

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ  
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

**ПОГОДЖЕНО  
Декан факультету**

**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ  
Завідувач кафедри інформаційних  
систем і технологій**

\_\_\_\_\_  
(підпис) **Ігор БОЛБОТ**

\_\_\_\_\_  
(підпис) **Михайло ШВИДЕНКО**

« \_\_\_\_ » \_\_\_\_\_ 2026 р.

« \_\_\_\_ » \_\_\_\_\_ 2026 р.

**БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

**на тему: «Інформаційна система контролю особистих фінансових ресурсів»**

Спеціальність «126 Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційно управляючі системи та технології»

**Гарант освітньої програми**

д.е.н., професор

\_\_\_\_\_  
(підпис) **Максим МОКРІЄВ**

**Керівник бакалаврської  
кваліфікаційної роботи**

д.ф. з к.н.

\_\_\_\_\_  
(підпис) **Роман ЗОЛОТУХА**

**Виконав**

\_\_\_\_\_  
(підпис) **Євгеній АНТІПОВ**

**Київ-2026**

# НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

**ЗАТВЕРДЖУЮ**  
Завідувач кафедри інформаційних систем і  
технологій

к.е.н., доцент \_\_\_\_\_ Михайло ШВИДЕНКО  
(підпис)

«\_\_\_» \_\_\_\_\_ 2026 р.

## **ЗАВДАННЯ** **ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ** **ЗДОБУВАЧУ**

**Антіпов Євгеній Олександрович**

Спеціальність 126 – «Інформаційні системи та технології»

ОП «Інформаційно управляючі системи та технології»

Тема бакалаврської кваліфікаційної роботи «Інформаційна система контролю особистих фінансових ресурсів» затверджена наказом від «19» грудня 2025 р. № 3205 «С2»

Термін подання завершеної роботи на кафедру «25» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): надання інформації про облік власних доходів та витрат у вигляді категорій, а також аналіз своїх фінансових даних у вигляді звітності

Перелік питань, що підлягають дослідженню:

- 1. Системний аналіз предметної області
- 2. Моделювання системи
- 3. Розробка системи
- 4. Результати дослідження.

Дата видачі завдання «06» січня 2026 р.

**Керівник бакалаврської  
кваліфікаційної роботи**

\_\_\_\_\_  
(підпис)

**Роман ЗОЛОТУХА**

**Завдання взяв до  
виконання**

\_\_\_\_\_  
(підпис)

**Євгеній АНТІПОВ**

# ЗМІСТ

|                                                                 |    |
|-----------------------------------------------------------------|----|
| ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ                                       | 5  |
| ВСТУП                                                           | 6  |
| 1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ                                     | 9  |
| 1.1 Опис предметної області                                     | 9  |
| 1.2 Огляд існуючих рішень                                       | 14 |
| 1.3 Постановка завдання                                         | 20 |
| 1.4 Функціональні та нефункціональні вимоги                     | 22 |
| 1.5 Вимоги до інтерфейсу користувача                            | 23 |
| 2 МОДЕЛЮВАННЯ СИСТЕМИ                                           | 26 |
| 2.1 Загальні відомості                                          | 26 |
| 2.2 Об'єктне та функціональне моделювання                       | 29 |
| 2.3 Огляд інструментів для реалізації завдань Data Mining       | 38 |
| 2.4 Структура джерела інформації для інтелектуального аналізу   | 40 |
| 3 РОЗРОБКА СИСТЕМИ                                              | 45 |
| 3.1 Логічна модель даних                                        | 45 |
| 3.2 Вибір системи управління базою даних та її реалізація       | 48 |
| 3.3 Архітектура програмного забезпечення                        | 53 |
| 3.4 Використання 1-Rule для класифікації                        | 56 |
| 3.5 Вибір інструментарію для створення програмного забезпечення | 60 |
| 3.6 Алгоритм оптимізації топології                              |    |
| 4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ                                        | 62 |
| 4.1 Вимоги до апаратного та програмного забезпечення            | 62 |
| 4.2 Тестування системи                                          | 65 |
| ВИСНОВКИ                                                        | 72 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ                                      | 74 |
| ДОДАТОК А                                                       | 77 |
| ДОДАТОК Б                                                       | 79 |

## ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

|       |   |                                   |
|-------|---|-----------------------------------|
| БД    | – | База даних                        |
| ОС    | – | Операційна система                |
| ПЗ    | – | Програмне забезпечення            |
| СКБД– |   | Система керування баз даних       |
| API   | – | Application Programming Interface |
| Bash  | – | Bourne Again SHell                |
| MVVM  | – | Model-View-ViewModel              |
| SQL   | – | Structed Query Language           |
| UI    | – | User Interface                    |

## ВСТУП

У сучасних умовах цифровізації суспільства ефективне управління особистими фінансами набуває особливої актуальності. Зростання кількості фінансових операцій, використання електронних платіжних систем та онлайн-банкінгу створюють значні обсяги даних, які потребують впорядкування, аналізу та контролю. Водночас більшість користувачів не володіє достатніми інструментами для систематичного відстеження доходів і витрат, що може призводити до неефективного використання фінансових ресурсів. У зв'язку з цим розробка програмного забезпечення для автоматизації обліку та аналізу особистих фінансів є актуальним завданням як з наукової, так і з практичної точки зору.

**Предметом** дослідження є методи, моделі та програмні засоби реалізації інформаційної системи контролю особистих фінансових ресурсів.

**Об'єктом** дослідження є процес контролю особистих фінансових ресурсів користувача.

**Метою** даної бакалаврської кваліфікаційної роботи є розробка інформаційної системи контролю особистих фінансових ресурсів, що забезпечує облік доходів і витрат, управління категоріями, контроль бюджетних лімітів, формування звітів та візуалізацію фінансової інформації користувача.

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

1. провести аналіз існуючих програмних рішень для обліку та управління особистими фінансами;
2. визначити функціональні та нефункціональні вимоги до розроблюваної системи;
3. розробити концептуальну модель системи та визначити її основні компоненти;

4. спроектувати архітектуру веб-застосунку та структуру бази даних;
5. реалізувати програмне забезпечення із використанням PHP, фреймворку Symfony та системи управління базами даних MySQL;
6. здійснити тестування розробленої системи та оцінити її ефективність.

У процесі дослідження використано **методи дослідження**, моделювання інформаційних систем, аналізу даних та об'єктно-орієнтованого проектування. Для реалізації програмного забезпечення застосовано сучасні підходи до розробки веб-додатків та проектування реляційних баз даних.

**Науково–практична новизна** полягає у розробленні структури веб-орієнтованої інформаційної системи для обліку доходів і витрат, контролю бюджетних лімітів та формування аналітичної звітності користувача.

Отримані результати підтверджують доцільність використання розробленої системи для підвищення ефективності управління особистими фінансами, а також демонструють її практичну значущість у сучасних умовах цифрової економіки.

Структура бакалаврської роботи включає такі основні розділи:

- системний аналіз предметної області – у даному розділі розглядаються існуючі системи обліку фінансів, аналізуються їх переваги та недоліки, а також визначаються основні вимоги до розроблюваного програмного забезпечення;
- проектування системи – у цьому розділі описано архітектуру системи, наведено моделі даних, діаграми та структуру бази даних;
- реалізація системи – розділ присвячений опису процесу розробки програмного забезпечення, використаних технологій та реалізації функціональних можливостей системи;

- аналіз результатів – у завершальному розділі наведено результати тестування системи, оцінено її ефективність та визначено напрями подальшого розвитку.

Бакалаврська робота містить 83 сторінки, включаючи 25 рисунків, 2 таблиці та 2 додатки. Список використаних джерел містить 35 посилань. У роботі використано наукові джерела, що охоплюють сучасні підходи до розробки інформаційних систем, веб-технологій та управління фінансами.

# 1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

## 1.1 Опис предметної області

У сучасному цифровому середовищі управління особистими фінансами стає важливою складовою повсякденного життя користувачів. Збільшення кількості фінансових операцій, використання банківських карток, електронних гаманців та онлайн-сервісів призводить до необхідності систематичного контролю фінансових потоків. Відсутність ефективних інструментів обліку та аналізу фінансів часто спричиняє нераціональне використання коштів, складність у плануванні бюджету та прийнятті фінансових рішень.

Предметна область дослідження охоплює процеси збору, зберігання, обробки та аналізу фінансової інформації користувачів, включаючи облік доходів, витрат, категоризацію транзакцій, формування звітності та візуалізацію даних. Основною метою таких систем є забезпечення користувачів зручними інструментами для контролю фінансового стану та підвищення фінансової грамотності.

Існуючі програмні рішення для обліку фінансів можна умовно поділити на декілька категорій: мобільні додатки, веб-сервіси та настільні програми. Мобільні додатки забезпечують швидкий доступ до фінансової інформації, однак часто мають обмежений функціонал або залежать від платформи. Веб-застосунки є більш універсальними, оскільки доступні з будь-якого пристрою з доступом до Інтернету, але можуть потребувати підключення до мережі. Настільні програми, у свою чергу, забезпечують високу продуктивність, проте мають обмежену мобільність та складність у синхронізації даних [1].

Аналіз сучасних систем обліку фінансів дозволяє виділити їх основні функціональні можливості: ведення обліку доходів і витрат, класифікація фінансових операцій за категоріями, формування статистичних звітів, побудова графіків та діаграм, планування бюджету, а також інтеграція з

банківськими сервісами. Проте значна частина існуючих рішень має ряд недоліків, серед яких складність інтерфейсу, обмежені можливості налаштування, відсутність гнучкої аналітики або необхідність платної підписки.

Важливим аспектом предметної області є питання безпеки та конфіденційності фінансових даних. Оскільки такі системи працюють із чутливою інформацією, необхідно забезпечити надійний захист даних користувачів, включаючи автентифікацію, авторизацію, шифрування та контроль доступу.

З урахуванням проведеного аналізу можна виділити основні вимоги до сучасної системи обліку фінансів: зручний та інтуїтивно зрозумілий інтерфейс, можливість швидкого введення даних, підтримка категоризації операцій, наявність аналітичних інструментів, адаптивність до різних пристроїв, а також забезпечення безпеки даних.

Основними задачами, які повинна вирішувати система відстеження фінансів, є:

- збір та збереження інформації про фінансові операції користувача;
- класифікація доходів і витрат за визначеними категоріями;
- аналіз фінансових потоків та виявлення тенденцій [2];
- формування звітів і візуалізація даних у зручному вигляді;
- планування бюджету та контроль його виконання;
- забезпечення захисту та конфіденційності фінансової інформації.

Існує потреба у створенні ефективного веб-орієнтованого програмного забезпечення для обліку та аналізу особистих фінансів, яке поєднує зручність використання, функціональність та високий рівень безпеки. Розробка такої системи дозволить підвищити ефективність управління фінансовими ресурсами користувачів та забезпечити підтримку прийняття обґрунтованих фінансових рішень.

Предметна область управління особистими фінансами охоплює сукупність процесів, пов'язаних із обліком, аналізом, плануванням та

контролем фінансових ресурсів користувача. У сучасних умовах цифрової економіки ці процеси дедалі більше автоматизуються за допомогою інформаційних систем, що дозволяє підвищити точність обліку, зменшити ймовірність помилок та забезпечити оперативний доступ до фінансової інформації.

Основними сутностями предметної області є фінансові операції, які включають доходи та витрати. Доходи характеризують надходження грошових коштів із різних джерел, таких як заробітна плата, інвестиції або інші фінансові надходження. Витрати, у свою чергу, відображають використання коштів на різні потреби, зокрема оплату послуг, придбання товарів, розваги та інші категорії. Для забезпечення більш детального аналізу фінансові операції класифікуються за категоріями, що дозволяє визначати структуру витрат і доходів.

Важливою складовою предметної області є процес категоризації фінансових операцій. Категорії можуть бути як стандартними, так і користувацькими, що забезпечує гнучкість системи та її адаптацію до індивідуальних потреб. Це дозволяє користувачам більш ефективно аналізувати власні фінансові звички та приймати обґрунтовані рішення щодо оптимізації витрат.

Ще одним ключовим елементом є бюджетування, яке передбачає планування фінансових ресурсів на визначений період часу. Бюджет формується на основі прогнозованих доходів і витрат та дозволяє контролювати фінансовий стан користувача. Система повинна забезпечувати можливість встановлення лімітів витрат для окремих категорій та відстеження їх виконання.

Не менш важливою складовою є аналітична обробка фінансових даних. Вона включає формування звітів, побудову графіків та діаграм, а також виявлення тенденцій у зміні фінансового стану користувача. Візуалізація даних сприяє кращому розумінню фінансових процесів та полегшує прийняття управлінських рішень.

У межах предметної області також важливу роль відіграють користувачі системи, які взаємодіють із програмним забезпеченням через інтерфейс. Основними вимогами до інтерфейсу є зручність, інтуїтивність та швидкість доступу до необхідної інформації. Система повинна забезпечувати можливість додавання, редагування та видалення фінансових операцій, а також надання зведеної інформації у зрозумілому вигляді.

Особливу увагу в предметній області приділяють питанням безпеки та конфіденційності даних. Оскільки система працює з персональною та фінансовою інформацією, необхідно забезпечити захист даних від несанкціонованого доступу, використовуючи механізми автентифікації, авторизації та шифрування.

Веб-додаток, розроблений у цій роботі, зосереджений на забезпеченні такої функціональності гнучким та інтуїтивно зрозумілим способом. На відміну від корпоративних бухгалтерських систем, які вимагають складних знань бухгалтерського обліку, ця система адаптована для звичайних користувачів з невеликим досвідом або без нього у фінансовій сфері. Вона дозволяє користувачам визначати власні категорії доходів і витрат, адаптувати систему до своїх фінансових звичок та отримувати доступ до фінансової інформації в режимі реального часу.

Детальний опис класів та атрибутів предметної області наведено у таблиці 1.1.

Таблиця 1.1

## Опис атрибутів класів предметної області

| Клас предметної області | Атрибут           | Опис                                                   |
|-------------------------|-------------------|--------------------------------------------------------|
| Користувач              | Ім'я користувача  | Ім'я або псевдонім, який обирає користувач             |
|                         | Електронна адреса | Адреса для входу в систему та комунікації              |
|                         | Секретний код     | Засіб захисту облікового запису (пароль)               |
|                         | Фото профілю      | Зображення, що персоніфікує обліковий запис            |
|                         | Дата реєстрації   | Коли користувач вперше зареєструвався в системі        |
| Категорія               | Назва             | Назва групи доходів або витрат, яку бачить користувач  |
|                         | Призначення       | Визначає, чи це категорія для доходів чи для витрат    |
|                         | Власник           | Користувач, який створив і використовує цю категорію   |
| Транзакція              | Сума              | Обсяг коштів, що надійшли або були витрачені           |
|                         | Дата операції     | День, коли відбулася відповідна фінансова дія          |
|                         | Опис              | Коротка примітка чи пояснення щодо транзакції          |
|                         | Категорія         | Група (доходів/витрат), до якої належить ця транзакція |
|                         | Автор запису      | Користувач, який додав цей запис у систему             |

## 1.2 Огляд існуючих рішень

Розглянемо існуючі рішення предметної області.

Одним із популярних програмних рішень у сфері управління особистими фінансами є мобільний застосунок CoinKeeper, який орієнтований на спрощення процесу обліку доходів і витрат та підвищення фінансової дисципліни користувачів. Даний застосунок реалізує інтуїтивний підхід до

управління фінансами, поєднуючи функціональність бюджетування з візуальними інструментами аналізу даних.

Основною особливістю системи є використання графічного інтерфейсу у вигляді «монет», що дозволяє користувачам швидко додавати фінансові операції. Такий підхід значно спрощує процес введення даних і робить взаємодію із системою більш наочною. Завдяки цьому користувач може оперативно фіксувати витрати та доходи без необхідності виконання складних дій.

Система підтримує широкий спектр функціональних можливостей, зокрема ведення обліку фінансових операцій, встановлення бюджетів та контроль витрат. Користувач має можливість задавати ліміти для окремих категорій витрат, а також отримувати повідомлення у випадку перевищення встановленого бюджету. Це сприяє більш ефективному плануванню фінансових ресурсів та запобіганню надмірним витратам [3].

Важливим компонентом CoinKeeper є аналітичний модуль, який забезпечує обробку фінансових даних і їх візуалізацію. Система формує статистичні звіти, графіки та діаграми, що дозволяють користувачам аналізувати структуру витрат, визначати основні категорії споживання та відслідковувати динаміку змін фінансового стану.

Крім того, застосунок підтримує багатоплатформеність та синхронізацію даних між пристроями, що дозволяє використовувати його як для особистого, так і для сімейного бюджетування. Користувачі можуть вести спільний облік фінансів, отримуючи доступ до актуальної інформації з різних пристроїв.

Серед додаткових можливостей системи варто виділити: нагадування про регулярні платежі, управління боргами, підтримку різних валют, експорт даних у форматі CSV та використання засобів захисту доступу, таких як пароль або біометрична автентифікація.

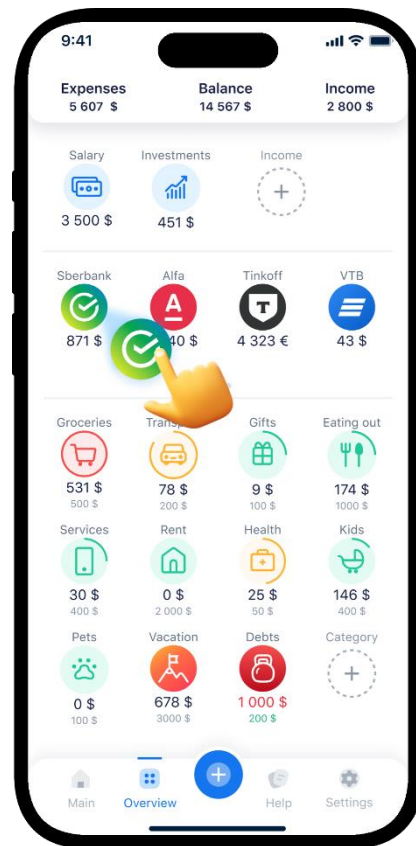


Рис. 1.1 Мобільний застосунок “CoinKeeper”

Разом із тим, незважаючи на значну функціональність, система має певні обмеження. Частина розширених можливостей доступна лише за платною підпискою, що може обмежувати використання для деяких користувачів. Також іноді відзначаються проблеми із синхронізацією даних або питання щодо конфіденційності інформації, що є важливим аспектом для фінансових застосунків.

Основні функціональні можливості системи CoinKeeper можна узагальнити наступним чином:

1. ведення обліку доходів і витрат із використанням зручного візуального інтерфейсу;
2. категоризація фінансових операцій та встановлення лімітів витрат;
3. формування аналітичних звітів і візуалізація фінансових даних;
4. підтримка бюджетування та прогнозування витрат;
5. синхронізація даних між пристроями та підтримка сімейного доступу;

## 6. забезпечення базових механізмів захисту даних користувача.

Таким чином, CoinKeeper є сучасним і зручним інструментом для управління особистими фінансами, який поєднує простоту використання з достатньо широким функціоналом. Його аналіз дозволяє визначити ключові підходи до реалізації систем обліку фінансів, а також виявити напрямки для подальшого вдосконалення при розробці власного програмного забезпечення.

Розглянемо інше програмне рішення на ринку.

Серед сучасних мобільних застосунків для обліку особистих фінансів окрему увагу привертає система 1Money, яка орієнтована на спрощення процесу відстеження витрат та доходів користувачів. Даний програмний продукт реалізує підхід до фінансового обліку, що базується на швидкому введенні даних, наочності інформації та зручності використання.

Основною ідеєю системи є мінімізація складності взаємодії користувача із фінансовими інструментами. Застосунок дозволяє додавати фінансові операції практично миттєво — користувачу достатньо ввести суму витрати або доходу, що значно спрощує процес обліку. Такий підхід сприяє регулярному використанню системи та формуванню звички контролю фінансів.

Інтерфейс системи характеризується простотою та інтуїтивністю. Використання сучасних принципів дизайну забезпечує зручну навігацію та швидкий доступ до основних функцій. Зокрема, фінансові дані відображаються у вигляді графіків та діаграм, що дозволяє користувачу отримати узагальнену інформацію про свої витрати та доходи [4].

Функціональні можливості 1Money охоплюють основні аспекти управління особистими фінансами. Система підтримує бюджетування, що дозволяє користувачам планувати доходи та витрати, встановлювати фінансові обмеження та уникати перевитрат. Крім того, застосунок забезпечує можливість відстеження заощаджень і боргів, що є важливим елементом фінансового планування.

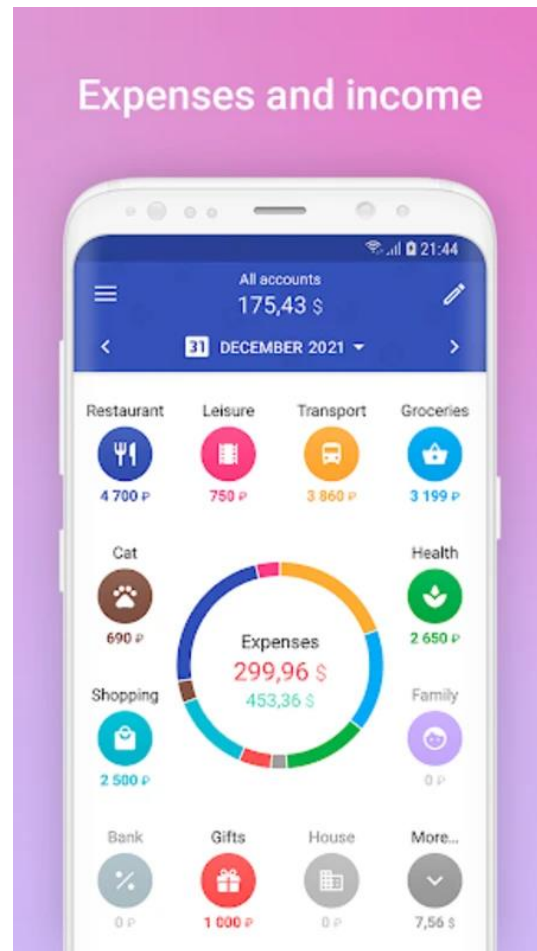


Рис. 1.2 Мобільний застосунок “1Money”

Однією з важливих характеристик системи є підтримка мультивалютності. Це дозволяє користувачам вести облік фінансів у різних валютах із автоматичним оновленням курсів, що особливо актуально в умовах глобалізації та міжнародних фінансових операцій .

Система також надає можливість категоризації фінансових операцій, що забезпечує більш детальний аналіз витрат і доходів. Завдяки цьому користувач може визначати основні напрями витрат та оптимізувати власний бюджет.

Разом із тим, 1Money має певні особливості, які відрізняють її від інших систем. Зокрема, вона орієнтована переважно на ручне введення даних, що з одного боку забезпечує контроль над інформацією, а з іншого — потребує регулярної участі користувача. У порівнянні з більш сучасними системами, що використовують автоматичну синхронізацію з банківськими рахунками або елементи штучного інтелекту, функціональність 1Money є більш базовою .

Крім того, у відгуках користувачів відзначаються окремі проблеми, пов'язані із синхронізацією даних та стабільністю роботи застосунку, що може впливати на надійність зберігання фінансової інформації. Водночас значна кількість користувачів позитивно оцінює простоту інтерфейсу та ефективність базових функцій обліку фінансів.

Основні функціональні можливості системи 1Money можна узагальнити наступним чином:

1. швидке додавання фінансових операцій із мінімальними витратами часу;
2. ведення обліку доходів і витрат;
3. категоризація фінансових операцій;
4. формування графіків і аналітичних звітів;
5. підтримка бюджетування та фінансового планування;
6. відстеження заощаджень, боргів і мультивалютних операцій.

Таким чином, 1Money є зручним інструментом для базового управління особистими фінансами, який орієнтований на простоту використання та швидкість введення даних. Його аналіз дозволяє визначити ефективні підходи до реалізації фінансових систем, а також виявити обмеження, які можуть бути враховані при розробці власного веб-застосунку для відстеження фінансів.

Детальний опис цих програмних застосунків наведено у таблиці 1.2.

Таблиця 1.2

Порівняння програмних застосунків

| Критерій | CoinKeeper | 1Money | Розроблювана система |
|----------|------------|--------|----------------------|
|----------|------------|--------|----------------------|

|                             |                                                                 |                                                        |                                                                             |
|-----------------------------|-----------------------------------------------------------------|--------------------------------------------------------|-----------------------------------------------------------------------------|
| Основне призначення         | Облік особистих фінансів та візуальне управління витратами      | Контроль доходів, витрат та планування бюджету         | Облік доходів/витрат, контроль бюджетних лімітів та аналітична звітність    |
| Інтерфейс користувача       | Яскравий та інтерактивний інтерфейс із перетягуванням категорій | Мінімалістичний та простий інтерфейс                   | Адаптивний веб-інтерфейс (Bootstrap)                                        |
| Ведення бюджету             | Підтримується встановлення бюджетів для категорій               | Підтримується створення бюджетів та фінансових лімітів | Встановлення лімітів на категорії із блокуванням транзакцій при перевищенні |
| Категорії доходів та витрат | Наявні готові категорії з можливістю редагування                | Гнучке створення та налаштування категорій             | Створення користувацьких категорій із вибором іконок                        |
| Аналітика та звіти          | Графіки, діаграми та статистика витрат                          | Детальні звіти та статистика транзакцій                | Звіти за обраний період, статистика транзакцій                              |
| Синхронізація даних         | Підтримка синхронізації між пристроями                          | Синхронізація через обліковий запис користувача        | Централізоване зберігання у БД (MySQL)                                      |
| Робота з кількома рахунками | Підтримується                                                   | Підтримується                                          | Підтримується (наприклад, "Картка", "Готівка")                              |
| Нагадування та сповіщення   | Частково реалізовані                                            | Наявні сповіщення та контроль лімітів                  | Системні попередження                                                       |
| Підтримка валют             | Мультимовна та мультивалютна підтримка                          | Підтримка декількох валют                              | Передбачена в структурі БД (DimCurrency)                                    |

|                |                                                       |                                             |                                                                 |
|----------------|-------------------------------------------------------|---------------------------------------------|-----------------------------------------------------------------|
| Переваги       | Зручна візуалізація фінансів та простота використання | Простий інтерфейс та детальна аналітика     | Доступність через веб-браузер, алгоритм 1-Rule для класифікації |
| Недоліки       | Частина функцій доступна лише у платній версії        | Менш виражена візуалізація фінансових даних | Потребує доступу до інтернету                                   |
| Платформи      | Android, iOS                                          | Android, iOS                                | Web (кросплатформна)                                            |
| Тип застосунку | Мобільний застосунок                                  | Мобільний застосунок                        | Веб-застосунок                                                  |

### 1.3 Постановка завдання

Розроблюване програмне забезпечення призначене для автоматизації процесу обліку та аналізу особистих фінансів користувача. Система забезпечує ведення обліку доходів і витрат, контроль фінансових потоків, планування бюджету та формування аналітичної інформації. Основною метою є створення ефективного інструменту, який дозволить користувачам отримувати повну та достовірну інформацію про власний фінансовий стан і приймати обґрунтовані фінансові рішення.

Програмна система передбачає створення єдиного інформаційного середовища для зберігання фінансових даних користувача, що включає відомості про транзакції, категорії витрат, бюджети та фінансові цілі. Система повинна забезпечувати простий і зручний механізм введення даних, їх редагування та подальшого аналізу.

Основна програмна система, що розробляється, надасть користувачам можливість швидко отримувати інформацію про:

- структуру та динаміку доходів і витрат;
- розподіл витрат за категоріями;
- виконання встановлених бюджетних обмежень;

- досягнення фінансових цілей та рівень заощаджень;
- виявлення перевищення витрат у визначених категоріях.

Система буде реалізована у вигляді веб-застосунку з інтуїтивно зрозумілим користувацьким інтерфейсом. Користувачі матимуть можливість взаємодіяти із системою через веб-браузер, використовуючи зручні форми введення даних та навігаційне меню. У процесі роботи система буде обробляти введену інформацію та відображати результати у вигляді звітів, таблиць і графічних візуалізацій.

На основі введених користувачем даних система формуватиме узагальнену інформацію, зокрема зведення доходів, аналіз витрат та аналітичні показники фінансової активності. Користувач також матиме можливість редагувати або видаляти раніше додані записи, що забезпечує гнучкість та актуальність даних.

Для забезпечення ефективного функціонування система повинна реалізовувати такі основні завдання:

- забезпечити реєстрацію та автентифікацію користувачів;
- реалізувати механізми додавання, редагування та видалення фінансових операцій;
- забезпечити категоризацію доходів і витрат;
- реалізувати функціонал бюджетування та контролю витрат;
- забезпечити формування аналітичних звітів і візуалізацію даних;
- гарантувати захист і конфіденційність фінансової інформації користувачів.

Така організація системи дозволить забезпечити ефективне управління особистими фінансами, підвищити фінансову дисципліну користувачів та створити умови для прийняття обґрунтованих фінансових рішень на основі актуальних даних.

## 1.4 Функціональні та нефункціональні вимоги

### ***Функціональні вимоги:***

1. реєстрація та автентифікація користувачів у системі;
2. створення, редагування та видалення фінансових операцій;
3. ведення обліку доходів і витрат із можливістю їх класифікації за категоріями;
4. створення та налаштування користувацьких категорій фінансових операцій;
5. реалізація механізму бюджетування з можливістю встановлення лімітів витрат [5];
6. формування аналітичних звітів щодо фінансового стану користувача;
7. візуалізація даних у вигляді графіків і діаграм;
8. пошук та фільтрація фінансових операцій за різними параметрами;
9. ведення обліку заощаджень та фінансових цілей;
10. можливість редагування профілю користувача;
11. експорт фінансових даних у зовнішні формати (наприклад, CSV);
12. забезпечення повідомлень або сповіщень щодо перевищення бюджету.

### ***Нефункціональні вимоги:***

1. зручність використання – система повинна мати інтуїтивно зрозумілий та адаптивний інтерфейс;
2. продуктивність – обробка запитів користувача повинна виконуватись із мінімальними затримками;
3. надійність – система повинна забезпечувати стабільну роботу без втрати даних;

4. масштабованість – можливість розширення функціоналу та обробки зростаючої кількості користувачів;
5. безпека – забезпечення захисту персональних і фінансових даних користувачів;
6. сумісність – коректна робота системи у різних веб-браузерах та на різних пристроях;
7. підтримуваність – можливість легкого внесення змін і оновлень у систему;
8. доступність – забезпечення доступу до системи у будь-який час через мережу Інтернет;
9. цілісність даних – гарантування коректності та узгодженості фінансової інформації;
10. резервне копіювання – забезпечення збереження даних у разі збоїв або втрати інформації.

Таким чином, визначені функціональні та нефункціональні вимоги формують основу для проєктування та реалізації програмного забезпечення для відстеження фінансів. Їх дотримання дозволить створити надійну, зручну та ефективну систему, що відповідатиме сучасним вимогам користувачів і забезпечуватиме високий рівень якості обробки фінансової інформації.

## **1.5 Вимоги до інтерфейсу користувача**

Користувацький інтерфейс є важливою складовою програмного забезпечення для відстеження фінансів, оскільки саме через нього здійснюється взаємодія користувача із системою. Ефективність роботи із системою значною мірою залежить від зручності, зрозумілості та логічності побудови інтерфейсу. Тому при розробці веб-застосунку необхідно враховувати сучасні принципи проєктування інтерфейсів та орієнтацію на потреби кінцевого користувача.

Інтерфейс системи повинен бути інтуїтивно зрозумілим та не вимагати тривалого навчання для початку роботи. Основні функції повинні бути доступні з головного екрана, а навігація між розділами має бути простою та логічною. Важливим аспектом є мінімізація кількості дій, необхідних для виконання основних операцій, таких як додавання фінансової транзакції або перегляд звітів [7].

Дизайн інтерфейсу повинен відповідати принципам консистентності, тобто однакові елементи управління повинні мати однаковий вигляд і поведінку у різних частинах системи. Використання єдиного стилю оформлення, кольорової гами та типографіки сприяє кращому сприйняттю інформації та підвищує зручність використання.

Інтерфейс повинен забезпечувати наочне відображення фінансових даних. Для цього доцільно використовувати графіки, діаграми та інші засоби візуалізації, що дозволяють швидко оцінити фінансовий стан користувача. Важливо забезпечити можливість фільтрації та сортування даних для отримання більш детальної інформації.

Особлива увага повинна приділятися адаптивності інтерфейсу. Система повинна коректно відображатися на різних пристроях, включаючи персональні комп'ютери, планшети та смартфони. Це забезпечує доступність системи незалежно від типу пристрою та розміру екрана.

Інтерфейс має підтримувати швидке введення даних. Форми для додавання фінансових операцій повинні бути максимально спрощеними, містити лише необхідні поля та передбачати використання підказок або автозаповнення. Це дозволяє зменшити час введення інформації та підвищити точність даних.

Крім того, інтерфейс повинен забезпечувати зворотний зв'язок із користувачем. Система має інформувати користувача про результати виконаних дій, наприклад, успішне додавання транзакції або виникнення помилки. Повідомлення повинні бути зрозумілими та інформативними.

## 1.6 Висновки до розділу

Виконано системний аналіз предметної області та обґрунтовано актуальність створення інформаційної системи контролю особистих фінансових ресурсів. За результатами аналізу зроблено такі висновки: Визначено сутність та ключові процеси управління особистими фінансами в умовах цифровізації, окреслено основні сутності (користувачі, транзакції, категорії, бюджети) та сформульовано жорсткі вимоги до безпеки й конфіденційності фінансових даних. Проведено аналіз існуючих мобільних рішень (CoinKeeper, 1Money) та виявлено їхні недоліки, такі як наявність платного функціоналу чи обмеження мобільних платформ. Обґрунтовано розробку власної системи «CashFlow» у вигляді веб-застосунку, що забезпечує кросплатформність та доступність через браузер. Сформульовано постановку завдання й визначено детальні функціональні вимоги (CRUD-операції з транзакціями й категоріями, звітність за період, жорсткий контроль лімітів із блокуванням перевитрат) та нефункціональні вимоги (адаптивність інтерфейсу на базі Bootstrap, використання СУБД MySQL). Обґрунтовано доцільність інтеграції інструментів інтелектуального аналізу даних, зокрема використання алгоритму 1-Rule для автоматизації первинної класифікації фінансових операцій за сумою транзакції, що дозволяє спростити ручне введення та покращити аналітичну обробку інформації.

## 2 МОДЕЛЮВАННЯ СИСТЕМИ

### 2.1 Загальні відомості

Уніфікована мова моделювання (Unified Modeling Language, UML) є стандартною мовою в галузі програмної інженерії, яка використовується для візуалізації, проєктування та документування інформаційних систем. Вона дозволяє описувати як структуру, так і поведінку програмного забезпечення за допомогою графічних діаграм, що значно спрощує процес розробки та підвищує зрозумілість системи для всіх учасників проєкту.

Основною метою UML є створення формалізованого представлення системи, яке відображає її архітектуру, логіку роботи та взаємодію між компонентами. Використання UML дає змогу на ранніх етапах розробки аналізувати вимоги, проєктувати структуру системи та виявляти потенційні помилки ще до етапу реалізації програмного коду.

UML охоплює різні типи діаграм, які умовно поділяються на структурні та поведінкові. Структурні діаграми використовуються для опису статичної організації системи, тоді як поведінкові діаграми відображають динамічні процеси та взаємодії всередині системи.

До найбільш поширених UML-діаграм належить діаграма варіантів використання, яка відображає взаємодію користувачів із системою та дозволяє визначити основні функції програмного забезпечення. Діаграма класів описує структуру системи, включаючи класи, їх атрибути, методи та зв'язки між ними. Діаграма послідовностей демонструє порядок взаємодії об'єктів у часі, що дозволяє зрозуміти логіку виконання операцій. Діаграма діяльності використовується для опису бізнес-процесів і потоків виконання, тоді як діаграма станів показує можливі стани об'єктів і переходи між ними. Окремо також застосовуються діаграми компонентів, які відображають структуру програмних модулів, та діаграми розгортання, що описують фізичне розміщення системи на апаратному забезпеченні.

У контексті розробки програмного забезпечення для відстеження фінансів UML відіграє важливу роль, оскільки дозволяє формалізувати вимоги до системи, визначити її архітектуру та структуру даних, а також описати сценарії взаємодії користувача з функціоналом системи. Це сприяє більш точному проєктуванню та зменшує ризики виникнення помилок на етапі реалізації.

Крім того, UML є універсальним інструментом, який не залежить від конкретної мови програмування чи технологічної платформи. Завдяки цьому він широко використовується у веб-розробці, де необхідно моделювати взаємодію між серверною частиною, базою даних та користувацьким інтерфейсом.

Таким чином, UML є важливим інструментом проєктування інформаційних систем, який забезпечує наочне представлення структури та поведінки програмного забезпечення. Його застосування у процесі розробки системи для відстеження фінансів дозволяє підвищити якість проєктування, полегшити реалізацію та забезпечити ефективний супровід програмного продукту.

**Інтелектуальний аналіз даних (Data Mining)** є однією з ключових технологій сучасної обробки інформації, яка спрямована на виявлення прихованих закономірностей, залежностей та структур у великих масивах даних. У контексті інформаційних систем для відстеження фінансів даний підхід дозволяє не лише зберігати та відображати фінансові дані, але й отримувати з них аналітичні висновки, що можуть бути використані для прийняття обґрунтованих рішень.

Основною ідеєю Data Mining є перетворення сирих даних у корисну інформацію та знання. Це досягається за допомогою застосування статистичних методів, алгоритмів машинного навчання та методів штучного інтелекту. У фінансових системах це може включати аналіз витрат користувача, прогнозування фінансової поведінки, виявлення аномалій у транзакціях та класифікацію фінансових операцій.

Процес інтелектуального аналізу даних зазвичай включає декілька етапів: збір та підготовку даних, їх очищення, трансформацію, застосування алгоритмів аналізу та інтерпретацію отриманих результатів. Особливе значення має етап підготовки даних, оскільки якість вихідної інформації безпосередньо впливає на точність аналітичних результатів.

У системах управління особистими фінансами Data Mining може використовуватися для визначення типових моделей витрат користувача, аналізу сезонності фінансової поведінки, а також для прогнозування майбутніх витрат на основі історичних даних. Наприклад, система може виявляти, що користувач регулярно перевищує бюджет у певних категоріях, і на основі цього формувати рекомендації щодо оптимізації витрат.

Окремим напрямом застосування інтелектуального аналізу даних є виявлення аномалій у фінансових операціях. Це дозволяє ідентифікувати нетипові транзакції, які можуть свідчити про помилки введення даних або потенційно підозрілу активність.

Сучасні підходи до Data Mining часто інтегруються з методами машинного навчання, що дозволяє створювати більш точні та адаптивні моделі аналізу. Такі моделі можуть навчатися на основі історичних даних користувача та поступово підвищувати точність прогнозування фінансових показників.

У межах програмного забезпечення для відстеження фінансів використання інтелектуального аналізу даних дозволяє перейти від простого обліку операцій до розумної системи підтримки прийняття рішень. Це забезпечує користувачам можливість не лише переглядати свої фінансові дані, але й отримувати рекомендації щодо покращення фінансового стану.

Таким чином, інтелектуальний аналіз даних є важливим інструментом сучасних фінансових інформаційних систем, оскільки забезпечує глибший рівень обробки даних, дозволяє виявляти приховані закономірності та сприяє підвищенню ефективності управління особистими фінансами.

## 2.2 Об'єктне та функціональне моделювання

**2.2.1 Діаграма прецедентів.** Діаграма варіантів використання (Use Case Diagram) є одним із ключових інструментів UML, який застосовується для моделювання функціональних вимог програмної системи з точки зору користувача. Вона відображає взаємодію між акторами (користувачами або зовнішніми системами) та функціональними можливостями системи, що дозволяє наочно представити її основні сценарії використання.

Основною метою діаграми варіантів використання є визначення того, які функції повинна виконувати система та хто саме буде взаємодіяти з цими функціями. У контексті програмного забезпечення для відстеження особистих фінансів така діаграма дозволяє описати взаємодію користувача із системою, включаючи облік доходів і витрат, формування звітів, управління бюджетами та фінансовими цілями.

На діаграмі варіантів використання основними елементами є актори та прецеденти (use cases). Актори представляють користувачів системи або зовнішні сервіси, які взаємодіють із нею. У системі фінансового обліку основним актором зазвичай виступає користувач, який здійснює всі операції з фінансовими даними. У деяких випадках можуть також враховуватися додаткові ролі, наприклад користувач системи.

Прецеденти описують окремі функціональні можливості системи, які доступні актору. До таких прецедентів у системі відстеження фінансів належать додавання фінансових транзакцій, перегляд історії операцій, аналіз витрат, формування звітів, налаштування бюджетів та управління категоріями витрат.

Діаграма варіантів використання дозволяє також відобразити взаємозв'язки між прецедентами, такі як включення (include) та розширення (extend). Це дає змогу структурувати функціональні можливості системи та показати залежності між окремими операціями. Наприклад, процес

формування фінансового звіту може включати попередній збір та обробку фінансових даних.

Розроблена діаграма прецедентів використання представлена на рис.

2.1.



Рис. 2.1 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Користувач”.

Актор «Користувач» включає такі прецеденти:

- створити категорію доходу;
- додати суму доходу;
- змінити іконку категорії;
- встановити ліміт на категорію;

- створити категорію витрати;
- додати суму витрати;
- перегляд статистики;
- обрання проміжку дат.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

**Сценарій:** Створення категорії доходу

**Виконавець:** Користувач

**Опис:** Користувач відповідає за підтримку структури фінансових категорій у системі, зокрема за додавання нових категорій доходу. Ця функція забезпечує можливість подальшого коректного обліку та класифікації джерел доходів користувачів. Завдяки цьому система дозволяє більш точно структурувати фінансові дані та підвищує зручність їх аналізу.

**Послідовність дій:**

1. користувач відкриває розділ управління категоріями доходів, де відображається перелік уже створених категорій;
2. користувач обирає функцію додавання нової категорії;
3. система відкриває форму для введення даних нової категорії доходу;
4. користувач вводить назву та, за необхідності, додатковий опис категорії;
5. після заповнення полів він підтверджує створення категорії шляхом натискання кнопки збереження;
6. система виконує перевірку введених даних на коректність, зокрема перевіряє заповненість полів та унікальність назви;
7. у разі успішної валідації система зберігає нову категорію доходу в базі даних;
8. після завершення операції система відображає повідомлення про успішне створення категорії;
9. новостворена категорія автоматично додається до загального

списку доступних категорій доходів.

**Альтернативні потоки:**

1. якщо введені дані некоректні (наприклад, відсутня назва або така категорія вже існує), система відхиляє запит та відображає повідомлення про помилку з поясненням причини;
2. якщо користувач не має достатніх прав доступу для виконання цієї операції, система блокує можливість створення категорії та повідомляє про відсутність відповідних дозволів.

У результаті виконання сценарію в системі створюється нова категорія доходу, яка стає доступною для подальшого використання користувачами при додаванні фінансових операцій, що забезпечує більш точну та структуровану класифікацію доходів.

Розглянемо ще один сценарій використання.

**Сценарій:** Встановлення ліміту для категорії

**Виконавець:** Користувач

**Опис:** Даний сценарій описує процес встановлення фінансового ліміту для певної категорії витрат. Функціональність дозволяє користувачу контролювати свої витрати, обмежуючи суму коштів, яка може бути витрачена в межах обраної категорії за визначений період. Це сприяє більш ефективному плануванню бюджету та запобігає перевищенню встановлених фінансових меж.

**Послідовність дій:**

1. користувач виконує вхід до системи через веб-інтерфейс;
2. він переходить до розділу управління бюджетом або категоріями витрат;
3. користувач обирає конкретну категорію, для якої необхідно встановити ліміт;
4. система відкриває форму налаштування ліміту для обраної категорії;
5. користувач вводить суму ліміту та, за необхідності, вибирає

- період його дії (наприклад, місяць);
6. після введення даних користувач підтверджує встановлення ліміту;
  7. система перевіряє коректність введених даних, зокрема перевіряє, чи є сума додатною та чи відповідає формат даних вимогам системи;
  8. у разі успішної валідації система зберігає встановлений ліміт у базі даних та прив'язує його до відповідної категорії;
  9. система відображає повідомлення про успішне встановлення ліміту;
  10. оновлена інформація про ліміт стає доступною в інтерфейсі користувача.

#### **Альтернативні потоки:**

1. якщо користувач вводить некоректні дані (наприклад, від'ємну суму або порожнє значення), система відображає повідомлення про помилку та пропонує виправити введену інформацію;
2. якщо у користувача відсутні необхідні права доступу або він не авторизований, система забороняє виконання операції та перенаправляє його на сторінку входу або відображає відповідне повідомлення;
3. якщо ліміт для даної категорії вже існує, система може запропонувати оновити поточне значення або скасувати дію.

Після успішного виконання сценарію для обраної категорії витрат встановлюється фінансовий ліміт, який використовується системою для контролю витрат користувача та формування попереджень у разі його перевищення.

**2.2.2 Діаграма послідовності.** Діаграма послідовності (Sequence Diagram) є одним із видів поведінкових діаграм UML, яка використовується для моделювання взаємодії між об'єктами системи у часовій послідовності. Вона дозволяє детально описати, як саме виконуються процеси в системі,

які повідомлення передаються між об'єктами та в якій черзі відбуваються ці взаємодії.

Основною метою діаграми послідовності є відображення сценарію виконання певного варіанту використання, де акцент робиться на часовому порядку обміну повідомленнями між користувачем та системою або між внутрішніми компонентами системи. Це дозволяє зрозуміти логіку роботи системи на рівні взаємодії її елементів.

У контексті програмного забезпечення для відстеження фінансів діаграма послідовності використовується для моделювання ключових операцій, таких як додавання фінансових транзакцій, створення категорій, встановлення бюджетів, формування звітів або аналіз витрат. Вона дозволяє наочно показати, як запит користувача проходить через різні рівні системи, включаючи інтерфейс, бізнес-логіку та базу даних.

Основними елементами діаграми послідовності є актори, об'єкти (або компоненти системи), повідомлення та лінії життя. Актори представляють зовнішніх користувачів або системи, які ініціюють взаємодію. Об'єкти відображають внутрішні компоненти програмної системи, такі як контролери, сервіси або модулі доступу до бази даних. Повідомлення описують взаємодію між цими елементами, а лінії життя показують тривалість їхнього існування у процесі виконання сценарію [8].

Діаграма послідовності дозволяє детально проаналізувати порядок виконання операцій та виявити можливі проблеми у логіці взаємодії компонентів. Вона також є важливим інструментом для розробників, оскільки допомагає краще зрозуміти структуру викликів методів та обмін даними між частинами системи.

Діаграма послідовності, зображена на рис. 2.2, включає наступні об'єкти:

- “Користувач”;
- “Доходи”;
- “Витрати”;

- “Ліміт”.

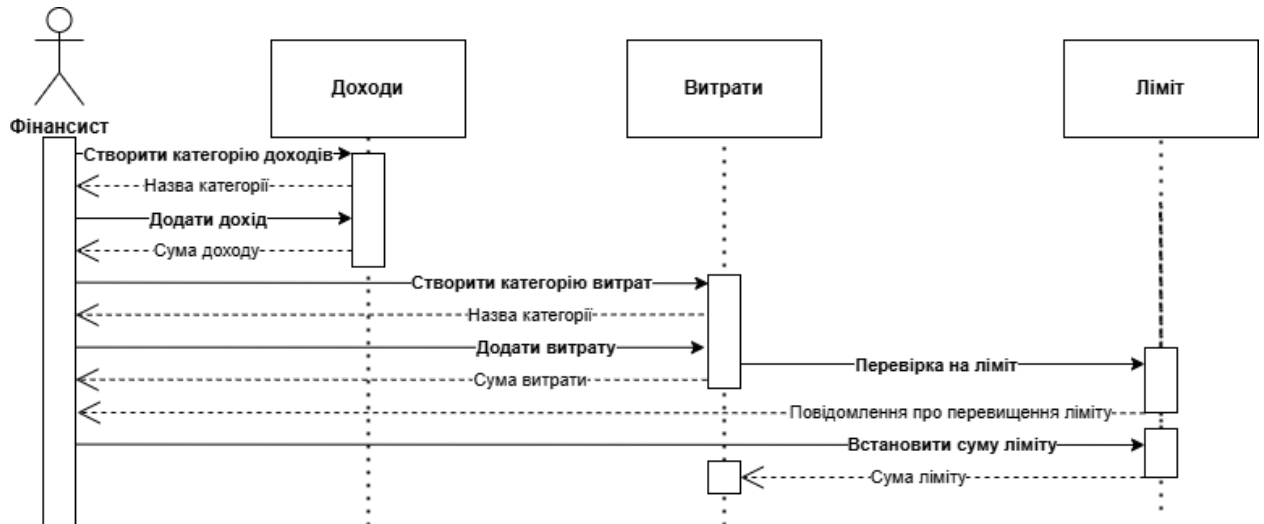


Рис. 2.2 Діаграма послідовності

Діаграма послідовності ілюструє робочий процес управління фінансами, зосереджуючись на категоризації та моніторингу доходів і витрат. Він починається зі створення фінансовим менеджером категорії для доходу, вказуючи її назву та суму, а потім додає відповідний дохід до цієї категорії. Далі аналогічний процес розгортається для витрат, де визначаються категорії, а витрати призначаються відповідно. Ці витрати потім перевіряються, щоб переконатися, що вони відповідають заздалегідь визначеним лімітам. Якщо витрати перевищують встановлені межі, спрацьовує сповіщення, щоб висвітлити цю проблему. Для підтримки фінансового контролю та дисципліни також можна проактивно встановлювати ліміти для категорій витрат. Цей процес забезпечує систематичну фінансову організацію та сприяє ефективному моніторингу та управлінню.

**2.2.3 Діаграма активності.** Діаграма діяльності (Activity Diagram) є одним із типів поведінкових діаграм UML, яка використовується для моделювання бізнес-процесів та алгоритмів виконання операцій у програмній системі. Вона відображає послідовність дій, умови переходів між ними та логіку виконання процесів, що дозволяє наочно представити роботу системи або окремого її модуля.

Основною метою діаграми діяльності є опис потоку виконання

операцій у системі від початкового стану до кінцевого результату. Вона дозволяє моделювати як прості, так і складні процеси, включаючи розгалуження, паралельне виконання дій та циклічні процеси [9].

У контексті програмного забезпечення для відстеження фінансів діаграма діяльності використовується для моделювання основних сценаріїв роботи системи, таких як додавання фінансових транзакцій, встановлення бюджету, формування звітів або аналіз витрат. Вона дозволяє чітко відобразити послідовність кроків, які виконує користувач або система для досягнення певного результату.

Основними елементами діаграми діяльності є початковий та кінцевий вузли, дії (activity states), переходи між діями, а також умовні та паралельні розгалуження. Дії представляють окремі кроки процесу, тоді як переходи визначають порядок їх виконання. Умовні розгалуження дозволяють моделювати альтернативні сценарії, залежно від виконання певних умов.

Діаграма діяльності особливо корисна для опису алгоритмічної логіки системи, оскільки вона дозволяє абстрагуватися від технічної реалізації та зосередитися на логіці бізнес-процесів. У системі для обліку фінансів вона може використовуватися для опису процесу обробки транзакцій, перевірки бюджетних обмежень або формування фінансових звітів.

Крім того, діаграма діяльності дозволяє відобразити паралельні процеси, наприклад одночасне оновлення балансу та формування статистики після додавання нової фінансової операції. Це робить її ефективним інструментом для моделювання сучасних багатокомпонентних систем.

Розроблена діаграма активності представлена на рис. 2.3

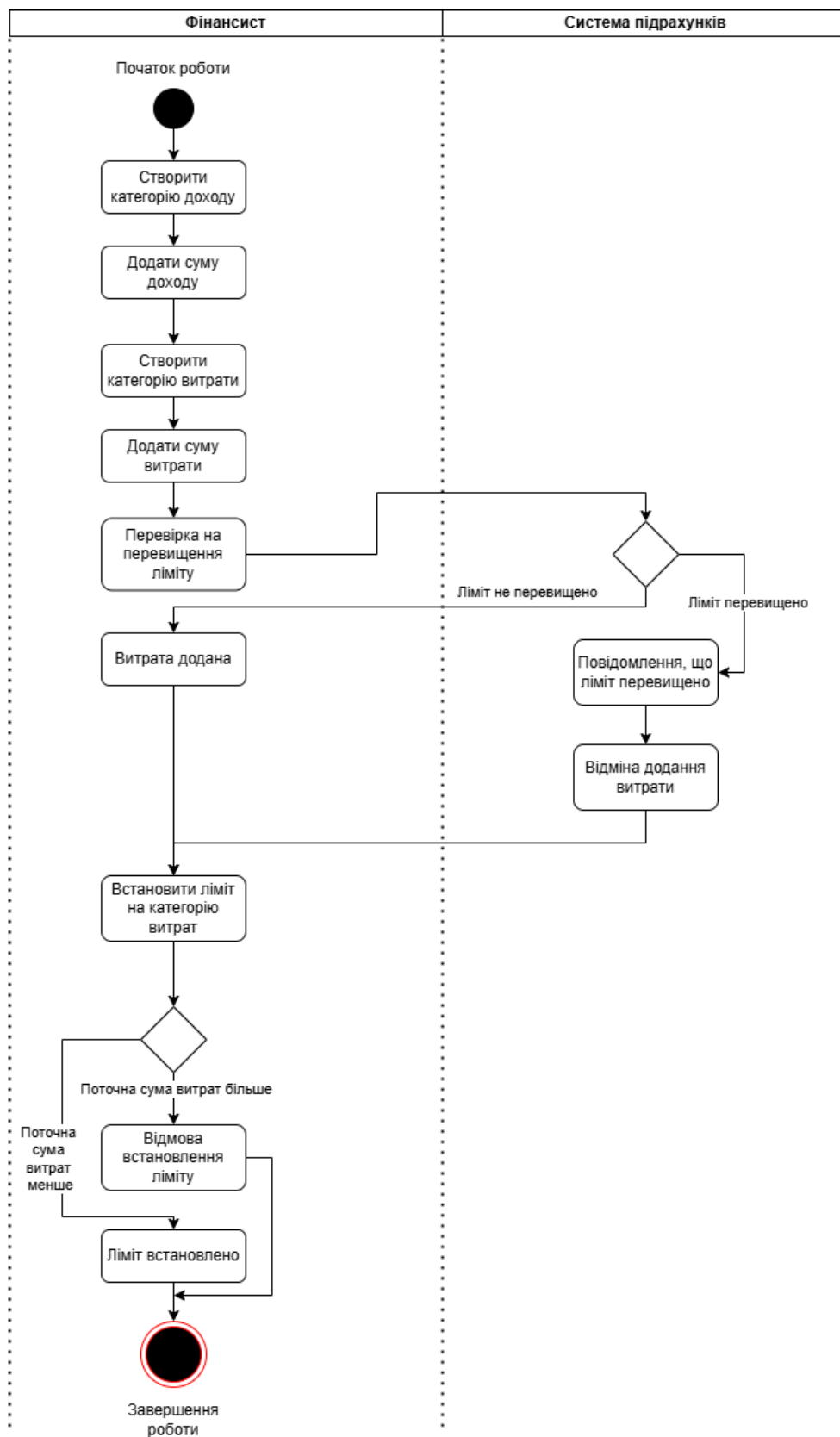


Рис. 2.3 Діаграма активності

Ця діаграма активності демонструє процес взаємодії користувача з системою підрахунків. Початковий крок – початок роботи, де користувач

створює категорії для доходів і витрат. У кожную категорію він додає відповідні суми.

Система підрахунків відіграє ключову роль у перевірці витрат щодо встановлених лімітів. Якщо витрати перевищують межі, система повідомляє про це і не дозволяє додати перевищену суму. Користувач також може встановлювати ліміти для витрат. У випадку, коли поточні витрати вже перевищують бажаний ліміт, встановлення буде скасовано. В іншому разі ліміт успішно додається.

Завершується процес закінченням роботи користувача, після того як усі дії виконано. Така діаграма чітко ілюструє розподіл відповідальностей між користувачем і системою, забезпечуючи контроль і дисципліну у фінансовому управлінні.

## **2.3 Огляд інструментів для реалізації завдань Data Mining**

Інтелектуальний аналіз даних (Data Mining) у сучасних інформаційних системах базується на використанні спеціалізованих програмних інструментів, які забезпечують обробку великих обсягів даних, виявлення закономірностей та побудову аналітичних моделей. У контексті програмного забезпечення для відстеження фінансів такі інструменти дозволяють здійснювати аналіз витрат і доходів користувачів, прогнозування фінансової поведінки та виявлення аномалій у транзакціях.

Одним із найпоширеніших інструментів є Python як екосистема для аналізу даних. Завдяки бібліотекам, таким як Pandas, NumPy та Scikit-learn, Python забезпечує широкий набір функцій для обробки, очищення та аналізу даних. Scikit-learn, зокрема, використовується для реалізації алгоритмів класифікації, кластеризації та регресії, що є основою багатьох задач Data Mining [10].

Важливу роль відіграє також мова програмування R, яка спеціалізується на статистичному аналізі даних. Вона широко застосовується

для побудови статистичних моделей, аналізу часових рядів та візуалізації даних. R містить велику кількість пакетів, що дозволяють реалізовувати складні аналітичні завдання без необхідності глибокої програмної реалізації алгоритмів.

Серед комплексних платформ для Data Mining варто відзначити RapidMiner. Це середовище дозволяє створювати аналітичні процеси без глибокого програмування, використовуючи візуальний інтерфейс. RapidMiner підтримує широкий спектр алгоритмів машинного навчання та є зручним для швидкого прототипування аналітичних моделей.

Іншим популярним інструментом є Weka, яка є відкритою платформою для машинного навчання. Вона надає набір алгоритмів для класифікації, кластеризації, регресії та асоціативного аналізу. Weka часто використовується в академічних дослідженнях завдяки своїй простоті та наочності реалізації алгоритмів.

Для більш складних і масштабованих задач Data Mining застосовуються Big Data технології, такі як Apache Spark. Spark дозволяє обробляти великі обсяги даних у розподіленому середовищі та підтримує бібліотеку MLlib для машинного навчання. Це особливо актуально для систем, які обробляють великі потоки фінансових транзакцій.

У веб-орієнтованих системах, таких як програмне забезпечення для відстеження фінансів, інструменти Data Mining часто інтегруються безпосередньо у серверну частину. PHP може використовуватися для базової обробки даних, однак для складної аналітики зазвичай застосовуються зовнішні сервіси або мікросервіси, реалізовані на Python або інших мовах, що спеціалізуються на аналізі даних [11].

Таким чином, інструменти Data Mining охоплюють як мови програмування та бібліотеки, так і спеціалізовані платформи та фреймворки. Їх використання у системі для відстеження фінансів дозволяє перейти від простого обліку даних до глибокого аналізу фінансової поведінки користувача, побудови прогнозів та прийняття більш обґрунтованих рішень на

основі даних.

Для реалізації даної бакалаврської роботи було обрано наступний підхід до Data Mining: у якості основного інструменту для аналітичної обробки даних використовується Python з бібліотеками Pandas та Scikit-learn, оскільки вони забезпечують необхідний функціонал для обробки фінансових транзакцій, їх класифікації та побудови базових прогнозних моделей. Дані з веб-застосунку, реалізованого на PHP Symfony, передаються до аналітичного модуля, де здійснюється їх обробка та формування результатів. Отримані аналітичні дані повертаються у систему для подальшої візуалізації та відображення користувачу у вигляді звітів та графіків. Такий підхід дозволяє поєднати зручність веб-інтерфейсу з можливостями інтелектуального аналізу даних та забезпечує гнучкість і масштабованість системи.

## **2.4 Структура джерела інформації для інтелектуального аналізу**

У процесі розробки програмного забезпечення для відстеження особистих фінансів важливе значення має правильна організація та структуризація джерел інформації, які використовуються для аналізу предметної області, проєктування системи та реалізації функціональних модулів. Якість і достовірність джерел безпосередньо впливають на обґрунтованість прийнятих технічних і архітектурних рішень.

Джерела інформації, що використовуються у межах даного дослідження, можна умовно поділити на кілька основних категорій: наукові джерела, технічна документація, онлайн-ресурси та практичні приклади існуючих програмних рішень. Кожна з цих категорій виконує окрему роль у процесі формування цілісного уявлення про предметну область.

Наукові джерела включають монографії, наукові статті та матеріали конференцій, які присвячені питанням розробки інформаційних систем, аналізу даних, фінансового моделювання та застосування алгоритмів

машинного навчання. Вони використовуються для теоретичного обґрунтування обраних підходів та методів реалізації системи.

Технічна документація охоплює офіційні матеріали, пов'язані з використаними технологіями, зокрема PHP, фреймворком Symfony та системою управління базами даних MySQL. До цієї категорії також належать документації бібліотек та інструментів, які застосовуються для реалізації аналітичних функцій та Data Mining.

Онлайн-ресурси включають офіційні вебсайти розробників програмного забезпечення, технічні блоги, професійні форуми та тематичні платформи. Вони використовуються для отримання актуальної інформації про сучасні підходи до розробки, практичні рекомендації та приклади реалізації функціональних модулів [12].

Окрему категорію становлять аналізовані існуючі програмні рішення для управління особистими фінансами. До них належать мобільні та веб-застосунки, які вже реалізують подібний функціонал. Їх вивчення дозволяє визначити сильні та слабкі сторони існуючих систем, а також сформулювати вимоги до власного програмного продукту.

Структура аналізу джерел передбачає послідовне опрацювання інформації від загальних теоретичних положень до конкретних практичних рішень. Спочатку здійснюється аналіз наукових підходів до обліку фінансів та обробки даних, після чого розглядаються технологічні інструменти реалізації, а завершальним етапом є дослідження існуючих програмних продуктів та їх функціональних можливостей.

Систематизована структура джерел інформації забезпечує комплексний підхід до аналізу предметної області, дозволяє сформулювати обґрунтовані вимоги до програмного забезпечення та створити ефективну архітектуру системи для відстеження особистих фінансів.

**Сховище даних (Data Warehouse)** є спеціалізованою інформаційною системою, призначеною для централізованого зберігання, інтеграції та обробки великих обсягів даних, що надходять із різних джерел. Основною

метою використання сховищ даних є забезпечення ефективного аналітичного опрацювання інформації, підтримка прийняття рішень та формування узагальнених звітів на основі історичних даних.

У контексті програмного забезпечення для відстеження особистих фінансів концепція Data Warehouse може бути застосована для накопичення фінансової інформації користувача, включаючи транзакції, категорії витрат, бюджети та інші пов'язані дані. Це дозволяє здійснювати довгостроковий аналіз фінансової поведінки та формувати більш точні аналітичні висновки.

Основними характеристиками сховища даних є орієнтація на предметну область, інтеграція даних із різних джерел, підтримка історичності та незмінності даних, а також оптимізація для аналітичних запитів. На відміну від оперативних баз даних, які використовуються для щоденної обробки транзакцій, сховища даних призначені для виконання складних аналітичних операцій та формування звітності.

Архітектура Data Warehouse зазвичай включає кілька рівнів. На першому рівні здійснюється збір даних із різних джерел, таких як транзакційні бази даних або зовнішні сервіси. Далі дані проходять процес ETL (Extract, Transform, Load), який включає їх вилучення, очищення, трансформацію та завантаження у сховище. На завершальному рівні дані використовуються для аналітичної обробки, побудови звітів та виконання запитів користувачів [13].

У системах управління фінансами Data Warehouse дозволяє накопичувати історію фінансових операцій користувача протягом тривалого часу. Це дає змогу аналізувати зміни у витратах, виявляти фінансові тенденції, оцінювати ефективність бюджетування та будувати прогнози майбутніх витрат.

Важливим аспектом використання сховищ даних є можливість інтеграції з інструментами Business Intelligence (BI), які забезпечують візуалізацію даних у вигляді графіків, діаграм та інтерактивних звітів. Це значно підвищує зручність аналізу фінансової інформації та сприяє кращому розумінню фінансового стану користувача.

У межах розроблюваного веб-застосунку для відстеження фінансів елементи концепції Data Warehouse можуть бути реалізовані у вигляді окремого аналітичного шару або модулю, який акумулює історичні дані та забезпечує їх подальшу обробку. Такий підхід дозволяє розділити оперативну обробку транзакцій і аналітичні запити, що підвищує продуктивність системи.

Щоб задовольнити різноманітні вимоги до даних, у сховищі даних було ретельно розроблено набір таблиць. Сховище даних зображено на Рис. 2.4

Ці таблиці служать репозиторіями для зберігання важливих даних, кожна з яких адаптована до конкретних аналітичних потреб і функціональних областей:

- DimUser;
- DimCategory;
- DimDate;
- DimCurrency;
- DimPaymentMethod;
- FactFinanceTransactions;
- FactBudgetAnalysis.

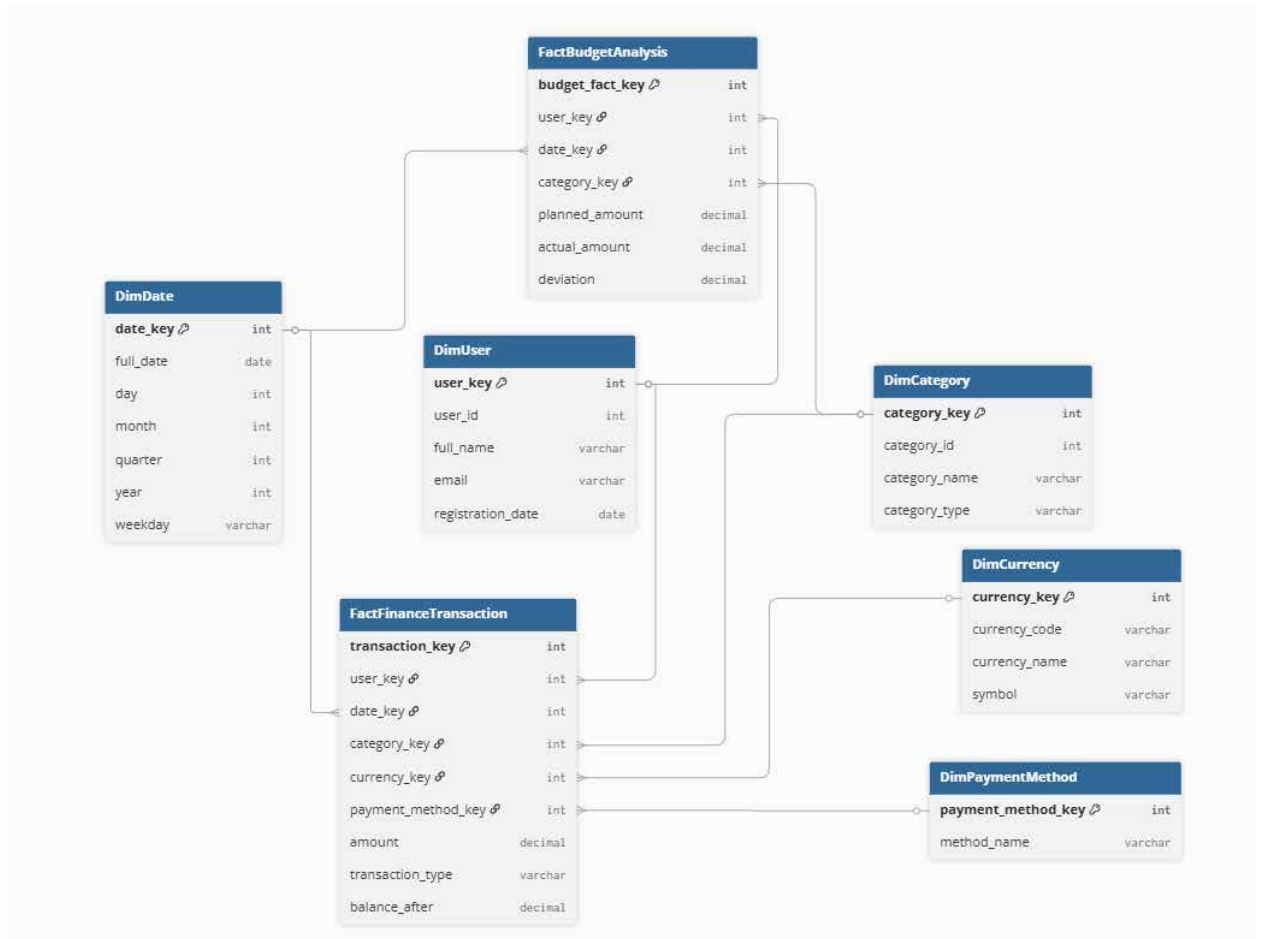


Рис. 2.4 Сховище даних

У цій моделі сховища даних факти та виміри розділені дуже чітко.

Факти — це таблиці, де зберігаються числові показники, які можна аналізувати. У таблиці бюджетного аналізу фіксуються планові та фактичні суми, а також відхилення між ними. Це дозволяє бачити, наскільки витрати відповідають запланованим. У таблиці фінансових транзакцій зберігаються всі операції користувачів: сума, тип транзакції, залишок після її проведення. Таким чином, факти відображають самі події та їхні кількісні результати.

Виміри — це таблиці-довідники, які додають контекст до цих фактів. Є календар, що дозволяє групувати дані за днями, місяцями чи роками. Є таблиця користувачів із їхніми ідентифікаторами, іменами та датами реєстрації. Є категорії, що класифікують витрати й доходи. Є валюти з кодами та символами, а також методи оплати, які уточнюють спосіб проведення транзакцій.

## 3 РОЗРОБКА СИСТЕМИ

### 3.1 Логічна модель даних

ER-діаграма (Entity-Relationship Diagram) є одним із ключових інструментів моделювання даних, який використовується для проєктування структури бази даних інформаційної системи. Вона дозволяє наочно представити сутності предметної області, їх атрибути та зв'язки між ними, що забезпечує чітке розуміння організації даних ще на етапі проєктування системи.

Основною метою ER-діаграми є формалізація структури даних таким чином, щоб забезпечити ефективне зберігання, обробку та взаємодію інформації в системі. У контексті програмного забезпечення для відстеження особистих фінансів ER-модель використовується для опису таких сутностей, як користувачі, фінансові транзакції, категорії витрат і доходів, бюджети та інші пов'язані елементи [13].

Основними елементами ER-діаграми є сутності (entities), атрибути (attributes) та зв'язки (relationships). Сутності представляють об'єкти предметної області, наприклад користувача або фінансову операцію. Атрибути описують властивості цих сутностей, такі як ім'я користувача, сума транзакції або дата операції. Зв'язки визначають взаємодію між сутностями, наприклад зв'язок між користувачем і його фінансовими транзакціями.

У системі для обліку фінансів ER-діаграма дозволяє структурувати дані таким чином, щоб забезпечити логічну цілісність та мінімізувати надлишковість інформації. Наприклад, один користувач може мати багато фінансових транзакцій, що реалізується через зв'язок типу «один-до-багатьох». Аналогічно, кожна транзакція належить до певної категорії, що дозволяє здійснювати подальший аналіз фінансових даних.

Важливим аспектом ER-моделювання є нормалізація бази даних, яка дозволяє зменшити дублювання даних та підвищити їх цілісність. Завдяки

правильному проєктуванню ER-діаграми забезпечується ефективна робота системи, швидкий доступ до даних та можливість масштабування.

ER-діаграма також є основою для подальшої реалізації фізичної структури бази даних у реляційній системі управління базами даних, такій як MySQL. Вона дозволяє перетворити концептуальну модель у конкретні таблиці, поля та зв'язки між ними.

Таким чином, ER-діаграма є важливим етапом проєктування інформаційної системи для відстеження фінансів, оскільки вона забезпечує формалізований опис структури даних, сприяє правильній організації бази даних та створює основу для подальшої реалізації програмного забезпечення.

ERwin Data Modeler є професійним інструментом для проєктування, моделювання та аналізу баз даних, який широко використовується у сфері розробки інформаційних систем. Він базується на концепції ER-моделювання (Entity-Relationship Modeling) і дозволяє створювати як концептуальні, так і фізичні моделі баз даних, забезпечуючи повний цикл проєктування структури даних.

Основною метою використання ERwin є спрощення процесу проєктування баз даних та забезпечення високого рівня узгодженості між логічною і фізичною моделями. Інструмент дозволяє описувати сутності, атрибути та зв'язки між ними у графічному вигляді, а також автоматично генерувати SQL-скрипти для створення структури бази даних у різних системах управління базами даних, зокрема MySQL.

ERwin підтримує методології нормалізації даних, що дозволяє зменшити надлишковість інформації та підвищити цілісність даних. Завдяки цьому забезпечується ефективне проєктування структурованих і масштабованих баз даних, що особливо важливо для інформаційних систем з великою кількістю взаємопов'язаних сутностей.

Однією з ключових переваг ERwin є можливість візуального моделювання складних структур даних. Користувач може створювати діаграми, які відображають взаємозв'язки між таблицями, визначати первинні

та зовнішні ключі, а також задавати обмеження цілісності даних. Це значно полегшує процес аналізу та перевірки коректності моделі.

У контексті розробки програмного забезпечення для відстеження особистих фінансів ERwin може бути використаний для проєктування структури бази даних, яка включає сутності користувачів, фінансових транзакцій, категорій доходів і витрат, бюджетів та аналітичних даних. Це дозволяє чітко визначити взаємозв'язки між компонентами системи ще на етапі проєктування.

Крім того, ERwin підтримує можливість зворотного інжинірингу, що дозволяє створювати модель бази даних на основі вже існуючої структури. Це є корисним при аналізі або модернізації існуючих систем.

Логічна модель системи представлена на рис. 3.1

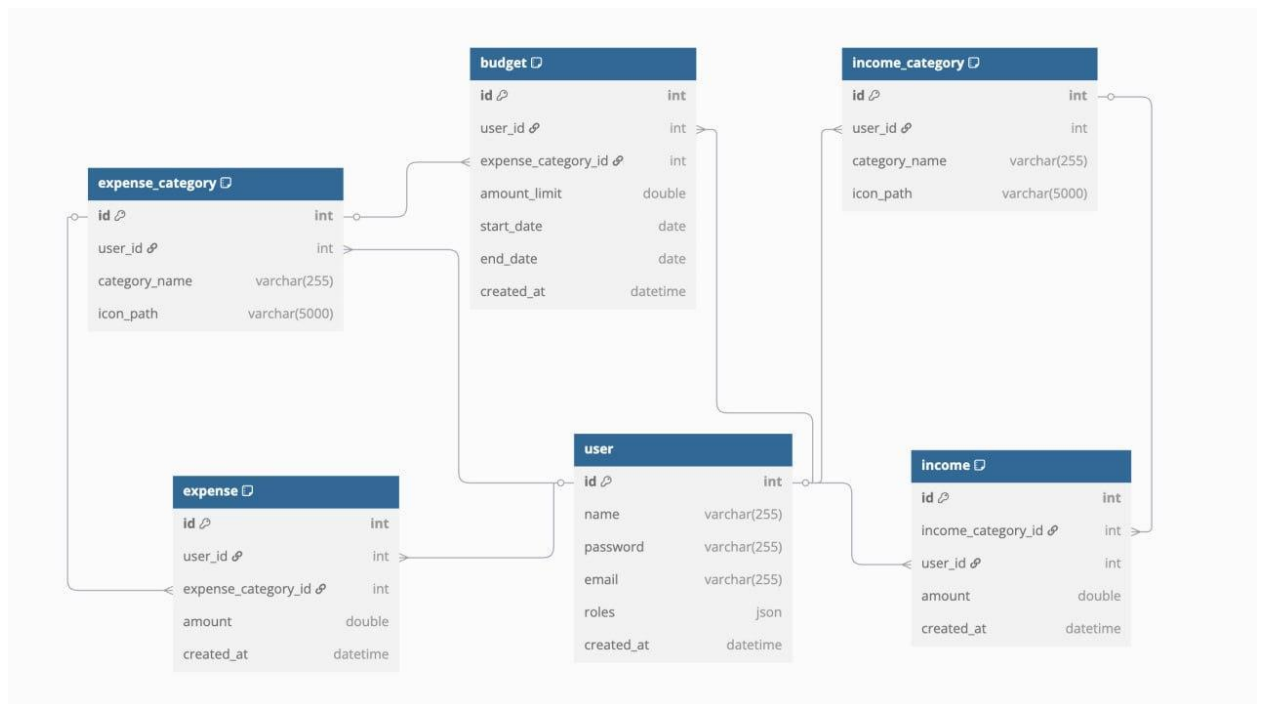


Рис. 3.1 ER-діаграма

Ця ER-діаграма відображає базу даних для управління фінансами, включаючи таблиці, їх атрибути та взаємозв'язки між ними.

#### Таблиці:

1. **expense\_category**: містить інформацію про категорії витрат із такими атрибутами, як **id**, **user\_id**, **category\_name**, **icon\_path**;

2. `expense`: зберігає витрати, описані через `id`, `user_id`, `expense_category_id`, `amount`, та `created_at`;
3. `budget`: відповідає за ліміти витрат, включаючи `id`, `user_id`, `expense_category_id`, `amount_limit`, `start_date`, `end_date`, і `created_at`;
4. `user`: представляє користувачів системи із атрибутами `id`, `name`, `password`, `email`, `roles`, `created_at`;
5. `income_category`: організує категорії доходів і включає `id`, `user_id`, `category_name`, `icon_path`;
6. `income`: деталізує доходи через `id`, `income_category_id`, `user_id`, `amount`, та `created_at`.

#### **Зв'язки:**

1. `expense_category` зв'язана з таблицею `expense` через атрибут `expense_category_id`;
2. `budget` має зв'язки з таблицею `expense_category` за допомогою `expense_category_id` і таблицею `user` через `user_id`;
3. `expense` також зв'язана з таблицею `user`, використовуючи атрибут `user_id`;
4. `income_category` під'єднана до таблиці `income` через `income_category_id`;
5. `income` має зв'язок із таблицею `user` за допомогою `user_id`.

Ця структура забезпечує чітке відображення фінансових операцій, їх класифікацію та контроль над лімітами.

## **3.2 Вибір системи управління базою даних та її реалізація**

Система управління базами даних (СУБД) є ключовим компонентом будь-якого сучасного інформаційного програмного забезпечення, оскільки забезпечує збереження, організацію та ефективну обробку даних. У контексті розробки програмного забезпечення для відстеження особистих фінансів

СУБД відіграє критичну роль, оскільки саме в ній зберігаються всі фінансові транзакції, категорії витрат і доходів, дані користувачів та аналітична інформація.

Існує велика кількість систем управління базами даних (СУБД), які застосовуються в сучасних інформаційних системах залежно від вимог до продуктивності, масштабованості, типу даних та архітектури програмного забезпечення. Умовно їх поділяють на реляційні та нереляційні (NoSQL) системи.

До реляційних СУБД належать системи, в яких дані організовані у вигляді таблиць із чітко визначеними зв'язками між ними. Найпоширенішими представниками цього класу є MySQL, PostgreSQL, MariaDB, Microsoft SQL Server, Oracle Database та SQLite. Такі системи широко використовуються у веб-розробці та корпоративних застосунках, оскільки забезпечують цілісність даних, підтримку складних SQL-запитів і надійну роботу з транзакціями.

MySQL є однією з найпопулярніших СУБД у сфері веб-застосунків завдяки простоті використання, високій продуктивності та зручній інтеграції з мовою PHP. PostgreSQL вважається більш функціонально розвиненою системою, яка підтримує складні типи даних, розширені можливості роботи з JSON та суворіше дотримання стандартів SQL. MariaDB є форком MySQL і пропонує покращення продуктивності та оптимізації. Microsoft SQL Server переважно використовується у корпоративних рішеннях на платформі Microsoft, тоді як Oracle Database застосовується у великих високонавантажених системах із підвищеними вимогами до надійності. SQLite є легкою вбудованою СУБД і використовується переважно у мобільних та невеликих застосунках [15].

Окрім реляційних рішень, існують NoSQL СУБД, які призначені для роботи з неструктурованими або слабоструктурованими даними. До них належать MongoDB, Redis, Cassandra, а також хмарні рішення, такі як Firebase Realtime Database. MongoDB зберігає дані у вигляді документів у форматі, подібному до JSON, Redis використовується як високошвидкісне сховище

даних у пам'яті, Cassandra призначена для розподілених систем із великими обсягами даних, а Firebase часто застосовується у веб- та мобільних застосунках завдяки простій інтеграції та хмарній архітектурі.

Для систем обліку фінансів найчастіше використовуються реляційні бази даних, оскільки такі системи потребують чіткої структури даних, підтримки зв'язків між сутностями та високого рівня цілісності інформації. Фінансові транзакції, категорії витрат і доходів, бюджети та дані користувачів природно описуються у табличній формі, що робить реляційні СУБД найбільш доцільним вибором. Зокрема, MySQL або PostgreSQL є оптимальними рішеннями для подібних проєктів завдяки стабільності, продуктивності та широким можливостям інтеграції з веб-технологіями.

Для реалізації даного проєкту була обрана реляційна система управління базами даних MySQL. Даний вибір обґрунтовується її високою продуктивністю, стабільністю, широкою підтримкою з боку спільноти розробників та інтеграцією з веб-технологіями, зокрема з мовою програмування PHP та фреймворком Symfony. MySQL також забезпечує підтримку транзакцій, механізмів цілісності даних та масштабованість, що є важливим для фінансових інформаційних систем.

Однією з ключових переваг MySQL є її здатність ефективно працювати з великими обсягами структурованих даних. Це особливо важливо для системи фінансового обліку, де кількість записів може швидко зростати внаслідок постійного додавання транзакцій користувачами. Крім того, MySQL підтримує індексацію даних, що дозволяє значно прискорити виконання запитів та підвищити загальну продуктивність системи.

Реалізація бази даних у межах проєкту здійснюється на основі попередньо розробленої ER-моделі та сховища даних. Структура бази даних включає таблиці для зберігання користувачів, фінансових операцій, категорій доходів і витрат, валют, способів оплати, а також аналітичних даних для подальшої обробки. Між таблицями встановлюються зв'язки за допомогою

первинних і зовнішніх ключів, що забезпечує цілісність даних та логічну узгодженість структури.

Особлива увага приділяється нормалізації бази даних, що дозволяє зменшити дублювання інформації та забезпечити ефективне оновлення даних. Це є важливим аспектом для фінансових систем, де точність та консистентність інформації мають критичне значення.

Інтеграція MySQL із фреймворком Symfony здійснюється за допомогою ORM (Object-Relational Mapping), зокрема Doctrine. Це дозволяє працювати з базою даних через об'єктно-орієнтовану модель, що спрощує розробку, підвищує читабельність коду та зменшує ймовірність помилок при роботі з SQL-запитами.

Таким чином, вибір MySQL як системи управління базою даних є обґрунтованим з точки зору продуктивності, надійності та інтеграції з обраним технологічним стеком. Реалізація бази даних на її основі забезпечує ефективне зберігання та обробку фінансової інформації, що є критично важливим для функціонування системи відстеження особистих фінансів.

Під час проєктування таблиць у фізичній моделі основою слугували сутності з логічної моделі. Атрибути цих сутностей були використані для розробки структури таблиці. Отриману схему бази даних зображено на рис. 3.2

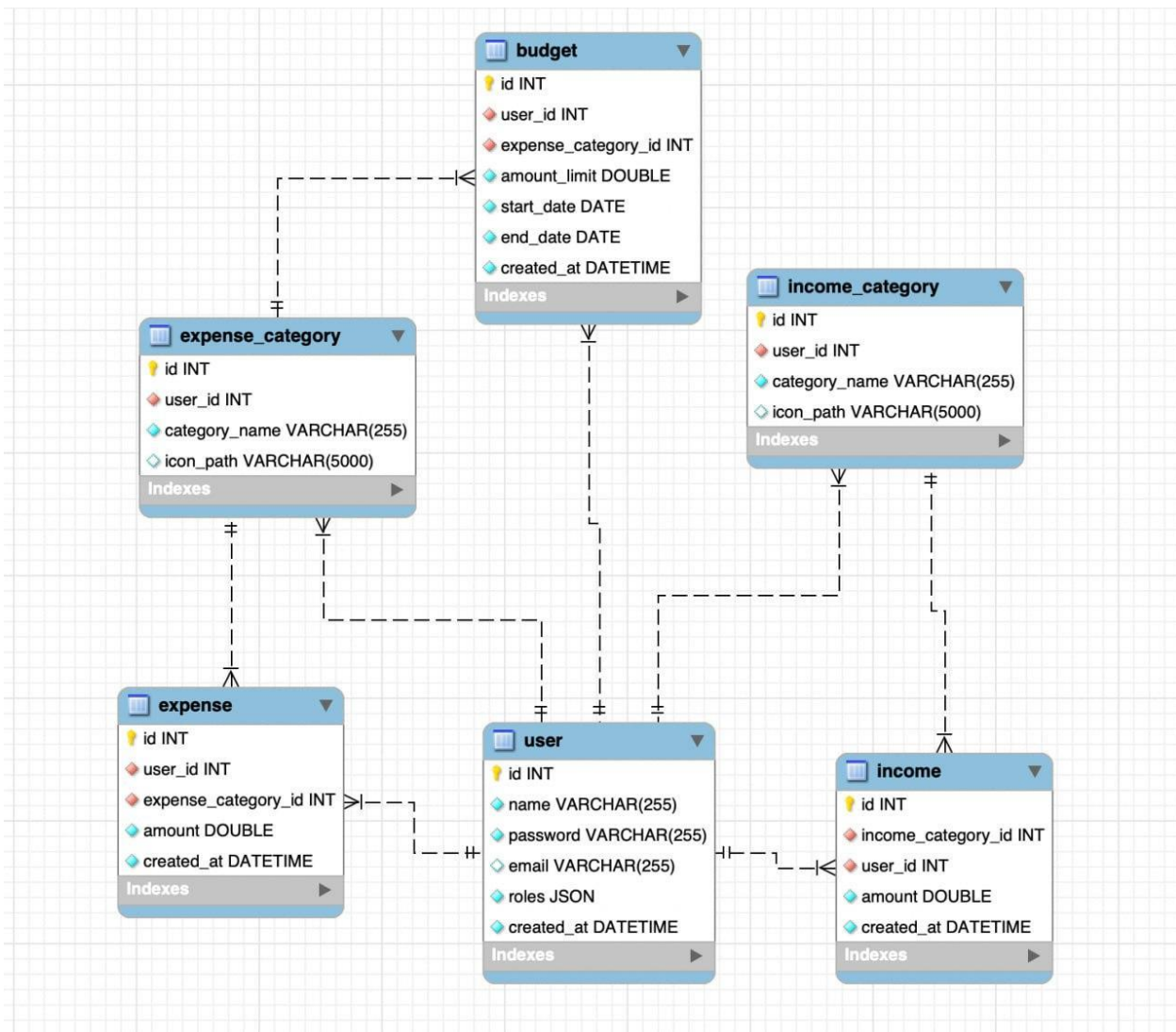


Рис. 3.2 База даних системи

У цій базі даних є кілька основних таблиць, які разом формують систему для управління фінансами користувачів. Центральною є таблиця користувачів, де зберігається інформація про ім'я, електронну пошту, пароль, ролі та дату створення запису. Всі інші сутності пов'язані з користувачем через його ідентифікатор.

Для витрат існує таблиця, що містить суму витрати, дату її створення та посилання на категорію. Категорії витрат описані окремо, кожен користувач може мати власні категорії з назвою та іконкою. Аналогічно побудована структура для доходів: є таблиця доходів із сумою та датою, а також таблиця категорій доходів, де визначаються назви та візуальні позначення.

Окремо виділена таблиця бюджету, яка дозволяє встановлювати фінансові ліміти. Вона пов'язана з користувачем і категорією витрат, містить дані про граничну суму, період дії бюджету та дату створення.

Таким чином, таблиці утворюють логічну систему: користувач має власні категорії доходів і витрат, може реєструвати всі фінансові операції та контролювати їх у межах заданих бюджетів. Зв'язки між таблицями забезпечують узгодженість даних і дозволяють легко отримувати аналітичну інформацію про фінансову діяльність.

### **3.3 Архітектура програмного забезпечення**

Архітектура програмного забезпечення для системи відстеження особистих фінансів визначає загальну структуру застосунку, принципи взаємодії між його компонентами та способи організації обробки даних. Вона є ключовим етапом проєктування, оскільки забезпечує масштабованість, підтримуваність та гнучкість майбутньої системи.

У межах даного проєкту обрана багаторівнева клієнт-серверна архітектура, яка є типовою для сучасних веб-застосунків. Такий підхід дозволяє розділити систему на логічно незалежні рівні, кожен з яких виконує власні функції та взаємодіє з іншими через чітко визначені інтерфейси.

Першим рівнем є презентаційний рівень (frontend), який відповідає за взаємодію з користувачем. Він реалізує користувацький інтерфейс, через який користувач може вводити фінансові дані, переглядати звіти, аналізувати витрати та керувати категоріями. У веб-застосунку цей рівень реалізується за допомогою HTML, CSS та JavaScript.

Другим рівнем є рівень бізнес-логіки (backend), який є основним компонентом системи. Він відповідає за обробку даних, реалізацію бізнес-правил, перевірку коректності введеної інформації та виконання фінансових розрахунків. У даному проєкті цей рівень реалізовано за допомогою PHP із

використанням фреймворку Symfony, що забезпечує структурованість коду та зручність масштабування.

Третім рівнем є рівень доступу до даних, який забезпечує взаємодію із системою управління базами даних. Він відповідає за збереження, оновлення та отримання інформації з бази даних MySQL. Для організації роботи з даними використовується ORM Doctrine, що дозволяє працювати з базою даних через об'єктно-орієнтовану модель і зменшує залежність від прямого SQL-коду.

Окремим важливим компонентом архітектури є база даних, яка реалізує зберігання всіх фінансових даних користувачів. Вона включає таблиці користувачів, транзакцій, категорій, бюджетів та аналітичних даних. Структура бази даних проєктується на основі ER-моделі та сховища даних, що дозволяє забезпечити цілісність і узгодженість інформації.

Також у межах архітектури може бути виділений аналітичний модуль, який відповідає за обробку даних для побудови звітів та виконання елементів інтелектуального аналізу даних (Data Mining). Цей модуль може бути інтегрований у бекенд або реалізований як окремий сервіс для виконання складних обчислень.

Таким чином, запропонована архітектура забезпечує чіткий поділ відповідальності між компонентами системи, спрощує подальшу підтримку та розвиток програмного забезпечення, а також створює основу для масштабування функціональності системи відстеження особистих фінансів.

Загалом, архітектурний підхід зосереджений на чіткості, модульності та масштабованості. Він підтримує ефективну розробку та забезпечує безпечне, орієнтоване на користувача середовище для ефективного управління особистими фінансами.

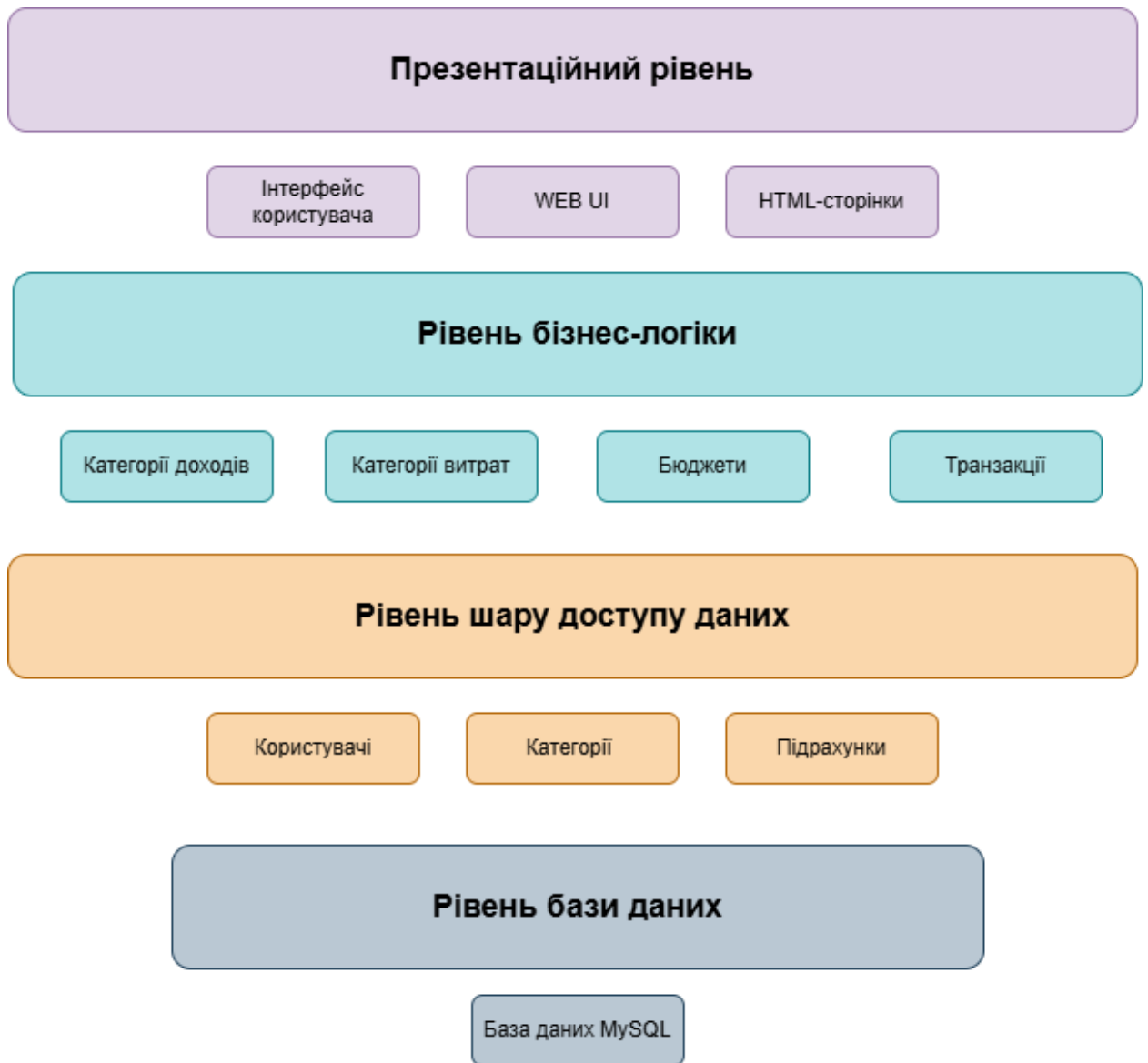


Рис. 3.3 Багатошарова архітектура

Ця архітектура програмного забезпечення побудована за принципом багаторівневої структури, забезпечуючи чітке розмежування функціональних шарів для полегшення розробки, обслуговування та масштабованості системи.

На презентаційному рівні знаходиться інтерфейс користувача, включаючи веб-інтерфейс (WEB UI) та HTML-сторінки, які відповідають за взаємодію користувача з системою. Цей рівень забезпечує доступність і зрозумілість функціоналу для користувача через зручний дизайн і інтерактивність.

Рівень бізнес-логіки відповідає за основні функції програми, такі як обробка категорій доходів та витрат, управління бюджетами та обробка

транзакцій. Він виконує ключову роль у реалізації правил і поведінки системи, забезпечуючи виконання основних завдань.

Рівень доступу до даних фокусується на управлінні збереженням і доступом до інформації. Він включає управління користувачами, обробку категорій та виконання необхідних розрахунків. Цей шар забезпечує ефективну взаємодію з базою даних і оптимізацію роботи з великим обсягом даних.

На рівні бази даних знаходиться MySQL, яка виконує роль надійного сховища для всієї інформації системи. Цей рівень працює з таблицями, що структурують та упорядковують дані, забезпечуючи їхню цілісність і доступність.

Така структура сприяє модульності, зручності в обслуговуванні й гнучкості системи, дозволяючи легко впроваджувати нові функціональні можливості або масштабувати систему.

### **3.4 Використання 1-Rule для класифікації**

У процесі розробки програмного забезпечення для відстеження особистих фінансів класифікація фінансових даних відіграє важливу роль, оскільки дозволяє структурувати інформацію та підвищити ефективність її подальшого аналізу. Одним із простих та ефективних підходів до класифікації є застосування методу 1-Rule, який являє собою правило прийняття рішень, що базується на використанні одного визначального атрибута для віднесення об'єкта до певної категорії. У межах даної системи 1-Rule може використовуватися для автоматичної класифікації фінансових транзакцій, категорій витрат та доходів відповідно до їх основних характеристик.

Принцип 1-Rule полягає в тому, що для кожної фінансової операції визначається один ключовий параметр, який має найбільший вплив на її класифікацію. Наприклад, таким атрибутом може виступати тип транзакції (дохід або витрата), категорія операції або спосіб платежу. На основі цього

параметра система відносить кожну операцію до відповідної групи, забезпечуючи просту та зрозумілу логіку класифікації без необхідності використання складних моделей [15].

Застосування 1-Rule у системі обліку фінансів дозволяє автоматизувати процес структурування даних та зменшити кількість ручних операцій з боку користувача. Наприклад, усі транзакції з негативним значенням суми можуть автоматично класифікуватися як витрати, тоді як операції з позитивним значенням — як доходи. Аналогічно, операції можуть бути згруповані за категоріями, такими як харчування, транспорт, житло або розваги, на основі одного визначеного атрибута.

Використання підходу 1-Rule також сприяє підвищенню ефективності аналітичної обробки даних. Після первинної класифікації фінансових операцій система може використовувати отримані категорії для формування звітів, аналізу витрат та виявлення фінансових тенденцій. Це дозволяє швидко отримувати узагальнену інформацію про фінансову поведінку користувача та приймати обґрунтовані рішення щодо управління бюджетом.

На малюнку 3.4 показано запит на прогнозування доходів або витрат.

```

1  <?php
2
3  class Transaction
4  {
5      public function __construct(
6          public float $amount,
7          public string $description = ''
8      ) {}
9  }
10
11 class OneRuleClassifier
12 {
13     private float $incomeThreshold;
14     private float $highIncomeThreshold;
15
16     public function __construct(
17         float $incomeThreshold = 0,
18         float $highIncomeThreshold = 10000
19     ) {
20         $this->incomeThreshold = $incomeThreshold;
21         $this->highIncomeThreshold = $highIncomeThreshold;
22     }
23
24     public function classify(Transaction $transaction): string
25     {
26         $amount = $transaction->amount;
27
28         if ($amount < $this->incomeThreshold) {
29             return 'expense';
30         }
31
32         if ($amount >= $this->highIncomeThreshold) {
33             return 'income_high';
34         }
35
36         return 'income_standard';
37     }
38 }
39
40 $classifier = new OneRuleClassifier();
41
42 $transactions = [
43     new Transaction(1500, 'Salary'),
44     new Transaction(-250, 'Groceries'),
45     new Transaction(12000, 'Bonus'),
46     new Transaction(300, 'Freelance'),
47     new Transaction(-75, 'Transport'),
48 ];
49
50 foreach ($transactions as $transaction) {
51     $category = $classifier->classify($transaction);
52
53     echo sprintf(
54         "Amount: %.2f | Description: %s | Class: %s\n",
55         $transaction->amount,
56         $transaction->description,
57         $category
58     );
59 }

```

Рис. 3.4 Запит на прогнозування доходів або витрат

Застосування методу 1-Rule у системі відстеження особистих фінансів забезпечує простий та ефективний механізм класифікації даних, підвищує

рівень автоматизації обробки фінансової інформації та створює основу для подальшого аналітичного опрацювання даних у межах системи.

Цей код реалізує простий алгоритм класифікації фінансових транзакцій на основі принципу 1-Rule, тобто прийняття рішення лише за одним атрибутом — сумою операції (amount).

```
Amount: 1500.00 | Description: Salary | Class: income_standard
Amount: -250.00 | Description: Groceries | Class: expense
Amount: 12000.00 | Description: Bonus | Class: income_high
Amount: 300.00 | Description: Freelance | Class: income_standard
Amount: -75.00 | Description: Transport | Class: expense
```

Рис. 3.5 Результат прогнозування

Спочатку визначається клас Transaction, який описує фінансову транзакцію. Він містить два основні поля: суму операції та її опис. Така структура імітує реальну доменну модель, яка може використовуватися у веб-застосунку на Symfony, де кожна транзакція представлена у вигляді окремого об'єкта.

Далі створюється клас OneRuleClassifier, який реалізує логіку класифікації. У конструкторі задаються порогові значення: базовий поріг (incomeThreshold) та поріг для високого доходу (highIncomeThreshold). Це дозволяє гнучко налаштовувати поведінку правила без зміни основного коду програми.

Основний метод classify() приймає об'єкт транзакції та аналізує лише одне його поле — amount. На основі значення суми приймається рішення щодо класифікації:

- якщо сума менша за нуль, транзакція відноситься до витрат (expense);
- якщо сума більша або дорівнює нулю, але менша за поріг високого доходу, вона класифікується як стандартний дохід (income\_standard);
- якщо сума перевищує встановлений високий поріг, транзакція відноситься до категорії високого доходу (income\_high).

Таким чином, незважаючи на різні результати класифікації, логіка прийняття рішення базується лише на одному параметрі, що повністю відповідає принципу 1-Rule.

У основній частині програми створюється екземпляр класифікатора та набір тестових транзакцій із різними значеннями сум. Далі виконується цикл, у якому кожна транзакція передається до методу класифікації, а результат виводиться разом із сумою та описом операції.

У підсумку програма автоматично присвоює кожній фінансовій операції відповідну категорію, демонструючи базовий принцип інтелектуальної обробки даних, який може використовуватися в системах обліку особистих фінансів.

### **3.5 Вибір інструментарію для створення програмного забезпечення**

У процесі розробки програмного забезпечення для відстеження особистих фінансів важливим етапом є вибір технологічного стеку, який визначає продуктивність, масштабованість та зручність подальшої підтримки системи. Обрана сукупність інструментів повинна забезпечувати ефективну реалізацію як серверної, так і клієнтської частини веб-застосунку, а також підтримку роботи з базою даних і можливості розширення функціональності.

Для реалізації серверної частини системи використовується мова програмування PHP, яка є однією з найпоширеніших технологій для веб-розробки. PHP забезпечує стабільну роботу з серверною логікою, обробку HTTP-запитів та інтеграцію з базами даних. У поєднанні з фреймворком Symfony дана мова дозволяє створювати структуровані, масштабовані та підтримувані веб-застосунки. Symfony забезпечує реалізацію архітектурного патерну MVC, що сприяє розділенню бізнес-логіки, представлення та доступу до даних.

Для збереження та обробки даних використовується система управління базами даних MySQL, яка забезпечує надійне зберігання структурованої інформації. Вона підтримує роботу з транзакціями, індексацією та складними SQL-запитами, що є важливим для фінансових систем, де необхідна висока точність та цілісність даних.

Клієнтська частина веб-застосунку реалізується за допомогою HTML, CSS та JavaScript. HTML використовується для побудови структури веб-сторінок, CSS — для стилізації та створення адаптивного інтерфейсу, а JavaScript забезпечує динамічну взаємодію користувача із системою.

Для покращення зовнішнього вигляду та прискорення розробки інтерфейсу використовується CSS-фреймворк Bootstrap, який надає готові компоненти інтерфейсу, такі як форми, кнопки, таблиці та модальні вікна. Це дозволяє забезпечити адаптивність системи та її коректне відображення на різних пристроях.

Додатково у проєкті використовується Node.js як допоміжна технологія для реалізації інструментів розробки, пакетного управління (npm), а також для побудови додаткових сервісів або модулів, пов'язаних із обробкою даних у реальному часі.

Обраний технологічний стек, що включає PHP, Symfony, MySQL, HTML, CSS, JavaScript, Bootstrap та Node.js, забезпечує повний набір інструментів для створення сучасного веб-застосунку для обліку особистих фінансів. Він дозволяє реалізувати надійну серверну логіку, зручний користувацький інтерфейс та ефективну роботу з даними, що є критично важливим для подібних інформаційних систем.

## **4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ**

### **4.1 Вимоги до апаратного та програмного забезпечення**

Для забезпечення коректної роботи програмного забезпечення для відстеження особистих фінансів необхідно визначити мінімальні та рекомендовані апаратні й програмні вимоги. Вони гарантують стабільну продуктивність системи, швидку обробку даних та зручну взаємодію користувача із веб-застосунком.

Апаратні вимоги залежать від характеру використання системи та навантаження на серверну частину. Оскільки розроблюване програмне забезпечення є веб-застосунком, основна обробка даних виконується на сервері, а клієнтська частина потребує лише браузера для доступу до системи.

Для клієнтської частини мінімальні вимоги включають наявність персонального комп'ютера, ноутбука або мобільного пристрою з доступом до інтернету. Рекомендовано використання пристроїв із сучасним процесором та щонайменше 4 ГБ оперативної пам'яті для забезпечення комфортної роботи з інтерфейсом системи.

Серверна частина системи повинна мати більш високі характеристики, оскільки саме вона відповідає за обробку запитів, роботу бізнес-логіки та взаємодію з базою даних. Мінімальні вимоги до сервера включають багатоядерний процесор, щонайменше 8 ГБ оперативної пам'яті та достатній обсяг дискового простору для зберігання даних користувачів і резервних копій.

Щодо програмних вимог, система розробляється із використанням мови програмування PHP, фреймворку Symfony та системи управління базами даних MySQL. Для коректного функціонування необхідна встановлена серверна платформа, яка підтримує виконання PHP-коду, наприклад Apache або Nginx у поєднанні з PHP-FPM.

Рекомендовані вимоги до апаратного та програмного забезпечення для успішного впровадження та роботи системи подано в таблиці 4.1.

Таблиця 4.1

## Вимоги до апаратного та програмного забезпечення

| Тип вимог              | Найменування                          | Мінімальні вимоги               | Рекомендовані вимоги                                |
|------------------------|---------------------------------------|---------------------------------|-----------------------------------------------------|
| Апаратне забезпечення  | Процесор (CPU)                        | Intel Core i3 або еквівалент    | Intel Core i5 / Ryzen 5 або вище                    |
|                        | Оперативна пам'ять (RAM)              | 4 ГБ                            | 8 ГБ або більше                                     |
|                        | Жорсткий диск (HDD/SSD)               | 10 ГБ вільного місця            | SSD на 50 ГБ або більше                             |
|                        | Мережева карта                        | Ethernet 100 Mbps               | Ethernet 1 Gbps                                     |
|                        | Монітор                               | Роздільна здатність 1280×720    | Роздільна здатність 1920×1080                       |
| Програмне забезпечення | Операційна система                    | Windows 10 / Linux Ubuntu 20.04 | Windows 11 / Ubuntu 22.04 LTS                       |
|                        | Система керування базами даних (СУБД) | MySQL 8.0                       | MySQL 8.0 або вище                                  |
|                        | Сервер середовища виконання           | PHP 8.2                         | PHP 8.2 з розширеннями для Symfony                  |
|                        | Фреймворк                             | Symfony 6                       | Symfony 6 або вище                                  |
|                        | Середовище розробки                   | Visual Studio Code              | Visual Studio Code з плагінами для PHP, MySQL       |
|                        | Система контейнеризації               | Docker Desktop                  | Docker Desktop останньої версії                     |
|                        | Додаткове ПЗ                          | Git, Composer, Node.js          | Git, Composer, Node.js, npm, браузер Chrome/Firefox |
|                        |                                       |                                 |                                                     |

Діаграма розгортання (Deployment Diagram) є одним із видів UML-діаграм, яка використовується для відображення фізичної архітектури програмної системи. Вона показує, як програмні компоненти розміщуються на апаратних вузлах, а також яким чином відбувається взаємодія між серверною та клієнтською частинами системи [19].

У контексті програмного забезпечення для відстеження особистих фінансів діаграма розгортання дозволяє візуалізувати структуру системи на рівні інфраструктури, включаючи клієнтські пристрої користувачів, веб-сервер, сервер застосунку та базу даних. Такий підхід дає змогу чітко визначити, де саме виконуються окремі частини програмної логіки та як організовано обмін даними між ними.

Основними елементами діаграми розгортання є вузли (nodes), артефакти (artifacts) та канали зв'язку. Вузли представляють фізичні або віртуальні середовища, такі як сервери або клієнтські пристрої. Артефакти відображають програмні компоненти, наприклад веб-застосунок, модулі бізнес-логіки або базу даних. Канали зв'язку визначають спосіб передачі даних між вузлами, зазвичай через HTTP/HTTPS протоколи.

У розроблюваній системі відстеження фінансів клієнтський вузол представлений веб-браузером користувача, через який здійснюється доступ до інтерфейсу системи. Серверний вузол включає веб-сервер та сервер застосунку, на якому виконується бізнес-логіка, реалізована за допомогою PHP та фреймворку Symfony. Окремим вузлом виступає сервер бази даних MySQL, який відповідає за збереження та обробку фінансової інформації.

Взаємодія між вузлами здійснюється через мережеві запити, які передаються за протоколом HTTPS. Користувач надсилає запити через браузер, сервер застосунку обробляє їх, виконує необхідні операції з базою даних і повертає результат у вигляді веб-сторінок або JSON-відповідей.

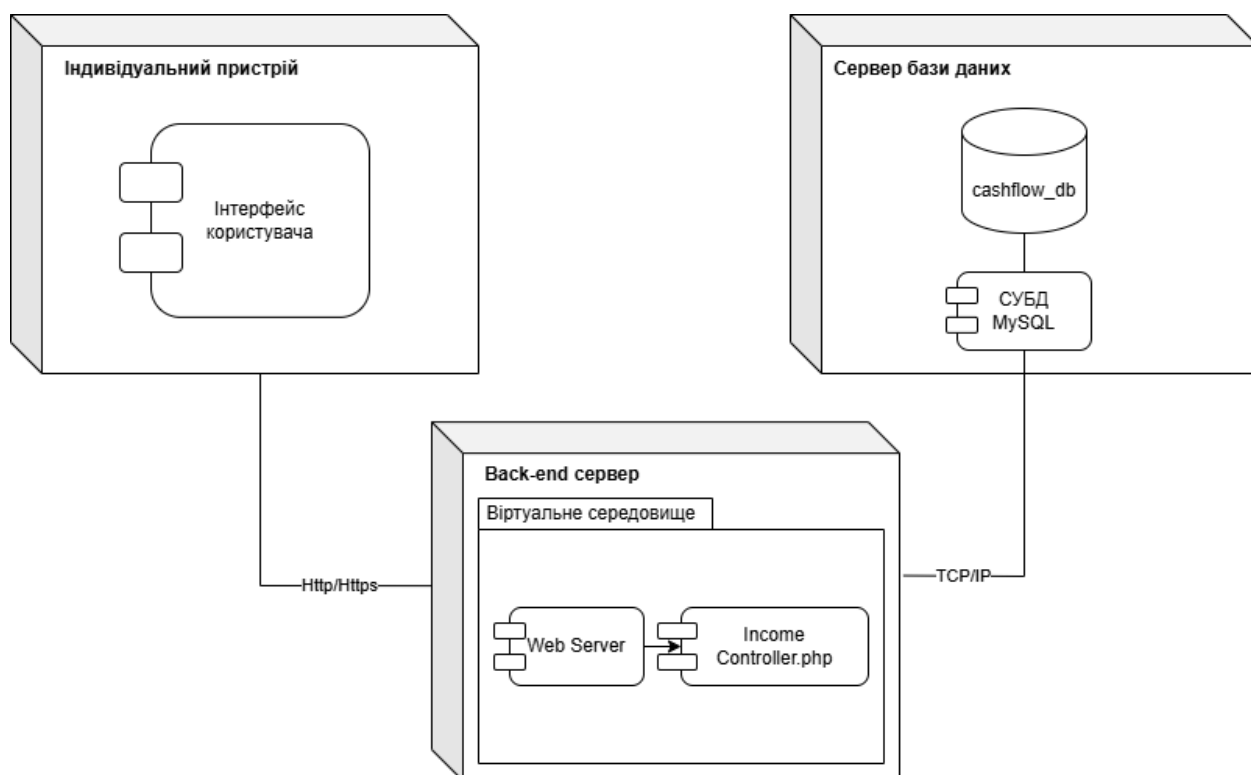


Рис. 4.1 Діаграма розгортання

## 4.2 Тестування системи

Запустивши систему, потрапляємо на форму авторизації, тут нам треба ввести логін та пароль користувача (рис. 4.2).

The screenshot shows the CashFlow login interface. At the top is the CashFlow logo. Below it is the heading "Увійдіть у свій акаунт" (Log in to your account) and the instruction "Введіть ваше ім'я користувача та пароль для входу" (Enter your username and password for login). There are two input fields: "Ім'я користувача" (Username) with an "@" symbol and "Пароль" (Password). A blue button labeled "Увійти" (Log in) is positioned below the fields. At the bottom, there is a link: "Ще не маєте акаунта? Створити акаунт" (Don't have an account? Create an account).

Рис. 4.2 Форма авторизації

Спочатку нас зустрічає основна сторінка. Тут показуються всі наші дані про фінанси у вигляді категорій доходів, категорій витрат, які суми на що витрачені, скільки зараз є грошей у володінні, який дохід та витрати у поточному місяці (Рис. 4.3).

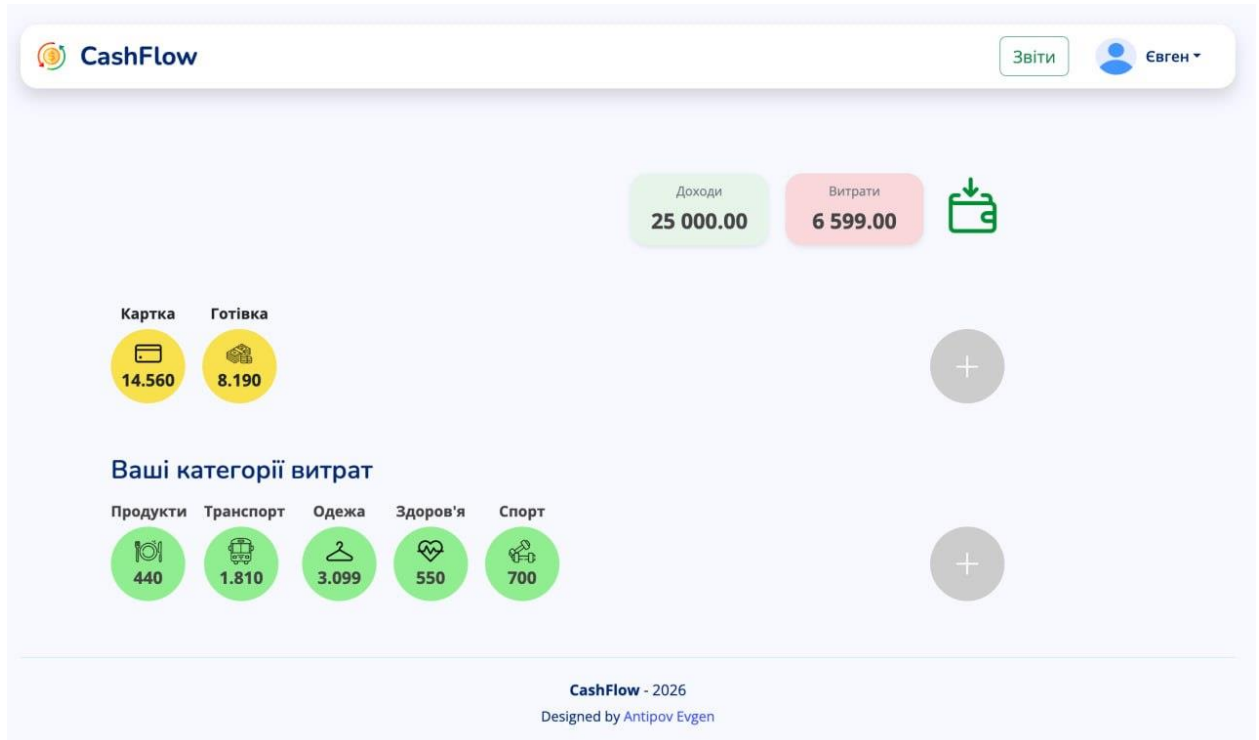


Рис. 4.3 Головна сторінка

Спробуємо зараз створити нову категорію доходів, для цього нам потрібно натиснути на сірий плюс з правого краю (Рис. 4.4) і у нас відкриється форма для заповнення даних про категорію доходу (Рис. 4.5).

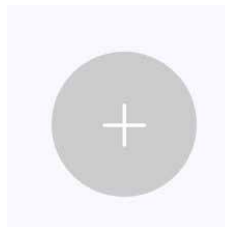
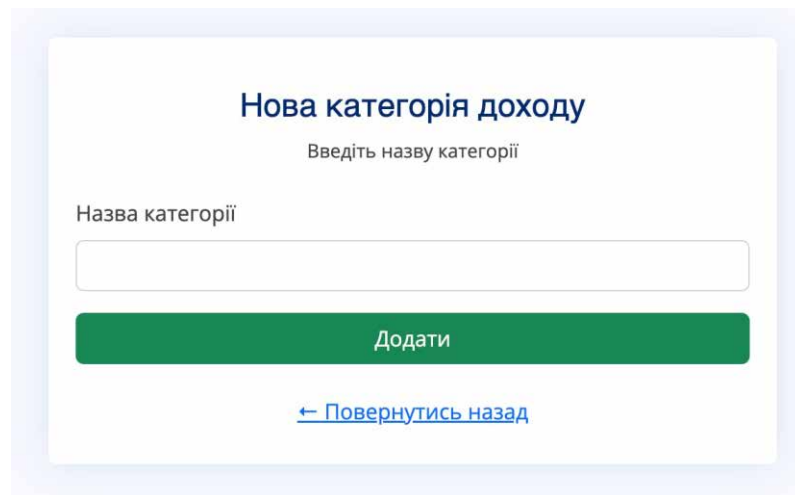


Рис. 4.4 Елемент створення



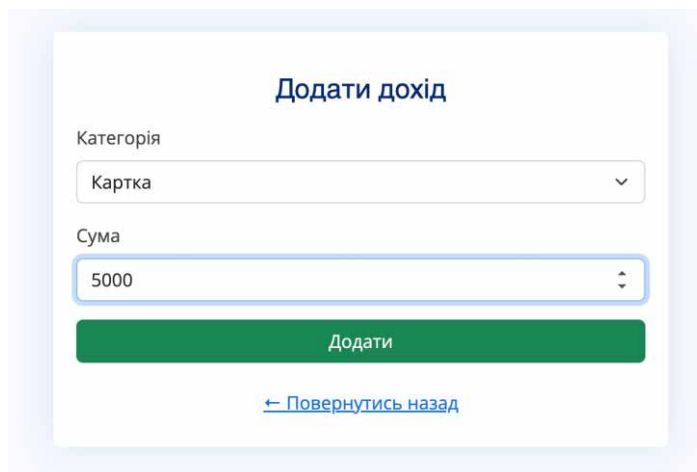
The form is titled "Нова категорія доходу" (New income category). Below the title is a subtitle "Введіть назву категорії" (Enter category name). There is a text input field labeled "Назва категорії" (Category name). Below the input field is a green button labeled "Додати" (Add). At the bottom of the form is a blue link labeled "← Повернутись назад" (← Go back).

Рис. 4.5 Форма створення нової категорії

Тепер ми можемо додати певну суму до нашої категорії доходу, для цього нам потрібно натиснути на іконку зеленого гаманця по правому краю (Рис. 4.6), яка відкриє нам форму внесення суми до обраної нами категорії (Рис. 4.7).



Рис. 4.6 Елемент додання доходу



The form is titled "Додати дохід" (Add income). It has two main sections: "Категорія" (Category) with a dropdown menu showing "Картка" (Card) and "Сума" (Sum) with a numeric input field showing "5000". Below these fields is a green button labeled "Додати" (Add). At the bottom of the form is a blue link labeled "← Повернутись назад" (← Go back).

Рис. 4.7 Форма додання нового доходу

Після додавання суми до категорії доходу, у нас оновиться число в даній категорії (Рис. 4.8).



Рис. 4.8 Наша створена категорія доходу

Якщо натиснути на категорію доходу, у нас відкриється її налаштування, тут ми можемо змінити назву або встановити іконку на наш вибір із запропонованих (Рис. 4.9).

 A white rectangular form with a light blue border. At the top, the title 'Редагувати категорію доходу: Картка' is centered in blue. Below it, the subtitle 'Виберіть іконку для категорії та інші параметри' is centered in gray. The form contains a text input field with the label 'Назва категорії' and the value 'Картка'. Below the input field is the label 'Оберіть іконку для категорії'. There are two icons: a stack of money on the left and a credit card on the right. At the bottom, there is a green button with the text 'Оновити категорію' and a blue link with a left arrow and the text 'Повернутись назад'.

Рис. 4.9 Форма редагування категорії доходу

Якщо натиснути на категорію витрати, у нас відкриється її сторінка де ми зможемо вказати суму витрати яку ми виробили, можемо також встановити ліміт для даної категорії вказавши період і саму суму ліміту. І так само є налаштування іконок для категорій, щоб візуально швидше знаходити потрібні нам категорії (Рис. 4.10)

**Спорт**  
Оберіть категорію доходів та суму витрати

Чим платити  
Оберіть категорію

Сума

Додати витрату

[← Повернутись назад](#)

**Встановити ліміт**  
Задайте ліміт витрат для категорії

Ліміт (€)

Початок Кінець  
ДД.ММ.ГГГГ ДД.ММ.ГГГГ

Зберегти ліміт

**Оберіть іконку для категорії**  
Виберіть одну з іконок, щоб додати до категорії витрат

Іконки: автобус, гаманець, одяг, їжа, серце, глобус

Підтвердити вибір іконки

Рис. 4.10 Форма категорії витрат

Спробуємо заздалегідь встановити ліміт для категорії, наприклад 1000 гривень, і спробувати ввести суму витрати  $>1000$ , то ми отримаємо повідомлення про те, що дана витрата перевищує ліміт і в результаті транзакція не буде записана (Рис. 4.11).

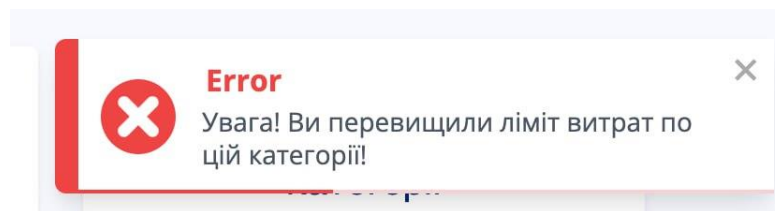


Рис. 4.11 Повідомлення про перевищення ліміту

І насамкінець, у цій програмі представлений функціонал звітності, нам потрібно перейти на сторінку звітів і тут вибрати діапазон дат, що цікавить нас, за яким буде проводиться підрахунок статистики. Тут у нас показується статистика з доходів і витрат для конкретного користувача (Рис. 4.2.12).

**CashFlow** Звіти Евген

**Звіт за період**

12.03.2026 30.04.2026 Сформувати

|                          |                                                   |
|--------------------------|---------------------------------------------------|
| <b>Доходи:</b>           | <b>Витрати:</b>                                   |
| Сума доходів: 25 000 грн | Сума витрат: 6 599 грн                            |
| Кількість транзакцій: 2  | Кількість транзакцій: 6                           |
|                          | Найпопулярніша категорія витрат: Одяг (3 099 грн) |

**CashFlow - 2026**  
Designed by Antipov Evgen

Рис. 4.12 Сторінка “Звіти”

### 4.3 Складу інсталяційного пакету

Інсталяційний пакет розробленого програмного забезпечення для відстеження особистих фінансів містить повний набір програмних компонентів, конфігураційних файлів та службових ресурсів, необхідних для розгортання, налаштування та експлуатації системи у локальному або серверному середовищі. Загальний обсяг інсталяційного пакета становить близько 350 МБ без урахування кешу та тимчасових файлів.

Програмна система реалізована з використанням мови програмування PHP 8.2, фреймворку Symfony 6.4 та системи управління базами даних MySQL 8.0. Для роботи frontend-частини використовуються HTML, CSS, JavaScript та Bootstrap 5. Для управління залежностями застосовується Composer та Node.js із менеджером пакетів npm.

До складу інсталяційного пакета входять такі компоненти:

Вихідний код програмної системи

Вихідний код розміщується у каталозі /src та містить:

- контролери для обробки HTTP-запитів;
- сервіси бізнес-логіки;
- моделі та сутності Doctrine ORM;
- репозиторії для роботи з базою даних;
- модулі аналітики фінансових операцій;
- класи для реалізації алгоритмів класифікації даних.

**Клієнтська частина системи**

Файли інтерфейсу користувача розміщуються у каталогах /templates, /assets та /public і включають:

- HTML/Twig-шаблони сторінок;
- CSS-стили;
- JavaScript-файли;
- Bootstrap-компоненти;
- графічні ресурси та іконки.

## Конфігураційні файли

- Конфігураційні параметри системи містяться у каталогах `/config` та `.env` файлах і включають:
- параметри підключення до бази даних;
- налаштування маршрутизації;
- конфігурацію безпеки;
- параметри середовища виконання;
- службові налаштування Symfony.

## База даних

- До складу пакета входить SQL-дамп бази даних, який містить:
- структуру таблиць;
- зв'язки між сутностями;
- початкові системні дані;
- тестові записи для перевірки роботи системи.

## Залежності та бібліотеки

Для забезпечення коректної роботи системи інсталяційний пакет містить:

- PHP-бібліотеки Composer;
- JavaScript-залежності npm;
- сторонні Symfony-компоненти;
- модулі для роботи з ORM та формами.

## Документація

До інсталяційного пакета також входить документація користувача яка містить:

- інструкцію зі встановлення;
- опис налаштування середовища;
- рекомендації щодо запуску системи.

## ВИСНОВКИ

У даній бакалаврській роботі було розглянуто процес проєктування та розробки програмного забезпечення для відстеження особистих фінансів, яке призначене для автоматизації обліку доходів і витрат, аналізу фінансових даних та підтримки прийняття користувацьких рішень щодо управління бюджетом.

У ході дослідження було проаналізовано предметну область фінансового обліку, розглянуто існуючі програмні рішення та визначено їхні основні переваги й недоліки. На основі цього сформовано вимоги до власної системи, які враховують необхідність зручного інтерфейсу, гнучкої структури даних та можливості аналітичної обробки фінансової інформації.

Було спроектовано архітектуру програмного забезпечення, що базується на багаторівневій клієнт-серверній моделі. Розроблено структуру бази даних із використанням концепцій ER-моделювання та сховища даних, що забезпечує цілісність, масштабованість та ефективність зберігання фінансових даних.

Для реалізації програмного продукту було обрано сучасний технологічний стек, який включає PHP, фреймворк Symfony, систему управління базами даних MySQL, а також HTML, CSS, JavaScript і Bootstrap для створення інтерфейсу користувача. Використані технології забезпечили стабільну роботу системи та можливість її подальшого розвитку.

Окрему увагу приділено застосуванню методів інтелектуальної обробки даних, зокрема класифікації фінансових транзакцій та базових алгоритмів аналізу даних, що дозволяє підвищити ефективність роботи з інформацією та покращити якість фінансової аналітики.

У процесі виконання бакалаврської роботи також було встановлено, що запропонований підхід до побудови системи дозволяє забезпечити достатній рівень гнучкості та адаптивності до змін вимог користувачів.

Завдяки використанню сучасного стеку технологій та модульної архітектури система може бути легко розширена новими функціональними можливостями, такими як інтеграція з банківськими сервісами, автоматичне імпортування транзакцій або розширена аналітика витрат із використанням методів Data Mining.

Окремо слід зазначити, що реалізація системи передбачає можливість подальшої оптимізації продуктивності та масштабування. Це досягається за рахунок використання реляційної бази даних, чіткої структуризації даних та розділення бізнес-логіки і рівня доступу до даних. Такий підхід дозволяє забезпечити стабільну роботу системи навіть при збільшенні обсягу фінансових операцій та кількості користувачів.

У результаті виконаної роботи можна сформулювати основні досягнення системи:

- реалізовано веб-застосунок для обліку особистих фінансів із можливістю управління доходами, витратами та категоріями;
- розроблено структуру бази даних, що забезпечує цілісність та ефективне зберігання фінансової інформації;
- впроваджено архітектурний підхід, який забезпечує масштабованість і зручність підтримки програмного продукту;
- застосовано базові методи інтелектуального аналізу даних для класифікації фінансових операцій та підвищення якості аналітики.

Розроблена система для відстеження особистих фінансів забезпечує автоматизацію ключових процесів фінансового обліку, підвищує зручність управління особистим бюджетом та створює основу для подальшого впровадження розширених аналітичних функцій. Отримані результати підтверджують доцільність використання обраних методів і технологій та їх ефективність у межах поставленого завдання.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kapoor J. R., Dlabay L. R., Hughes R. J. Особисті фінанси. McGraw-Hill Education, 2020.
2. Garman E. T., Forgue R. E. Особисті фінанси. Cengage Learning, 2018.
3. Lusardi A., Mitchell O. S. Економічне значення фінансової грамотності: теорія та докази. *Journal of Economic Literature*. 2014.
4. OECD. Рекомендація щодо фінансової грамотності. URL: <https://www.oecd.org/finance/financial-education/> (дата звернення: 22.05.2026).
5. National Endowment for Financial Education. Основи особистих фінансів. URL: <https://www.nefe.org/> (дата звернення: 22.05.2026).
6. Hilgert M. A., Hogarth J. M., Beverly S. G. Управління фінансами домогосподарств: зв'язок між знаннями та поведінкою. *Бюлетень Федеральної резервної системи*. 2003.
7. Комісія з фінансової грамотності та освіти США. Звіт Комісії з фінансової грамотності та освіти. URL: <https://home.treasury.gov/> (дата звернення: 22.05.2026).
8. Гітман Л. Дж., Джоенк М. Д. Особисте фінансове планування. Cengage Learning, 2013.
9. Агентство Канади з питань фінансового захисту прав споживачів. Управління вашими грошима. URL: <https://www.canada.ca/en/financial-consumer-agency.html> (дата звернення: 22.05.2026).
10. Х'юстон С. Дж. Вимірювання фінансової грамотності. *Журнал у справах споживачів*. 2010.
11. CoinKeeper. URL: <https://about.coinkeeper.me> (дата звернення: 22.05.2026).
12. 1Money. URL: <https://1money.uptodown.com/android> (дата звернення: 22.05.2026).

- 13.Посібник з PHP. URL: <https://www.php.net/manual/en/> (дата звернення: 22.05.2026).
- 14.Symfony Framework. URL: <https://symfony.com/> (дата звернення: 22.05.2026).
- 15.Документація MySQL. URL: <https://dev.mysql.com/doc/> (дата звернення: 22.05.2026).
- 16.Doctrine ORM. URL: <https://www.doctrine-project.org/projects/orm.html> (дата звернення: 22.05.2026).
- 17.HTML Living Standard. URL: <https://html.spec.whatwg.org/> (дата звернення: 22.05.2026).
- 18.CSS: Каскадні таблиці стилів. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (дата звернення: 22.05.2026).
- 19.Посібник з JavaScript. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide> (дата звернення: 22.05.2026).
- 20.PlantUML. URL: <https://plantuml.com/> (дата звернення: 22.05.2026).
- 21.Інструмент для створення діаграм yUML. URL: <https://yuml.me/> (дата звернення: 22.05.2026).
- 22.Інструмент для створення діаграм Draw.io. URL: <https://app.diagrams.net/> (дата звернення: 22.05.2026).
- 23.ПРОЄКТУВАННЯ ER-ДІАГРАМИ. URL: <https://nationalteam.worldskills.ua/skills/proektirovanie-er-diagrammy/> (дата звернення: 22.05.2026).
- 24.Діаграма варіантів використання (UseCase diagram). URL: [https://flexberry.github.io/ua/fd\\_use-case-diagram.html](https://flexberry.github.io/ua/fd_use-case-diagram.html) (дата звернення: 22.05.2026).
- 25.ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ. URL: <https://nationalteam.worldskills.ua/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/> (дата звернення: 22.05.2026).

26. What is Sequence Diagram? URL: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/> (дата звернення: 22.05.2026).
27. UML - Activity Diagrams. Tutorialspoint. URL: [https://www.tutorialspoint.com/uml/uml\\_activity\\_diagram.htm](https://www.tutorialspoint.com/uml/uml_activity_diagram.htm) (дата звернення: 22.05.2026).
28. Entity Relationship Diagram - Data Modeling. URL: <https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html> (дата звернення: 22.05.2026).
29. Документація Bootstrap (офіційний сайт). URL: <https://www.getbootstrap.com/documentation/> (дата звернення: 22.05.2026).
30. Що таке Data Mining? Ключові концепції, як це працює? URL: <https://www.upgrad.com/blog/what-is-datamining-key-concepts-how-does-it-work/> (дата звернення: 22.05.2026).
31. Інструменти Data Mining для кращого аналізу даних. URL: <https://www.ionos.co.uk/digitalguide/online-marketing/webanalytics/a-comparison-of-data-mining-tools/> (дата звернення: 22.05.2026).
32. Системи OLAP. URL: <https://pidru4niki.com/1670032447784/informatika/olap-sistemi> (дата звернення: 22.05.2026).
33. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. 3-тє вид. Morgan Kaufmann Publishers, 2011.
34. Наївні баєсівські класифікатори. URL: <https://www.geeksforgeeks.org/naive-bayes-classifiers/> (дата звернення: 22.05.2026).
35. Agrawal R., Imielinski T., Swami A. Правила асоціації між наборами елементів у великих базах даних. *ACM SIGMOD Record*. 1993. Т. 22, № 2. С. 207–216.

## ДОДАТОК А

## Фрагменти програмного коду. Функція створення категорії

```

#[Route('/income/category/add', name: 'add_income_category')]
public function addIncomeCategory(Request $request, EntityManagerInterface
$entityManager): Response
{
    if ($request->isMethod('POST')) {
        $categoryName = $request->request->get('name');
        $user = $this->getUser();

        $category = new IncomeCategory();
        $category->setCategoryName($categoryName);
        $category->setUser($user);

        $entityManager->persist($category);
        $entityManager->flush();

        flash()->success('Ви успішно створили категорію доходів');
        return $this->redirectToRoute('app_home');
    }

    return $this->render('income/add_income_category.html.twig');
}

#[Route('/expense/category/add', name: 'add_expense_category')]
public function addExpenseCategory(Request $request, EntityManagerInterface
$entityManager): Response
2 days ago

created authorization
{
yesterday

created money functions
    if ($request->isMethod('POST')) {
        $categoryName = $request->request->get('name');
        $user = $this->getUser();

        $category = new ExpenseCategory();
        $category->setCategoryName($categoryName);
        $category->setUser($user);

        $entityManager->persist($category);

```

```

        $entityManager->flush();

        flash()->success('Ви успішно створили категорію витрат');
        return $this->redirectToRoute('app_home');
    }

    return $this->render('expense/add_expense_category.html.twig');
}

#[Route('/income/add', name: 'add_income')]
public function addIncome(
    Request $request,
    EntityManagerInterface $entityManager,
    IncomeCategoryRepository $incomeCategoryRepository,
    Security $security
): Response {
    $user = $security->getUser();
    $categories = $incomeCategoryRepository->findBy(['user' => $user]);

    if ($request->isMethod('POST')) {
        $amount = (float)$request->request->get('amount');
        $categoryId = (int)$request->request->get('category');
        $category = $incomeCategoryRepository->find($categoryId);

        if ($category && $category->getUser() === $user) {
            $income = new Income();
            $income->setAmount($amount);
            $income->setIncomeCategory($category);
            $income->setUser($user);
            $income->setCreatedAt(new \DateTimeImmutable());

            $entityManager->persist($income);
            $entityManager->flush();

            flash()->success('Новий дохід успішно додано');
            return $this->redirectToRoute('app_home');
        }
    }

    return $this->render('income/add_income.html.twig', [
        'categories' => $categories
    ]);
}

```

## ДОДАТОК Б

### Фрагменти програмного коду. Підрахунки власних фінансів

```
<?php

namespace App\Repository;

use App\Entity\Income;
yesterday

created money functions
use App\Entity\IncomeCategory;
use App\Entity\User;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @extends ServiceEntityRepository<Income>
 */
class IncomeRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Income::class);
    }

    public function getTotalIncomeByUser(User $user): float
    {
        return $this->createQueryBuilder('i')
            ->select('SUM(i.amount)')
            ->andWhere('i.user = :user')
```

```

        ->setParameter('user', $user)
        ->getQuery()
        ->getSingleScalarResult() ?: 0;
    }

```

```

public function getIncomeSumByCategory(User $user): array
{
    $result = $this->createQueryBuilder('i')
        ->select('IDENTITY(i.incomeCategory) AS categoryId, SUM(i.amount) AS total')
        ->andWhere('i.user = :user')
        ->setParameter('user', $user)
        ->groupBy('i.incomeCategory')
        ->getQuery()
        ->getResult();

    return array_column($result, 'total', 'categoryId');
}

```

```

public function getIncomeSum(User $user, IncomeCategory $incomeCategory): float
{
    return $this->createQueryBuilder('i')
        ->select('SUM(i.amount)')
        ->where('i.incomeCategory = :category')
        ->andWhere('i.user = :user')
        ->setParameter('category', $incomeCategory)
        ->setParameter('user', $user)
        ->getQuery()
        ->getSingleScalarResult() ?: 0;
}

```

```

public function getIncomeSumByDateRange(User $user, \DateTimeImmutable $startDate,
\DateTimeImmutable $endDate): float
{
    $result = $this->createQueryBuilder('i')
        ->select('SUM(i.amount) AS total')
        ->andWhere('i.user = :user')
        ->andWhere('i.createdAt BETWEEN :start AND :end')
        ->setParameter('user', $user)
        ->setParameter('start', $startDate)
        ->setParameter('end', $endDate)
        ->getQuery()
        ->getSingleScalarResult();

    return (float) $result;
}

```

```
public function getIncomeTransactionCountByDateRange(User $user, \DateTimeImmutable
$startDate, \DateTimeImmutable $endDate): int
{
    $result = $this->createQueryBuilder('i')
        ->select('COUNT(i.id) AS total')
        ->andWhere('i.user = :user')
        ->andWhere('i.createdAt BETWEEN :start AND :end')
        ->setParameter('user', $user)
        ->setParameter('start', $startDate)
        ->setParameter('end', $endDate)
        ->getQuery()
        ->getSingleScalarResult();

    return (int) $result;
}
```

**Додаток Б**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І  
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ  
Факультет інформаційних технологій Декларація про академічну  
добročесність та використання ШІ**

Я, Антіпов Євгеній Олександрович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Інформаційна система контролю особистих фінансових ресурсів» 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:  
інструменти ШІ Gemini були використані для:

- + перекладу іншомовних джерел;
- + стилістичного редагування та перевірки граматики;
- + допомоги у написанні/відлагодженні програмного коду;
- + інше: пошуку додаткових інструментів у розробці програмного продукту.

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

«22» Травня 2026 р. \_\_\_\_\_

**Євгеній АНТІПОВ**