1. Аналіз вимог

Функціональні вимоги

1. **Управління проектами:** Система повинна дозволяти користувачам створювати нові проекти, редагувати їх опис та видаляти проекти (CRUD операції для сутності "Проект").
2. **Управління ролями:** Система повинна забезпечувати можливість призначення ролей учасникам команди, зокрема: "Лідер", "Розробник", "Тестувальник", із різними правами доступу.
3. **Трекінг завдань:** Система повинна дозволяти створювати завдання, призначати їх конкретним виконавцям та змінювати статус виконання.
4. **Кабінет викладача:** Система повинна мати окремий інтерфейс для викладача з можливістю перегляду всіх проектів та виставлення оцінок за виконану роботу.
5. **Автентифікація та авторизація:** Система повинна дозволяти студентам та викладачам реєструватися та входити в систему, розмежовуюче доступ до функцій залежно від типу користувача.

Нефункціональні вимоги

1. **Продуктивність:** Час відгуку системи на дії користувача (наприклад, відкриття сторінки проекту) не повинен перевищувати 2 секунди при навантаженні до 100 одночасних користувачів.
2. **Безпека даних:** Паролі користувачів повинні зберігатися в базі даних виключно у вигляді хеш-суми, а з'єднання з сервером має відбуватися через захищений протокол HTTPS.
3. **Сумісність (Usability):** Інтерфейс системи повинен коректно відображатися та функціонувати в останніх версіях браузерів як на десктопних, так і на мобільних пристроях.

Обґрунтування моделі життєвого циклу

Agile (Гнучка методологія), зокрема фреймворк Scrum.

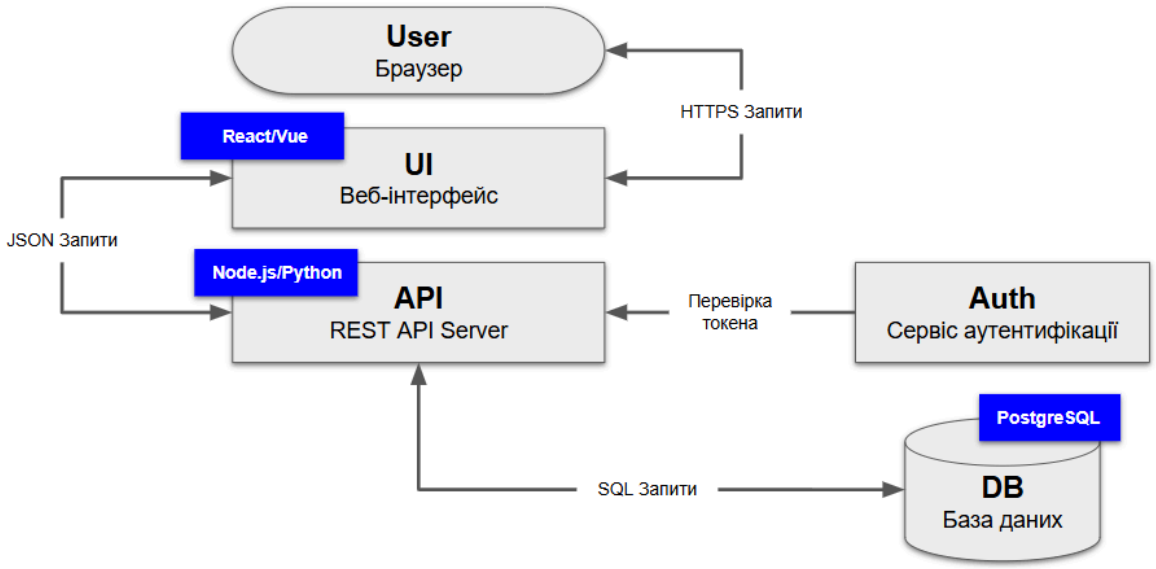
Ітеративність: Студентські проекти зазвичай розробляються поетапно. Scrum дозволяє розбити роботу на спринти (наприклад, по 1-2 тижні), що допомагає команді бачити прогрес.

Зворотний зв'язок: Регулярні демонстрації результатів викладачу (після кожного спринту) дозволяють швидко вносити корективи в вимоги та оцінювати проект поетапно.

Гнучкість ролей: Оскільки студенти вчаться, ролі можуть змінюватися або доповнюватися, що легше підтримувати в Agile.

2. Проєктування архітектури

Схема основних компонентів



Обґрунтування архітектурного підходу

Вибір: Монолітна архітектура (Monolithic Architecture).

Простота розробки: Для студентського проекту мікросервіси є надлишковими. Моноліт дозволяє тримати весь код в одному репозиторії, що спрощує налагодження та запуск.

Швидкість розгортання: Легше розгорнути один сервер, ніж налаштовувати оркестрацію для кількох мікросервісів.

Цілісність даних: В системі управління проектами зв'язки між таблицями є тісними, тому єдина реляційна база даних в моноліті працюватиме ефективніше.