**1. Проаналізувати предметну область, визначити групи користувачів, випадки використання та компоненти ПЗ.**

*Аналіз предметної області.*

Предметна область охоплює логопедичну терапію для дітей із порушеннями мовлення, а саме:

* Постановку, автоматизацію та контроль правильності вимови звуків.
* Формування звичок правильної артикуляції.
* Мотивацію дитини до регулярних занять через ігрові механіки.
* Взаємодію логопеда і дитини в єдиній системі.

Система спрощує роботу логопеда, робить процес тренувань більш приємним для дитини, а також застосовується модуль штучного інтелекту, який оцінює правильність вимови та допомагає відстежувати прогрес учня.

*Основні групи користувачів.*

* Дитина – виконує артикуляційні вправи, повторює звуки, отримує підказки та візуальний зворотний зв’язок.
* Логопед – створює індивідуальні плани занять, перевіряє результати, додає нові вправи та аналізує прогрес.

*Основні функціональні сценарії (випадки використання).*

* Реєстрація та авторизація користувачів.
* Виконання вправ учнем і надсилання результатів.
* Автоматичний аналіз вимови за допомогою AI-модуля.
* Отримання оцінки й підказок для покращення вимови.
* Створення та редагування вправ логопедом.
* Перегляд статистики занять і прогресу учнів.
* Адміністративне керування системою.

*Основні компоненти програмного забезпечення:* Mobile Client (React Native), User Service, Exercise Service, Speech AI Service, Progress Service, API Gateway, Database.

**2. Визначити архітектурний стиль та спроєктувати рівні взаємодії.**

Для системи обрано мікросервісну клієнт–серверну архітектуру. Вона дозволяє розподілити логіку між незалежними сервісами, що спрощує масштабування, підтримку та подальший розвиток проєкту.

Система складається з таких основних рівнів:

* 1. Клієнтський рівень – мобільний застосунок на React Native, який надає інтерфейс користувача та взаємодіє із сервером через REST API.
  2. Рівень API Gateway – приймає всі запити від клієнта, виконує маршрутизацію, автентифікацію та передає дані у відповідний мікросервіс.
  3. Рівень мікросервісів – набір незалежних сервісів (User, Exercise, Speech AI, Progress), кожен із яких реалізує певну бізнес-функцію системи. Комунікація між ними здійснюється через HTTP.
  4. Рівень даних – кожен мікросервіс має власну базу даних (UserDB, ExerciseDB, ProgressDB), що реалізує принцип Database per Service.  
     Це забезпечує ізоляцію даних, незалежність модулів та спрощує масштабування.

Такий підхід дозволяє створити масштабовану систему з чітким розділенням відповідальностей між компонентами, що полегшує її розробку та супровід.

**3. Описати компоненти ПЗ.**

* 1. Mobile Client (React Native) – реалізує інтерфейс для учня та логопеда, забезпечує реєстрацію, авторизацію, виконання вправ і перегляд результатів. Клієнт взаємодіє з бекендом через REST API та отримує дані у форматі JSON.
  2. User Service – обробляє запити реєстрації, автентифікації та керування обліковими записами. Забезпечує роботу з JWT-токенами та визначення ролей користувачів.
  3. Exercise Service – надає можливість створювати, редагувати та видаляти вправи. Забезпечує доступ логопеда до методичних матеріалів і планів занять.
  4. Speech AI Service – отримує запит від Exercise Service, проводить аналіз вимови, повертає результат у вигляді оцінки точності та рекомендацій. Не взаємодіє з клієнтом або базами даних напряму.
  5. Progress Service – відповідає за збереження результатів аналізу після виконання вправ, отриманих від Exercise Service, і надає логопеду або користувачу статистику та звіти через API Gateway.
  6. API Gateway – приймає всі зовнішні запити, виконує перевірку авторизації, маршрутизує їх до відповідних сервісів (User, Exercise, Speech AI або Progress) та повертає узгоджену відповідь клієнту.
  7. Database – набір окремих баз даних (UserDB, ExerciseDB, ProgressDB), що відповідають за ізольоване зберігання даних у кожному мікросервісі. Speech AI може використовувати зовнішнє сховище або інтеграцію з Azure / Google Speech API.

**4. Побудувати UML-діаграми:**

На діаграмі компонентів (рис. 1) показано основні частини системи та взаємозв’язки між ними. Mobile Client взаємодіє з API Gateway через REST API. Gateway маршрутизує запити до User Service, Exercise Service та Progress Service.  
Exercise Service викликає Speech AI Service для аналізу аудіо й надсилає результати у Progress Service для збереження. Кожен мікросервіс має власну базу даних, що забезпечує ізольованість і стабільність системи.

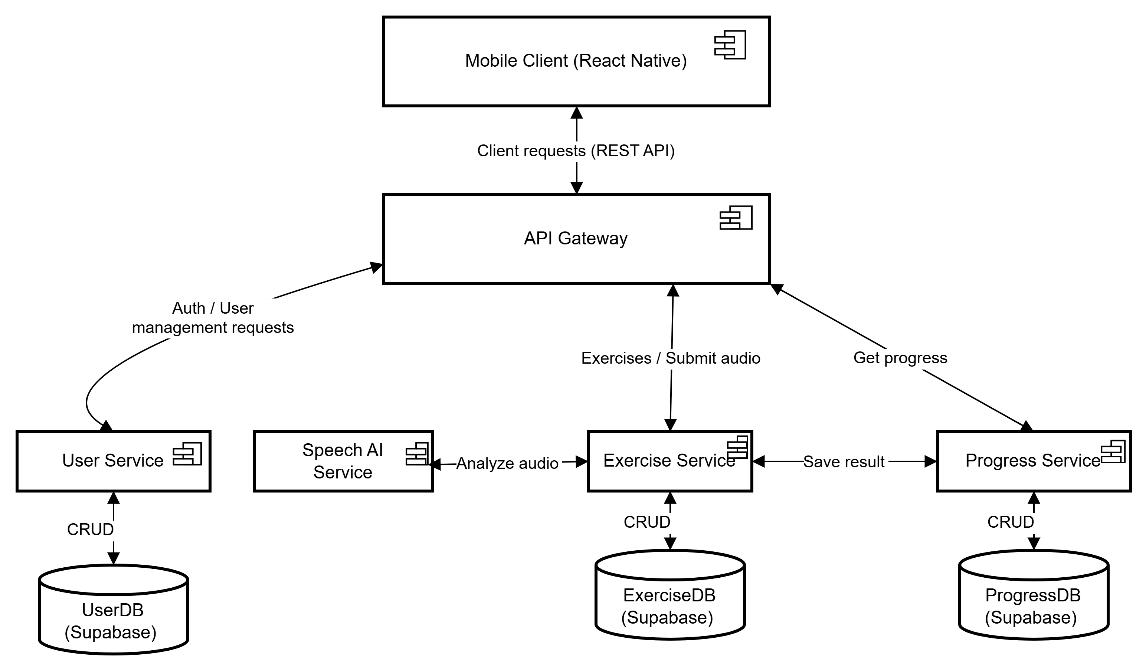


Рис. 1. Діаграма компонентів

Діаграма взаємодії відображає послідовність виконання вправи користувачем. Після авторизації користувач отримує вправу, записує аудіо й надсилає його через API Gateway у Exercise Service. Exercise Service передає аудіо в Speech AI Service для аналізу вимови. Після отримання оцінки правильності Exercise Service зберігає результат у Progress Service та повертає оновлену статистику користувачу. Логопед може переглядати прогрес і формувати звіти через ті самі сервіси, використовуючи API Gateway.

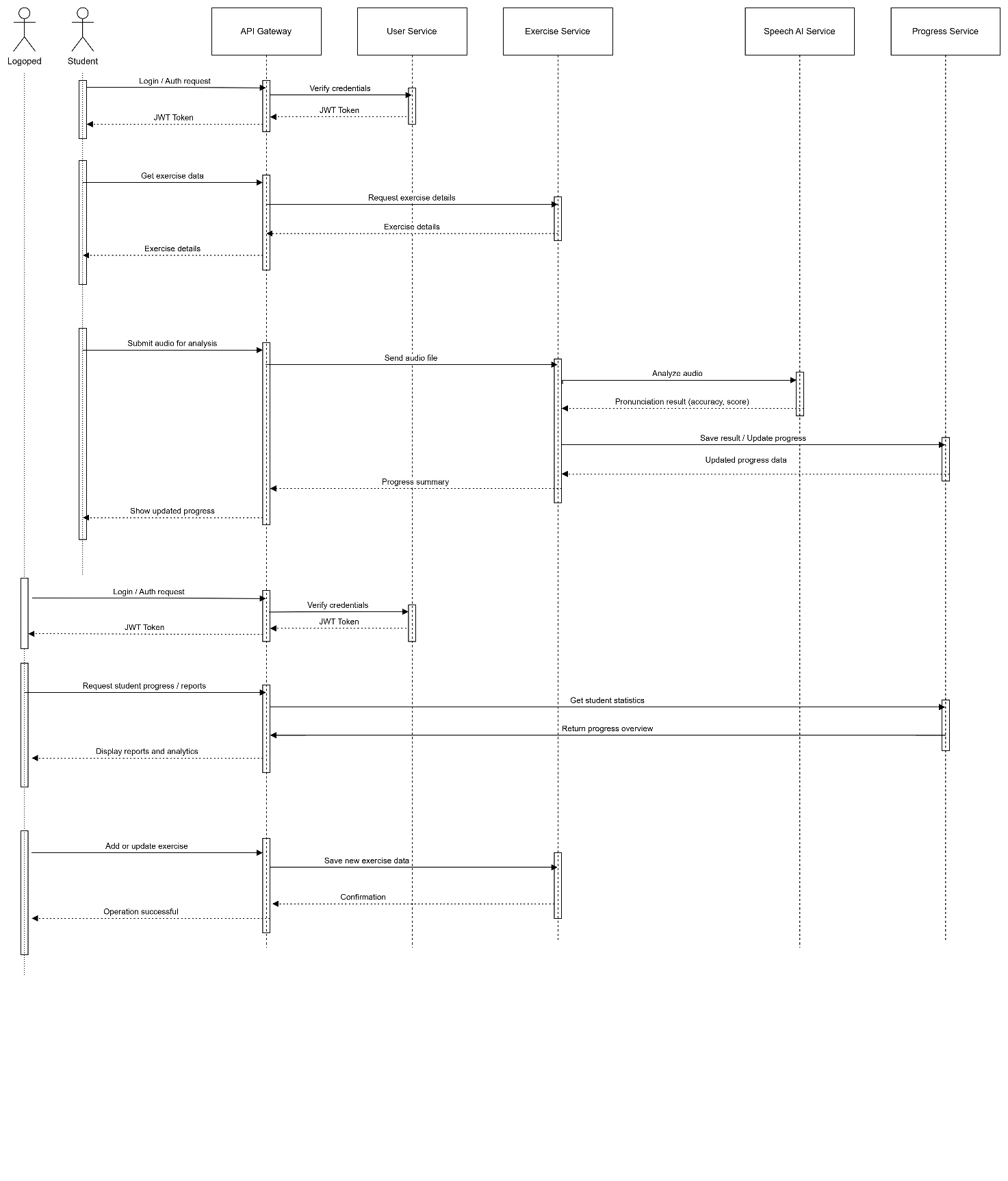


Рис. 2. Діаграма взаємодії

На діаграмі розгортання показано фізичну структуру системи.  
Усі серверні компоненти розгорнуті в середовищі Docker Compose: API Gateway, User Service, Exercise Service та Progress Service — кожен має власну базу даних.  
Speech AI Service представлений як зовнішній API (Azure / Google Speech), який викликається з Exercise Service для аналізу вимови.  
Mobile Client (React Native) взаємодіє з API Gateway через REST API, а Gateway маршрутизує запити до відповідних сервісів.

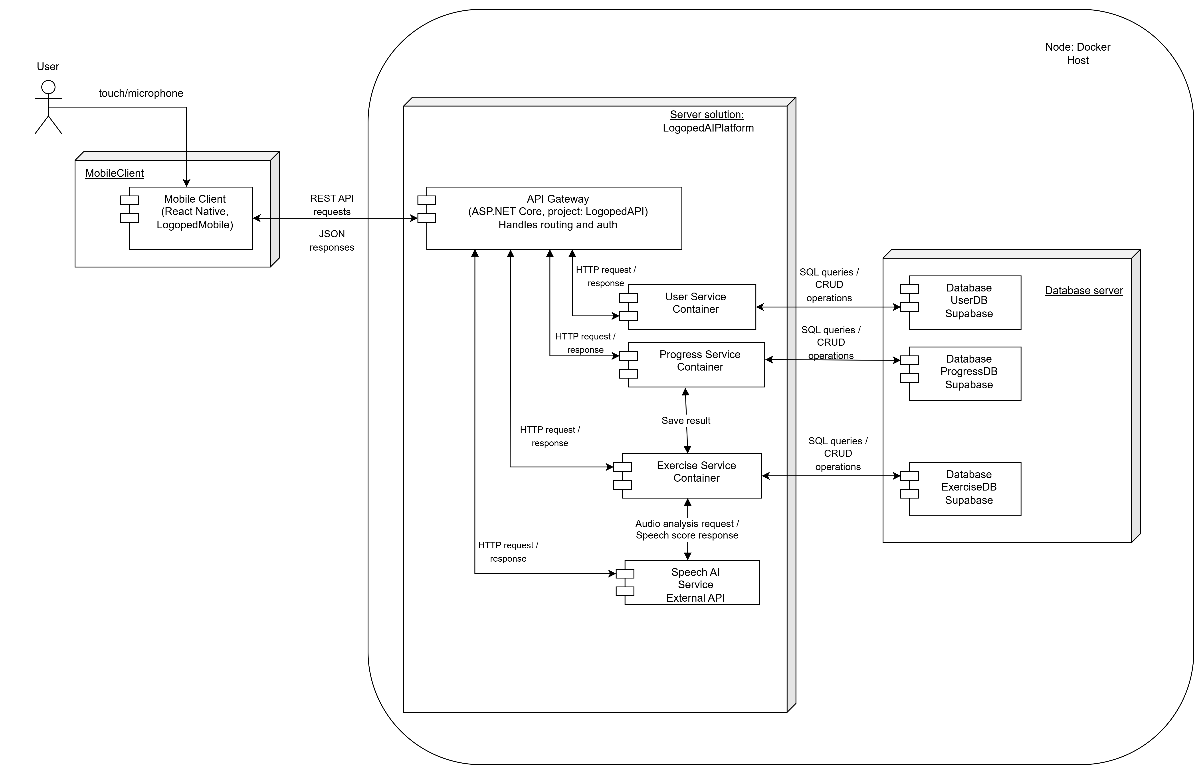


Рис. 3. Діаграма розгортання

**5. Обґрунтувати вибір архітектури й шаблонів проєктування**

Обрана мікросервісна архітектура забезпечує гнучкість, модульність і простоту масштабування. Кожен сервіс виконує одну бізнес-функцію, що полегшує оновлення та подальший розвиток системи.

Основними шаблонами проєктування виступають:

* Repository – для розділення бізнес-логіки від доступу до даних у кожному сервісі.
* Dependency Injection – для зручного керування залежностями у .NET-сервісах.
* API Gateway Pattern – для централізованої маршрутизації запитів і захищеного доступу до внутрішніх сервісів.
* Adapter – для взаємодії з зовнішнім сервісом розпізнавання мовлення.

Обрана структура забезпечує зрозумілу логіку, ізольованість модулів і стабільну роботу системи з можливістю подальшого розширення функціональності.