1.1. Загальний опис системи

Проєкт являє собою веб-систему для управління студентськими проєктами. Вона дозволяє студентам створювати команди, керувати завданнями, призначати ролі, а також отримувати оцінку від викладача.
Система передбачає наявність користувачів із різними правами доступу (лідер, розробник, тестувальник, викладач) і має забезпечити зручну взаємодію між ними.

1.2. Функціональні вимоги

1. Система користувачів

- Реєстрація нових користувачів.
- Можливість редагувати власний профіль.
- Можливість адміністратору видаляти користувачів або змінювати їхні ролі.

2. Система проектів

- Створення, редагування та видалення проєктів.
- Призначення учасників у команду.
- Можливість переглядати всі активні та завершені проєкти.

3. Система ролей у команді

- Призначення ролей (лідер, розробник, тестувальник).
- Зміна ролей лідером команди або адміністратором.
- Перевірка прав доступу залежно від ролі користувача.

4. Система завдань і прогресу

- Додавання та редагування завдань у межах конкретного проєкту.
- Встановлення статусу виконання (нове, у процесі, виконано).
- Автоматичне підрахування прогресу виконання проєкту.

5. Система оцінювання

- Викладач може виставляти оцінки за проєкт.
- Система зберігає історію оцінок і коментарі викладача.
- Студенти можуть переглядати свої оцінки та відгуки.

1.3. Нефункціональні вимоги

1. Інтерфейс має бути інтуїтивним, простим у використанні й зрозумілим для студентів та викладачів.

2. Система повинна мати модульну структуру, щоб її можна було легко розширювати новими функціями.

3. Дані користувачів і проєктів зберігаються у захищеній базі даних, із можливістю резервного копіювання.

1.4. Обґрунтування вибору моделі життєвого циклу

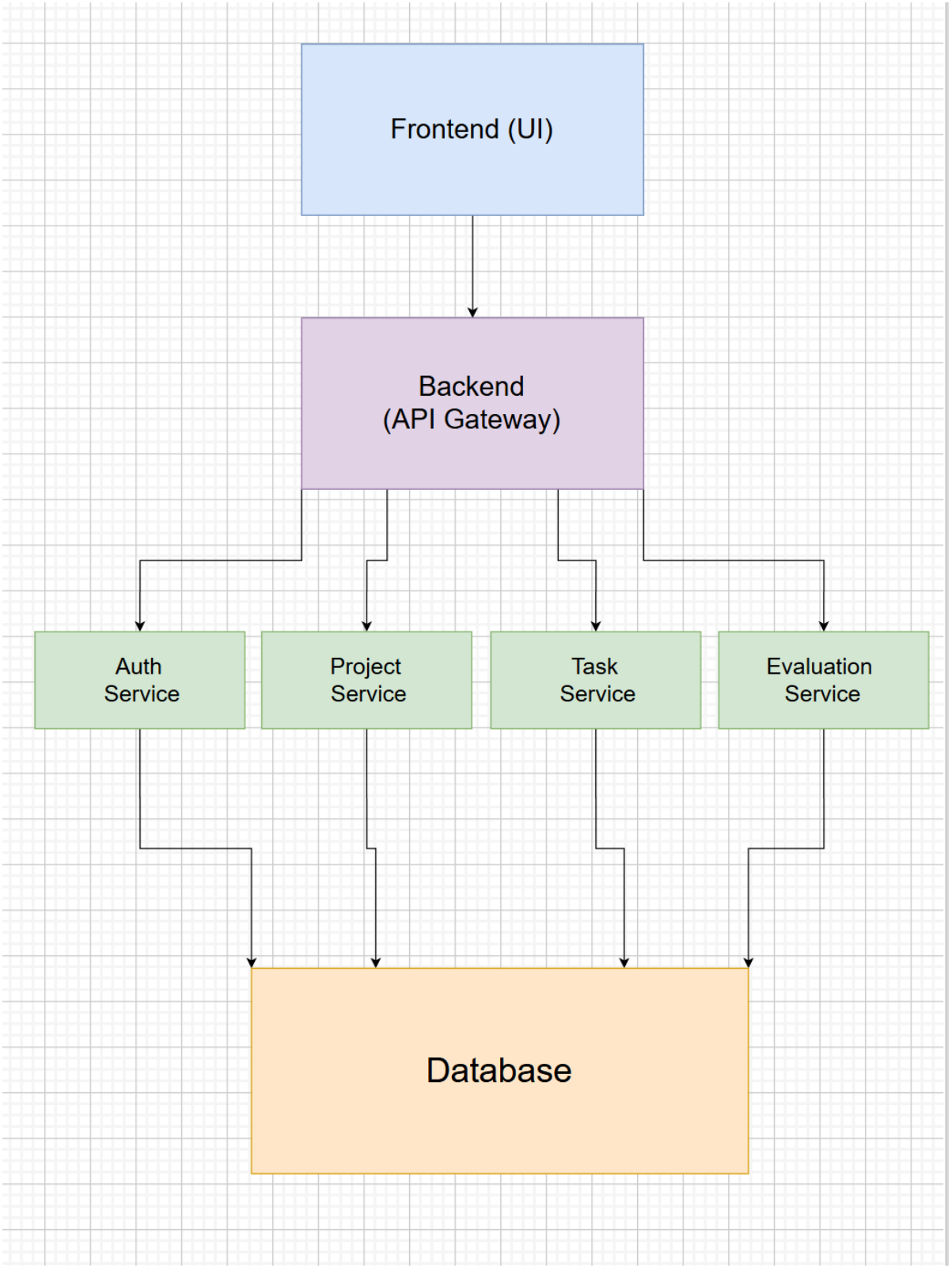
Для розробки системи управління студентськими проектами обрано ітераційну модель життєвого циклу. Ця модель дозволяє поступово розробляти систему частинами (ітераціями), тестувати кожен модуль окремо та вдосконалювати його на основі відгуків користувачів.

Оскільки система має кілька незалежних компонентів (модулі користувачів, проєктів, ролей, оцінювання), ітераційний підхід дозволяє гнучко додавати нові функції без необхідності повного переписування коду. Також це зручно для командної роботи - різні учасники команди можуть паралельно працювати над своїми частинами системи.

Такий підхід спрощує тестування, покращує масштабованість і зменшує ризики появи критичних помилок на пізніх етапах розробки.

2. Проєктування архітектури

Система базується на мікросервісній архітектурі, що дозволяє розділити логіку програми на окремі незалежні сервіси.
Це дає можливість кільком розробникам працювати одночасно над різними частинами проєкту, а також полегшує масштабування й оновлення системи в майбутньому.



2.1. Основні мікросервіси

1. **Auth Service (сервіс авторизації)**

    - Реєстрація, вхід і керування користувачами.
    - Видача токенів доступу (JWT).
    - Керування ролями користувачів.

2. **Project Service (сервіс проєктів)**

    - Створення, редагування, видалення проєктів.
    - Призначення учасників і збереження інформації про команду.

3. **Task Service (сервіс завдань)**

    - Керування завданнями в межах конкретного проєкту.
    - Зміна статусів завдань, обчислення прогресу.

4. **Evaluation Service (сервіс оцінювання)**

    - Призначення оцінок і коментарів викладачами.
    - Перегляд історії оцінок студентами.

5. **Gateway / API Gateway**

    - Центральна точка входу до системи.
    - Приймає запити від користувача та перенаправляє їх до потрібного мікросервісу.

6. **Frontend Service**

    - Інтерфейс користувача (HTML, CSS, JavaScript).
      Зв’язок із Gateway через REST API.

7. **Database Layer**

    - Кожен мікросервіс має власну таблицю в базі (наприклад, PostgreSQL або MySQL).
    - Дані розділені за контекстами (користувачі, проєкти, завдання тощо).

2.2. Взаємодія між компонентами

1. Користувач працює з інтерфейсом (Frontend), який надсилає запити до API Gateway.

2. Gateway визначає, якому сервісу передати запит (наприклад, створення завдання - до Task Service).

3. Обраний сервіс обробляє дані, звертається до своєї бази та повертає результат назад через Gateway.

4. Frontend отримує відповідь і оновлює інтерфейс користувача.

2.3. Переваги мікросервісної архітектури

- Можна розробляти окремі частини проєкту незалежно.

- Легке масштабування - при збільшенні навантаження можна розширити конкретний сервіс.

- Простота оновлень - зміни в одному модулі не впливають на роботу інших.

- Зручність командної розробки: кожен розробник відповідає за свій сервіс.

2.4. Технологічний стек (приклад)

- **Frontend:** React або Vue.js (JavaScript, HTML, CSS)

- **Backend:** Node.js / Python (FastAPI, Flask або Express)

- **Database:** PostgreSQL / MySQL

- **API комунікація:** REST або gRPC

- **Авторизація:** JWT-токени