МІНІСТЕРСТВО НАУКИ ТА ОСВІТИ УКРАЇНИ

Київський авіаційний інститут

Факультет комп’ютерних наук та технологій

Кафедра прикладної математики

Лабораторна робота №4

Тема: «Функціональне моделювання бізнес-процесів (UML)»

З дисципліни «Проєктування Інформаційних систем»

Виконали студенти групи

Б-122-23-1-ШІ:

Щербинський Дмитро

Ніколенко Дмитро

Абд Ель Вахаб Юсіф

Прийняв:

Бандурін Владислав

Київ-2025

**Зміст**

1. Мета лабораторної роботи…………………..……………………………..
2. Постановка задачі………………………………………………….….........
3. Теоритичні відомості………………………………………………………
4. Практична частина………………………………………………...……….
5. Use Case Diagram
6. Activity Diagram
7. Class Diagram
8. Sequence Diagram
9. Аналіз побудованих UML-діаграм…………………….....……………..
10. Висновки…………………………………………………………………..

**1. Мета лабораторної роботи**

Опанувати навички побудови UML-діаграм (діаграми варіантів використання, діаграми активностей, діаграми класів та діаграми послідовності) для функціонального моделювання бізнес-процесів в інформаційних системах.

**2. Теоретичні відомості**

Unified Modeling Language (UML) — мова моделювання, яка застосовується для візуалізації, опису, побудови і документування артефактів програмних систем.

Для функціонального моделювання бізнес-процесів використовують наступні типи UML-діаграм:

* **Діаграма варіантів використання (Use Case Diagram)** — відображає функціональні вимоги до системи, акторів та їх взаємодію із системою.
* **Діаграма активностей ([Activity Diagram](https://www.lucidchart.com/pages/uml-activity-diagram?usecase=uml))** — описує послідовність дій, що відбуваються в межах одного або кількох процесів.
* **Діаграма класів (Class Diagram)** — описує структуру системи через класи, їх атрибути, методи та взаємозв’язки.

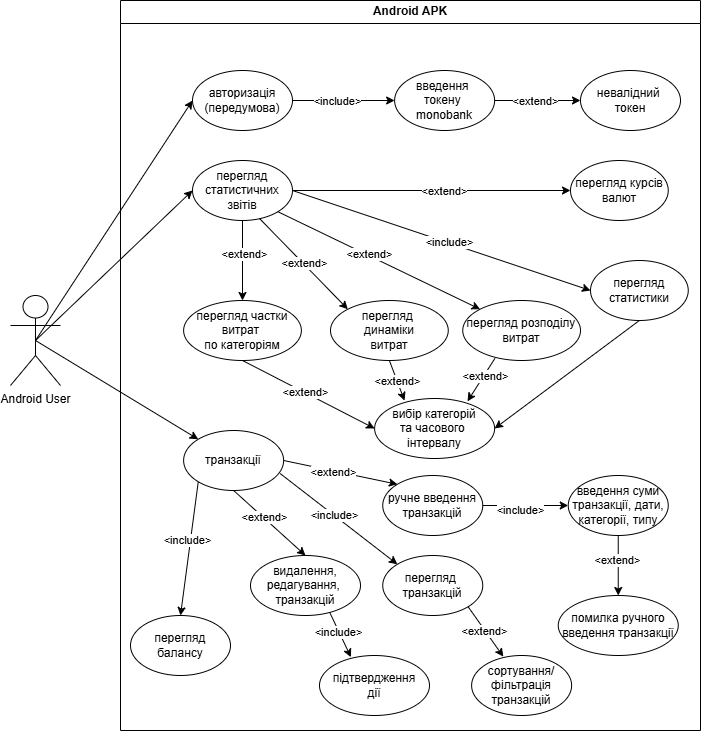
**Діаграма послідовності (Sequence Diagram)** — описує взаємодію між об’єктами у часі для реалізації певного функціоналу.

**3. Постановка задачі**

1. **Вибір бізнес-процесу для моделювання**
   * Виберіть конкретний бізнес-процес, який буде змодельовано за допомогою UML.
   * Складіть текстовий опис цього бізнес-процесу.
2. **Побудова діаграми варіантів використання (Use Case Diagram)**
   * Визначте акторів та функціональні вимоги до обраного процесу.
   * Побудуйте діаграму варіантів використання, зобразивши всі визначені взаємодії.
3. **Побудова діаграми активностей (Activity Diagram)**
   * Виділіть ключові дії та послідовності в обраному бізнес-процесі.
   * Побудуйте детальну діаграму активностей, вказавши всі необхідні переходи та рішення.
4. **Побудова діаграми класів (Class Diagram)**
   * Визначте основні класи, їх атрибути та методи для обраного бізнес-процесу.
   * Побудуйте діаграму класів, включаючи зв'язки між класами.
5. **Побудова діаграми послідовності (Sequence Diagram)**
   * Оберіть один із варіантів використання та змоделюйте його детально за допомогою діаграми послідовності.
6. **Опис і аналіз побудованих UML-діаграм**
   * Підготуйте докладний опис кожної створеної UML-діаграми.
   * Проведіть аналіз діаграм на логічність, повноту та відповідність вимогам бізнес-процесу.
7. **Виявлення проблемних зон та рекомендації щодо оптимізації**
   * Вкажіть проблемні місця у функціонуванні бізнес-процесу, виявлені при побудові UML-діаграм.
   * Сформулюйте рекомендації щодо можливих покращень і оптимізації процесу.

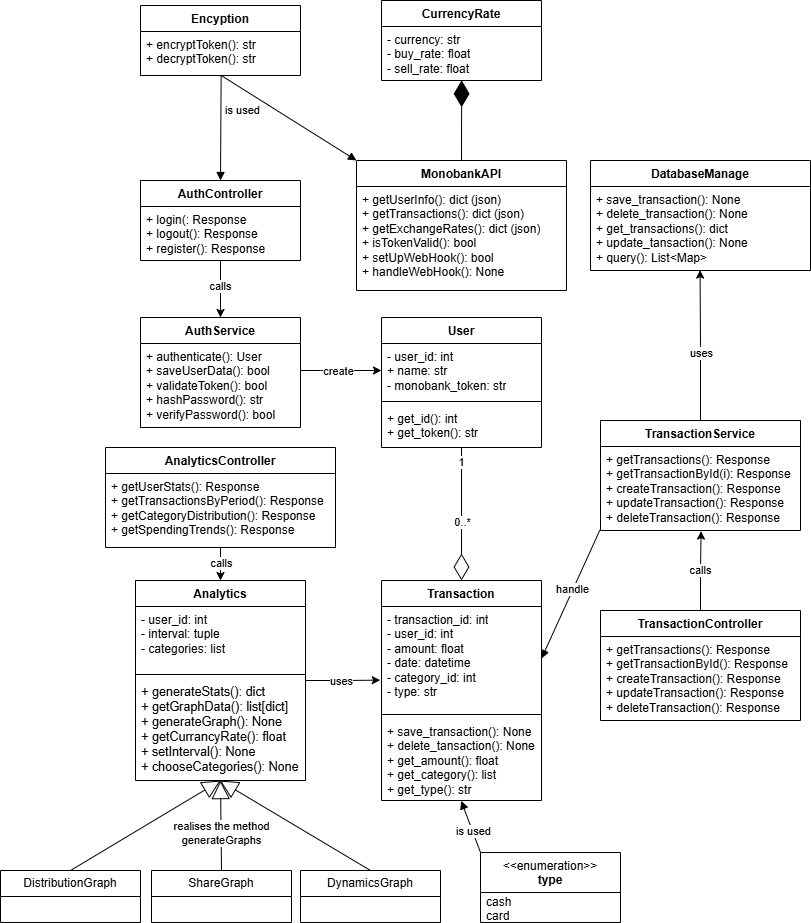
**4. Практична частина**

**Use Cases**

****

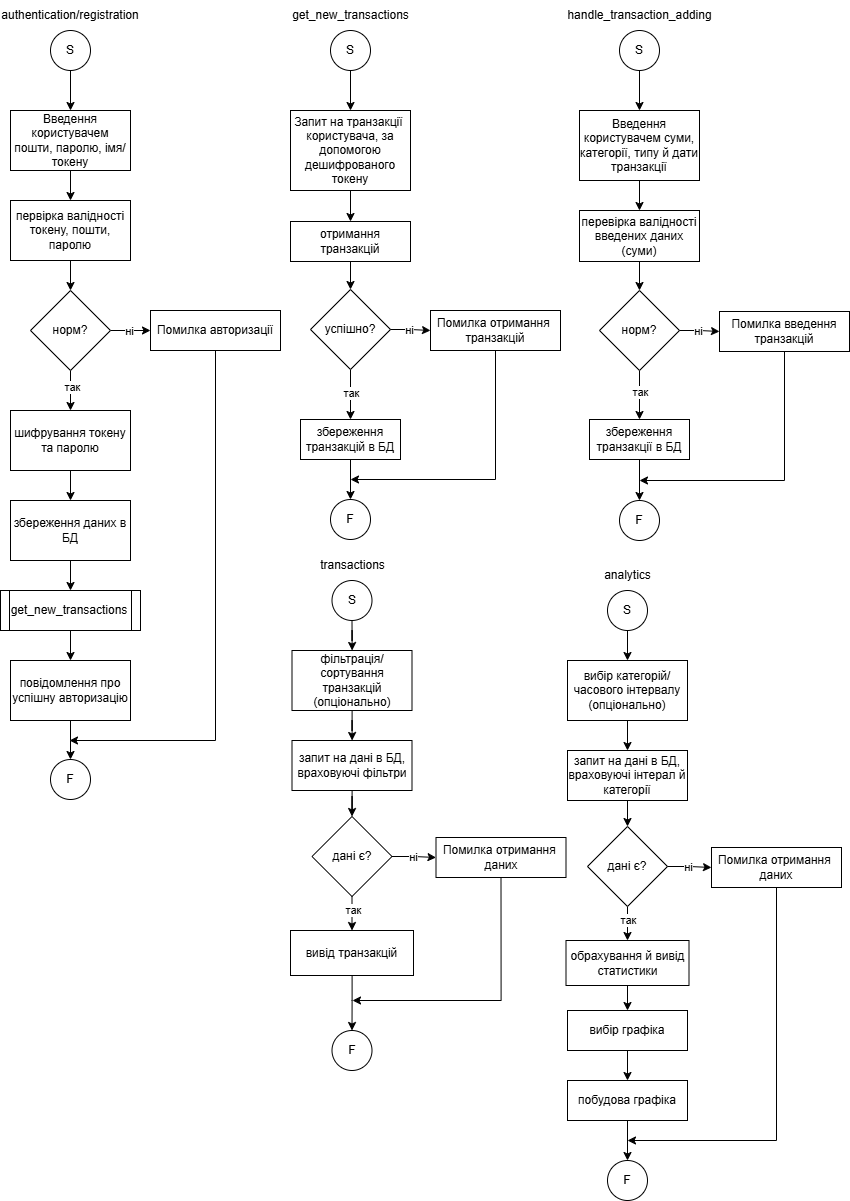
Діаграма варіантів використання розкриває взаємодію користувача з системою. Android-користувач має широкий спектр можливостей: від базової авторизації до складних операцій з транзакціями. Система дозволяє переглядати статистику витрат, курси валют, здійснювати ручне введення транзакцій та керувати фінансовими даними.

**Class**

****

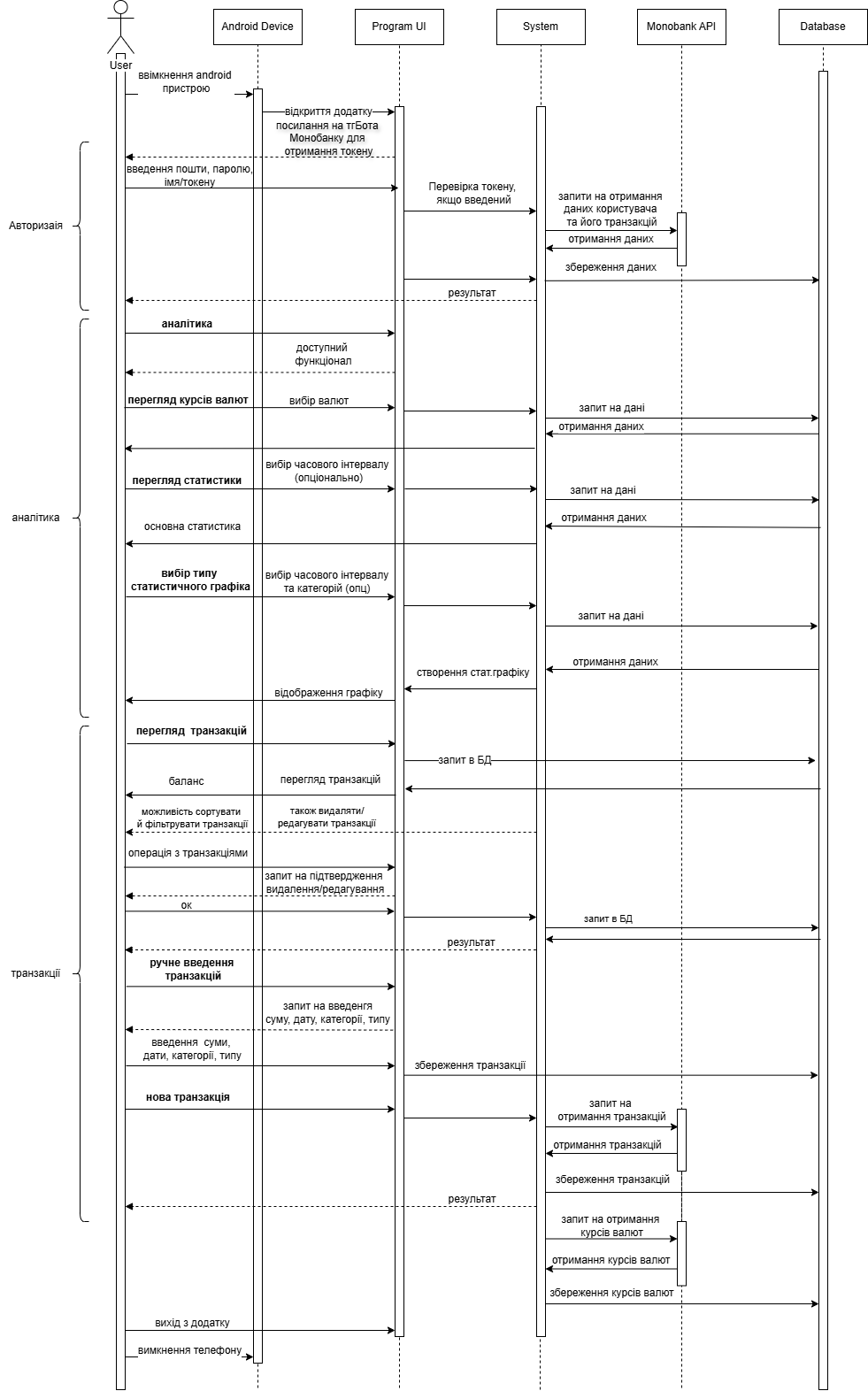
Діаграма класів демонструє складну внутрішню структуру системи. Ключовими компонентами є контролери автентифікації, сервіси транзакцій, модулі аналітики та управління базою даних. Особливістю архітектури є чітке розмежування відповідальності між класами: AuthController відповідає за авторизацію, MonobankAPI здійснює комунікацію з банківською системою, TransactionService керує операціями з транзакціями.

**Activity**

****

Діаграма активностей описує внутрішні процеси додатку. Вона розкриває послідовність дій під час авторизації, отримання транзакцій, генерації аналітики та обробки помилок. Кожен процес супроводжується перевірками та альтернативними сценаріями, що забезпечує стійкість роботи застосунку.

**Sequence**

****

Діаграма послідовності ілюструє взаємодію між компонентами системи. Вона показує повний цикл роботи - від моменту авторизації користувача до генерації фінансової статистики. Простежується складний механізм комунікації між Android-пристроєм, системними компонентами та зовнішнім API.

**5. Аналіз побудованих UML-діаграм**

**Діаграма варіантів використання (Use Case Diagram):**  
Діаграма варіантів використання описує взаємодію Android User із системою через основні сценарії: авторизація, перегляд статистики, обробка транзакцій, вибір категорій витрат та перегляд графіків. Зовнішня сутність Monobank API відображає інтеграцію із зовнішніми сервісами. Використання зв’язків <<extend>> (наприклад, "перевірка токену" для авторизації) та <<include>> (наприклад, "введення суми транзакції" для обробки транзакцій) чітко визначає залежності між сценаріями. Проте діаграма не враховує сценарії обробки помилок, таких як збої під час авторизації чи обробки транзакцій, що може негативно вплинути на користувацький досвід.

**Діаграма класів (Class Diagram):**  
Діаграма класів детально описує статичну структуру системи. Клас User пов’язаний із класом Transaction, що відображає зв’язок між користувачем і його транзакціями. Клас MonobankAPI забезпечує методи для отримання даних про користувача та транзакції, а DatabaseManager відповідає за збереження та оновлення даних у базі. Клас Analytics генерує статистику та графіки, що відповідає аналітичному модулю. Проте діаграма не включає класи для обробки помилок або кешування даних, що може ускладнити масштабування. Також взаємодія між TransactionController та TransactionService виглядає надмірно складною через дублювання методів, таких як getTransactions().

**Діаграма послідовності (Sequence Diagram):**  
Діаграма послідовності описує динамічну поведінку системи під час авторизації, обробки транзакцій та аналітики. Вона показує, як Android User взаємодіє з інтерфейсом, який звертається до Monobank API, зберігає дані в базі та повертає статистику. Обробка помилок частково врахована (наприклад, перевірка токену), але немає механізму повторних спроб у разі збою API. Використання локальної бази даних для кешування є позитивним аспектом, однак численні запити до бази під час аналітики можуть створювати навантаження.

**Діаграма активності (Activity Diagram):**  
Діаграма активності детально описує робочі процеси системи, поділені на чотири основні потоки: авторизація/реєстрація, отримання нових транзакцій, обробка транзакцій та аналітика.

* **Авторизація/реєстрація:** Процес починається з введення користувачем логіну, пароля або токену. Система перевіряє, чи є токен дійсним. Якщо ні, відбувається помилка авторизації; якщо так, токен шифрується та зберігається в базі даних.
* **Отримання нових транзакцій:** Система запускає запит на транзакції до Monobank API, отримує їх, фільтрує та сортує (за необхідності), після чого зберігає в базі даних. У разі помилки API система повертає помилку доступу.
* **Обробка транзакцій:** Користувач вводить суму, дату, категорію та тип транзакції. Система перевіряє, чи є транзакція нормальною (наприклад, чи відповідає сума введеним даним). Якщо ні, видається помилка; якщо так, транзакція зберігається в базі.
* **Аналітика:** Система отримує дані з бази, враховує категорії та інтервали, генерує статистику та повертає її користувачу. У разі помилки (наприклад, якщо даних немає), система видає повідомлення про помилку.  
  Діаграма активності чітко показує розгалуження (наприклад, перевірка токену, нормальність транзакції), але не враховує повторні спроби у разі збоїв API або кешування даних для аналітики. Також немає чіткого механізму для асинхронної обробки запитів, що може призводити до затримок.

Виявлені проблеми стосуються обробки помилок, продуктивності, безпеки. Можна запропонувати такі рішення, як додавання кешування, покращення обробки помилок, забезпечення асинхронності та шифрування даних, дозволять зробити систему більш ефективною, безпечною та зручною для користувача. Ці зміни також полегшать подальше масштабування системи.

**6. Висновки**

Розроблені UML-діаграми демонструють комплексний підхід до проєктування мобільного фінансового додатку. Система має чітку архітектуру, розподілену на відповідальні компоненти, та забезпечує широкий функціонал для користувача.

Під час роботи над проєктом ми набули практичних навичок моделювання складних інформаційних систем, вивчили особливості проєктування мобільних додатків з інтеграцією зовнішніх сервісів. Діаграми показали важливість детального опрацювання внутрішніх процесів, механізмів обробки даних та комунікації між компонентами.

Подальший розвиток системи вбачається у вдосконаленні механізмів безпеки, оптимізації продуктивності та розширенні аналітичного функціоналу.