізувати існуючі рішення у сфері обліку побутових витрат та керування особистим бюджетом. У межах аналізу розглянуто такі програмні продукти: Money Manager Expense & Budget, Monefy, AndroMoney, Wallet by BudgetBakers та Money Lover. Ці застосунки мають подібне призначення, однак відрізняються структурою інтерфейсу, способом введення операцій, рівнем аналітики, підтримкою бюджетів, лімітів, звітів і синхронізації даних. [26; 27; 28]

Money Manager Expense & Budget є одним із найбільш функціональних застосунків для ведення особистих фінансів. Його інтерфейс орієнтований на детальний перегляд щоденних операцій, доходів, витрат і підсумкового балансу за вибраний період, що показано на рисунку 1.1. У системі передбачено перегляд тижневих і місячних підсумків, бюджетування, фільтрацію транзакцій, календар операцій, діаграми витрат, збереження фото чеків і роботу з активами. Перевагою Money Manager є висока деталізація фінансового обліку, однак для частини користувачів інтерфейс може бути перевантаженим через значну кількість функцій і режимів перегляду. [26]

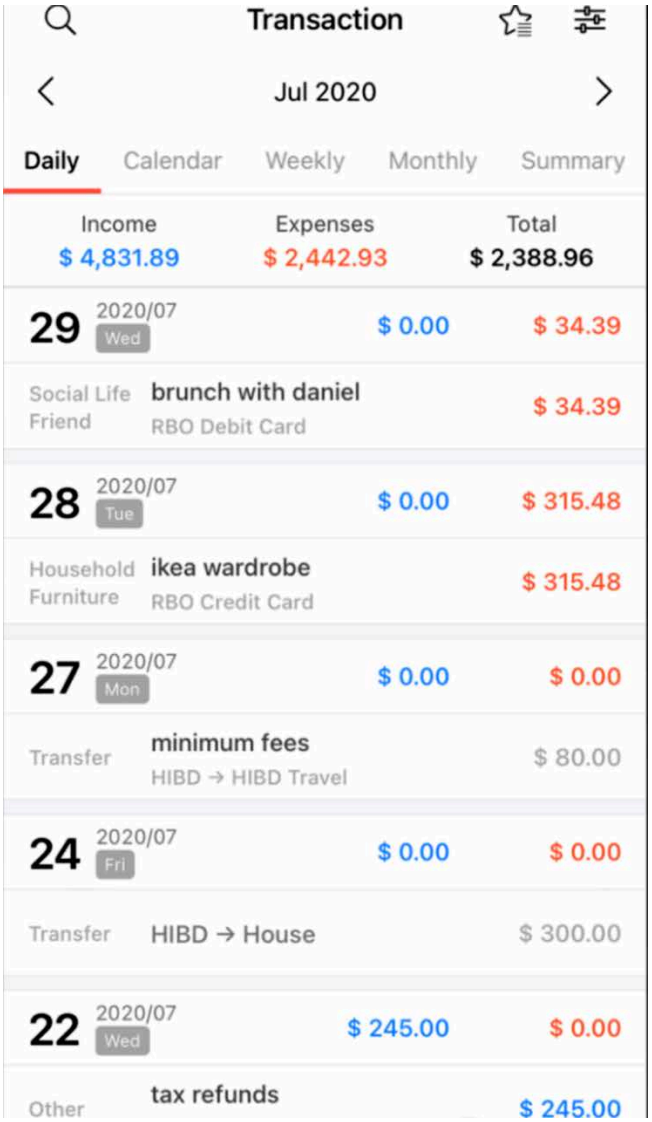


Рис. 1.1 – Інтерфейс Money Manager Expense & Budget для перегляду щоденних фінансових операцій

Monefy є простішим рішенням, основний акцент якого зроблено на швидкому додаванні доходів і витрат. Інтерфейс застосунку побудований навколо категорій, балансу та двох основних дій: додавання доходу або витрати. Приклад такого підходу наведено на рисунку 1.2. Перевагою Monefy є зручність щоденного використання, мінімальна кількість зайвих елементів і швидкий доступ до основних фінансових категорій. Водночас така простота обмежує можливості глибокого аналізу бюджету, контролю лімітів, формування розгорнутих звітів і роботи зі складнішою структурою даних. [27]

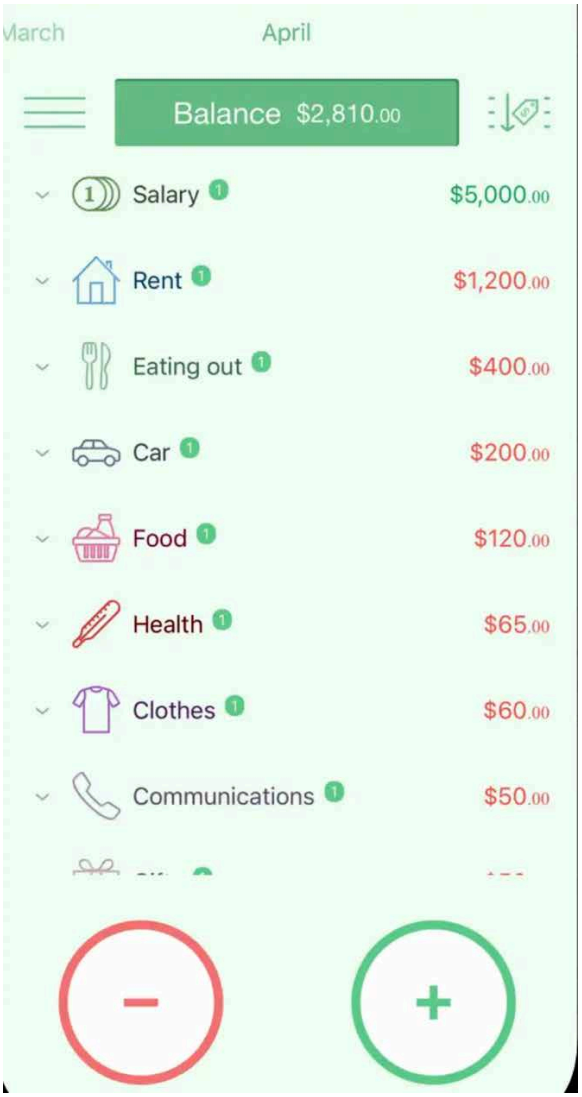


Рис. 1.2 – Інтерфейс Monefy для категоризованого обліку доходів і витрат

AndroMoney орієнтований на більш комплексний облік особистих фінансів. Застосунок підтримує кілька рахунків, перекази між ними, різні валюти, ієрархічні категорії, прості та деталізовані бюджети, синхронізацію між пристроями, резервне копіювання, захист паролем, а також побудову кругових, стовпчикових і трендових діаграм. На рисунку 1.3 показано приклад інтерфейсу, де поєднано облік рахунків і перегляд структури витрат. Перевагою AndroMoney є широка функціональність, однак для розроблюваної системи така кількість можливостей є частково надлишковою, оскільки основний акцент має бути зроблений на побутових витратах, бюджеті, лімітах і звітах.

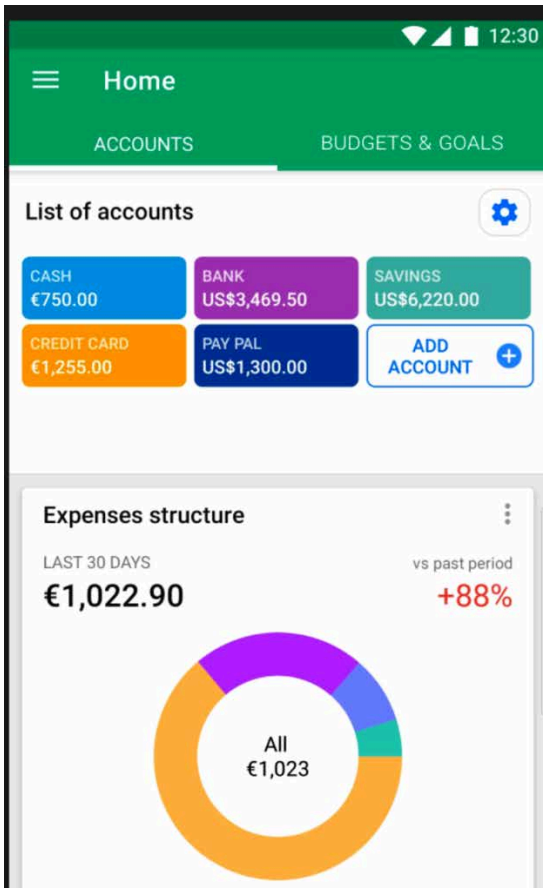


Рис. 1.3 – Інтерфейс AndroMoney для перегляду рахунків і структури витрат

Wallet by BudgetBakers є сучасним багатоплатформним рішенням для персонального фінансового планування. Система підтримує ручне введення операцій, банківську синхронізацію, автоматичну категоризацію транзакцій, створення бюджетів, фінансові цілі, статистику, звіти, спільне ведення фінансів і роботу з різними пристроями. Приклад аналітичного інтерфейсу Wallet наведено на рисунку 1.4. Сильною стороною цього рішення є розвинена аналітика та інтеграція з банківськими сервісами. Недоліком для навчального програмного проєкту є залежність частини функцій від зовнішніх сервісів, облікового запису та синхронізації даних. [28]

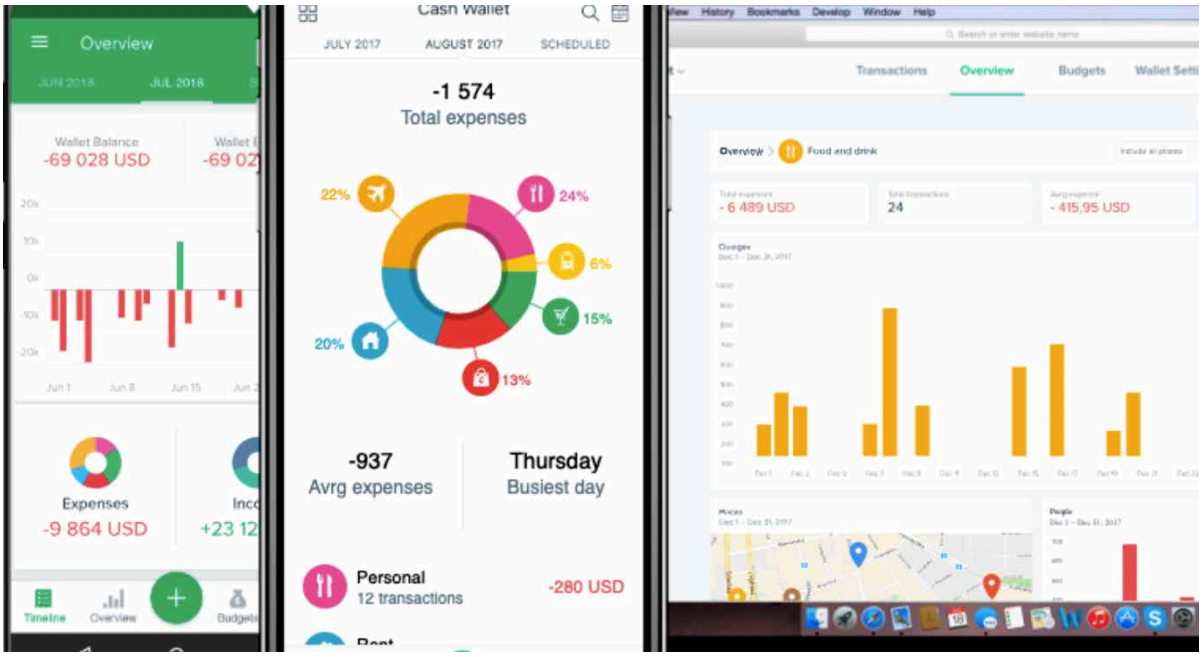


Рис. 1.4 – Інтерфейси Wallet by BudgetBakers для перегляду аналітики витрат

Money Lover є застосунком для відстеження щоденних операцій, категоризації доходів і витрат, створення бюджетів, перегляду звітів, синхронізації між пристроями та роботи з регулярними платежами. У системі реалізовано візуальне подання фінансових показників, що дає змогу користувачу швидко оцінити поточний стан особистих фінансів. Приклад узагальненого фінансового екрана наведено на рисунку 1.5. Перевагою Money Lover є поєднання простого обліку з бюджетуванням і звітністю, однак частина можливостей, зокрема необмежена кількість бюджетів, гаманців, експорт CSV та доступ до вебверсії, може залежати від преміум-режиму.

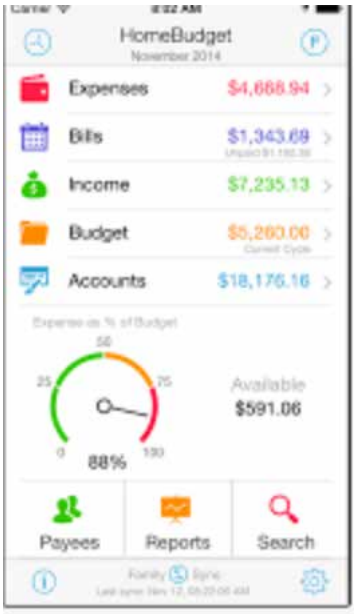


Рис. 1.5 – Інтерфейс Money Lover для перегляду узагальненого фінансового стану

Порівняльну характеристику розглянутих програмних рішень наведено в таблиці 1.2.

Таблиця 1.2

Порівняльна характеристика існуючих рішень у сфері обліку побутових витрат

Програмне рішення	Основні функції	Переваги	Недоліки
Money Manager Expense & Budget	Облік доходів і витрат, бюджетування, календар операцій, фільтри, діаграми, облік активів	Детальний журнал транзакцій, зручний перегляд за днями й місяцями, підтримка бюджетів і графіків	Інтерфейс може бути перевантаженим для користувача, якому потрібен простий побутовий облік
Monefy	Швидке додавання доходів і витрат, категоризація операцій, перегляд балансу	Простий інтерфейс, швидке внесення операцій, зручне групування витрат за категоріями	Обмежені можливості глибокої аналітики, звітності та контролю лімітів
AndroMoney	Кілька рахунків, бюджети, валюти, діаграми, синхронізація, резервне копіювання	Розвинений облік рахунків, підтримка складної структури категорій, наявність графіків і бюджетів	Надлишкова функціональність для простої локальної системи обліку побутових витрат

Продовження таблиці 1.2

Wallet by BudgetBakers	Банківська синхронізація, ручне введення операцій, бюджети, цілі, статистика, звіти	Сильна аналітика, автоматична категоризація, підтримка різних пристроїв і банківських рахунків	Залежність частини функцій від зовнішніх сервісів, синхронізації та облікового запису
Money Lover	Облік операцій, категорії, бюджети, регулярні платежі, звіти, синхронізація, експорт	Поєднання простого обліку, бюджетування та візуальної звітності	Частина розширених можливостей доступна лише в преміум-режимі

Як видно з таблиці 1.2, кожне з розглянутих рішень має власну логіку організації фінансового обліку. Money Manager Expense & Budget найбільше підходить для детального ведення журналу транзакцій, Monefy є зручним для швидкого щоденного внесення операцій, AndroMoney орієнтований на роботу з кількома рахунками та складнішою структурою фінансів, Wallet by BudgetBakers має розвинену синхронізацію й аналітику, а Money Lover поєднує бюджетування, регулярні платежі та звітність.

Проведений аналіз дозволяє визначити функції, які доцільно реалізувати у розроблюваному програмному забезпеченні. Із Money Manager доцільно врахувати детальний журнал операцій і перегляд фінансових показників за періодами. Із Monefy варто використати ідею швидкого додавання доходів і витрат через простий інтерфейс. Із AndroMoney доцільно взяти підтримку категорій, бюджетів і графічного аналізу. Із Wallet by BudgetBakers варто врахувати підхід до аналітики, звітів і контролю фінансових показників. Із Money Lover доцільно використати поєднання бюджетування, сповіщень і зрозумілого узагальненого фінансового екрана.

Отже, аналіз конкретних існуючих рішень показав, що розроблюване програмне забезпечення повинно поєднувати простоту щоденного введення операцій, зрозумілу категоризацію доходів і витрат, створення бюджетів, контроль лімітів, формування сповіщень і підготовку звітів. На відміну від частини розглянутих аналогів, основний акцент у системі доцільно зробити на

локальному збереженні даних, простому інтерфейсі, цілісній базі даних і функціях, безпосередньо пов’язаних з обліком побутових витрат та аналізом бюджету.

1.3 Моделювання предметної області

Для формалізації предметної області використано UML-моделювання. Воно дозволяє представити систему з різних точок зору: функціональної, поведінкової та структурної. У межах роботи побудовано діаграму прецедентів, діаграму послідовності та діаграму активності, які відображають основні сценарії взаємодії користувача й адміністратора із системою. [5; 11]

Діаграма прецедентів наведена на рисунку 1.6. Вона відображає межі системи обліку побутових витрат, зовнішніх акторів і перелік функцій, доступних кожному з них. Основним актором є користувач, який авторизується, веде доходи і витрати, переглядає бюджет та аналітику, а також формує звіти. Адміністратор має додаткові можливості керування категоріями, лімітами і службовою звітністю.

Рис. 1.6 – Діаграма прецедентів інформаційної системи обліку побутових витрат

Як видно з рисунка 1.1, функціональність системи згрупована навколо процесів авторизації, ведення операцій, аналізу бюджету, формування звітності

та адміністративного керування. Така модель є достатньо компактною і водночас охоплює всі основні сценарії роботи програмного забезпечення.

На рисунку 1.7 наведено діаграму послідовності, яка деталізує взаємодію користувача та адміністратора під час основного сценарію роботи із системою. У моделі показано надсилання запиту на авторизацію, підтвердження доступу, передавання даних про доходи і витрати, запит перегляду бюджету, формування аналітики та службової звітності.

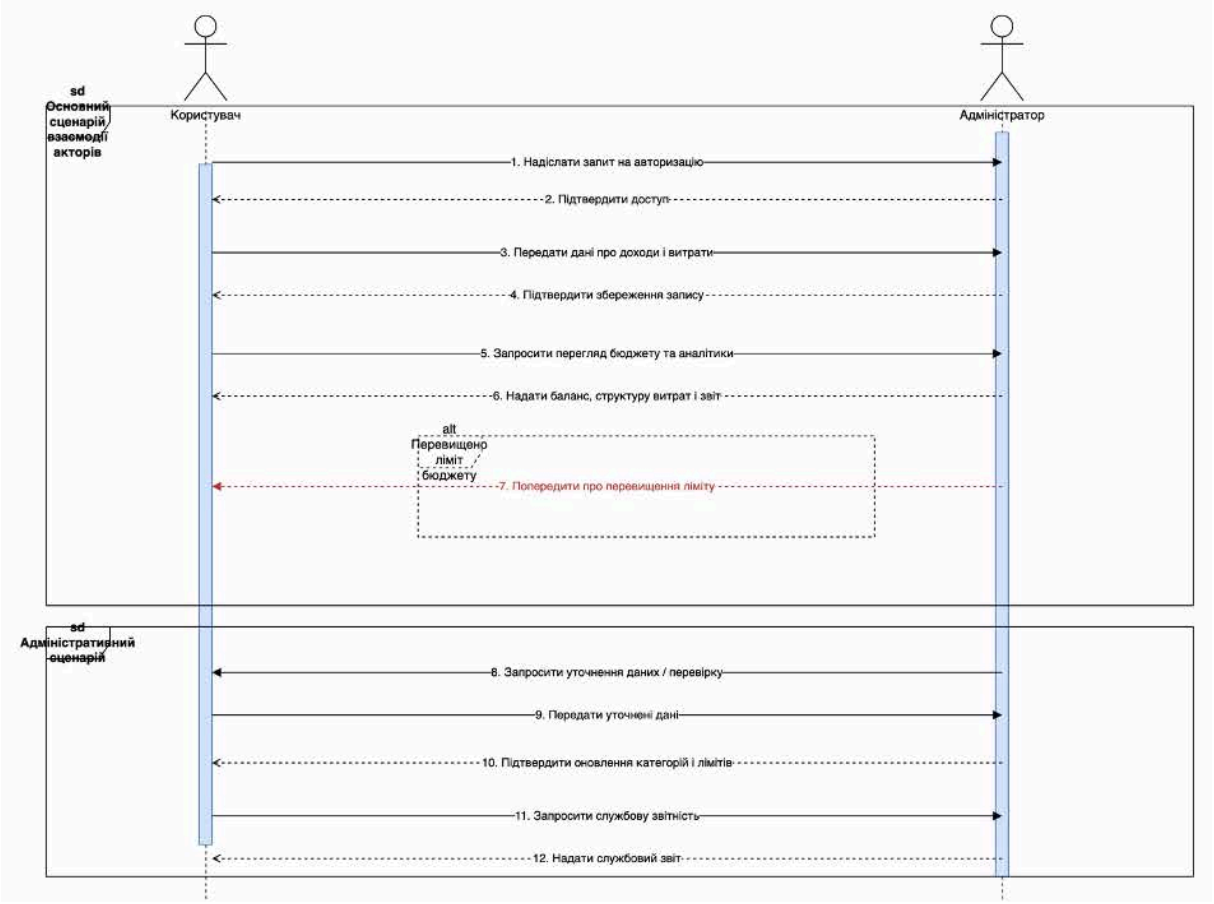


Рис. 1.7 – Діаграма послідовності основного сценарію взаємодії акторів

Діаграма послідовності демонструє, що після авторизації користувач передає дані про фінансові операції, а система підтверджує їх збереження і формує аналітичну відповідь. Умовний блок перевірки ліміту показує альтернативний сценарій, коли користувач отримує попередження про перевищення бюджету. Це відповідає основній логіці розроблюваної системи.

Алгоритмічний перебіг взаємодії між користувачем і адміністратором подано на діаграмі активності, що наведена на рисунку 1.8. Вона відображає

послідовність дій від авторизації до перегляду бюджету, уточнення даних, підтвердження оновлення категорій і отримання службового звіту.

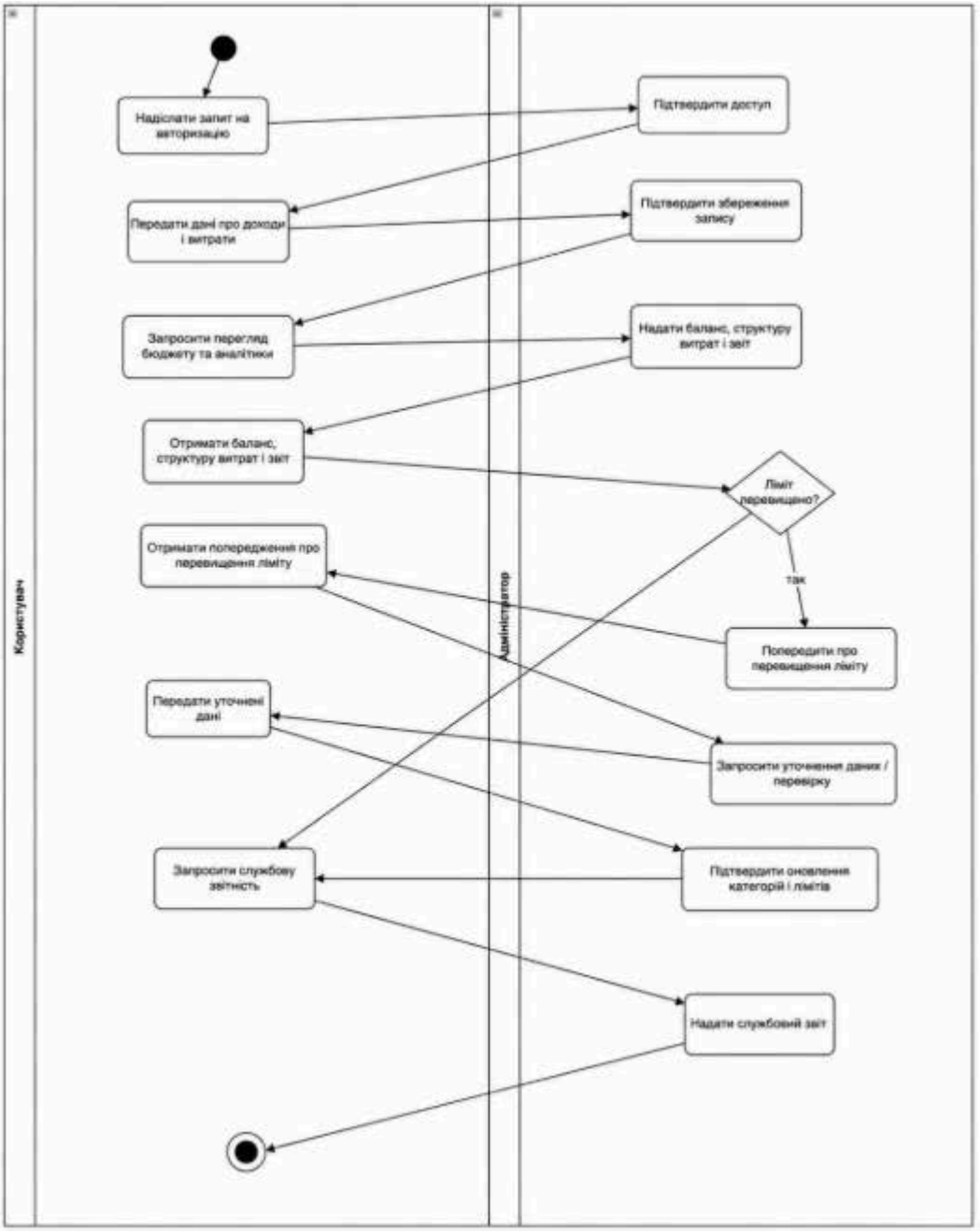


Рис. 1.8 – Діаграма активності процесу роботи із системою обліку побутових витрат

Діаграма активності дозволяє простежити логіку прийняття рішень у системі. Якщо встановлений ліміт не перевищено, користувач продовжує роботу з аналітичними показниками. Якщо ліміт перевищено, система формує попередження і ініціює уточнення даних або перегляд бюджетних обмежень.

Побудовані UML-моделі є основою для подальшого проєктування структури програмного забезпечення.

1.4 Аналіз вимог розроблюваної системи

Розроблення програмного забезпечення для обліку побутових витрат з аналізом бюджету потребує попереднього визначення вимог, які описують функціональне призначення системи, очікувану якість її роботи, технічні обмеження та механізми захисту даних. Аналіз вимог дає змогу узгодити потреби користувача, структуру предметної області та подальшу програмну реалізацію. [1; 2; 12]

Основним призначенням системи є забезпечення локального ведення доходів і витрат, створення особистих бюджетів, групування фінансових операцій за категоріями, встановлення лімітів витрат, формування сповіщень, відображення аналітики та підготовка звітів. Система повинна бути зрозумілою для користувача, забезпечувати швидке введення фінансових операцій і надавати актуальну інформацію про стан бюджету.

Вхідними даними системи є облікові дані користувача, параметри бюджету, назви категорій, суми доходів і витрат, дати операцій, типи транзакцій, описи платежів і встановлені ліміти. Вихідними даними є розраховані фінансові показники, залишок бюджету, попередження про перевищення лімітів, таблиці операцій, аналітичні підсумки та звіти за вибраний період.

Функціональні вимоги визначають основні дії, які повинні виконувати користувач, адміністратор і сама система. Перелік функціональних вимог наведено в таблиці 1.3.

Таблиця 1.3

Функціональні вимоги до програмного забезпечення

№	Функціональна вимога	Опис
1	Авторизація користувача	Система повинна забезпечувати вхід користувача за логіном або e-mail і паролем
2	Реєстрація користувача	Система повинна дозволяти створення нового облікового запису з подальшим збереженням даних у базі
3	Ведення доходів	Користувач повинен мати можливість додавати, переглядати та редагувати записи про доходи
4	Ведення витрат	Користувач повинен мати можливість додавати, переглядати та редагувати записи про витрати
5	Керування бюджетом	Система повинна підтримувати створення бюджету на вибраний період із зазначенням планової суми
6	Керування категоріями	Система повинна дозволяти створення, перегляд і редагування категорій доходів та витрат
7	Контроль лімітів	Система повинна перевіряти витрати за категоріями відповідно до встановлених користувачем меж
8	Формування попереджень	У разі наближення до ліміту або його перевищення система повинна створювати відповідне сповіщення
9	Аналітика бюджету	Система повинна розраховувати доходи, витрати, баланс, залишок бюджету та частку витрат за категоріями
10	Формування звітів	Система повинна формувати звіти за вибраний період на основі збережених фінансових операцій
11	Експорт даних	Система повинна підтримувати підготовку даних у табличному форматі для подальшого збереження або аналізу
12	Адміністрування	Адміністратор повинен мати доступ до службових налаштувань, категорій, лімітів і звітної інформації

Як видно з таблиці 1.3, функціональні вимоги охоплюють повний цикл роботи з побутовими фінансами: від створення облікового запису та введення операцій до аналізу бюджету, контролю лімітів і формування звітності. Їх виконання забезпечує практичну придатність системи для щоденного використання.

Окрім функціональних можливостей, важливе значення мають нефункціональні вимоги, які визначають якість роботи програмного забезпечення. Вони стосуються зручності інтерфейсу, швидкодії, стабільності,

збереження даних і можливості подальшого розвитку системи. Основні нефункціональні вимоги наведено в таблиці 1.4. [2; 4]

Таблиця 1.4

Нефункціональні вимоги до програмного забезпечення

№	Нефункціональна вимога	Характеристика
1	Зручність інтерфейсу	Елементи керування повинні бути зрозумілими для користувача без спеціальної підготовки
2	Швидкість роботи	Перемикання розділів, збереження операцій і оновлення показників мають виконуватися без помітних затримок
3	Надійність	Некоректне введення даних не повинно призводити до аварійного завершення роботи програми
4	Локальність роботи	Базові функції системи повинні працювати без обов’язкового підключення до хмарного сервісу
5	Супроводжуваність	Програмний код повинен бути поділений на модулі інтерфейсу, роботи з даними та бізнес-логіки
6	Масштабованість	Структура системи повинна дозволяти додавання нових категорій, звітів, правил контролю та аналітичних показників
7	Збереження даних	Фінансові записи мають залишатися доступними після перезапуску програми
8	Узгодженість даних	Дані про бюджети, категорії, операції, ліміти та звіти повинні зберігатися у взаємопов’язаній структурі
9	Зрозумілість результатів	Аналітичні показники мають подаватися у формі, придатній для швидкого оцінювання фінансового стану користувача

Наведені в таблиці 1.4 нефункціональні вимоги визначають якість використання системи. Для локального програмного забезпечення особливо важливими є стабільність роботи, простота інтерфейсу, коректне збереження інформації та можливість подальшого розширення функціональності без повної зміни архітектури.

Окрему групу становлять вимоги до безпеки, оскільки система працює з персональними фінансовими даними користувача. Навіть у разі локального збереження інформації необхідно передбачити автентифікацію, захист паролів, перевірку введених даних і підтримання цілісності бази даних. Вимоги до безпеки наведено в таблиці 1.5. [14; 15]

Таблиця 1.5

Вимоги до безпеки програмного забезпечення

№	Вимога до безпеки	Опис
1	Автентифікація	Доступ до системи повинен надаватися лише після введення правильних облікових даних
2	Захист паролів	Паролі користувачів повинні зберігатися у вигляді хешу, а не відкритого тексту
3	Розмежування ролей	Система повинна відрізняти звичайного користувача та адміністратора
4	Контроль введення	Сума операції, дата, категорія, бюджет і тип транзакції повинні перевірятися перед збереженням
5	Цілісність даних	Операції повинні бути пов’язані з існуючими користувачами, бюджетами та категоріями
6	Збереження історії	Фінансові операції, звіти та сповіщення повинні зберігатися для подальшого перегляду й аналізу
7	Обмеження некоректних дій	Система не повинна дозволяти створення операцій із від’ємною сумою, порожньою категорією або неіснуючим бюджетом
8	Захист від втрати даних	Дані повинні зберігатися в локальній базі після завершення роботи програми

Як видно з таблиці 1.5, вимоги до безпеки спрямовані не лише на захист доступу до програми, а й на забезпечення коректності фінансових записів. Перевірка введених даних, використання зв’язків між таблицями та збереження історії операцій дають змогу підтримувати цілісність інформації та уникати помилок під час роботи користувача із системою.

Отже, визначені функціональні, нефункціональні та безпекові вимоги є основою для подальшого проєктування програмного забезпечення. Вони дозволяють перейти до розроблення моделей системи, структури бази даних, алгоритмів оброблення фінансових операцій і тестування програмних модулів. Сукупність наведених вимог забезпечує відповідність майбутньої системи поставленому завданню та потребам користувача.

1.5 Постановка завдання

Постановка завдання полягає у проєктуванні та розробленні інформаційної системи обліку побутових витрат, яка автоматизує ведення доходів, витрат, бюджету, категорій, лімітів і звітів користувача. Система повинна працювати як локальний застосунок із графічним інтерфейсом і забезпечувати збереження даних у структурованій базі.

У межах роботи необхідно реалізувати модулі авторизації та реєстрації, керування бюджетами, введення доходів і витрат, ведення категорій, контролю лімітів, формування попереджень, побудови аналітичних показників і створення звітів. Дані мають зберігатися у базі SQLite, що дозволяє запускати систему без окремого серверного середовища.

Результатом виконання завдання має бути програмна система, яка дає змогу користувачу вести особистий бюджет, швидко оцінювати доходи і витрати, аналізувати структуру побутових витрат і контролювати перевищення встановлених лімітів. Система повинна бути придатною для навчального використання, демонстрації принципів проєктування інформаційних систем і подальшого розширення функціоналу.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Проєктування інформаційного забезпечення починається з побудови логічної моделі даних. Логічна модель описує сутності предметної області, їхні атрибути та зв’язки без прив’язки до конкретної СКБД. Для системи обліку побутових витрат ключовими сутностями є User, Budget, Transaction, Category, Budget_Category_Limit, Notification, Report, Income_Transaction та Expense_Transaction.

На рисунку 2.1 подано логічну модель даних. Вона відображає зв’язки між користувачем, бюджетами, фінансовими операціями, категоріями, лімітами, повідомленнями та звітами. Така модель дозволяє зберігати як загальні транзакції, так і спеціалізовану інформацію про доходи та витрати.

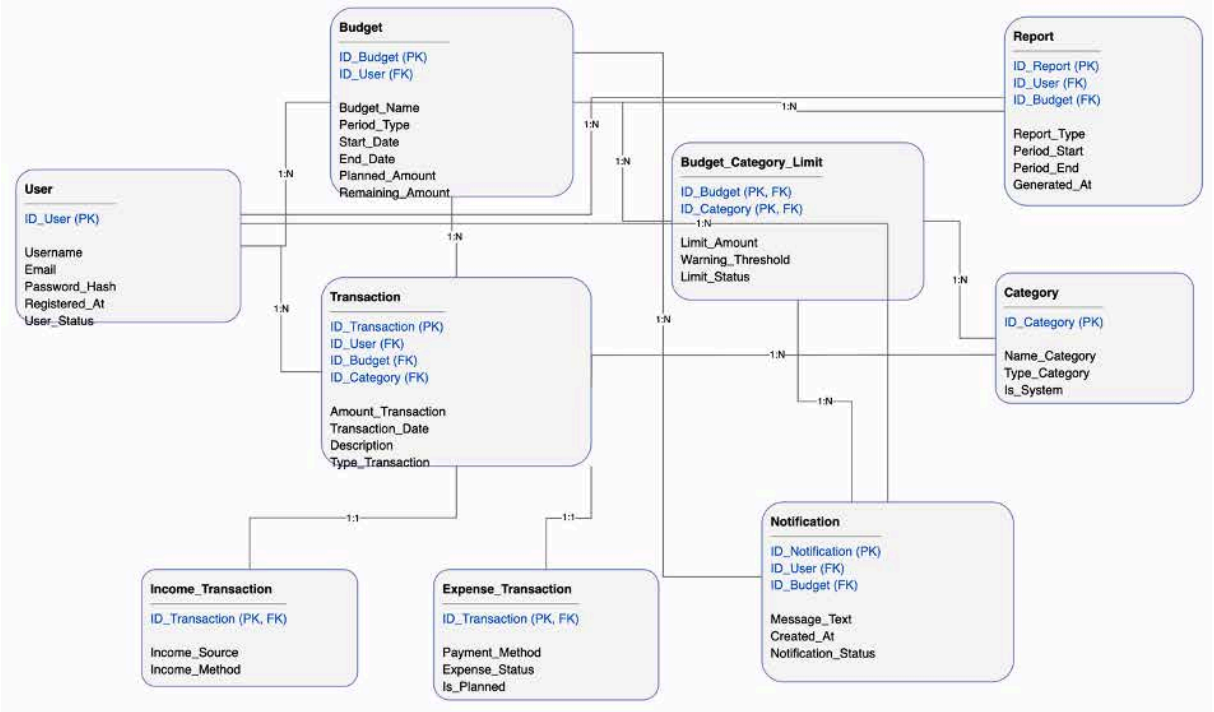


Рис. 2.1 – Логічна модель даних інформаційної системи обліку побутових витрат

Сутність User містить дані про користувача системи: ідентифікатор, логін, електронну пошту, хеш пароля, дату реєстрації та статус. Сутність Budget зберігає інформацію про бюджетний період, планову суму та залишок. Зв’язок між User і Budget має кратність один-до-багатьох, оскільки один користувач може створювати декілька бюджетів за різні періоди.

Сутність Transaction є центральною для обліку фінансових операцій. Вона містить суму, дату, опис і тип операції. Для деталізації типу операції використовуються сутності Income_Transaction та Expense_Transaction. Такий підхід дає змогу не дублювати загальні атрибути, але водночас зберігати специфічні параметри доходів і витрат.

Сутність Category використовується для групування фінансових операцій. За допомогою сутності Budget_Category_Limit встановлюються ліміти витрат для конкретних категорій у межах бюджету. Якщо фактичні витрати перевищують попереджувальний поріг або повний ліміт, система створює запис у Notification. Звіти зберігаються у сутності Report і пов’язуються з користувачем та бюджетом.

Таблиця 2.1

Сутності логічної моделі даних

Сутність	Призначення
User	Зберігає облікові дані користувачів системи
Budget	Описує бюджет користувача на вибраний період
Transaction	Фіксує загальні дані доходу або витрати
Income_Transaction	Уточнює дані операцій доходу
Expense_Transaction	Уточнює дані операцій витрати
Category	Містить категорії доходів і витрат
Budget_Category_Limit	Зберігає обмеження витрат за категоріями
Notification	Містить повідомлення про перевищення лімітів
Report	Зберігає сформовані звіти за період

Логічна модель відповідає вимогам предметної області та забезпечує основу для побудови фізичної моделі даних. Вона є достатньо деталізованою для реалізації обліку доходів, витрат, бюджетів, категорій і звітності без надмірного ускладнення структури.

2.2 Діаграма класів і кооперації

Для опису об’єктної структури програмної системи побудовано діаграму класів, наведену на рисунку 2.2. Вона показує основні класи предметної області та операції, які виконуються над ними. Діаграма відображає зв’язок між користувачем, бюджетом, доходами, витратами, категоріями та звітами.

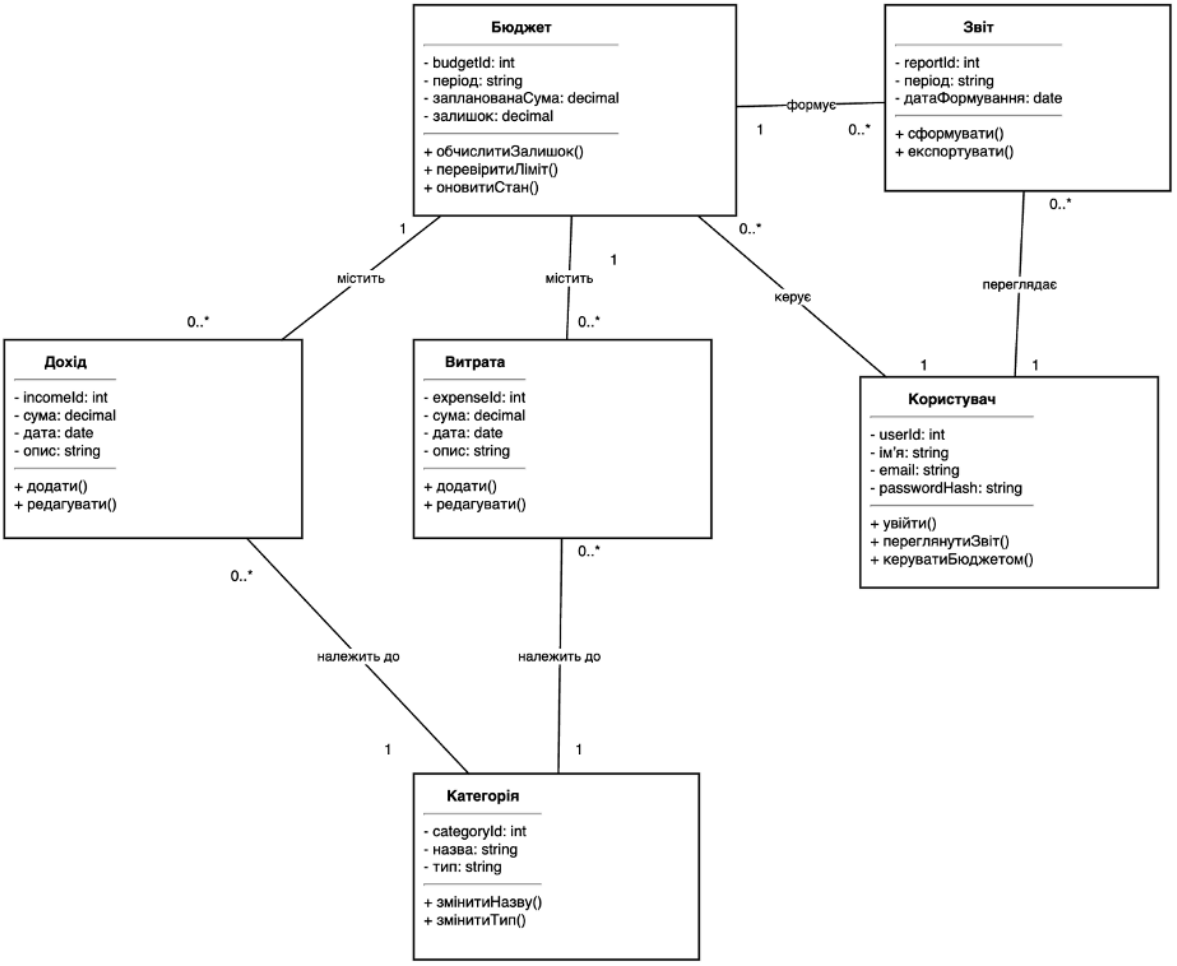


Рис. 2.2 – Діаграма класів інформаційної системи обліку побутових витрат

Клас Користувач відповідає за дії входу до системи, перегляду звітів і керування бюджетом. Клас Бюджет містить планову суму, залишок і методи обчислення залишку та перевірки ліміту. Класи Дохід і Витрата описують фінансові операції, а клас Категорія забезпечує їх групування. Клас Звіт використовується для формування та експорту підсумкової інформації.

Для деталізації взаємодії об’єктів побудовано три діаграми кооперації. Перша кооперація описує процес додавання витрати і наведена на рисунку 2.3.

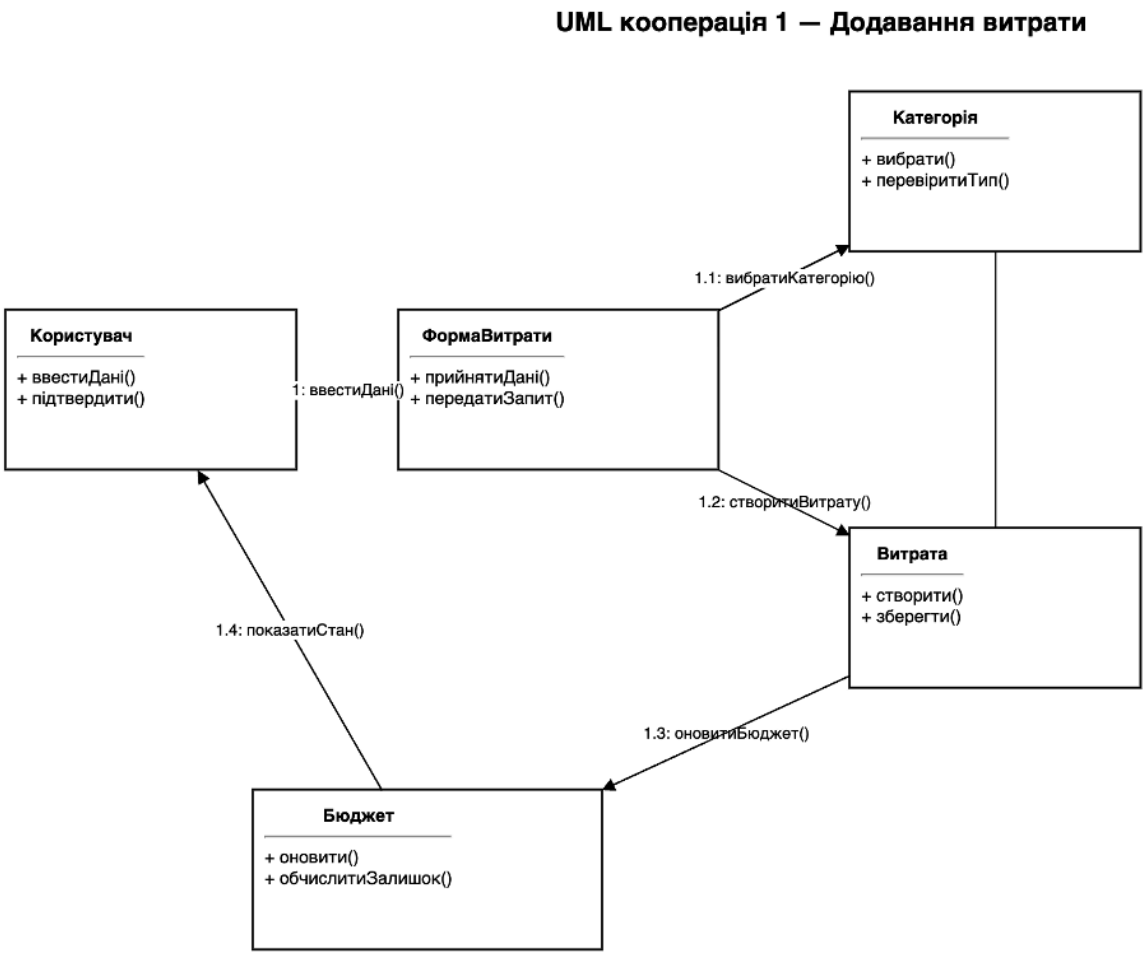


Рис. 2.3 – Діаграма кооперації процесу додавання витрати

Під час додавання витрати користувач вводить дані у форму, обирає категорію, після чого створюється об’єкт витрати. Далі бюджет оновлює свій стан і повертає користувачу результат виконання операції. Така взаємодія відповідає реальному сценарію внесення витрат у систему.

Друга кооперація описує аналіз бюджету, що наведено на рисунку 2.4. У цьому сценарії користувач ініціює аналіз, модуль аналітики отримує стан бюджету, звертається до операцій витрат і формує підсумковий звіт.

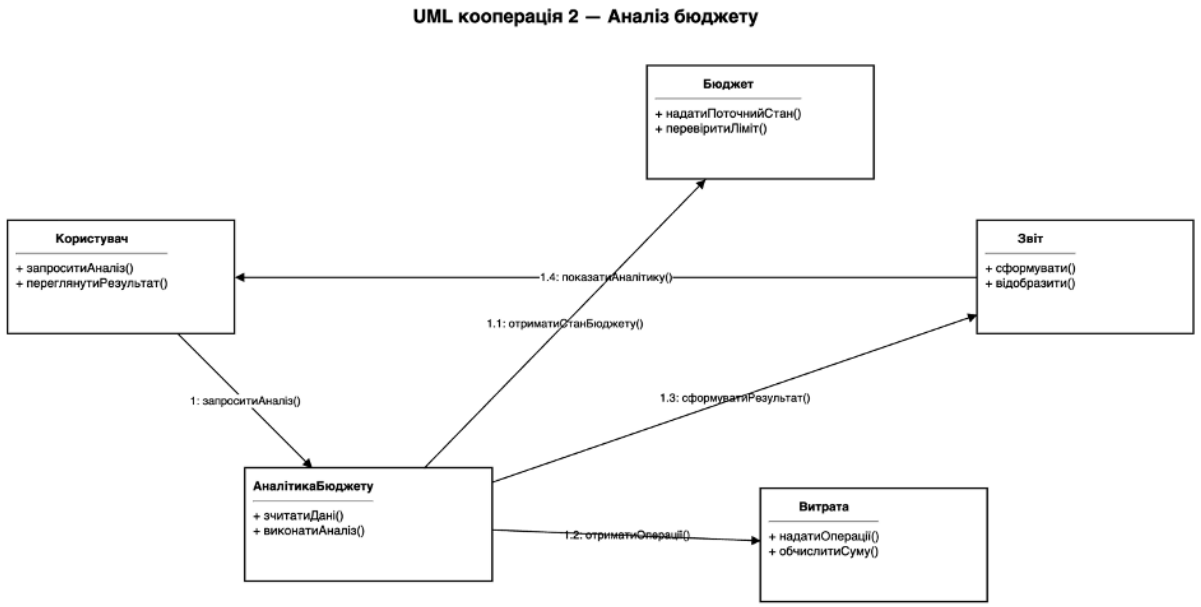


Рис. 2.4 – Діаграма кооперації процесу аналізу бюджету

Результатом аналізу є відображення користувачу балансу, структури витрат і показників використання бюджету. Діаграма показує, що аналітична логіка відокремлена від введення операцій і відображення звіту, що покращує модульність системи.

Третя кооперація описує процес керування лімітами і звітністю. Вона наведена на рисунку 2.5 та відображає взаємодію адміністратора, налаштувань, категорій, бюджету і звіту.

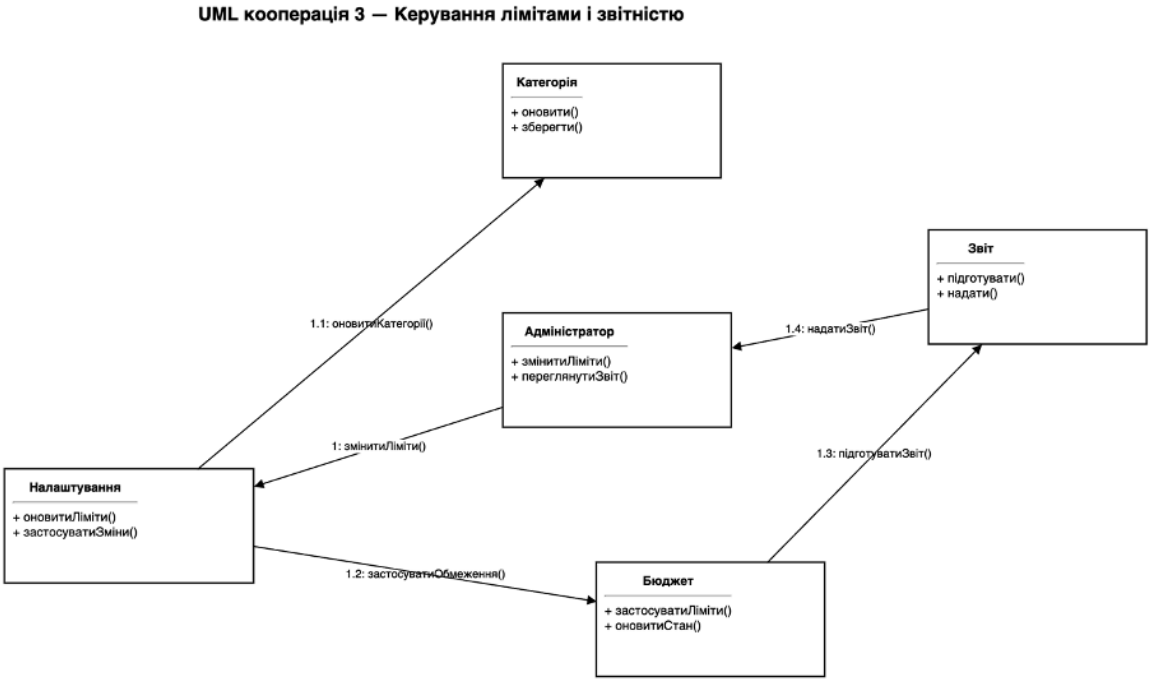


Рис. 2.5 – Діаграма кооперації процесу керування лімітами і звітністю

У межах цього сценарію адміністратор змінює ліміти, оновлює категорії та отримує службовий звіт. Побудовані діаграми кооперації деталізують поведінку системи на рівні об’єктів і підтверджують узгодженість класів із функціональними сценаріями.

2.3 Діаграма компонентів

Діаграма компонентів використовується для подання програмної системи як сукупності взаємопов’язаних модулів. Для інформаційної системи обліку побутових витрат виділено клієнтський рівень, серверний рівень, рівень даних і зовнішній сервіс повідомлень. Діаграму компонентів наведено на рисунку 2.6.

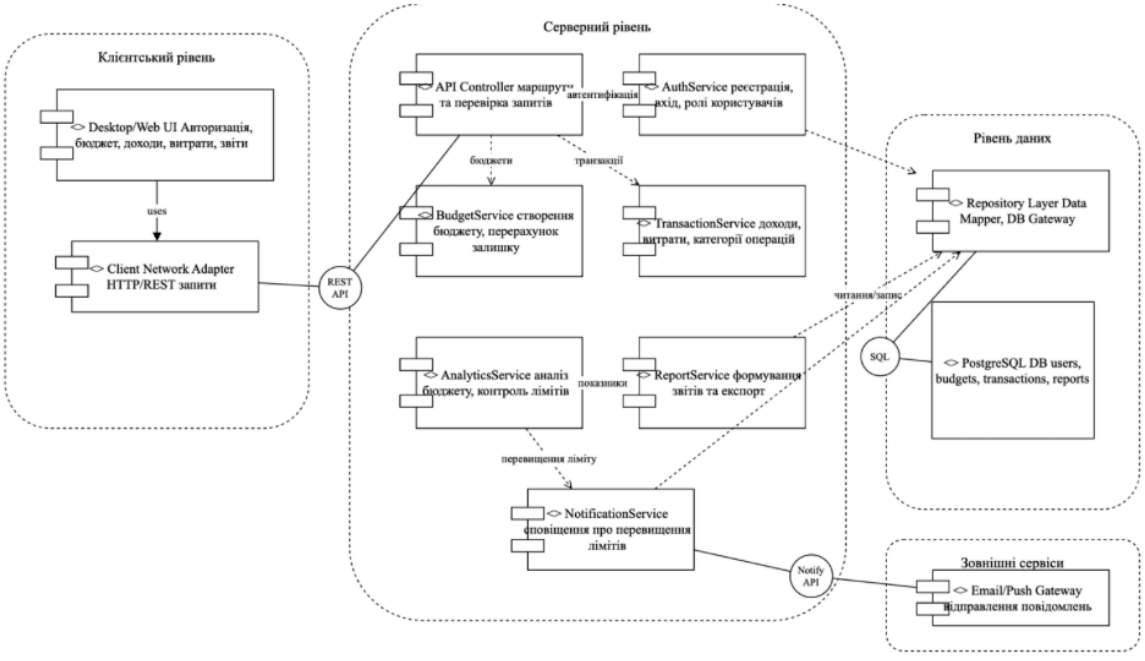


Рис. 2.6 – Діаграма компонентів інформаційної системи обліку побутових витрат

Клієнтський рівень містить Desktop/Web UI та мережевий адаптер запитів. Серверний рівень включає API Controller, AuthService, BudgetService, TransactionService, AnalyticsService, ReportService та NotificationService. Рівень даних представлений Repository Layer і базою даних PostgreSQL або SQLite

залежно від варіанта реалізації. Такий розподіл компонентів дає змогу підтримувати модульність і розширюваність системи.

2.4 Діаграма пакетів

Пакетна структура відображає логічну організацію програмного коду. Вона дозволяє розділити інтерфейс, прикладну логіку, доменні класи, аналітичні функції та доступ до даних. Діаграма пакетів наведена на рисунку 2.7.

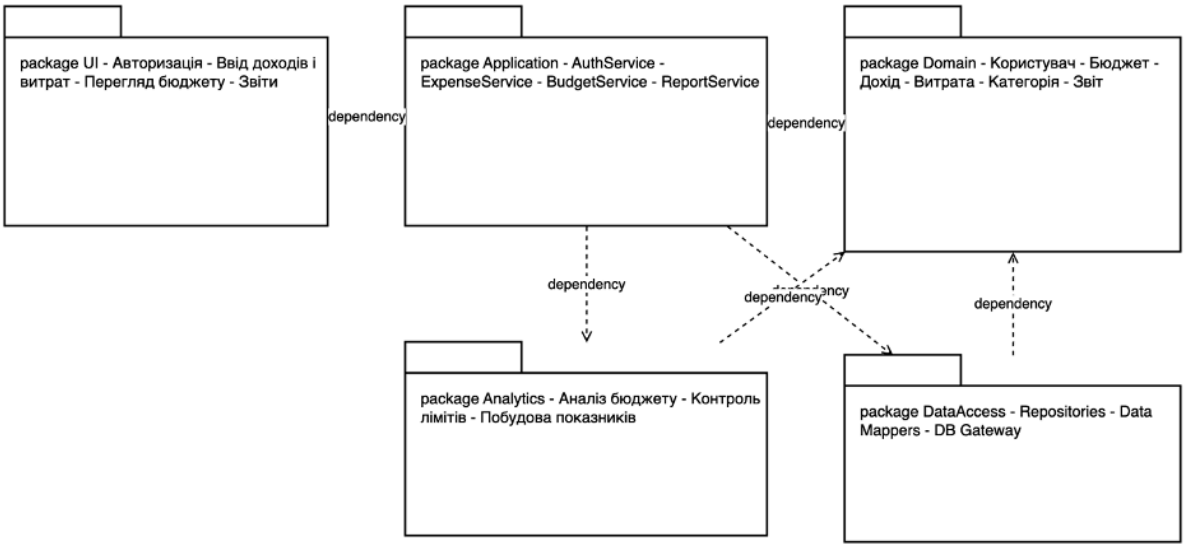


Рис. 2.7 – Діаграма пакетів програмного забезпечення

Пакет UI містить екрани авторизації, введення доходів і витрат, перегляду бюджету та звітів. Пакет Application об’єднує сервіси AuthService, ExpenseService, BudgetService і ReportService. Пакет Domain зберігає доменні класи користувача, бюджету, доходу, витрати, категорії та звіту. Пакет Analytics відповідає за аналіз бюджету, контроль лімітів і побудову показників, а пакет DataAccess реалізує роботу з репозиторіями та базою даних.

2.5 Висновки до другого розділу

У другому розділі виконано проєктування інформаційного та програмного забезпечення системи. Побудовано логічну модель даних, діаграму класів, три діаграми кооперації, компонентну модель і пакетну структуру. Отримані моделі формалізують структуру даних, поведінку об’єктів і архітектурну організацію програмної системи. Це створює основу для переходу до фізичного проєктування бази даних, алгоритмізації модулів і практичної реалізації системи обліку побутових витрат.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологічних засобів та інструментів реалізації

Для реалізації інформаційної системи обліку побутових витрат доцільно використати технологічний стек, який забезпечує швидку розробку графічного інтерфейсу, локальне збереження даних, простоту встановлення та можливість подальшого розвитку. З урахуванням цих вимог обрано мову програмування Python, бібліотеку PyQt6 для побудови графічного інтерфейсу та SQLite як локальну базу даних. [6; 7; 9]

Python є придатним для реалізації навчального і прикладного програмного забезпечення завдяки читабельному синтаксису, наявності бібліотек для роботи з базами даних, файлами, таблицями та графічними інтерфейсами. PyQt6 дозволяє створити повноцінний настільний інтерфейс із вікнами, формами, таблицями, вкладками, кнопками та діалогами. SQLite не потребує окремого сервера, що спрощує розгортання системи на робочій станції користувача.

Архітектура програмного забезпечення передбачає поділ на модулі: інтерфейс користувача, сервіси бізнес-логіки, моделі даних і модуль доступу до бази даних. Така структура відповідає діаграмам компонентів і пакетів, наведеним у другому розділі, та забезпечує супроводжуваність програмного коду.

Таблиця 3.1

Вибір технологічних засобів реалізації

Технологія	Призначення у системі	Обґрунтування вибору
Python	Реалізація прикладної логіки	Простий синтаксис, достатня продуктивність для локального застосунку, підтримка SQLite
PyQt6	Створення графічного інтерфейсу	Підтримка вкладок, таблиць, форм введення, кнопок і діалогових вікон
SQLite	Локальне збереження даних	Не потребує окремого сервера, зберігає дані у файлі, зручна для настільного застосунку

Продовження таблиці 3.1

CSV	Експорт звітів	Простий табличний формат, який можна відкрити в електронних таблицях
UML	Моделювання системи	Забезпечує формалізацію сценаріїв, класів, компонентів і розгортання

Обраний технологічний стек відповідає меті роботи, оскільки дозволяє реалізувати локальну систему без складної серверної інфраструктури. При цьому структура програми залишається придатною для подальшого розширення, наприклад додавання імпорту банківських операцій, резервного копіювання або синхронізації даних.

3.2 Фізична модель даних

Фізична модель даних конкретизує логічну ER-модель на рівні таблиць, типів даних, первинних і зовнішніх ключів. Вона визначає структуру бази даних, яка використовується програмною системою під час збереження користувачів, бюджетів, категорій, транзакцій, лімітів, повідомлень і звітів. Фізична модель наведена на рисунку 3.1.

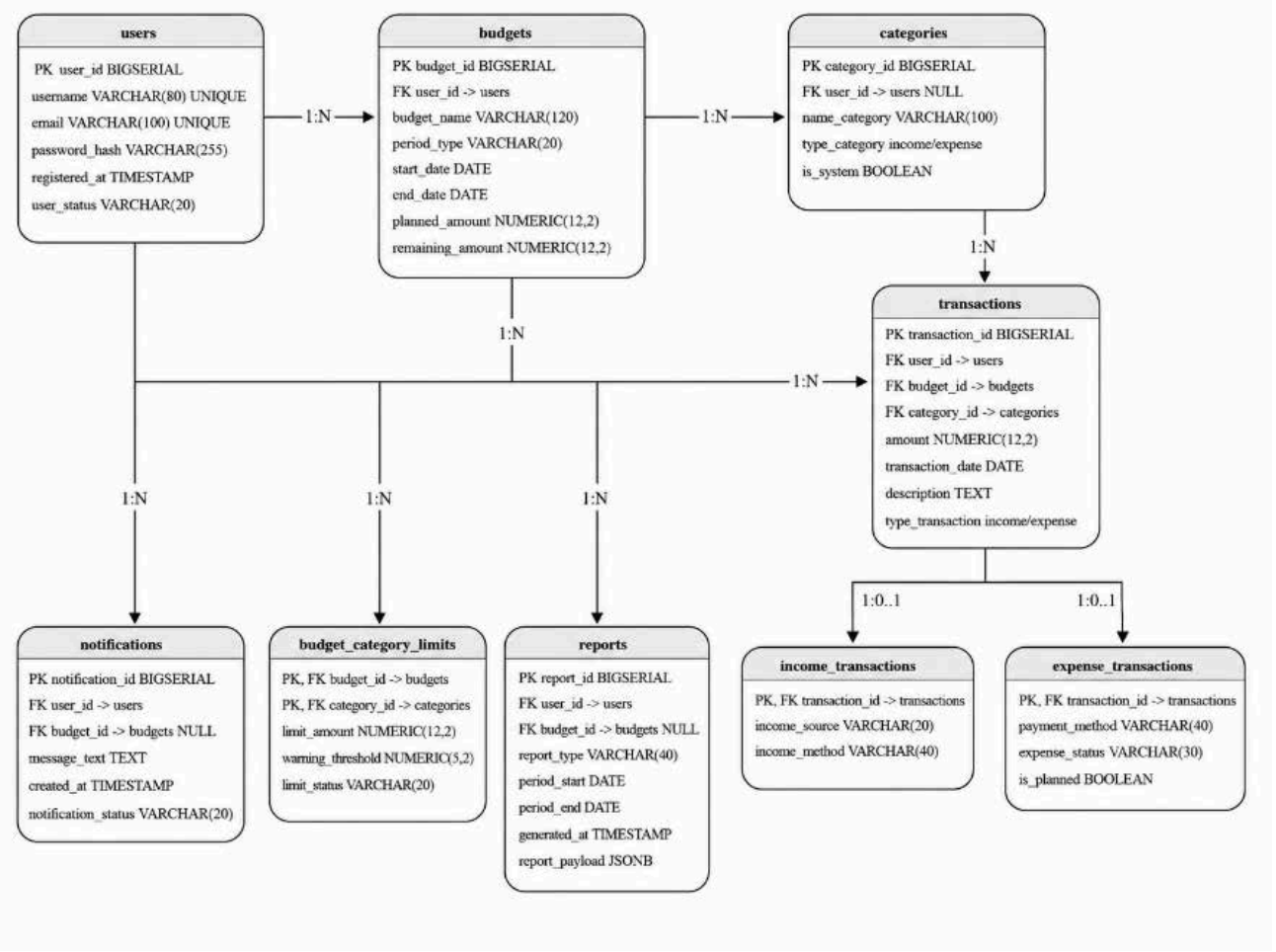


Рис. 3.1 – Фізична модель даних інформаційної системи обліку побутових витрат

У фізичній моделі таблиця users містить облікові записи користувачів. Таблиця budgets зберігає бюджетні періоди та планові суми. Таблиця categories використовується для довідника категорій, а transactions містить усі фінансові операції. Таблиці income_transactions і expense_transactions деталізують доходи та витрати. Таблиця budget_category_limits використовується для встановлення лімітів за категоріями. Таблиці notifications і reports забезпечують збереження попереджень і звітів.

Таблиця 3.2

Дані які зберігає система

Таблиця	Ключові поля	Призначення
users	user_id, username, email, password_hash	Збереження облікових записів користувачів

budgets	budget_id, user_id, period_type, planned_amount	Збереження бюджетів користувача
categories	category_id, name_category, type_category	Довідник категорій доходів і витрат
transactions	transaction_id, user_id, budget_id, category_id, amount	Збереження фінансових операцій
income_transactions	transaction_id, income_source, income_method	Деталізація операцій доходу
expense_transactions	transaction_id, payment_method, expense_status	Деталізація операцій витрати
budget_category_limits	budget_id, category_id, limit_amount, warning_threshold	Контроль лімітів за категоріями
notifications	notification_id, user_id, budget_id, message_text	Збереження попереджень користувача
reports	report_id, user_id, budget_id, period_start, period_end	Збереження сформованих звітів

Фізична модель забезпечує цілісність даних завдяки зовнішнім ключам між користувачами, бюджетами, категоріями й операціями. Така структура відповідає вимогам нормалізації та дозволяє уникати дублювання інформації, зберігаючи зв’язки між усіма основними об’єктами системи.

3.3 Алгоритмізація програмних модулів системи

Алгоритмізація програмних модулів дає змогу формально описати логіку виконання основних операцій системи. Для розроблюваного застосунку найбільш важливими є алгоритми авторизації користувача, додавання доходу або витрати, аналізу бюджету, контролю лімітів і формування звіту.

Алгоритм авторизації користувача наведено на рисунку 3.2. Він передбачає введення логіна або e-mail і пароля, перевірку заповнення полів, пошук користувача у таблиці users, перевірку хешу пароля та відкриття головного вікна у разі успішного входу.

Блок-схема 5 – Алгоритм формування звіту та експорту

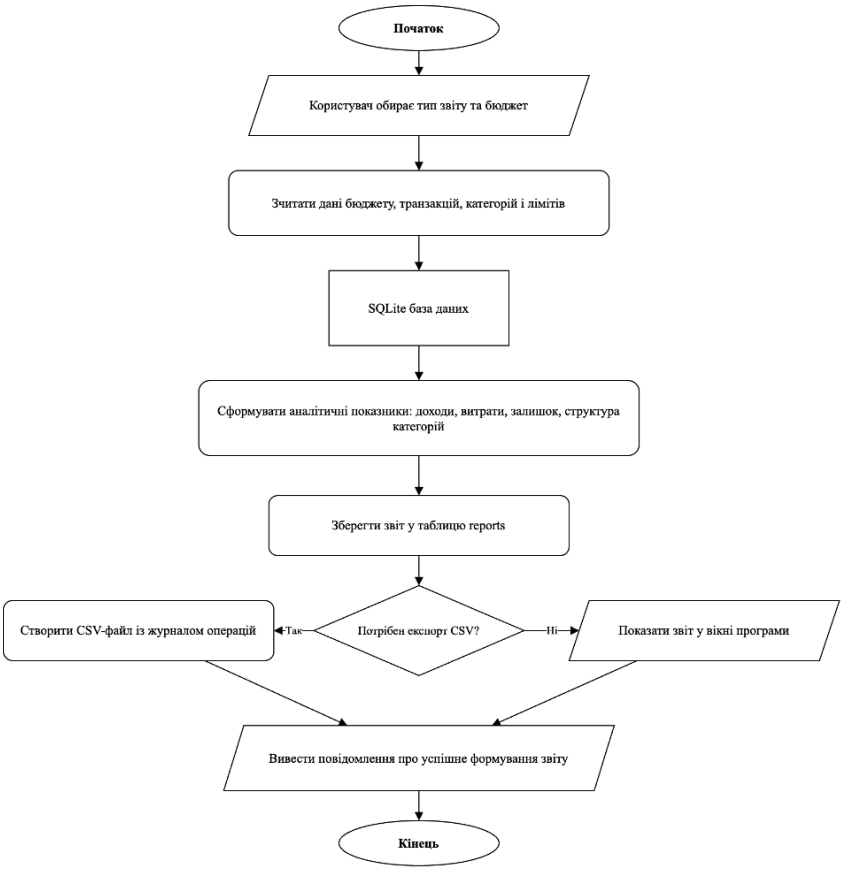


Рис. 3.2 – Алгоритм авторизації користувача

Якщо користувача не знайдено або пароль не відповідає збереженому хешу, система виводить повідомлення про помилку. Такий підхід забезпечує базовий рівень захисту доступу до персональних фінансових даних.

Алгоритм додавання доходу або витрати наведено на рисунку 3.3. Користувач обирає бюджет, тип операції, категорію, суму і дату. Далі система перевіряє коректність даних і, якщо введення є валідним, створює запис у таблиці transactions.

Рис. 3.3 – Алгоритм додавання доходу або витрати

Після створення базового запису система визначає тип операції. Якщо це дохід, додається деталізація до таблиці income_transactions; якщо витрата - до таблиці expense_transactions. Завершальним етапом є перерахунок залишку бюджету.

Алгоритм аналізу бюджету наведено на рисунку 3.4. Він починається з вибору активного бюджету, після чого система отримує дані з таблиць budgets і transactions, розраховує суму доходів, суму витрат, залишок та відсоток використання бюджету.

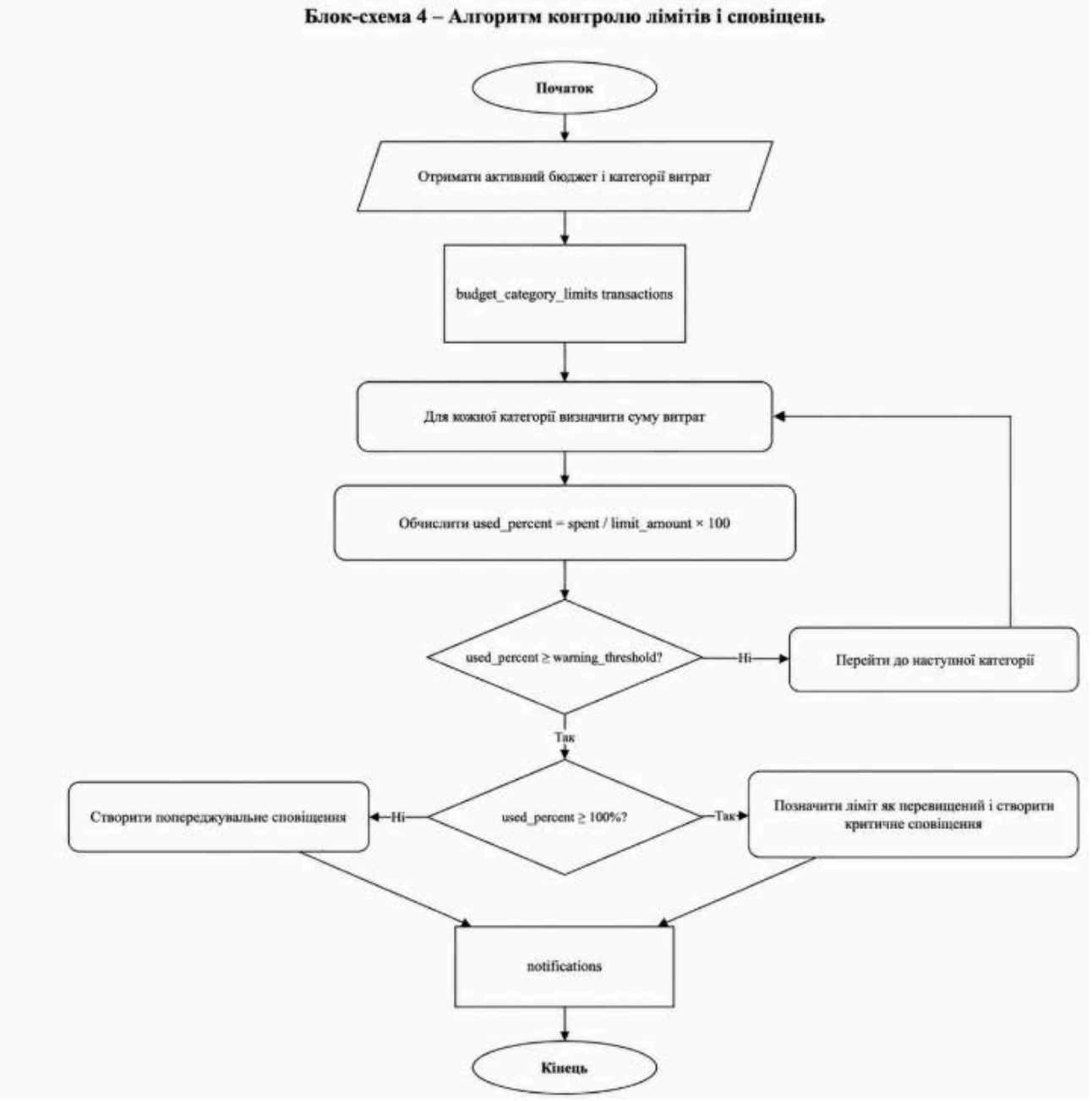


Рис. 3.4 – Алгоритм аналізу бюджету

Якщо витрати перевищують доступний бюджет, система позначає бюджет як ризиковий і готує попередження. Якщо перевищення немає, формується нормальний стан бюджету. У будь-якому разі користувач отримує картки показників і таблицю витрат за категоріями.

Алгоритм контролю лімітів і сповіщень наведено на рисунку 3.5. Він передбачає отримання активного бюджету і категорій витрат, обчислення суми витрат за кожною категорією та порівняння відсотка використання з попереджувальним і критичним порогами.

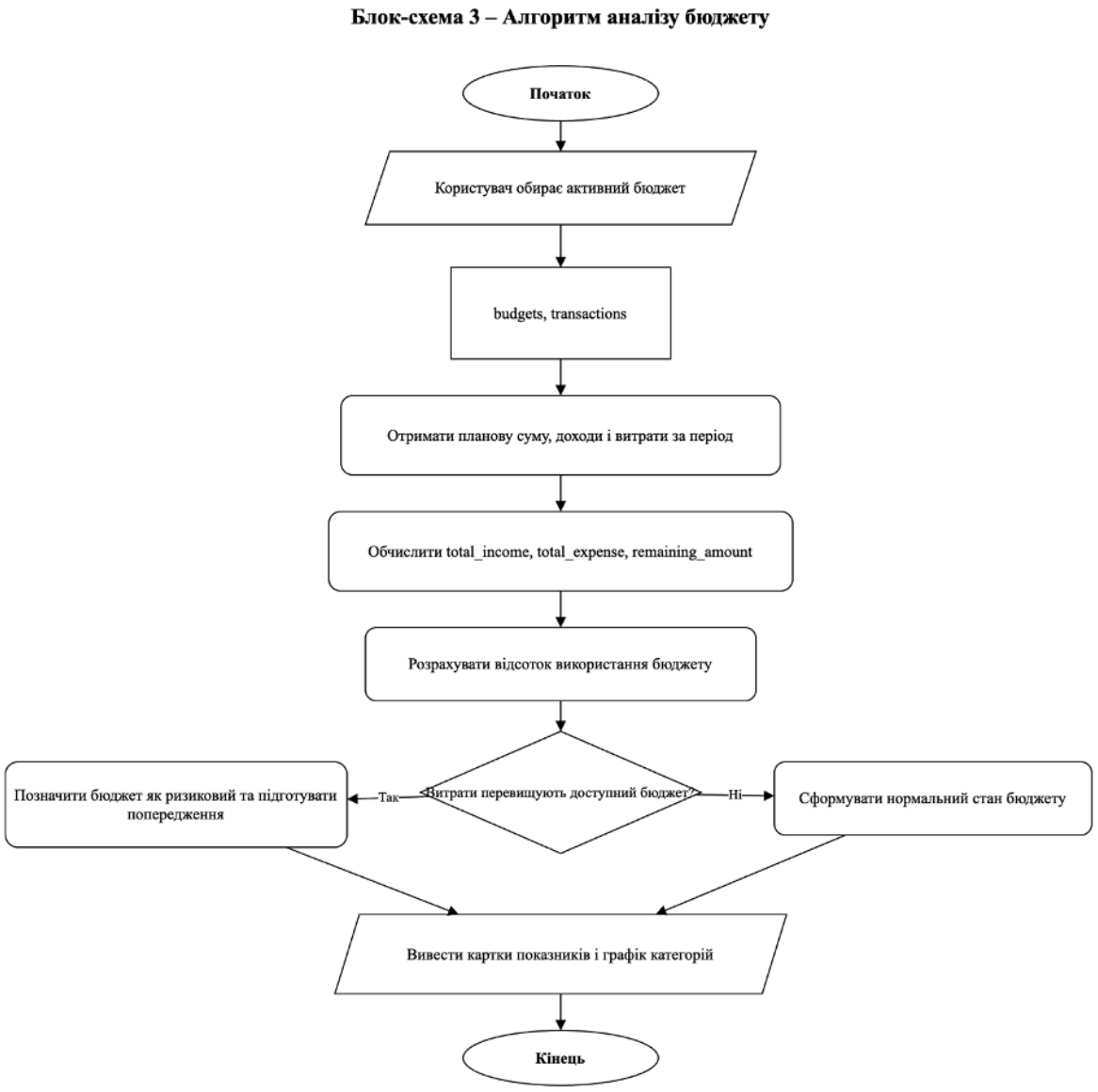


Рис. 3.5 – Алгоритм контролю лімітів і сповіщень

Якщо встановлений warning_threshold перевищено, система створює попереджувальне повідомлення. Якщо витрати досягли або перевищили 100 % ліміту, формується критичне сповіщення. Це дозволяє користувачу своєчасно реагувати на перевищення витрат.

Алгоритм формування звіту та експорту наведено на рисунку 3.6. Він передбачає вибір типу звіту і бюджету, зчитування даних бюджету, транзакцій, категорій і лімітів, формування аналітичних показників та збереження звіту у таблиці reports.

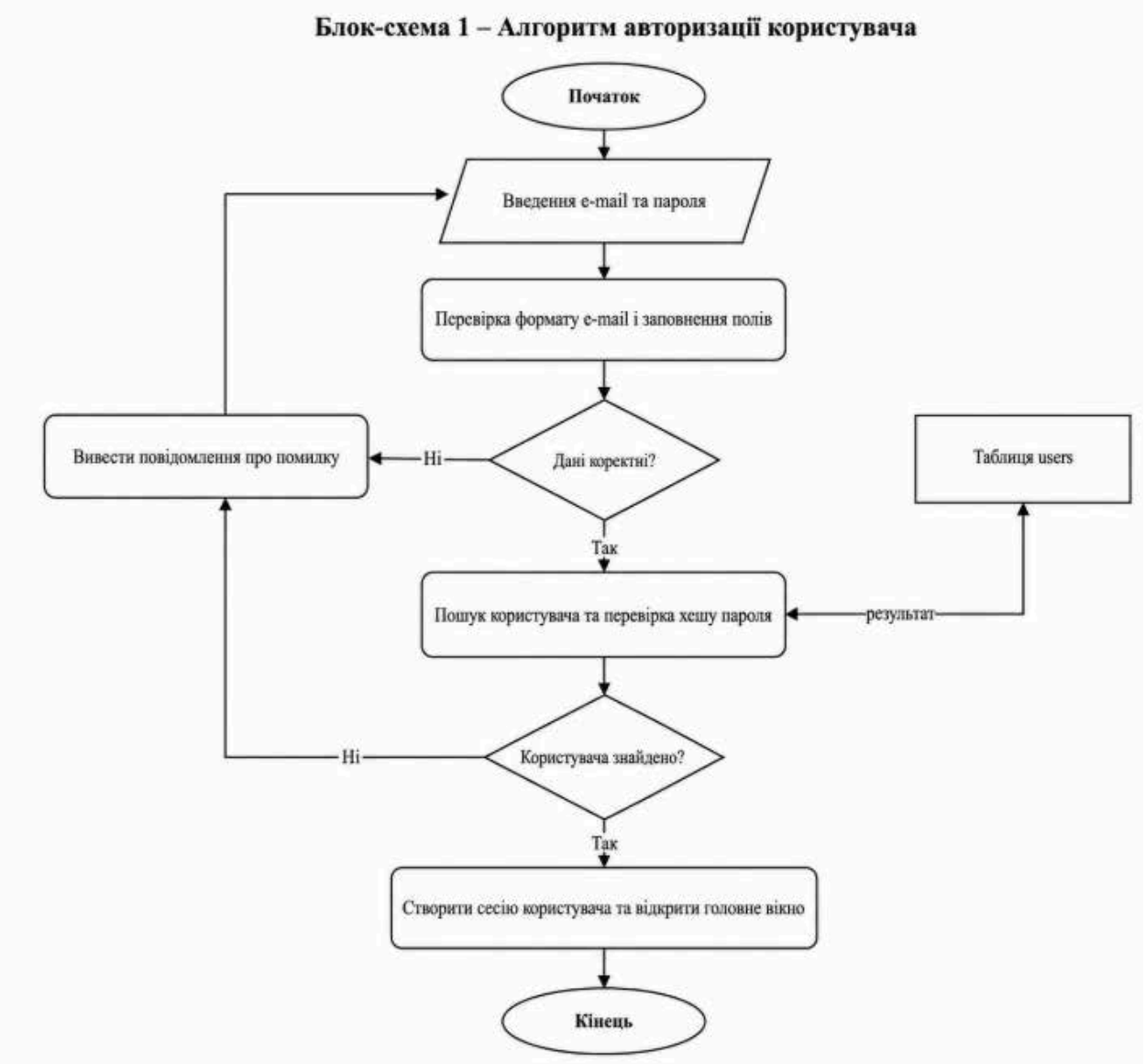


Рис. 3.6 – Алгоритм формування звіту та експорту

За потреби система створює CSV-файл із журналом операцій. Якщо експорт не потрібен, звіт відображається у вікні програми. Побудовані алгоритми охоплюють основні функції системи та забезпечують послідовну реалізацію програмних модулів. [25]

3.4 Представлення архітектури розгортання системи

Для опису розміщення програмних компонентів у середовищі виконання побудовано діаграму розгортання, наведену на рисунку 3.7. Вона показує

клієнтський комп’ютер, сервер застосунку, сервер бази даних і зовнішній сервіс повідомлень.

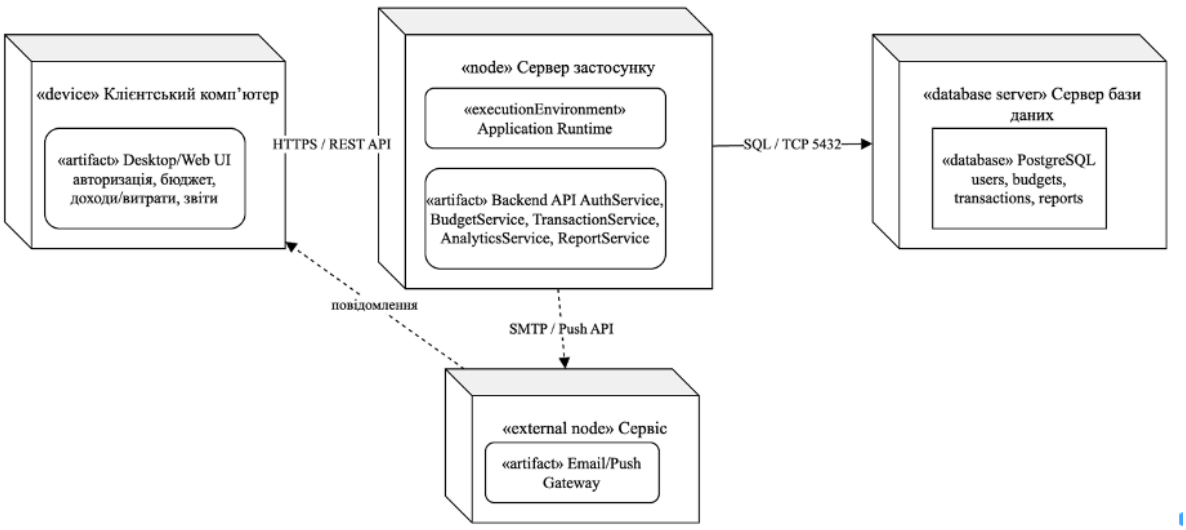


Рис. 3.7 – Діаграма розгортання інформаційної системи обліку побутових витрат

Для локального навчального варіанта клієнтський і серверний рівні можуть бути розміщені на одному комп’ютері. Проте діаграма розгортання демонструє можливість подальшого розвитку системи до клієнт-серверної архітектури, у якій клієнт взаємодіє з сервером через HTTPS/REST API, а сервер звертається до бази даних через SQL-з’єднання.

3.5 Висновки до третього розділу

У третьому розділі обґрунтовано вибір технологічних засобів реалізації, описано фізичну модель даних, розроблено алгоритми основних програмних модулів і подано архітектуру розгортання системи. Запропонована структура забезпечує локальну роботу застосунку, цілісне збереження даних і можливість подальшого розширення функціоналу.

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

Тестування програмного забезпечення для обліку побутових витрат з аналізом бюджету є завершальним етапом перевірки працездатності розробленої системи. Його основна мета полягає у встановленні відповідності реалізованого застосунку сформованим функціональним, нефункціональним і безпековим вимогам, а також у перевірці правильності роботи основних програмних модулів. [12; 13; 24]

Розроблена система реалізована як настільний застосунок із графічним інтерфейсом користувача та локальною базою даних. Вона забезпечує реєстрацію й авторизацію користувачів, ведення доходів і витрат, створення бюджетів, керування категоріями, встановлення лімітів, формування сповіщень, перегляд аналітичних показників і створення звітів. Тому тестування повинно охоплювати не лише окремі функції, а й повний сценарій роботи користувача із системою.

Під час тестування перевіряються такі основні аспекти роботи програмного забезпечення: коректність введення та збереження даних, правильність фінансових розрахунків, стабільність інтерфейсу, взаємодія програмних модулів із базою даних, формування попереджень про перевищення лімітів і створення звітів. План тестування програмних модулів наведено в таблиці 4.1.

Таблиця 4.1

План тестування програмних модулів системи

Модуль системи	Що перевіряється	Очікуваний результат

Модуль авторизації	Вхід користувача за тестовими обліковими даними, а також перевірка введення неправильного логіна або пароля	За правильних даних система відкриває головне вікно, за неправильних даних виводить повідомлення про помилку
Модуль реєстрації	Створення нового облікового запису із заповненням імені користувача, e-mail або логіна та пароля	Новий користувач зберігається в базі даних і може використовувати створений обліковий запис для входу
Головна панель	Відображення планового бюджету, доходів, витрат, залишку коштів і відсотка використання бюджету	Показники коректно оновлюються після додавання або зміни фінансових операцій
Модуль операцій	Додавання доходів і витрат із зазначенням суми, дати, категорії, опису та способу оплати	Операція зберігається в базі даних, відображається в журналі та впливає на фінансові показники бюджету
Модуль бюджету	Створення бюджету на вибраний період із зазначенням планової суми	Бюджет зберігається, стає доступним для вибору та використовується під час додавання операцій
Модуль категорій	Додавання категорій доходів і витрат, перевірка їх доступності під час створення операції	Категорії зберігаються в системі та відображаються у відповідних списках вибору
Модуль лімітів	Встановлення обмежень витрат для окремих категорій у межах бюджету	Ліміти зберігаються та враховуються під час аналізу витрат за категоріями
Модуль аналітики	Розрахунок доходів, витрат, залишку бюджету, частки витрат за категоріями та відсотка використання бюджету	Аналітичні показники відповідають даним, збереженим у базі даних
Модуль сповіщень	Контроль наближення до ліміту або його перевищення	У разі досягнення встановленого порогу система формує відповідне повідомлення для користувача
Модуль звітів	Формування звіту за вибраний період на основі доходів, витрат, бюджету та категорій	Звіт створюється з коректними підсумковими показниками
Модуль експорту	Підготовка даних у табличному форматі для подальшого збереження	Дані експортуються у файл без втрати основної інформації про операції
База даних	Збереження користувачів, бюджетів, категорій, операцій, лімітів, сповіщень і звітів	Дані залишаються доступними після перезапуску програми

Як видно з таблиці 4.1, план тестування охоплює всі ключові частини системи. Особливу увагу приділено модулям, які безпосередньо впливають на правильність фінансового обліку: операціям, бюджетам, категоріям, лімітам, аналітиці та звітам. Це дозволяє перевірити повний цикл роботи програмного забезпечення: від входу користувача до системи до формування підсумкового звіту.

Методика тестування передбачає виконання функціональних, інтеграційних, інтерфейсних і приймальних перевірок. Функціональне тестування використовується для перевірки окремих можливостей системи: авторизації, реєстрації, додавання операцій, створення бюджету, встановлення лімітів і формування звітів. Інтеграційне тестування дає змогу перевірити правильність взаємодії між графічним інтерфейсом, програмною логікою та базою даних. Інтерфейсне тестування спрямоване на оцінювання зручності форм введення, таблиць, вкладок, кнопок і повідомлень. Приймальне тестування підтверджує придатність системи для виконання основних користувацьких сценаріїв. [12; 24]

Для оцінювання результатів тестування визначено критерії, які дозволяють встановити, чи відповідає програмне забезпечення поставленим вимогам. Основні критерії оцінювання наведено в таблиці 4.2.

Таблиця 4.2

Критерії оцінювання результатів тестування

Критерій оцінювання	Метод перевірки	Допустимий результат
Коректність авторизації	Введення правильних і неправильних облікових даних	Правильні дані відкривають систему, неправильні викликають повідомлення про помилку
Коректність реєстрації	Створення нового користувача з унікальним логіном або e-mail	Дані користувача зберігаються без дублювання облікових записів
Правильність додавання доходів	Внесення тестового доходу із сумою, датою та категорією	Дохід зберігається в журналі операцій і збільшує загальну суму доходів
Правильність додавання витрат	Внесення тестової витрати із сумою, датою та категорією	Витрата зберігається в журналі операцій і збільшує загальну суму витрат
Точність фінансових розрахунків	Порівняння ручних розрахунків із показниками системи	Сума доходів, витрат, залишку та відсотка використання бюджету збігається з очікуваною

Продовження таблиці 4.2

Робота лімітів	Додавання витрат у категорію з установленим лімітом	У разі наближення до ліміту або його перевищення формується попередження

Стабільність інтерфейсу	Багаторазове перемикання вкладок, відкриття форм і виконання операцій	Інтерфейс не зависає та не завершує роботу аварійно
Збереження даних	Перезапуск програми після створення бюджетів, категорій і операцій	Раніше внесені дані залишаються доступними
Формування звітів	Створення звіту після внесення доходів і витрат	Звіт містить коректні дані за вибраний період
Експорт даних	Формування табличного файлу з журналом операцій	Експортований файл містить основні дані про операції без спотворення інформації
Контроль введення	Спроба зберегти операцію з порожніми або некоректними даними	Система не зберігає некоректний запис і повідомляє користувача про помилку

З таблиці 4.2 видно, що оцінювання результатів тестування охоплює як технічну коректність роботи системи, так і зручність її практичного використання. Перевіряється не лише факт виконання окремої функції, а й правильність зміни пов’язаних даних: наприклад, після додавання витрати має оновитися журнал операцій, загальна сума витрат, залишок бюджету, стан лімітів і відповідні аналітичні показники.

Тестування виконується на основі типових сценаріїв використання системи. Спочатку перевіряється створення або вхід до облікового запису користувача. Після цього створюється бюджет на вибраний період, додаються категорії доходів і витрат, встановлюються ліміти для окремих категорій. Далі користувач додає кілька фінансових операцій, переглядає головну панель, аналізує структуру витрат, перевіряє наявність сповіщень і формує звіт. Такий порядок тестування дає змогу перевірити систему в умовах, наближених до реальної експлуатації.

Окремо перевіряється робота з некоректними даними. До таких випадків належать введення неправильного пароля, спроба створити операцію без вибору категорії, введення нульової або від’ємної суми, вибір некоректної дати, створення дубльованого користувача або збереження операції без прив’язки до бюджету. У кожному з цих випадків система повинна відхилити некоректну дію та повідомити користувача про помилку без аварійного завершення роботи.

4.2 Тестування інтерфейсних модулів системи

Тестування інтерфейсних модулів інформаційної системи обліку побутових витрат проводилося шляхом послідовного виконання основних сценаріїв користувача. Спочатку було перевірено вікно авторизації, після цього виконано перехід до форми реєстрації, відкриття головного вікна системи, перегляд фінансових показників, роботу з вкладками та відображення аналітичних блоків.

Початковим елементом роботи із системою є вікно авторизації користувача. На ньому розміщено назву програмної системи, поля введення логіна і пароля, кнопки входу та переходу до реєстрації. Також передбачено демонстраційні облікові записи user/user123 та admin/admin123, що спрощує перевірку системи під час тестування. Вигляд вікна авторизації наведено на рисунку 4.1.

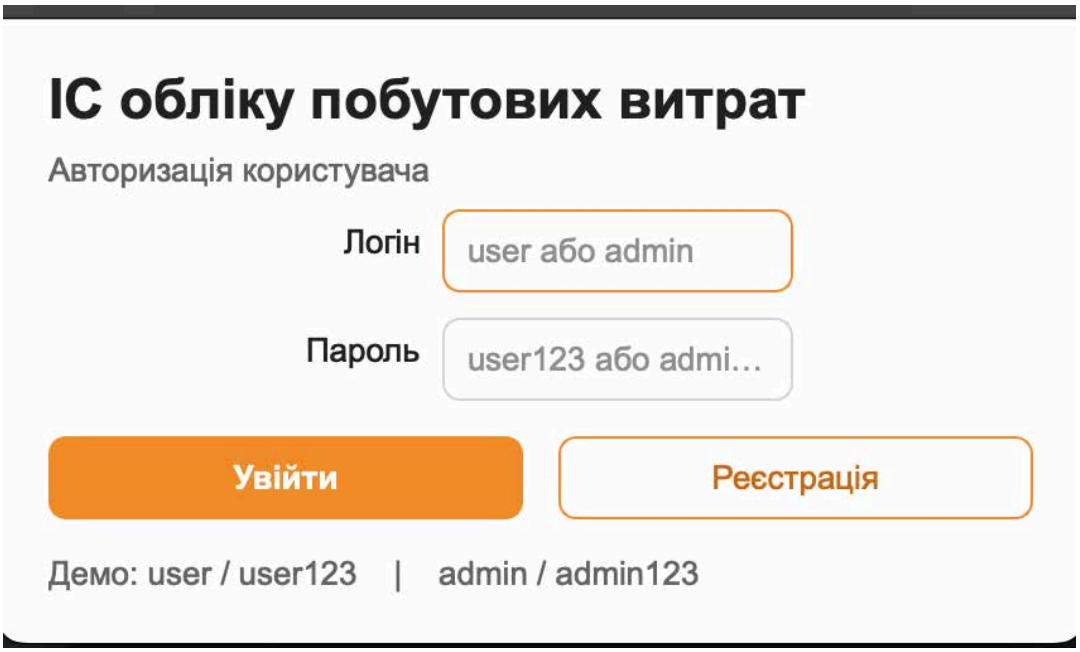


Рис. 4.1 – Вікно авторизації користувача інформаційної системи обліку побутових витрат

Під час тестування було перевірено два сценарії входу: звичайного користувача та адміністратора. За правильного введення логіна і пароля система відкривала головне вікно. У разі введення некоректних даних користувач

отримував повідомлення про помилку. Така перевірка підтверджує працездатність базового механізму автентифікації.

Наступним етапом перевірено форму реєстрації нового користувача, яка наведена на рисунку 4.2. Вона містить поля для введення ПІБ, логіна та пароля. Після натискання кнопки «Зареєструвати» система перевіряє заповнення полів, створює запис користувача та зберігає його в базі даних.

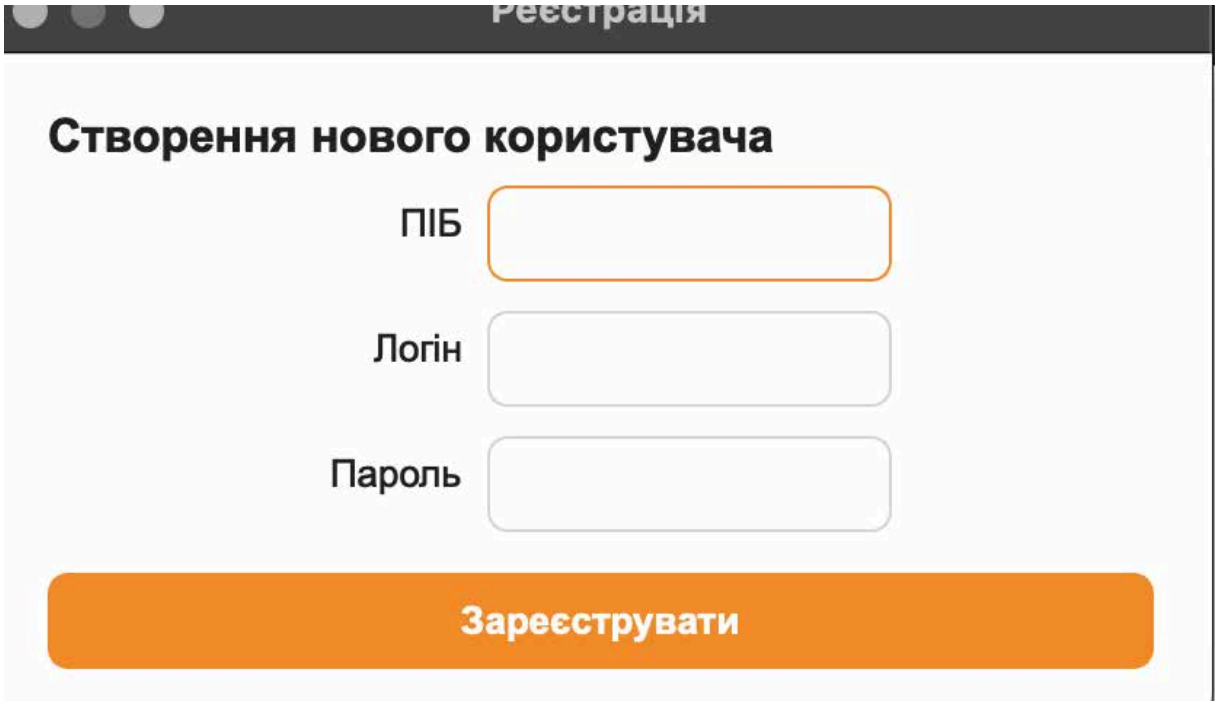


Рис. 4.2 – Вікно створення нового користувача

Під час тестування форми реєстрації перевірено коректність заповнення полів, реакцію системи на порожні значення та можливість створення нового облікового запису. Після успішної реєстрації користувач може виконати вхід до системи з новими даними. Це підтверджує зв’язок між інтерфейсом реєстрації, модулем авторизації та таблицею користувачів у базі даних.

Після входу відкривається головне вікно системи. Його інтерфейс побудовано за вкладковим принципом. У верхній частині розміщено вкладки «Головна», «Операції», «Бюджет і ліміти», «Категорії» та «Звіти». Така структура забезпечує швидкий доступ до основних функціональних частин системи. Головну панель користувача наведено на рисунку 4.3.

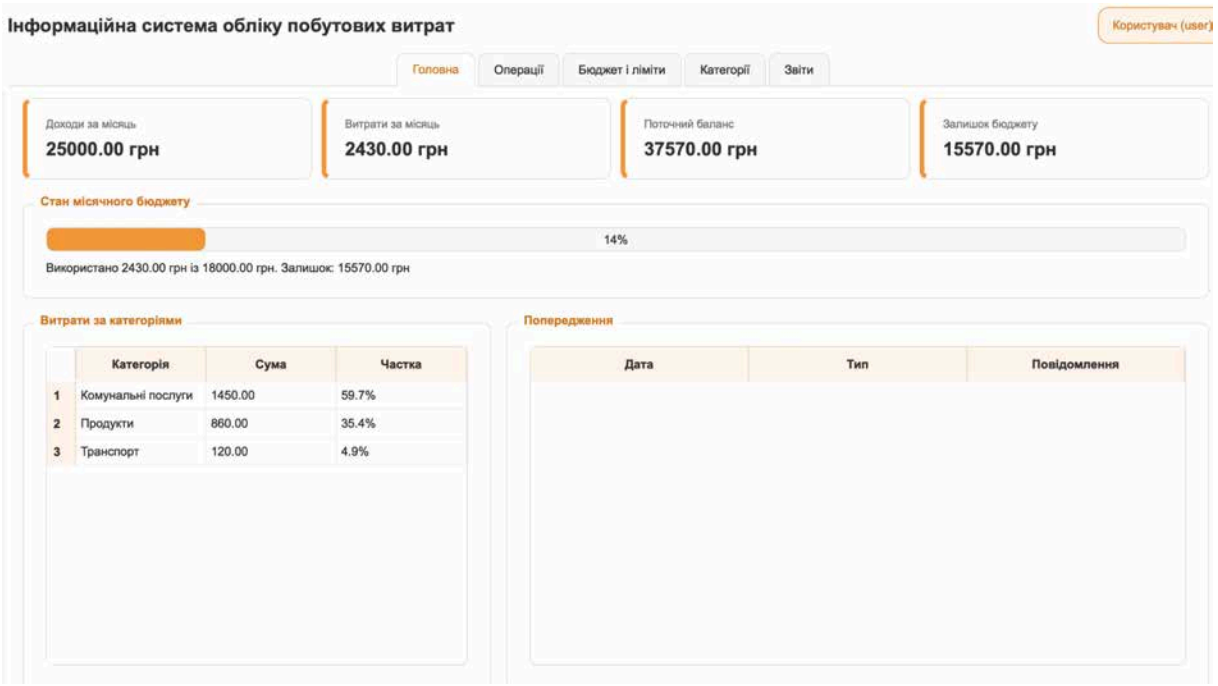


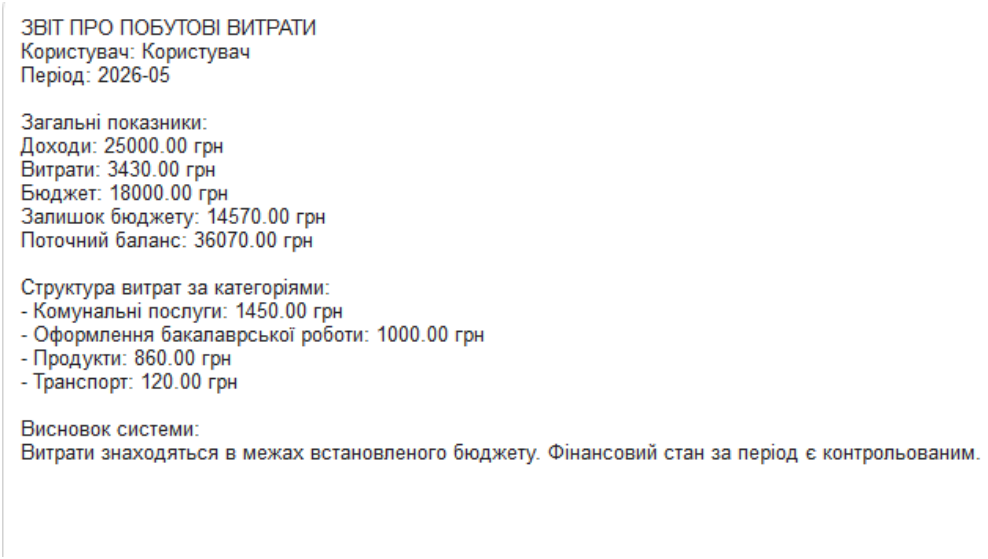
Рис. 4.3 – Головна панель інформаційної системи обліку побутових витрат

На головній панелі відображаються ключові фінансові показники: доходи за місяць, витрати за місяць, поточний баланс і залишок бюджету. У наведеному прикладі система показує доходи за місяць 25000,00 грн, витрати 2430,00 грн, поточний баланс 37570,00 грн та залишок бюджету 15570,00 грн. Також відображено стан місячного бюджету у вигляді індикатора виконання, де показано використання 14 % від планового бюджету. Наявність таких елементів дає користувачеві змогу швидко оцінити поточний фінансовий стан.

Окремо перевірено блок «Витрати за категоріями». У ньому система групує витрати за напрямами, наприклад комунальні послуги, продукти та транспорт. Для кожної категорії відображається сума та частка від загальних витрат. Це дозволяє користувачеві не лише бачити загальний обсяг витрат, а й аналізувати їхню структуру. У правій частині головної панелі розміщено блок попереджень, який призначений для інформування про перевищення встановлених лімітів або інші важливі події.

Було перевірено роботу вкладки «Звіти», яка призначена для формування підсумкової інформації за обраний місяць. Користувач може вибрати період,

натиснути кнопку «Сформувати звіт» та отримати узагальнені дані про доходи, витрати, бюджет, залишок коштів і структуру витрат за категоріями. Також передбачено можливість збереження сформованого звіту у текстовому форматі.


ЗВІТ ПРО ПОБУТОВІ ВИТРАТИ
Користувач: Користувач
Період: 2026-05

Загальні показники:
Доходи: 25000.00 грн
Витрати: 3430.00 грн
Бюджет: 18000.00 грн
Залишок бюджету: 14570.00 грн
Поточний баланс: 36070.00 грн

Структура витрат за категоріями:
- Комунальні послуги: 1450.00 грн
- Оформлення бакалаврської роботи: 1000.00 грн
- Продукти: 860.00 грн
- Транспорт: 120.00 грн

Висновок системи:
Витрати знаходяться в межах встановленого бюджету. Фінансовий стан за період є контрольованим.

Рис. 4.4 – Вкладка формування звітів

Також перевірено вкладку «Категорії», яка використовується для перегляду та керування категоріями доходів і витрат. У цій частині системи відображається список наявних категорій, їх тип та ознака системності. Користувач може додавати нові категорії, редагувати або видаляти вже створені записи, що забезпечує гнучке налаштування структури обліку побутових витрат.

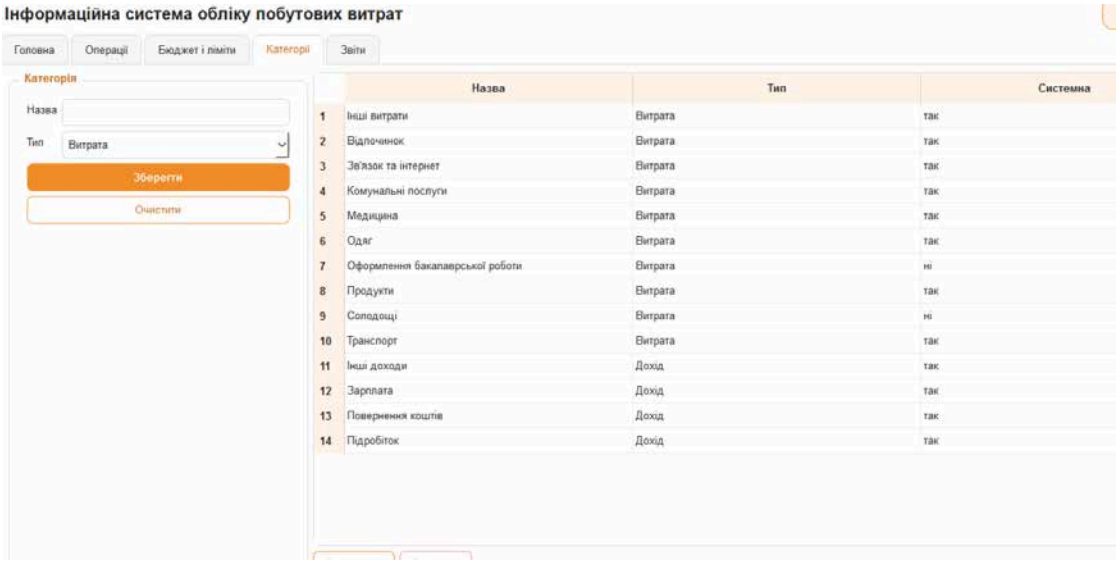


Рис. 4.5 – Вкладка керування категоріями

Після виконання наведених сценаріїв підтверджено, що інтерфейс системи є логічним і придатним для практичного використання. Основні функції винесено в окремі вкладки, фінансові показники подано у вигляді карток, а аналітичні дані згруповано в таблиці. Це спрощує роботу користувача та зменшує кількість дій, потрібних для отримання інформації про стан бюджету.

4.3 Результати тестування та аналіз ефективності систем

За результатами проведеного тестування встановлено, що програмне забезпечення для обліку побутових витрат з аналізом бюджету виконує основні функції, визначені на етапі аналізу вимог. Система забезпечує реєстрацію та авторизацію користувача, створення бюджетів, додавання доходів і витрат, керування категоріями, встановлення лімітів, формування сповіщень, перегляд аналітичних показників і створення звітів за вибраний період.

Під час перевірки основних сценаріїв роботи не виявлено критичних помилок, які блокують використання програмного забезпечення. Інтерфейс коректно реагує на дії користувача, перемикання між розділами виконується без помітних затримок, а дані на головній панелі оновлюються після внесення нових фінансових операцій.

У процесі тестування перевірялися основні модулі системи: авторизація, реєстрація, робота з бюджетами, доходами, витратами, категоріями, лімітами, сповіщеннями, аналітикою, звітами та базою даних. Узагальнені результати перевірки наведено в таблиці 4.3.

Таблиця 4.3

Результати тестування основних функцій системи

Функція системи	Результат перевірки	Оцінка
Авторизація користувача	Вхід до системи виконується після введення правильних облікових даних	Працює
Реєстрація нового користувача	Новий обліковий запис створюється після заповнення форми реєстрації	Працює

Головна панель	Плановий бюджет, доходи, витрати, залишок і відсоток використання бюджету відображаються коректно	Працює
Створення бюджету	Користувач може створити бюджет із назвою, періодом і плановою сумою	Працює
Додавання доходів	Дохід зберігається в базі даних і враховується під час розрахунку фінансових показників	Працює
Додавання витрат	Витрата зберігається в журналі операцій і зменшує залишок бюджету	Працює
Керування категоріями	Нові категорії доходів і витрат додаються та доступні під час створення операцій	Працює
Контроль лімітів	Система перевіряє витрати за категоріями відповідно до встановлених обмежень	Працює
Формування сповіщень	У разі наближення до ліміту або його перевищення створюється повідомлення для користувача	Працює
Аналітика бюджету	Система розраховує доходи, витрати, залишок і структуру витрат за категоріями	Працює
Формування звітів	Звіт створюється на основі збережених операцій, бюджету та категорій	Працює
Експорт даних	Журнал операцій може бути збережений у табличному форматі	Працює
База даних	Дані користувачів, бюджетів, категорій, операцій, лімітів, сповіщень і звітів зберігаються після перезапуску програми	Працює

Як видно з таблиці 4.3, усі основні функції системи пройшли перевірку та виконуються коректно. Особливо важливим результатом є правильна взаємодія між інтерфейсом користувача, програмною логікою та локальною базою даних. Після додавання доходу або витрати система автоматично оновлює журнал операцій, перераховує залишок бюджету, змінює аналітичні показники та перевіряє встановлені ліміти.

Ефективність розробленої системи полягає в автоматизації процесів, які при ручному веденні обліку потребують значних витрат часу. У разі використання паперових записів або звичайних електронних таблиць користувач повинен самостійно підраховувати суму доходів, витрат, залишок коштів, частку витрат за категоріями та рівень використання бюджету. У розробленому застосунку ці показники формуються автоматично на основі

внесених операцій, що зменшує ризик арифметичних помилок і підвищує зручність контролю особистих фінансів.

Для оцінювання ефективності системи було проаналізовано її основні переваги порівняно з ручним веденням обліку. Результати такого порівняння наведено в таблиці 4.4.

Таблиця 4.4

Оцінювання ефективності розробленої системи

Критерій оцінювання	Ручний облік	Розроблена система
Внесення доходів і витрат	Записи виконуються вручну в зошиті або таблиці	Операції додаються через форму з вибором категорії, дати та суми
Розрахунок залишку бюджету	Користувач виконує підрахунок самостійно	Залишок бюджету обчислюється автоматично
Контроль категорій	Потребує ручного групування витрат	Витрати автоматично прив’язуються до вибраних категорій
Аналіз структури витрат	Необхідно самостійно підсумовувати витрати за категоріями	Система відображає структуру витрат на основі збережених операцій
Контроль лімітів	Користувач самостійно порівнює витрати з планом	Система формує попередження при наближенні до ліміту або його перевищенні
Збереження історії	Є ризик втрати або пошкодження записів	Дані зберігаються в локальній базі даних
Формування звітів	Потребує ручного підрахунку та оформлення	Звіт формується автоматично за вибраний період
Зручність використання	Пошук і підрахунок даних займають більше часу	Основні показники доступні на головній панелі

З таблиці 4.4 видно, що розроблена система є ефективнішою за ручний спосіб ведення фінансового обліку, оскільки автоматизує основні операції з даними. Користувач отримує не лише журнал доходів і витрат, а й готові підсумкові показники, аналітику за категоріями, контроль лімітів і звітність. Це скорочує час роботи з фінансовими записами та робить процес керування бюджетом більш наочним.

Окремо було перевірено стабільність роботи програмного забезпечення. Під час багаторазового відкриття розділів, створення операцій, зміни активного

бюджету, додавання категорій і формування звітів аварійного завершення роботи програми не спостерігалося. У випадках некоректного введення даних система не зберігає помилковий запис, а повідомляє користувача про необхідність виправлення введеної інформації. Це підтверджує відповідність застосунку вимогам надійності та зручності використання.

Склад інсталяційного пакету розробленої системи наведено в таблиці 4.5. До нього входять основні програмні файли, модулі інтерфейсу, модулі роботи з базою даних, файл залежностей і службова документація для запуску застосунку.

Таблиця 4.5

Склад інсталяційного пакету програмного забезпечення

Елемент пакету	Призначення
run.py	Головний файл запуску програмного застосунку
requirements.txt	Перелік бібліотек Python, необхідних для роботи системи
app/main.py	Модуль ініціалізації застосунку та запуску графічного інтерфейсу
app/database.py	Модуль створення локальної бази даних і виконання операцій збереження та читання даних
app/auth.py	Модуль реєстрації, авторизації та перевірки паролів користувачів
app/ui.py	Основний модуль графічного інтерфейсу користувача
app/styles.py	Файл стилів інтерфейсу, що визначає оформлення вікон, кнопок, таблиць і карток
data/budget_app.db	Локальна база даних SQLite, яка створюється автоматично під час першого запуску
README.md	Інструкція із встановлення залежностей, запуску програми та короткого опису функцій
exports/	Каталог для збереження експортованих звітів або табличних файлів

Наведений у таблиці 4.5 склад пакету є достатнім для розгортання системи на робочій станції користувача. Для запуску програми необхідно встановити Python, додати залежності з файлу requirements.txt і виконати головний файл run.py. Локальна база даних створюється автоматично, тому користувачу не потрібно окремо встановлювати сервер баз даних або налаштовувати мережеве з’єднання.

Таким чином, результати тестування підтвердили працездатність і практичну придатність розробленого програмного забезпечення. Система забезпечує основні функції обліку побутових витрат, автоматизує розрахунок фінансових показників, підтримує контроль бюджету та лімітів, зберігає дані в локальній базі й надає користувачу зручні засоби аналізу власних фінансів. Це дає підстави вважати, що розроблений застосунок відповідає поставленій меті та може використовуватися для щоденного ведення особистого фінансового обліку.

4.4 Висновки до четвертого розділу

У четвертому розділі проведено тестування інформаційної системи обліку побутових витрат та оцінено результати її роботи. Було сформовано план тестування програмних модулів, визначено критерії оцінювання результатів і перевірено основні інтерфейсні сценарії користувача.

Під час тестування перевірено вікно авторизації, форму реєстрації нового користувача та головну панель системи. На основі рисунків 4.1-4.3 показано, що система має зрозумілу структуру інтерфейсу, підтримує роботу з обліковими записами та відображає основні фінансові показники у зручному вигляді. Головна панель забезпечує швидкий перегляд доходів, витрат, поточного балансу, залишку бюджету, стану місячного бюджету, витрат за категоріями та попереджень.

Результати тестування підтвердили працездатність основних функцій програмної системи. Авторизація, реєстрація, навігація між вкладками, розрахунок фінансових показників, аналіз витрат за категоріями та відображення попереджень працюють відповідно до поставлених вимог. Розроблена система може використовуватися як локальний застосунок для ведення обліку побутових витрат, контролю бюджету та отримання узагальненої фінансової інформації.

Практична цінність системи полягає у спрощенні процесу ведення особистого бюджету. Користувач отримує єдине програмне середовище для внесення фінансових операцій, перегляду стану бюджету, аналізу витрат і формування звітів. У подальшому систему можна розширити модулями імпорту банківських операцій, побудови детальніших графіків, резервного копіювання бази даних і синхронізації між кількома пристроями.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи спроєктовано та розроблено інформаційну систему обліку побутових витрат. Розроблена система забезпечує автоматизоване ведення доходів і витрат, створення бюджетів, групування операцій за категоріями, контроль лімітів, відображення попереджень і формування звітної інформації.

У першому розділі досліджено предметну область обліку побутових витрат, визначено основні інформаційні об’єкти та проаналізовано існуючі програмні рішення. Встановлено, що готові застосунки мають корисні функції, але не завжди забезпечують відкритість внутрішньої структури, локальну адаптацію та навчальну демонстрацію алгоритмів обліку. На основі аналізу сформовано вимоги до розроблюваної системи.

У другому розділі виконано проєктування інформаційного та програмного забезпечення. Побудовано логічну модель даних, діаграму класів, діаграми кооперації, компонентну та пакетну структуру. Побудовані UML-моделі дали змогу формалізувати ролі користувачів, сценарії роботи, структуру класів і взаємодію програмних модулів.

У третьому розділі обґрунтовано вибір технологічних засобів реалізації, зокрема Python, PyQt6 і SQLite. Сформовано фізичну модель даних, розроблено алгоритми авторизації, додавання доходів і витрат, аналізу бюджету, контролю лімітів і формування звітів. Також подано діаграму розгортання, яка показує можливість як локального, так і клієнт-серверного варіанта використання системи.

У четвертому розділі проведено тестування програмних модулів та інтерфейсних сценаріїв. Перевірено авторизацію, реєстрацію, головну панель, розрахунок фінансових показників, аналіз витрат за категоріями та роботу попереджень. Результати тестування підтвердили працездатність системи та її відповідність поставленим вимогам.

Практичне значення розробки полягає у створенні локального програмного засобу для щоденного контролю побутових фінансів. Система може бути використана як навчальний прототип або як основа для подальшого розвитку повноцінного застосунку особистого фінансового обліку. Перспективами розвитку є імпорт банківських операцій, підтримка кількох бюджетів, розширені графіки, резервне копіювання, синхронізація між пристроями та мобільна версія інтерфейсу.

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

7. Riverbank Computing. PyQt6 Reference Guide. URL: https://www.riverbankcomputing.com/static/Docs/PyQt6/.

8. The Qt Company. Qt 6 Documentation. URL: https://doc.qt.io/qt-6/.

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

10. Welling L., Thomson L. PHP and MySQL Web Development. 5th ed. Boston : Addison-Wesley, 2016.

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

12. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 9th ed. New York : McGraw-Hill, 2020.

13. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2016.

14. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. Boston : Prentice Hall, 2017.

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

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

17. Date C. J. An Introduction to Database Systems. 8th ed. Boston : Pearson, 2003.

18. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 13th ed. Boston : Cengage Learning, 2019.

19. Beazley D., Jones B. K. Python Cookbook. 3rd ed. Sebastopol : O'Reilly Media, 2013.

20. Lutz M. Learning Python. 5th ed. Sebastopol : O'Reilly Media, 2013.

21. Pilgrim M. Dive Into Python 3. New York : Apress, 2009.

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

23. Rubin K. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Boston : Addison-Wesley, 2012.

24. IEEE Computer Society. Guide to the Software Engineering Body of Knowledge (SWEBOK Guide). Version 4.0. Los Alamitos : IEEE Computer Society, 2024.

25. Microsoft. CSV file format and data exchange basics. URL: https://support.microsoft.com/.

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

27. Monefy. Money Manager App. URL: https://monefy.me/.

28. Wallet. BudgetBakers. URL: https://budgetbakers.com/.

29. Spendee. Personal finance app. URL: https://www.spendee.com/.

30. HomeBudget. Anishu, Inc. URL: https://anishu.com/homebudget/.

ДОДАТОК А

Фрагмент програмного коду створення бази даних

```
import sqlite3
from datetime import datetime

DB_NAME = "household_expenses.db"

class DatabaseManager:
    def __init__(self, db_path: str = DB_NAME):
        self.db_path = db_path
        self.create_tables()

    def connect(self):
        connection = sqlite3.connect(self.db_path)
        connection.row_factory = sqlite3.Row
        connection.execute("PRAGMA foreign_keys = ON")
        return connection

    def create_tables(self):
        with self.connect() as conn:
            cursor = conn.cursor()
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS users (
                    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    username TEXT NOT NULL UNIQUE,
                    email TEXT UNIQUE,
                    password_hash TEXT NOT NULL,
                    registered_at TEXT NOT NULL,
                    user_status TEXT NOT NULL DEFAULT 'active'
                )
            """)
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS budgets (
                    budget_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    user_id INTEGER NOT NULL,
                    budget_name TEXT NOT NULL,
                    period_type TEXT NOT NULL,
                    start_date TEXT NOT NULL,
                    end_date TEXT NOT NULL,
                    planned_amount REAL NOT NULL,
                    remaining_amount REAL NOT NULL,
                    FOREIGN KEY (user_id) REFERENCES users(user_id)
                )
            """)
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS categories (
                    category_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    name_category TEXT NOT NULL,
                    type_category TEXT NOT NULL,
                    is_system INTEGER NOT NULL DEFAULT 0
                )
            """)
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS transactions (
                    transaction_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    user_id INTEGER NOT NULL,
                    budget_id INTEGER NOT NULL,
                    category_id INTEGER NOT NULL,
                    amount REAL NOT NULL,
                    transaction_date TEXT NOT NULL,
                    description TEXT,
                    type_transaction TEXT NOT NULL,
                    FOREIGN KEY (user_id) REFERENCES users(user_id),
                    FOREIGN KEY (budget_id) REFERENCES budgets(budget_id),
                    FOREIGN KEY (category_id) REFERENCES categories(category_id)
                )
            """)
            conn.commit()
```

ДОДАТОК Б

Фрагмент програмного коду сервісу обліку операцій і контролю бюджету

```
class TransactionService:
    def __init__(self, database):
        self.database = database

    def add_transaction(self, user_id, budget_id, category_id, amount,
                        transaction_date, description, operation_type):
        if amount <= 0:
            raise ValueError("Сума операції повинна бути більшою за нуль")

        with self.database.connect() as conn:
            cursor = conn.cursor()
            cursor.execute("""
                INSERT INTO transactions (
                    user_id, budget_id, category_id, amount,
                    transaction_date, description, type_transaction
                )
                VALUES (?, ?, ?, ?, ?, ?, ?)
            """, (
                user_id, budget_id, category_id, amount,
                transaction_date, description, operation_type
            ))
            transaction_id = cursor.lastrowid

            if operation_type == "income":
                cursor.execute("""
                    INSERT INTO income_transactions (transaction_id, income_source,
income_method)
                    VALUES (?, ?, ?)
                """, (transaction_id, "manual", "cash"))
            else:
                cursor.execute("""
                    INSERT INTO expense_transactions (transaction_id, payment_method,
                                                      expense_status, is_planned)
                    VALUES (?, ?, ?, ?)
                """, (transaction_id, "cash", "confirmed", 0))

            self.recalculate_budget(cursor, budget_id)
            self.check_category_limit(cursor, user_id, budget_id, category_id)
            conn.commit()

    def recalculate_budget(self, cursor, budget_id):
        cursor.execute("""
            SELECT
                COALESCE(SUM(CASE WHEN type_transaction='income' THEN amount ELSE 0 END), 0) AS
income,
                COALESCE(SUM(CASE WHEN type_transaction='expense' THEN amount ELSE 0 END), 0) AS
expense
            FROM transactions
            WHERE budget_id = ?
        """, (budget_id,))
        row = cursor.fetchone()
        remaining = row["income"] - row["expense"]
        cursor.execute("UPDATE budgets SET remaining_amount = ? WHERE budget_id = ?",
                       (remaining, budget_id))
```

ДОДАТОК В

Фрагмент програмного коду формування аналітики та звіту

```
import csv
from datetime import datetime

class ReportService:
    def __init__(self, database):
        self.database = database

    def build_budget_summary(self, user_id, budget_id):
        with self.database.connect() as conn:
            cursor = conn.cursor()
            cursor.execute("""
                SELECT type_transaction, SUM(amount) AS total
                FROM transactions
                WHERE user_id = ? AND budget_id = ?
                GROUP BY type_transaction
            """, (user_id, budget_id))
            rows = cursor.fetchall()

        income = 0.0
        expense = 0.0
        for row in rows:
            if row["type_transaction"] == "income":
                income = row["total"] or 0.0
            if row["type_transaction"] == "expense":
                expense = row["total"] or 0.0

        balance = income - expense
        return {
            "income": income,
            "expense": expense,
            "balance": balance,
            "created_at": datetime.now().isoformat(timespec="seconds")
        }

    def export_csv_report(self, user_id, budget_id, file_path):
        with self.database.connect() as conn:
            cursor = conn.cursor()
            cursor.execute("""
                SELECT t.transaction_date, c.name_category, t.type_transaction,
                       t.amount, t.description
                FROM transactions t
                JOIN categories c ON t.category_id = c.category_id
                WHERE t.user_id = ? AND t.budget_id = ?
                ORDER BY t.transaction_date
            """, (user_id, budget_id))
            rows = cursor.fetchall()

        with open(file_path, "w", newline="", encoding="utf-8-sig") as csv_file:
            writer = csv.writer(csv_file, delimiter=";")
            writer.writerow(["Дата", "Категорія", "Тип", "Сума", "Опис"])
            for row in rows:
                writer.writerow([
                    row["transaction_date"],
                    row["name_category"],
                    row["type_transaction"],
                    row["amount"],
                    row["description"] or ""
                ])
```

ДОДАТОК Г

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

Я, Гановський Артем Сергійович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему “Програмне забезпечення для обліку побутових витрат з аналізом бюджету”.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
   - o [ ] інструменти генеративного ШІ не використовувалися.
   - o [✓] 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. _____ **Артем ГАНОВСЬКИЙ**

(підпис)