ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку для моніторингу навчального навантаження студентів. Спец-частина: Розробка Backend-частини застосунку»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр БОНДАРЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Олександр БОНДАРЕНКО

Керівник: Ірина ЗАМРІЙ
д.т.н., професор

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Бондаренку Олександру Олександровичу

1. Тема кваліфікаційної роботи: «Розробка застосунку для моніторингу навчального навантаження студентів. Спец-частина: Розробка Backend-частини застосунку»
керівник кваліфікаційної роботи д.т.н., професор Ірина ЗАМРІЙ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості щодо навчального навантаження, дослідження існуючих методів його моніторингу та їх обмежень, технічний огляд інструментів розробки з супровідною документацією.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз основних методів моніторингу навчального навантаження студентів.

2. Проєктування архітектури застосунку для моніторингу навчального навантаження студентів
3. Програмна реалізація та опис функціонування застосунку для моніторингу навчального навантаження студентів.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма розгортання.
6. Екранні форми.
7. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Огляд існуючих засобів для оцінювання та моніторингу навчального навантаження студентів	14.03-19.03.2025	
4	Проєктування застосунків для керування оцінюваннями навчального навантаження та для проведення оцінювань.	20.03.-01.04.2025	
5	Програмна реалізація застосунку	02.04.-28.04.2025	
6	Оформлення роботи: вступ, висновки, реферат	29.04-09.05.2025	
7	Розробка демонстраційних матеріалів	12.05-18.05.2025	
8	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Олександр БОНДАРЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 65 стор., 1 табл., 28 рис., 21 джерел.

Мета роботи – спрощення моніторингу навчального навантаження студентів на основі використання програмного забезпечення для вирішення проблеми збору даних про навантаженість студентів.

Об'єкт дослідження – процес збору даних та моніторингу навчального навантаження студентів у закладах вищої освіти.

Предмет дослідження – програмне забезпечення для оцінювання та моніторингу навчального навантаження студентів.

Короткий зміст роботи: В роботі проаналізовано методи моніторингу навчального навантаження. Проаналізовано інструментальні засоби збору даних про навчальне навантаження: Google Forms, SurveyMonkey, Moodle та Canvas. Розроблено методику проведення моніторингу навчального навантаження, програмно реалізовані ключові функціональні можливості, зокрема: керування оцінюваннями навчального навантаження, проведення оцінювання навчального навантаження та імпортування навчальної програми дисциплін з документів силабусів. Проведено тестове розгортання додатка. В роботі використано фреймворк ASP.NET Core для розробки API, Python для інтеграції великої мовної моделі Gemini та .NET Aspire для оркестрування мікросервісної архітектури.

Сферою використання застосунку є організація процесу моніторингу навчального навантаження в закладах вищої освіти

КЛЮЧОВІ СЛОВА: НАВЧАЛЬНЕ НАВАНТАЖЕННЯ, ОБРОБКА ДОКУМЕНТІВ, C#, ASP.NET, PYTHON, FLASK API, AUTH0, GEMINI, .NET ASPIRE,

ЗМІСТ

ВСТУП

У сучасних умовах цифровізації освіти більшої актуальності набуває питання моніторингу навчального навантаження студентів. Застосування змішаної та дистанційної форм навчання створюють додаткове навантаження для учасників освітнього процесу та ускладнюють процеси, пов’язані з моніторингом якості освіти. У зв’язку з цим зростає потреба у створенні програмних рішень, які б дозволили автоматизувати процес моніторингу навчального навантаження.

Моніторинг навантаження студентів є важливою складовою ефективного освітнього процесу. Він дозволяє контролювати обсяг і структуру навчальної діяльності та забезпечувати рівномірне розподілення завдань, що призводить до кращого засвоєння матеріалу.

Наявні методи, що застосовується для вимірювання навчального навантаження студентів здебільшого орієнтовані на загальний облік відвідуваності та кількості виконаних робіт, проте відсутній інструмент, що дозволяє оцінювати справжнє навантаження студентів.

Об’єктом дослідження є процес збору даних та моніторингу навчального навантаження студентів у закладах вищої освіти.

Предметом дослідження є програмне забезпечення для оцінювання та моніторингу навчального навантаження студентів.

Метою роботи є спрощення моніторингу навчального навантаження студентів на основі використання програмного забезпечення для вирішення проблеми збору даних про навантаженість студентів.

Наукова новизна роботи визначається в аналізі методик проведення моніторингу навчального навантаження студентів та огляді наявний програмних рішень для оцінювання навчального навантаження студентів.

Також робота досліджує метод використання штучного інтелекту, для обробки даних з цифрових документів.

Практична значущість результатів полягає в розробленій системі для моніторингу навчального навантаження студентів, що може бути використана у закладах вищої освіти для покращення навчального процесу. Окрім цього, робота досліджує метод використання великих мовних моделей, а саме Gemini 2.0 flash та Gemma 3 для автоматичної обробки документів, що може бути використаний для автоматизації у подібних сценаріях.

1. ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Аналіз задачі моніторингу навчального навантаження студентів

Процес моніторингу навчального навантаження студентів полягає в систематичному зборі, аналізі та оцінці інформації про обсяг і структуру навчальної діяльності студентів. Моніторинг є важливим інструментом для своєчасного виявлення перенавантаження або недостатнього навантаження студентів, що може негативно впливати на якість засвоєння матеріалу й мотивацію до навчання.

Процес моніторингу ґрунтується на зборі даних про навчальне навантаження студентів. Оцінювання навчального навантаження студентів – це процес визначення кількісних і якісних показників обсягу навчальної діяльності, яку виконує студент у межах освітньої програми.

Основними одиницями вимірювання навчального навантаження є одиниці часу в академічних годинах або в кредитах, як у кредитно-модульній системі ECTS.

Академічні години – це традиційна для України система обліку навчального навантаження, що базується на кількості навчальних занять (аудиторного часу). Зазвичай одна академічна година становить 45 хвилин. Навантаження оцінюється на основі кількості годин лекцій, практичних, лабораторних занять без урахування повного обсягу самостійної роботи студента.

Кредитно-модульна система ECTS (European Credit Transfer and Accumulation System) – це європейський стандарт вимірювання навчального навантаження, який орієнтується не лише на кількість занять, але й на загальний обсяг зусиль студента. Один кредит ECTS зазвичай відповідає 25–30 годинам загальної праці студента (включаючи лекції, самостійну роботу,

підготовку до іспитів тощо). Система створена для уніфікації та порівняння освітніх програм між країнами та університетами. Використання кредитно-модульної системи має ряд переваг над системою, що традиційно використовувалася в Україні [1]

Формальні одиниці вимірювання, такі як ECTS і академічні години, не завжди точно відображають реальне навчальне навантаження студентів. У різних дисциплінах витрати часу можуть суттєво відрізнятися від нормативів, тому важливо збирати фактичні дані про зусилля студентів. Це завдання вирішується за допомогою комплексних методів, які враховують як кількісні, так і якісні показники. Тому вибір методики збору інформації про навчальне навантаження, залежить від організації навчального процесу в конкретному закладі. Загалом можна виділити наступні методи збору інформації про навчальне навантаження студентів.

Опитування або анкетування є найпоширенішим методом збору інформації про навчальне навантаження студентів. Під час анонімного опитування студенти повідомляють, скільки часу витрачають на опанування окремих дисциплін, які форми навчальної діяльності потребують найбільше зусиль (наприклад, лекції, самостійна підготовка, робота над завданнями, підготовка до іспитів), чи відчувають вони перенавантаження, або труднощі у процесі навчання. Хоча цей метод надає суб’єктивні дані, він є надзвичайно цінним для розуміння студентської перспективи. Цей метод дає суб’єктивну, але дуже важливу інформацію з перспективи студента, та дозволяє оцінювати безпосередньо відповідність навчального навантаження навчальному плану [2].

Аналіз активності у цифрових платформах. У разі використання систем управління навчанням (LMS), таких як Moodle, Google Classroom, Canvas, адміністрація може збирати дані про: час перебування студента на платформі; кількість і частоту виконання завдань; активність у форумах, тестах, відеолекціях тощо. Такі дані допомагають оцінити рівень залучення студента, хоча й не враховують усю позаплатформену діяльність.

Моніторинг успішності та термінів виконання та порівняння з нормативами. Показниками навчального навантаження в даному методі можуть виступати: якість і своєчасність виконаних робіт; частота звернень студентів за допомогою; рівень успішності на контрольних заходах, та порівняння аналогічних показників інших груп, курсів або закладів. Недолік такого підходу в тому, що результати навчання залежать від багатьох інших факторів, а нормативи є формальними та не враховують індивідуальних відмінностей студентів, якості викладання матеріалів, інших факторів, упущення яких може призводити до неточних висновків.

1.2. Аналіз існуючих рішень для оцінки та моніторингу навчального навантаження студентів

Як було зазначено раніше, одним із найпоширеніших способів моніторингу навчального навантаження студентів є проведення опитувань через онлайн-платформи, такі як Google Forms або SurveyMonkey.

Крім того, джерелом даних про рівень навантаження студентів може слугувати аналіз активності студентів у цифрових освітніх системах, таких як Moodle та Canvas.

1.2.1. Аналіз платформи для проведення опитувань Google Forms

Google Forms – безплатний вебінструмент від компанії Google для створення різноманітних форм збору даних [3], включаючи анкети, тести та опитування, наприклад як на рисунку 1.1. Сервіс пропонує широкий спектр функціональних можливостей:

- конструювання адаптованих анкет, опитувальників та тестів з використанням варіативних типів запитань для якісного та кількісного збору даних;

- автоматична інтеграція з Google Sheets для подальшого аналізу масивів даних;
- зберігання форм у хмарному середовищі, з можливістю спільної роботи над проєктом та доступність з будь-якого пристрою за наявності підключення до мережі Інтернет.

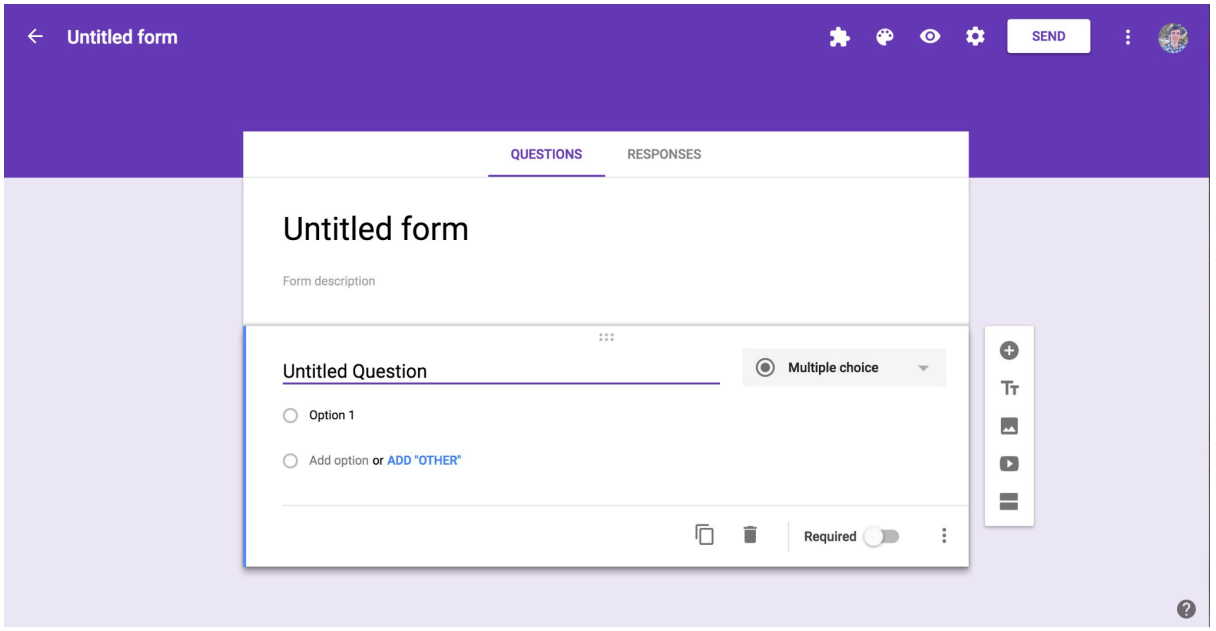


Рис. 1.1. Редагування форм в Google Forms

Різноманітні типи запитань Google Forms дозволяють викладачам збирати відгуки студентів стосовно кількості завдань, часу, витраченого на навчання, та інших аспектів. Можливість синхронізації з Google Таблицями надає можливість автоматизувати аналіз зібраних даних опитування.

Проте, функціонал Google Forms у частині автоматизації є обмеженим. Відповідно, при необхідності проведення моніторингу навантаження окремих академічних дисциплін, вимагає ручного внесення даних для кожного опитування. Це може призвести до значних часових витрат, особливо в умовах масштабування на велику кількість дисциплін та студентів.

1.2.2. Аналіз платформи для проведення опитувань Survey Monkey

SurveyMonkey – це онлайн-сервіс від компанії Momentive, призначений для розробки досліджень та опитувань [4]. Платформа фокусується на спрощенні процесів збору, опрацювання та візуалізації інформації через створення різноманітних анкет та форм зворотнього зв'язку як на рисунку 1.2. SurveyMonkey має наступні переваги:

- підтримує різноманітні методи розповсюдження опитувань, зокрема через прямі посилання, електронну пошту, інтеграцію на вебсайти та соціальні мережі;
- надає інструменти для створення складних опитувальників з різноманітними типами запитань, застосуванням логіки переходів та використанням готових шаблонів для специфічних завдань;
- надає інструменти для автоматизованого збору, систематизації та аналізу отриманих відповідей, надаючи можливості для генерації детальних звітів та візуалізації даних у режимі реального часу;
- надає змогу інтеграції з іншими програмними продуктами та сервісами, розширюючи функціонал платформи для автоматизації процесів та поглибленого аналізу даних;
- забезпечує можливість створювати форми опитувань за допомогою штучного інтелекту, задаючи запит на створення опитування природньою мовою.

SurveyMonkey демонструє більш досконалі засоби автоматизації, зокрема у сфері генерації запитань для опитувань за допомогою штучного інтелекту. Однак, етапи формулювання запитань та їх валідація залишаються обов'язковими. Суттєвим фактором, що перешкоджає повній автоматизації процесу створення запитань, є брак чіткої асоціації між специфікою програми навчальної дисципліни та структурою формалізованих запитань опитування.

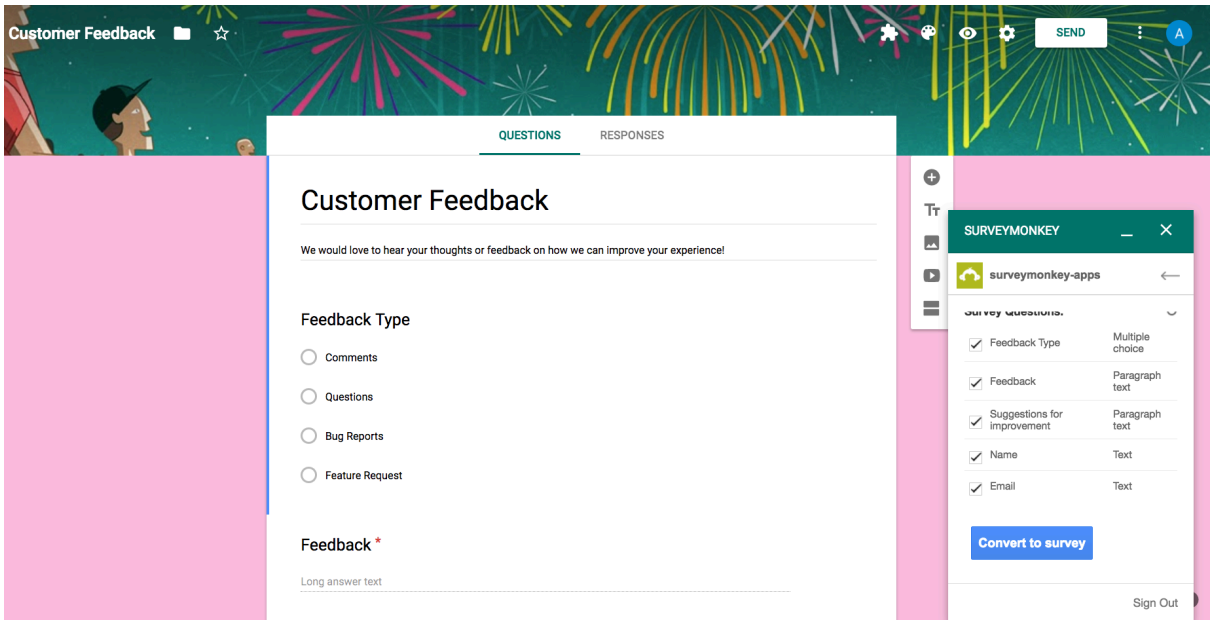


Рис. 1.2. Редагування форм у SurveyMonkey

1.2.3. Аналіз засобів для моніторингу залученості студентів з курсом на платформі Moodle

Moodle – це провідна система для керування навчальним процесом (Learning Management System, LMS) [5] з відкритим кодом, що має наступні переваги:

- система дозволяє структурувати, розміщувати та керувати доступом до різноманітних навчальних матеріалів у цифровому форматі. Приклад навчального курсу можна побачити на рисунку. 1.3.;
- надає інструменти для проведення оцінювання навчальних досягнень, включаючи автоматизовану перевірку тестів, та формування деталізованих звітів про успішність;
- легко адаптується до специфічних потреб закладів освіти та може бути розширений шляхом використання додаткових модулів (плагінів).

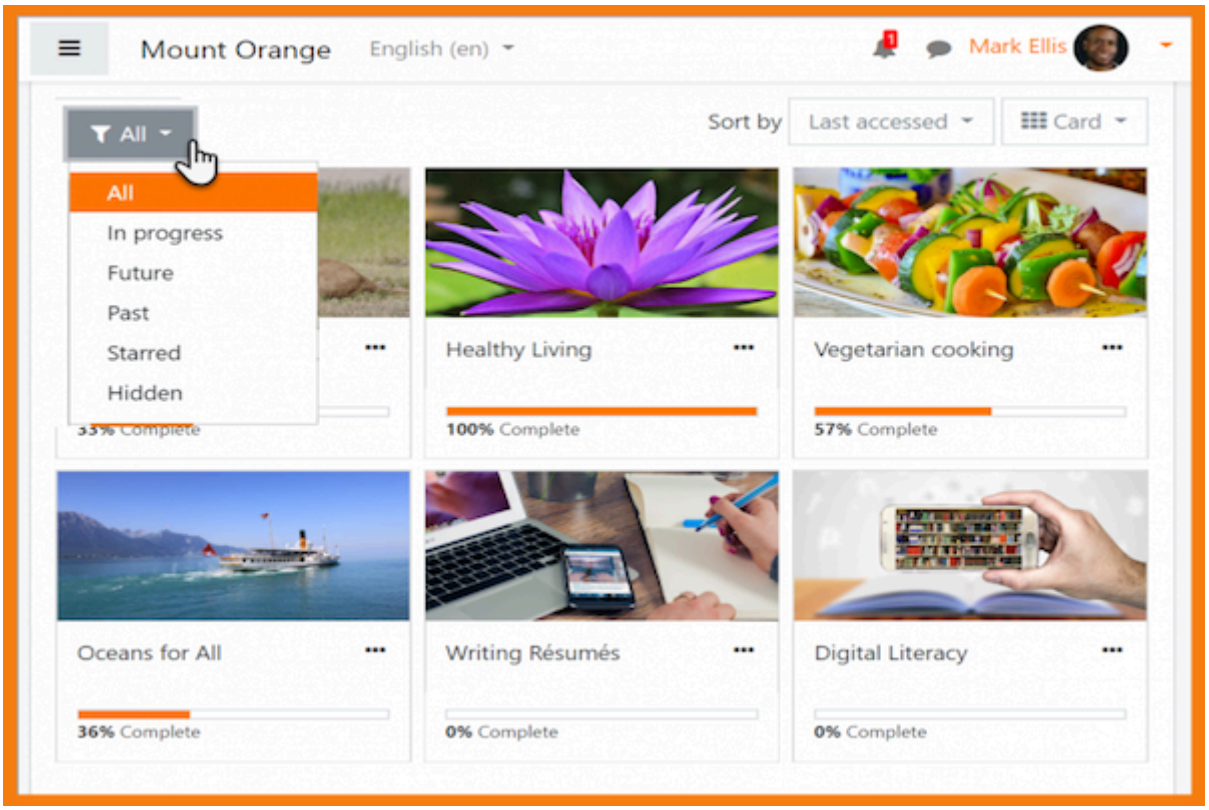


Рис. 1.3. Сторінка курсу в системі Moodle

Course Dedication – плагін для Moodle, створений для вимірювання фактичного часу залучення студентів до курсу. Цей інструмент надає адміністраторам статистику про активність студентів в онлайн-середовищі, як на рисунку 1.4. Ключові можливості Course Dedication охоплюють:

- оцінка тривалості залучення у межах окремого курсу на основі їхніх дій у системі;
- інтеграція з інструментом побудови звітів Moodle, такі як Custom Report Builder, що дозволяє створювати узагальнені звіти про залучення на рівні всього сайту Moodle;
- фільтрація даних звітів за певними критеріями, такими як приналежність до групи або роль користувача в системі.

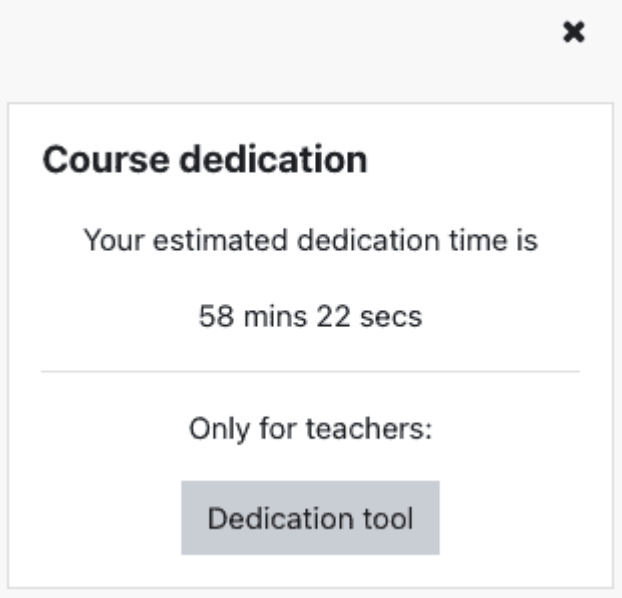


Рис. 1.4. Дані про залученість студента з плагіну Course Dedication

Використання платформи Moodle спільно з плагіном Course Dedication є можливим методом моніторингу активності студентів що користуються платформою, забезпечуючи автоматизований збір даних про час взаємодії з елементами курсу. Проте, слід зазначити, що дані виключно про активність в межах електронного курсу не є вичерпно об'єктивним показником загального навчального навантаження студента або достовірним інструментом оцінки трудомісткості самого курсу.

1.2.4. Аналіз засобів для моніторингу залученості студентів з курсом на платформі Canvas

Canvas – платформа управління навчанням компанії Instructure, яка широко використовується освітніми закладами по всьому світу. Створена компанією у 2011 році, система являє собою сучасне рішення для організації освітнього процесу [6], приклад курсу можна побачити на рисунку 1.5.

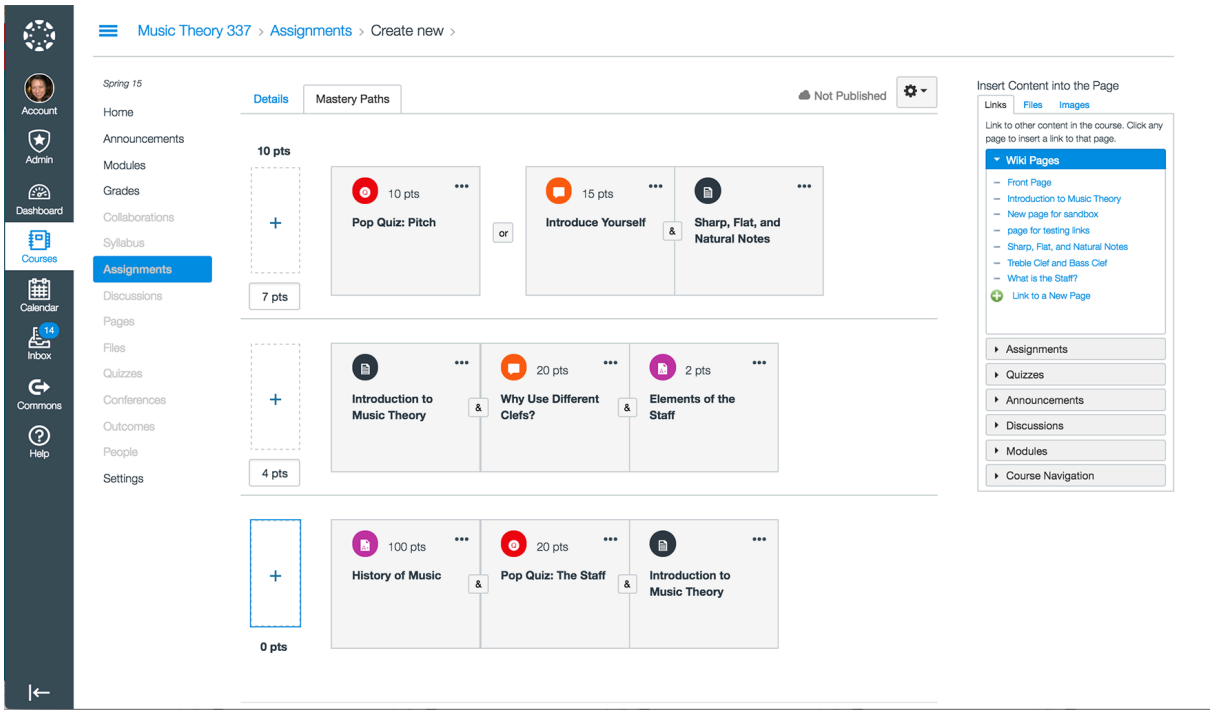


Рис. 1.5. Інтерфейс перегляду курсів Canvas

Student Access Data – частина інструментарію Canvas, який дозволяє адміністрації відстежувати доступ студентів до навчальних матеріалів та активність у курсах. Основні функції Student Access Data включають:

1. Відстеження переглядів конкретних сторінок, ресурсів та елементів курсу, надаючи кількісні показники взаємодії з навчальними матеріалами;
2. Моніторинг участі в діяльності в інтерактивних компонентах курсу, таких як форуми, тести та завдання, відображаючи рівень їх залучення до навчальної взаємодії;
3. Фільтрація аналітичних даних за різними критеріями та можливість їх подальшого експортування для поглибленого аналізу або інтеграції з іншими аналітичними системами.

Student Access Data надає можливість доступу до інформації щодо активності студентів в межах освітніх курсів, що є корисним для аналізу та внесення коректив у навчальний процес. Але важливо враховувати, що факт

доступу до матеріалів, не є достовірним відображенням сукупного навчального навантаження студентів.

1.2.5. Результати порівняння засобів для вимірювання та моніторингу навчального навантаження студентів

Таблиця 1.1

Зведені результати аналізу характеристик програмних рішень для збору та моніторингу навчального навантаження

Характеристика	**Google forms**	**SurveyMonkey**	**Moodle**	**Canvas**	**TaskMon**
Форма збору даних про навантаженість студентів	Опитування (оцінювання) студентів	Опитування (оцінювання) студентів	На основі взаємодії з курсом	На основі взаємодії з курсом та відвідуваності	Оцінювання студентів
Формування питань оцінювання	Вручну	Вручну, за допомогою ШІ	-	-	З навчального плану
Аналітика результатів	Прості діаграми	Прості діаграми	Гістограма та прості діаграми	Гістограма та прості діаграми	Динаміка навантаженості
Експорт аналітики	+	+	-	-	+
Обмеження	Необхідно мати електронну пошту Gmail	-	Використання Moodle в організації навчання	Використання Canvas в організації навчання	-

1.3. Визначення вимог до застосунку для моніторингу навчального навантаження студентів

В результаті аналізу предметної області та існуючих рішень для вимірювання та моніторингу навчального навантаження студентів, ключовими недоліками існуючих систем є те, що вони є загальними інструментами та

безпосередньо не орієнтовані на вирішення задачі моніторингу навчального навантаження студентів.

Отже, варто спроєктувати застосунок, що дозволяють виконувати всі ключові процеси, необхідні для проведення моніторингу навчального навантаження студентів. Спроєктовано потенційний варіант взаємодії адміністрації навчального закладу та студентів з таким застосунком на рисинку 1.6.:

1. Адміністрація навчального закладу створює оцінювання, за яким буде проводитися оцінювання навчального навантаження, та надає його студентам;
2. Студенти проводять оцінювання навчального навантаження, відповідаючи на питання у формі, наданій адміністрацією;
3. За підсумками оцінювання адміністрація отримує статистичну інформацію про навчальне навантаження, що дозволяє за потреби внести корективи в навчальний процес на основі отриманих результатів.

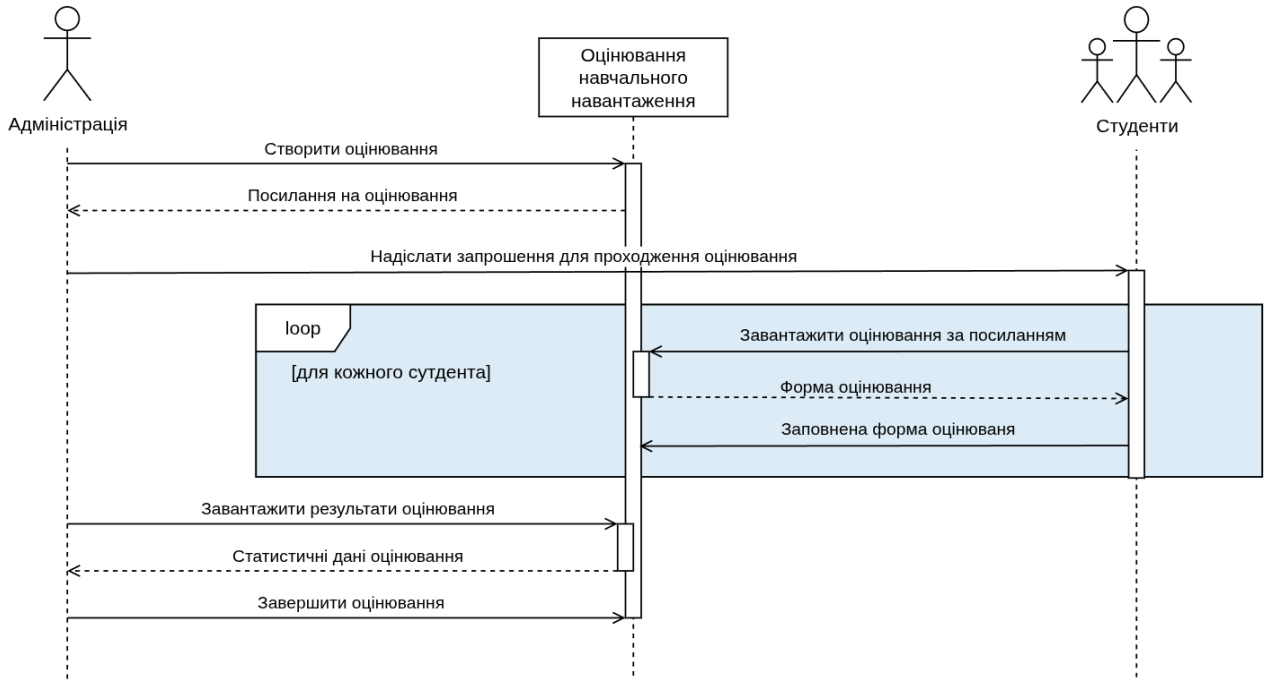


Рис. 1.6. Діаграма взаємодії адміністрації навчального закладу та студентів для проведення моніторингу навчального навантаження

Слід підкреслити, що в цьому випадку йдеться не про оцінювання знань або навчальних досягнень студентів, а про оцінювання ними навчального навантаження. Тобто студенти самостійно оцінюють рівень навчального навантаження.

З огляду цих процесів, стає зрозуміло, що застосунок має два контексти використання які жодним чином не перетинаються. Тобто адміністрація не дає оцінку навчального навантаження, а студенти не мають доступ до керування оцінювання та перегляду результатів. Отже, надалі варто розглядати не один, а два окремих застосунки: застосунок для керування оцінювання та моніторингу, та застосунок для проходження оцінювання навчального навантаження.

1.3.1. Визначення вимог до застосунку для керування оцінюваннями та моніторингу навчального навантаження

Застосунок для керування оцінюваннями та моніторингом навчального навантаження призначений для створення оцінювання навчального навантаження студентів, аналізу результатів оцінювання та моніторингу навчального навантаження на основі цих результатів.

Даний застосунок є ключовою частиною системи, тому потребує особливої уваги. Враховуючи недоліки існуючих рішень для оцінки та моніторингу навчального навантаження студентів, додатково була визначена функціональність, що полегшує процес створення опитувань.

Ключовим недоліком існуючих рішень є те, що вони не орієнтовані безпосередньо на вирішення задачі моніторингу навчального навантаження студентів. У таких системах відсутній прямий зв’язок між завданнями опитування та навчальною програмою. Такі зв'язки моделюються особою, відповідальною за складання форм, що ускладнює користування системою, потребує додаткових часових витрат, та збільшує кількість можливих помилок. Отже, додаток має орієнтуватися на план навчальної дисципліни, як ключовий

об’єкт предметної області, таким чином, щоб адміністрація виконувала задачу заповнення плану навчальної дисципліни, а не задачу створення опитування.

Враховуючи орієнтацію на навчальний план дисципліни, було вирішено автоматизувати задачу занесення даних в систему, оскільки кожен навчальний заклад має визначений навчальний план кожної дисципліни у документах університету. Таким документом зазвичай є силабус навчальної дисципліни.

Таким чином, майбутній програмний продукт повинен включати наступні ключові функції: проведення оцінювання навчального навантаження студентів, та порівняння результатів з очікуваними; автоматичне внесення інформації про навчальну програму з документів.

Функціональні вимоги:

- внесення даних про програму навчальної дисципліни;
- проведення оцінювань навчального навантаження студентів за програмою навчальної дисципліни;
- спостереження за динамікою навчального навантаження у групі студентів та порівняння з очікуваним навантаженням;
- можливість створювати групи оцінювань;
- імпортування програми дисциплін з документів;

Нефункціональні вимоги:

- захист від несанкціонованого доступу до адміністративної частини застосунку;
- забезпечити можливість функціонування застосунку адміністрації, незалежно від функціонування застосунку студентів.

Відповідно до визначених вимог, було спроєктовано наступні варіанти використання застосунком на рисунку 1.7.:

- керування дисциплінами, що включає:

  додавання дисциплін;

  додавання навчальних планів дисциплін;

  перегляд архіву навчальних планів дисциплін;

імпортування навчальних планів з документів.

- створення оцінювань, що включає:

  створення груп оцінювань;

  запрошення студентів на проходження опитування.

- перегляд результатів опитування:

  перегляд динаміки навантаження студентів;

  перегляд розподілу студентів за ступенем навантаження.

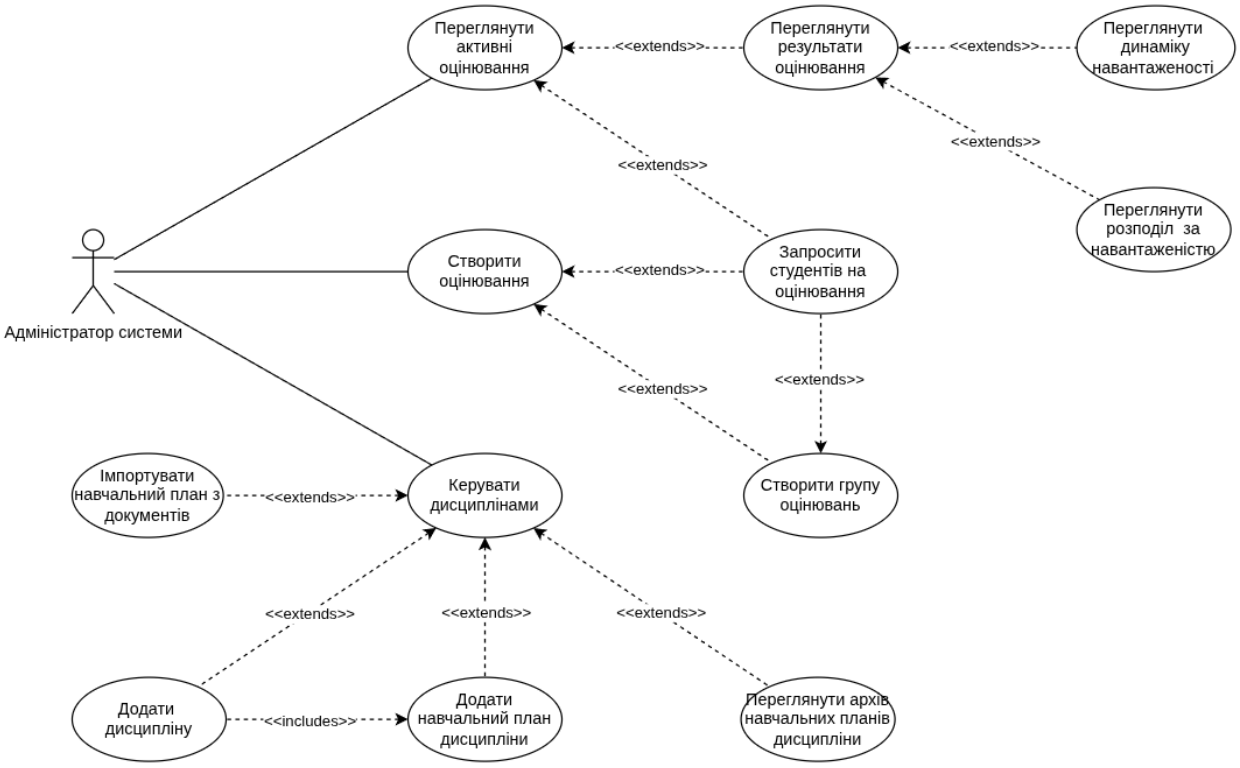


Рис. 1.7. Варіанти використання застосунку для адміністрації

1.3.2. Визначення вимог до застосунку для проходження оцінювання навчального навантаження

Застосунок для проходження оцінювання навчального навантаження призначений для перегляду та заповнення форм оцінювання, створених в застосунку адміністрації.

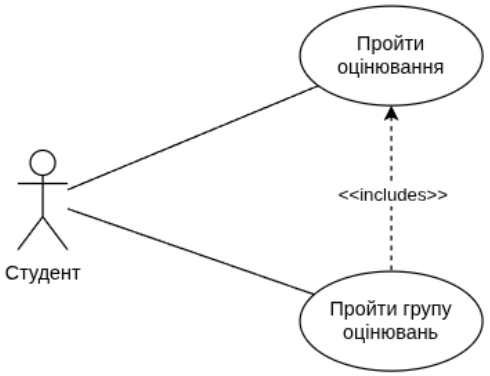


Рис. 1.8. Варіанти використання застосунку для студентів

Функціональні вимоги:

- проходження оцінювань навчального навантаження;
- проходження груп оцінювань навчального навантаження.

Нефункціональні вимоги:

- анонімність відповідей студентів.

Відповідно до визначених вимог, було спроєктовано наступні варіанти використання застосунком на рисунку 1.8.:

- пройти оцінювання навчального навантаження;
- пройти групу оцінювань навчального навантаження.

2. АНАЛІЗ ЗАСОБІВ РОЗРОБКИ

2.1. Огляд технічних викликів при реалізації серверної частини.

Оглянувши вимоги до застосунків, було виділено наступні технічні виклики, що постають для реалізації серверної частини:

- оскільки розробляються два окремі додатки, важливо забезпечити незалежність їх функціонування. Система має бути побудована за принципами модульності;
- система доступу для адміністративної частини, має бути захищена від неавторизованого доступу завдяки процесу аутентифікації. Також необхідно забезпечити механізм збору анонімних відповідей, який унеможливлює ідентифікацію студентів, навіть адміністраторами;
- імпорт навчальних програм з файлів DOCX/PDF вимагає обробки документів, що не завжди мають чітко визначено структуру, та значною мірою відрізняються, залежно від навчального закладу;
- реалізація збору відповідей студентів про реальне навантаження передбачає обробку динамічних даних та побудову аналітики з порівнянням очікуваного і фактичного навантаження, що вимагає складної схеми агрегування;
- з огляду на складність і гнучкість навчальних планів, слід ретельно спроєктувати модель даних – із можливим використанням реляційного сховища для основних даних і документного для збереження імпортованої інформації.

2.2. Аналіз засобів розробки

2.2.1. Огляд C# та ASP.NET Core для розробки API

C# – це об'єктноорієнтована, типізована мова програмування, розроблена компанією Microsoft як частина платформи .NET [7]. Вона підтримує сучасні парадигми, включаючи асинхронність, функціональне програмування та розробку на основі інтерфейсів. Завдяки сильній типізації, багатим можливостям для архітектурного дизайну та інтеграції з .NET-інфраструктурою, C# є популярною мовою для створення надійних, масштабованих вебдодатків, десктопних програм, мобільних застосунків і, зокрема, API. До основних переваг C# відносять наступне:

- застосунки, написані на C#, демонструють високу швидкість виконання, що важливо для забезпечення позитивного досвіду користування застосунком;
- об'єктно-орієнтований підхід дозволяє створювати добре структурований, модульний та масштабований код, що є важливим для середніх та великих проєктів;
- платформа .NET надає багатий набір бібліотек для роботи з базами даних, мережевими протоколами, криптографією та іншими завданнями, що прискорює процес розробки.

ASP.NET Core – це кросплатформенний, високопродуктивний вебфреймворк від Microsoft для створення API, вебзастосунків та мікросервісів. Він побудований на основі .NET Core і дозволяє запускати застосунки на Windows, Linux та macOS. ASP.NET Core є модульним, легким та ідеально підходить для розробки сучасних API з підтримкою REST, авторизації, масштабованості та безпеки завдяки наступним перевагам:

- ASP.NET Core є одним із найшвидших вебфреймворків, що забезпечує низьку затримку та високу пропускну здатність API;

- фреймворк надає зручні інструменти для створення стандартизованих та ефективних API, що є основою для взаємодії backend-частини з іншими сервісами та frontend-частиною;
- можливість розгортання застосунку на різних операційних системах (Windows, macOS, Linux) надає гнучкість у виборі інфраструктури;
- механізм Dependency Injection (DI), що спрощує створення зв'язаних компонентів, що полегшує тестування та підтримку коду.

2.2.2. Огляд Auth0 для забезпечення аутентифікації користувачів

Auth0 – це хмарна платформа для керування автентифікацією та авторизацією користувачів у веб та мобільних застосунках [8]. Вона надає готові сервіси для логіну, реєстрації, соціальної автентифікації, багатофакторної перевірки, керування сесіями, токенами доступу та правами користувачів. Auth0 підтримує сучасні протоколи безпеки, такі як OAuth 2.0, OpenID Connect та SAML, і дозволяє розробникам інтегрувати захист у додаток без потреби реалізовувати складні механізми самостійно.

Auth0 пропонує ефективну та швидку інтеграцію завдяки наявності готових SDK і прикладів коду для ASP.NET Core, що дозволяє швидко впровадити аутентифікацію користувачів.

Гнучке налаштування автентифікації є ще однією перевагою Auth0. Система дозволяє організувати доступ для анонімних користувачів. Це корисно для ресурсів, де персоналізація не є критичною, але все ще потрібна базова перевірка доступу.

Auth0 підтримує багатофакторну автентифікацію (MFA), що суттєво підвищує рівень безпеки, особливо для адміністративних користувачів. Як хмарне рішення з гарантованим рівнем доступності, Auth0 зменшує ризики, пов’язані з власною інфраструктурою, і передає відповідальність за її підтримку на надійного провайдера, дозволяючи зосередитися на розвитку продукту.

2.2.3. Огляд великої мовної моделі Gemini для автоматичного імпортування навчальних планів з документів

Gemini – це сучасна велика мовна модель (LLM), розроблена для розуміння, аналізу та генерації тексту різної складності. Вона здатна працювати з різними форматами документів, витягувати структуровану інформацію та адаптуватися до специфіки текстів різних доменів, включно з навчальними планами. Gemini використовує передові алгоритми глибинного навчання, що дозволяють точно і швидко розпізнавати контекст і формати, навіть якщо документи мають різну структуру або неузгодженості.

Перевагами використання великих мовних моделей над традиційними методами обробки природної мови (NLP) є:

- адаптивності до різноманітних форматів даних. Мовні моделі не потребують жорстких шаблонів або чіткої структури вхідного тексту, тому легко працюють із документами різних навчальних закладів, які можуть суттєво відрізнятися за оформленням;
- великі мовні моделі демонструють високу ефективність у завданнях автоматичного витягання ключових даних. Вони можуть швидко і точно розпізнавати й класифікувати важливу інформацію – як-от предмети, теми, кількість годин і типи занять – навіть у складно структурованих або неформалізованих текстах;
- зручність інтеграції великих мовних моделей у сучасну інфраструктуру через API. Це дозволяє легко вбудовувати їх у серверну логіку для потокової обробки імпортованих документів, що робить можливим масштабування рішень та автоматизацію типових задач без потреби в окремих етапах попередньої обробки чи перенавчання.

2.2.4. Огляд Python та Flask API для роботи з великими мовними моделями

Python – це високорівнева, інтерпретована мова програмування, відома своєю простотою і читабельністю коду [9]. Завдяки багатій екосистемі бібліотек для машинного навчання, обробки природної мови та інтеграції з API, Python є одним із найпопулярніших виборів для розробки сервісів, що працюють з великими мовними моделями. Він чудово підходить для швидкого створення прототипів AI-рішень.

Переваги мови програмування Python для роботи зі штучним інтелектом включають:

- розвинена екосистема та активна підтримка з боку спільноти. Його бібліотеки постійно оновлюються, забезпечуючи сучасні можливості для машинного навчання. Широка база знань, прикладів і документації робить Python зручним вибором як для початківців, так і для досвідчених розробників;
- здатність легко інтегруватися зі сторонніми сервісами, REST API та хмарними платформами. Це дозволяє швидко підключати LLM до зовнішніх джерел даних, інтерфейсів або аналітичних систем, спрощуючи розгортання повноцінних продуктів. Така гнучкість робить Python зручним вибором для створення інтерактивних застосунків з LLM;
- Python підходить для швидкої розробки завдяки своїй зрозумілій синтаксичній структурі та широким можливостям для обробки тексту й даних., розробники можуть швидко реалізовувати та тестувати ідеї.

Flask API – це легкий вебфреймворк для Python, який дозволяє швидко створювати RESTful API та вебсервіси. Його мінімалістичний підхід дає змогу будувати прості, масштабовані та розширювальні серверні додатки, ідеальні для інтеграції з великими мовними моделями, надаючи API для обробки запитів, управління сесіями та маршрутизації. Flask API є популярним вибором для розробки API завдяки наступним перевагам:

- простий синтаксис і мінімалістична структура дозволяють швидко налаштувати обробку HTTP-запитів без зайвого навантаження. Це робить FlaskAPI ідеальним для проєктів, де існує необхідність розробки простого бекенду;
- гнучкість реалізації. Розробник має повний контроль над маршрутизацією, middleware та обробниками запитів, що дає змогу легко адаптувати API під специфічні вимоги проєкту. Це дозволяє ефективно реалізовувати як прості, так і складні сценарії взаємодії з клієнтами;
- FlaskAPI добре масштабується завдяки можливості підключення додаткових розширень та сторонніх бібліотек. Його легко інтегрувати з базами даних, системами кешування, сервісами черг і навіть з іншими фреймворками, якщо виникає потреба в масштабуванні. Існують численні бібліотеки, які розширюють можливості безпеки, підтримують JWT, OAuth та інші сучасні методи захисту;
- такоє сильною стороною FlaskAPI є малий обсяг початкового коду. Розробник може розгорнути повноцінний API буквально з кількох рядків, що суттєво зменшує час розробки.

2.2.5. Огляд PostgreSQL для зберігання реляційних даних

PostgreSQL – об’єктно-реляційна система управління базами даних (СУБД), відома своєю надійністю, розширюваністю і відповідністю стандартам SQL. Вона підтримує складні типи даних, транзакції, індекси та різноманітні механізми контролю цілісності, що робить її ідеальним вибором для зберігання структурованих і взаємопов’язаних даних у складних інформаційних системах. До основних переваг PostgreSQL відносять:

- сувора відповідність ACID-принципам надає стабільну платформу для зберігання критично важливої інформації, зокрема у фінансових, наукових та аналітичних системах;

- гнучка архітектура, що дозволяє розширювати можливості системи за допомогою власних типів даних, користувацьких функцій, індексів і модулів. Це дає змогу адаптувати базу даних під специфічні задачі та створювати унікальні рішення, не виходячи за межі самої СУБД;
- активну і досвідчену спільноту, яка постійно вдосконалює систему, випускає оновлення та надає підтримку. Завдяки цьому вона залишається однією з найстабільніших і найнадійніших відкритих СУБД.

2.2.6. Огляд MongoDb для зберігання не реляційних даних

MongoDB – це документно-орієнтована база даних, яка зберігає дані у форматі BSON (розширений формат JSON). Вона добре підходить для зберігання гнучких, нереляційних структур.

MongoDB надає гнучкість у зберіганні даних завдяки відсутності суворої схеми. Це означає, що кожен документ у колекції може мати різну структуру, що ідеально підходить для швидкої розробки, прототипування та роботи з динамічними або слабо структурованими даними.

Система масштабування MongoDB дозволяє легко обробляти великі обсяги даних через горизонтальне масштабування. Це забезпечує високу продуктивність навіть при зростанні навантаження. Крім того, MongoDB має вбудовані інструменти для агрегації, що дозволяє обробляти та аналізувати дані безпосередньо в базі даних.

2.2.7. Огляд Swagger UI для документування API

Swagger UI – це інтерактивний інструмент для візуалізації, тестування та документування REST API, заснований на специфікації OpenAPI. Він автоматично генерує вебінтерфейс документації, який дозволяє розробникам переглядати доступні ендпоінти, вхідні/вихідні дані та взаємодіяти з API безпосередньо з браузера. Swagger UI часто використовується разом із

серверними фреймворками, такими як ASP.NET Core або Flask, для спрощення роботи з API під час розробки та інтеграції.

Swagger UI значно спрощує процес створення документації для API завдяки автоматичній генерації на основі коду. Це усуває потребу у веденні документації вручну, зменшуючи ризик розбіжностей між реалізацією й описом. Розробники можуть зосередитись на логіці API, покладаючи документування на інтегровані інструменти.

Документація Swagger UI надає можливість надсилати запити до API прямо з браузера. Це зручно як для налагодження, так і для демонстрації функціональності без використання сторонніх клієнтів. Такий підхід пришвидшує виявлення помилок і прискорює розробку. А також покращує взаємодію в команді, оскільки створює візуально зрозумілу і доступну структуру API. Аналітики, тестувальники та інші члени команди можуть швидко ознайомитися з можливостями сервісу, що сприяє кращій координації та ефективнішій розробці.

2.2.8. Огляд .NET Aspire для оркестрування мікросервісів

.NET Aspire – це новий набір інструментів та шаблонів від Microsoft, створений для спрощення розробки, налаштування та оркестрування мікросервісної архітектури в екосистемі .NET [10]. Aspire дозволяє створювати взаємопов’язані сервіси, визначати залежності (наприклад, бази даних, кеші, черги повідомлень) та запускати все разом у вигляді керованого проєкту. Він поєднує можливості локального запуску, конфігурації, моніторингу та забезпечує плавний перехід до хмарної інфраструктури.

.NET Aspire надає централізоване управління конфігурацією, що значно полегшує роботу з секретами, змінними середовища та параметрами запуску. Завдяки цьому розробники можуть керувати налаштуваннями всіх сервісів в одному місці, зменшуючи ризики помилок і спрощуючи розгортання в різних середовищах.

Розробка завдяки готовим шаблонам і глибокій інтеграції з популярними технологіями, такими як ASP.NET Core, gRPC та Entity Framework. Це дозволяє створювати складні розподілені системи з мінімальними зусиллями, використовуючи знайомі інструменти та практики .NET-екосистеми. .NET Aspire забезпечує наступні ключові функції:

- забезпечує запуск усіх компонентів у єдиному контексті, що спрощує локальну розробку, тестування та налагодження розподілених рішень;
- надає можливість організувати повноцінний моніторинг, логування та трасування між сервісами без додаткових налаштувань, забезпечуючи прозорість роботи системи й пришвидшуючи виявлення проблем у продуктивному середовищі.

2.3. Організація процесу розробки

Розробка застосунку велась у команді, що складалась з двох учасників: один відповідав за бекенд (серверна логіка та API), другий – за фронтенд (користувацький інтерфейс та взаємодію з API). Основний акцент був зроблений на чітке розмежування відповідальності та синхронізацію дій через інструменти командної співпраці.

Для організації робочого процесу використовувався GitHub як платформа для зберігання коду, система контролю версій.

GitHub – це популярна вебплатформа для зберігання та керування вихідним кодом, яка базується на системі контролю версій Git. Вона дозволяє розробникам ефективно співпрацювати над проєктами, використовуючи гілки (branches), pull request'и, код-рев’ю та автоматизацію CI/CD. GitHub підтримує відкриті та приватні репозиторії, забезпечуючи зручне середовище для розробки як в командах, так і індивідуально.

Координація завдань здійснювалась через GitHub Projects з шаблоном Kanban-дошки. Завдання ділилися на стадії “To Do”, “In Progress” і “Done”, що забезпечувало прозору організацію роботи, дозволяло слідкувати за прогресом та ефективно розподіляти навантаження. Обговорення задач, визначення пріоритетів та планування спринтів проводились усно.

GitHub Projects – це вбудований інструмент керування проєктами, який дозволяє планувати, відстежувати та організовувати робочі процеси безпосередньо в GitHub. Він інтегрується з Issues і Pull Requests, забезпечуючи прозору візуалізацію прогресу, розподіл обов’язків і гнучке управління розробкою в реальному часі.

3. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Огляд принципів мікросервісної архітектури

Мікросервіси – це стиль архітектури, за якого великі та складні програмні застосунки складаються з одного або кількох сервісів. Кожен мікросервіс може розгортатися незалежно від інших і є слабо пов’язаним з ними. Кожен із цих мікросервісів зосереджений лише на виконанні одного завдання – і виконує його дуже добре. У всіх випадках це завдання відповідає невеликій бізнес-функції [11].

Можна виділити наступні властивості сервісів в мікросервісній архітектурі:

- кожен мікросервіс виконує одну чітко визначену бізнес-функцію і робить це максимально ефективно. Такий підхід полегшує підтримку, тестування та розгортання сервісу, а також дозволяє легко змінювати його логіку без впливу на всю систему;
- мікросервіси мінімально залежать один від одного й взаємодіють через чітко визначені інтерфейси (API). Це забезпечує незалежність у розробці, тестуванні, масштабуванні та оновленні, що значно підвищує стабільність та гнучкість всієї системи;
- кожен сервіс може бути реалізований будь-якою мовою програмування, що найкраще підходить для конкретного завдання. Це відкриває можливість використовувати найкращі інструменти для кожної задачі;
- кожен мікросервіс працює у власному логічно відокремленому контексті, який не перетинається з іншими. Це дозволяє уникати конфліктів у бізнес-логіці та даних, підтримувати чітке розділення відповідальності й

полегшує моделювання складних доменів, наприклад за допомогою доменно-орієнтованого програмування (Domain-Driven Design).

Використання мікросервісної архітектури забезпечує низку ключових переваг, які роблять її популярним вибором для сучасної розробки [12]:

- кожен сервіс може розроблятися з використанням тієї технології, яка найкраще підходить до його завдань і навантаження. Такий підхід забезпечує максимальну ефективність кожного компонента системи;
- сервіси можна оновлювати та розгортати автономно, без необхідності координувати зміни у всій системі. Це знижує ризики та дозволяє підтримувати різні цикли випуску;
- кожен сервіс масштабується окремо, що дає змогу ефективніше розподіляти ресурси, особливо в хмарному середовищі. Це зменшує витрати та підвищує продуктивність;
- команди організовуються навколо бізнес-функцій, а не технічних шарів, що сприяє більшій ефективному прийняттю рішень і прискоренню розробки.

Використання мікросервісної архітектури в цьому проєкті надає суттєві переваги завдяки розділенню системи на незалежні, спеціалізовані компоненти. Це прискорює розробку та тестування, а також полегшує впровадження змін без впливу на всю систему, та дозволяє використовувати різні технологій для розробки (C# та Python). Крім того, мікросервіси сприяють підвищенню надійності: відмова одного компонента не призводить до зупинки всього застосунку.

Хоча використання однієї бази даних в мікросервісній архітектурі вважається поганою практикою [13], що лише збільшує зв’язаність сервісів, усе одно було вирішено використовувати цей підхід для полегшення розробки та розгортання, використання спільної предметної області, й економії ресурсів, оскільки в разі розділення схем доведеться зберігати багато повторюваних даних.

3.2. Проєктування чистої архітектури (Clean architecture)

Чиста архітектура (Clean Architecture) – це підхід до проєктування програмного забезпечення, запропонований Робертом Мартіном [14], що ставить у центрі бізнес-логіку та забезпечує її незалежність від інфраструктури, інтерфейсу. Основна ідея полягає у розділенні коду на концентричні кола залежностей, де внутрішні шари не залежать від зовнішніх.

Це дозволяє змінювати інфраструктуру або технології (наприклад, UI-фреймворк чи СУБД) без впливу на ядро системи.

У чистій архітектурі програму поділено на шари відповідальності, які мають чіткі правила: взаємодії, що зображені на рисунку 3.1. [15]:

- предметна область (Domain або Entities) – відповідає за визначення сутностей, які відображають бізнес-об'єкти. Цей шар не має залежностей від інших частин програми, що забезпечує його максимальну стабільність і незалежність від інфраструктурних змін;
- додаток (Application) або варіанти використання (Use cases) містить бізнес-логіку, яка керує виконанням конкретних сценаріїв. Тут реалізуються сервіси та варіанти використання, які працюють із сутностями з предметної області та взаємодіють з інтерфейсами зовнішніх систем;
- інфраструктура (Infrastructure) – реалізує залежності, описані в контрактах попередніх шарів. Тут знаходяться класи, що працюють з базами даних, логування, доступ до зовнішніх API тощо;
- представлення (Presentation) відповідає за взаємодію з користувачем. Він отримує запити, передає їх у шар Application і повертає відповіді результати.

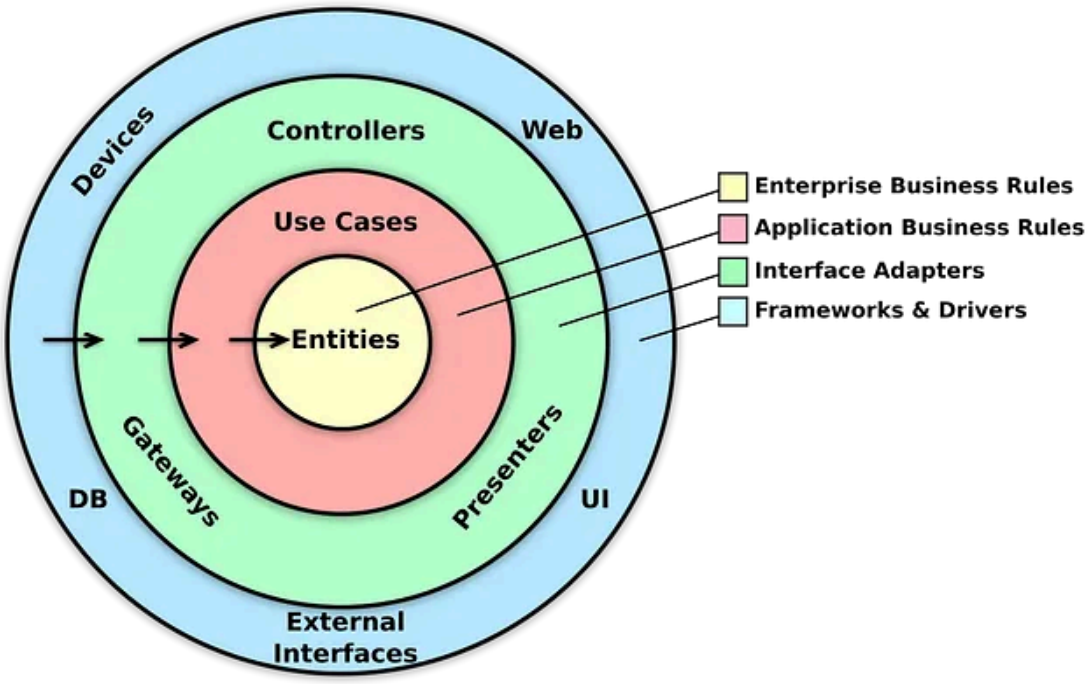


Рис. 3.1. Шари програми в чистій архітектурі

Реалізація Clean Architecture у проєктах .NET зазвичай передбачає створення окремих проєктів. Зв’язок між шарами забезпечується через інверсію залежностей, що в C# реалізується через впровадження залежностей (DI-контейнер) [16].

Використання чистої архітектури дозволяє чітко відокремити бізнес-логіку від інфраструктури та зовнішніх залежностей, що корисно для проєкту з декількома підсистемами. Такий підхід забезпечує високу модульність, тестованість і розширюваність, дозволяє легко змінювати базу даних, API або зовнішні сервіси без впливу на ядро програми. Принципи чистої архітектури застосовувалися лише для сервісів, написаних на C# в цьому проєкті.

3.3. Огляд протоколу OAuth 2.0 для процесу авторизації

OAuth 2.0 – це протокол авторизації, який дозволяє стороннім додаткам отримувати обмежений доступ до ресурсів користувача без передачі пароля, що забезпечує високий рівень безпеки як і додатків, так і користувачів [17]. У протоколі OAuth 2.0 беруть участь чотири основні учасники процесу аутентифікації:

- власник ресурсу (Resource Owner) – це суб’єкт (зазвичай користувач), яка володіє захищеними ресурсами та має право вирішувати, які додатки можуть отримати до них доступ. Він надає дозвіл (авторизацію) клієнтському додатку через сервер авторизації;
- клієнтський додаток (Application) – це додаток, який хоче отримати доступ до ресурсів користувача на сервері ресурсів. Він діє від імені користувача, запитує дозвіл у сервера авторизації та, отримавши токен доступу, використовує його для взаємодії з ресурсами;
- сервер авторизації (Authorization Server) – це сервер, який аутентифікує власника ресурсу (користувача), обробляє запити авторизації від клієнтів і видає токени доступу (та оновлення), що дозволяють клієнту отримувати захищені дані;
- сервер ресурсів (Resource Server) – це сервер, який зберігає та захищає ресурси користувача, наприклад API або дані. Він приймає запити з токенами доступу, перевіряє їх дійсність і, якщо все правильно, надає доступ до відповідних ресурсів.

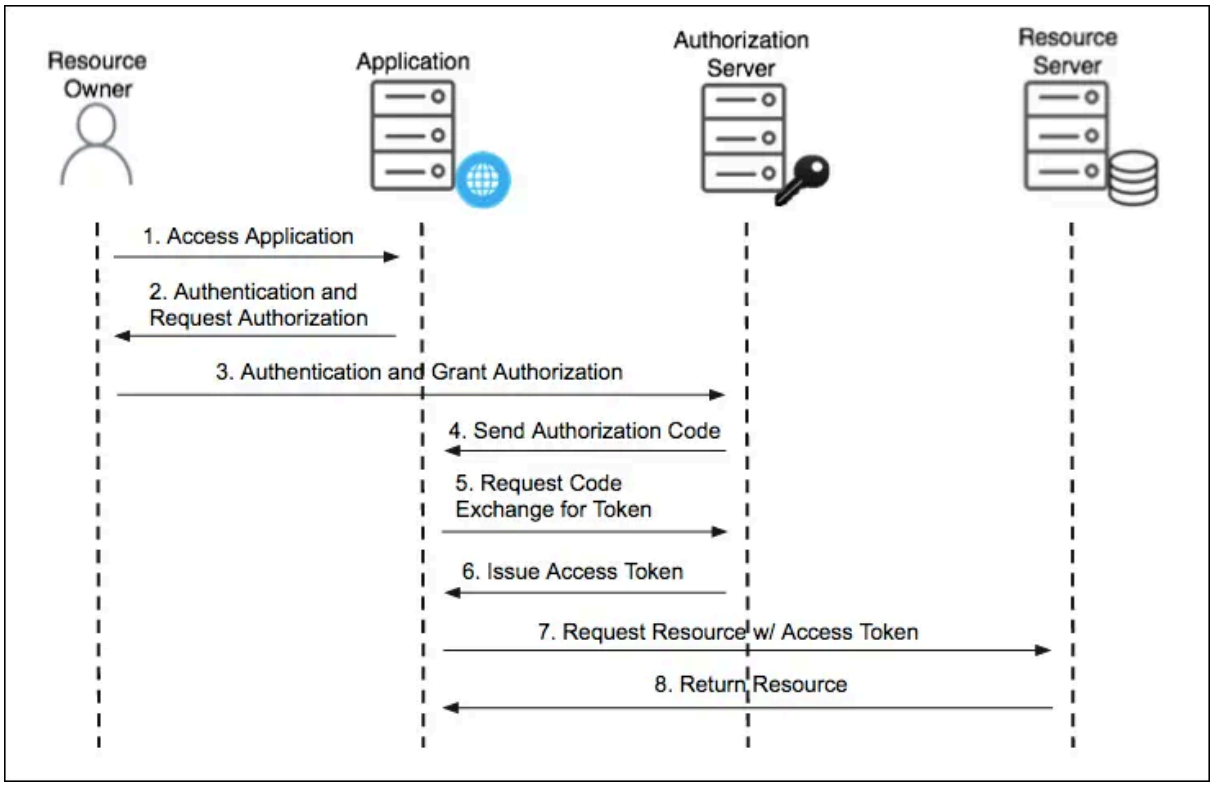


Рис. 3.2. Процес аутентифікації користувача за протоколом OAuth 2.0

Процес аутентифікації користувача за протоколом OAuth 2.0 відбувається за кроками, що зображені на рисунку 3.2. [18]:

1. Доступ до додатка (Access Application): Користувач відкриває додаток, який хоче отримати доступ до його даних;

2. Аутентифікація та запит дозволу (Authentication and Request Authorization): Додаток перенаправляє користувача на сервер авторизації, де він проходить аутентифікацію (вводить логін і пароль) і підтверджує, що надає додатку дозвіл на доступ до певних даних;

3. Аутентифікація та надання дозволу (Authentication and Grant Authorization): Користувач успішно проходить аутентифікацію та погоджується надати додатку доступ до запитуваних ресурсів;

4. Надання коду авторизації (Send Authorization Code): Сервер авторизації перенаправляє користувача назад до додатка, передаючи код авторизації у параметрах запиту;

5. Обмін коду на токен (Request Code Exchange for Token): Додаток надсилає отриманий код авторизації до сервера авторизації, щоб обміняти його на токен доступу;

6. Видача токена доступу (Issue Access Token): Сервер авторизації перевіряє код і видає токен доступу, який дозволяє додатку отримувати доступ до ресурсів користувача;

7. Запит до ресурсу з токеном доступу (Request Resource w/ Access Token): Додаток надсилає запит до сервера ресурсів, додаючи токен доступу до заголовків, щоб отримати доступ до захищених даних.

8. Повернення ресурсу (Return Resource): Сервер ресурсів перевіряє дійсність токена і повертає запитувані дані додатка.

3.4. Огляд процесу розгортання застосунку

Процес розгортання сервісів охоплює створення готових до запуску артефактів, налаштування конфігурацій (змінні середовища, секрети, інтеграції), підготовку інфраструктури (сервери, БД, шлюзи), безпосереднє розміщення компонентів у цільовому середовищі, а також перевірку їхньої взаємодії для забезпечення стабільної роботи системи.

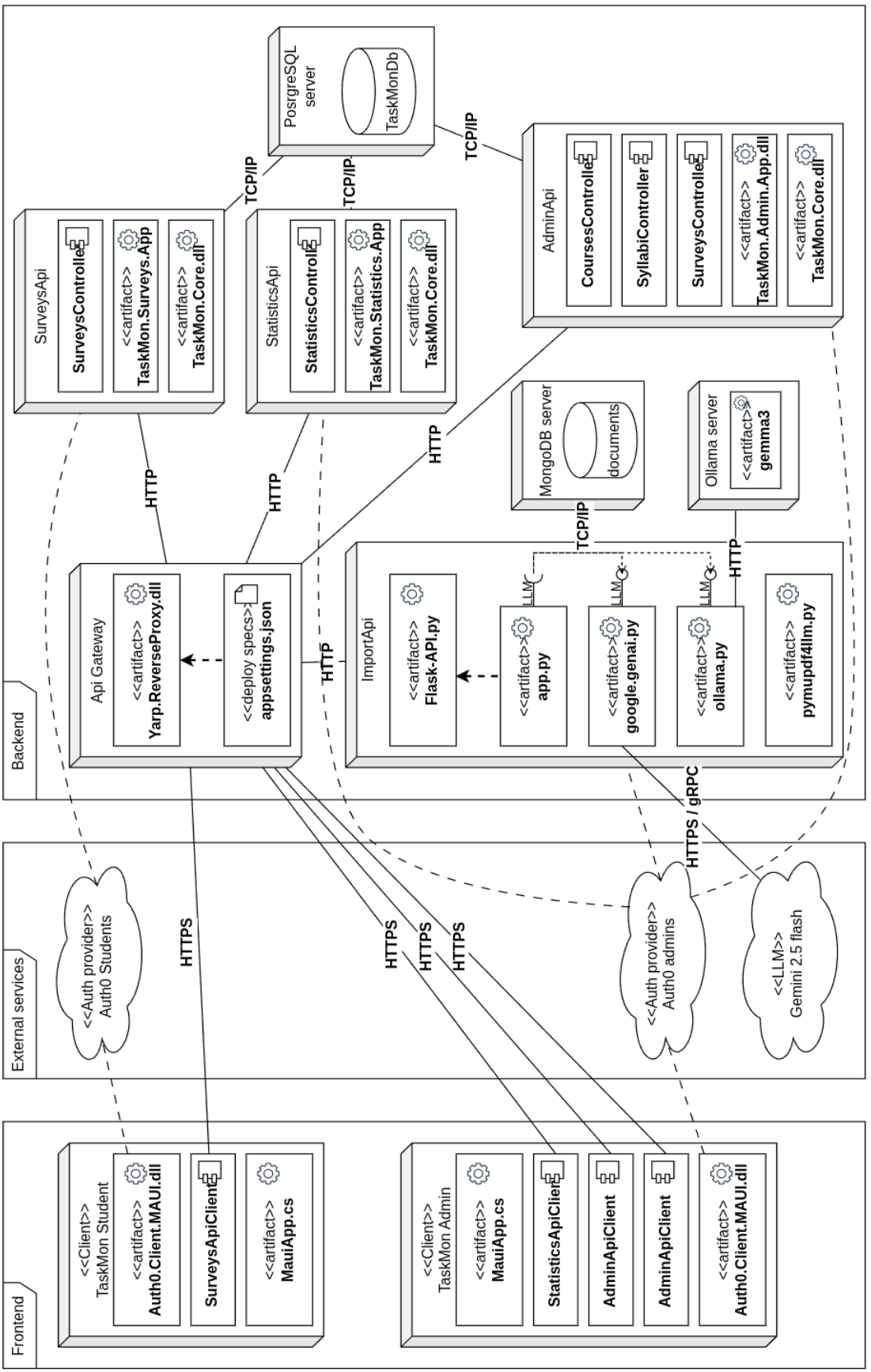


Рис. 3.3. Діаграма розгортання проєкту

Загальна архітектура проєкту побудована за принципом мікросервісів і розподілена на обчислювальні вузли, що зображені на рисунку 3.3.

- клієнтські додатки (MAUI) – студенти й адміністратори взаємодіють із застосунком через два окремих MAUI-додатки. Вони автентифікуються через Auth0 і надсилають запити до API. Ці додатки використовують бібліотеки клієнтів для взаємодії з відповідними сервісами.
- сервіси:

  AdminApi – забезпечує керування навчальними курсами, силабусами (навчальними програмами) та опитуваннями. Реалізує авторизований доступ, відокремлюючи адміністративні можливості від студентських;

  SurveysApi – відповідає за створення, оновлення, отримання та видалення опитувань. Він надає API-методи для клієнтів (зокрема, студентського додатка), які дозволяють проходити опитування або переглядати їхній зміст;

  StatisticsApi – обробляє дані, пов’язані зі статистикою проходження опитувань або навчального прогресу. Він агрегує інформацію, виконує підрахунки та формує аналітику, яка доступна адміністраторам через відповідний клієнт;

  ImportApi – відповідає за обробку PDF-документів, імпорт змісту та його аналіз за допомогою LLM-моделей. Після обробки витягнута інформація зберігається у форматі документів у MongoDB;
- API-шлюз (ApiGateway), який приймає всі зовнішні HTTP(S) запити з клієнтів (MAUI-додатків) і перенаправляє їх до відповідних мікросервісів у бекенді;
- сховища даних:

  PostgreSQL використовується як головне джерело структурованих даних у системі. Вона зберігає інформацію про користувачів, курси, силабуси, опитування, відповіді студентів і статистику;

MongoDB використовується як неструктуроване документоорієнтоване сховище, куди зберігаються дані, отримані за допомогою ImportApi;

- зовнішні сервіси:

Auth0 виконує роль зовнішнього провайдера автентифікації та авторизації, що реалізує стандартний протокол OAuth2. У системі TaskMon є два незалежні контексти автентифікації: Auth0 Students – для студентів, Auth0 Admins – для адміністративних користувачів;

LLM-сервіси – для обробки тексту й PDF-файлів використовується Gemini, але альтернативно може бути розгорнутий та використаний в локальному середовищі сервіс Ollama [19] з моделлю Gemma3 [20];

4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Загальний огляд розробленого рішення

У процесі розробки було реалізовано комплексне архітектурне рішення відповідно до принципів чистої архітектури (Clean Architecture) та принципів мікросерісів, що дозволяє легко масштабувати й підтримувати систему. Рішення складається з 18-ти основних проєктів, логічно згрупованих за рівнями відповідальності та сервісами, що входять до складу системи. Їх взаємозв’язки можна побачити на рисунку 4.1.

До ядра (Core) належать два ключові проєкти:

- Domain – містить сутності, value-об’єкти, інтерфейси. Він повністю ізольований і не залежить від жодного іншого проєкту;
- Application – визначає інтерфейси до зовнішніх інфраструктурних компонентів що використовуватимуться у всіх сервісах.

Infrastructure – це єдиний проєкт, який реалізує інфраструктурні залежності для всіх сервісів: реалізацію інтерфейсів з Application/Domain, конфігурацію доступу до баз даних (наприклад, через EF Core), сервіси логування, надсилання повідомлень тощо. Він залежить від Application і Domain, але сам по собі не використовується напряму кінцевими користувачами.

Кожен із чотирьох функціональних сервісів має власну трирівневу структуру, що складається з:

- [Назва сервісу].App – модуль із варіантами використання, які специфічні для конкретного сервісу, і які можуть взаємодіяти з ядром або реалізовувати додаткову бізнес-логіку.

- [Назва сервісу] – проєкт на основі ASP.NET Core Web API, що відповідає за маршрутизацію HTTP-запитів. Він делегує логіку до відповідного App модулю.
- [Назва сервісу].Client – бібліотека-клієнт, яка надає строго типізовані методи доступу до API сервісу за допомогою бібліотеки Refit [21], що надалі будуть використані в мобільному застосунку на .NET MAUI.

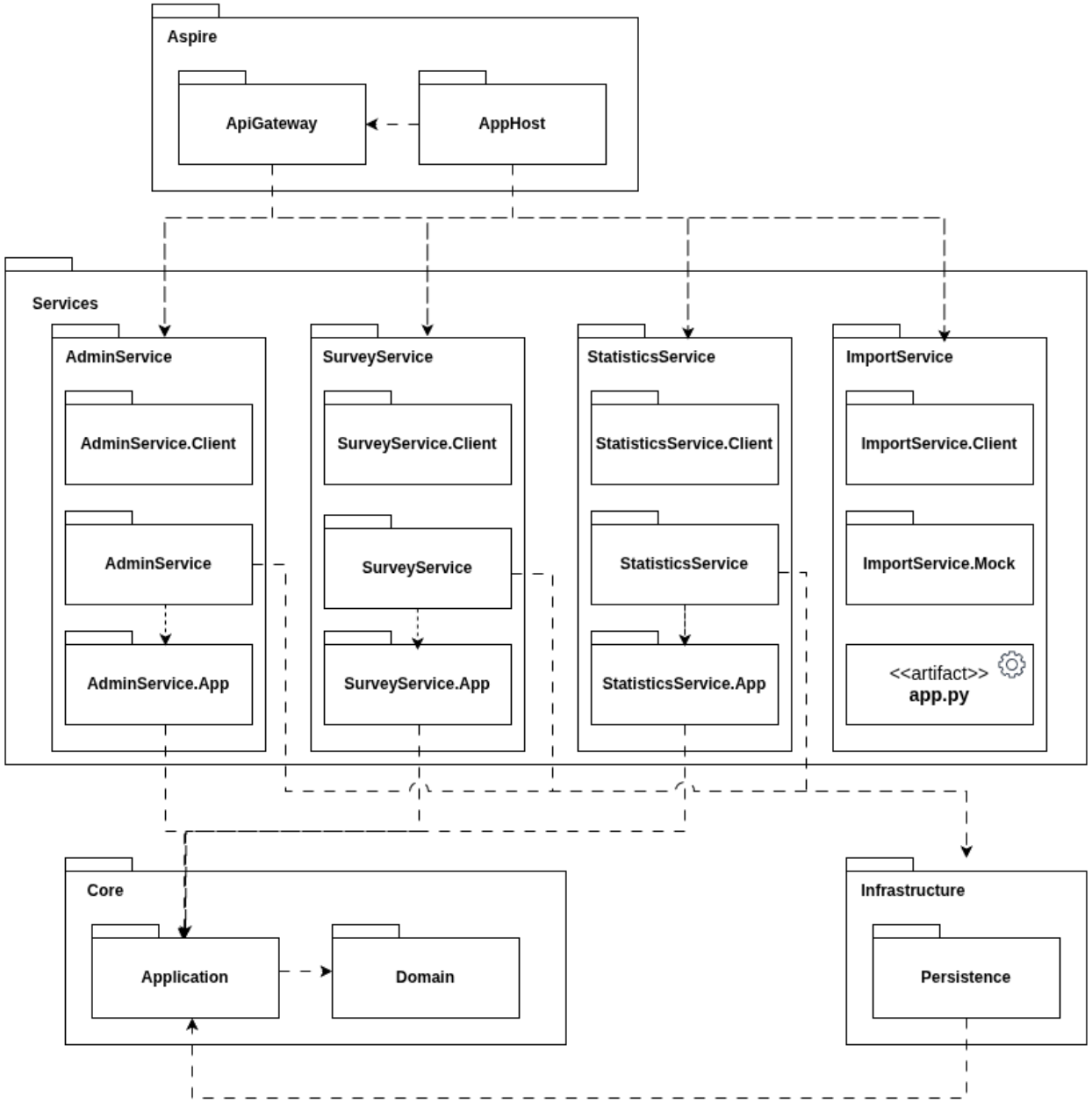


Рис. 4.1. Діаграма пакетів розробленого рішення

Виключенням до цієї схеми є сервіс імпортування, оскільки він написаний за допомогою мови програмування python. Тому він не має залежностей до ядра, але усе одно визначає клієнтську бібліотеку ImportServie.Client, та для зручності тестування інтеграції з клієнтом було розроблено макет Api, що імітує поведінку сервісу ImporService.Mock

Останнім шаром в системі є високорівневі проєкти оркестрування (Aspire):

- ApiGateway – забезпечує єдину точку доступу до всієї системи для зовнішніх застосунків, виступаючи як фасад для взаємодії з окремими сервісами.
- AppHost – це проєкт оркестрування .NET Aspire, який виконує роль центральної точки керування всіма компонентами рішення. Він забезпечує єдину конфігурацію та запуск мікросервісів, інфраструктурних служб і допоміжних проєктів у локальному середовищі, значно спрощуючи процес розробки та розгортання.

4.2. Розробка сервісу для створення та керування оцінюваннями

Розроблено сервіс для керування створенням та керуванням оцінюваннями навчального навантаження, що відповідає за адміністрування організацію процесу оцінювання. Зокрема, він забезпечує створення та редагування навчальних дисциплін і їхніх навчальних планів, а також формування груп оцінювань і запуск опитувань серед студентів. Код ініціалізації сервісу можна побачити на рисунку 4.2.

```
var builder = WebApplication.CreateBuilder(args);

builder.AddServiceDefaults();
builder.Services.AddOpenApi();

builder.Services.AddControllers();

builder.AddNpgsqlDbContext<ApplicationContext>(connectionName: "postgresdb");
builder.Services.AddTransient<IApplicationContext>(s :IServiceProvider => s.GetRequiredService<ApplicationContext>());

builder.Services.AddScoped<CoursesService>();
builder.Services.AddScoped<SurveysService>();
builder.Services.AddScoped<SyllabiService>();

var app :WebApplication = builder.Build();

app.MapOpenApi();
app.UseSwaggerUI(options => options.SwaggerEndpoint( url: "/openapi/v1.json", name: "AdminService"));
app.MapScalarApiReference();

app.UseHttpsRedirection();

app.MapControllers();

app.Run();
```

Рис. 4.2. Код ініціалізації сервісу (Program.cs)

Цей сервіс реалізує наступні ендпоінти, що можна побачити в документації Swagger UI, що зображено на рисунку 4.3.:

- GET /Courses – отримати всі курси;
- POST /Courses – створити новий курс;
- DELETE /Courses/{courseId} – видалити курс;
- PATCH /Courses/{courseId}/title – оновити назву курсу;
- GET /Surveys – отримати всі опитування;
- POST /Surveys – створити опитування;
- GET /Surveys/groups – отримати групи опитувань;
- POST /Surveys/groups – створити групу опитувань;
- POST /Surveys/courses/{courseId} – створити опитування для курсу;
- GET /courses/{courseId}/syllabuses/current – отримати поточний силабус курсу;

- POST /courses/{courseId}/syllabuses/current – створити або оновити поточний силабус;
- GET /courses/{courseId}/syllabuses – отримати всі силабуси курсу;
- GET /courses/{courseId}/syllabuses/{syllabusId} – отримати конкретний силабус.

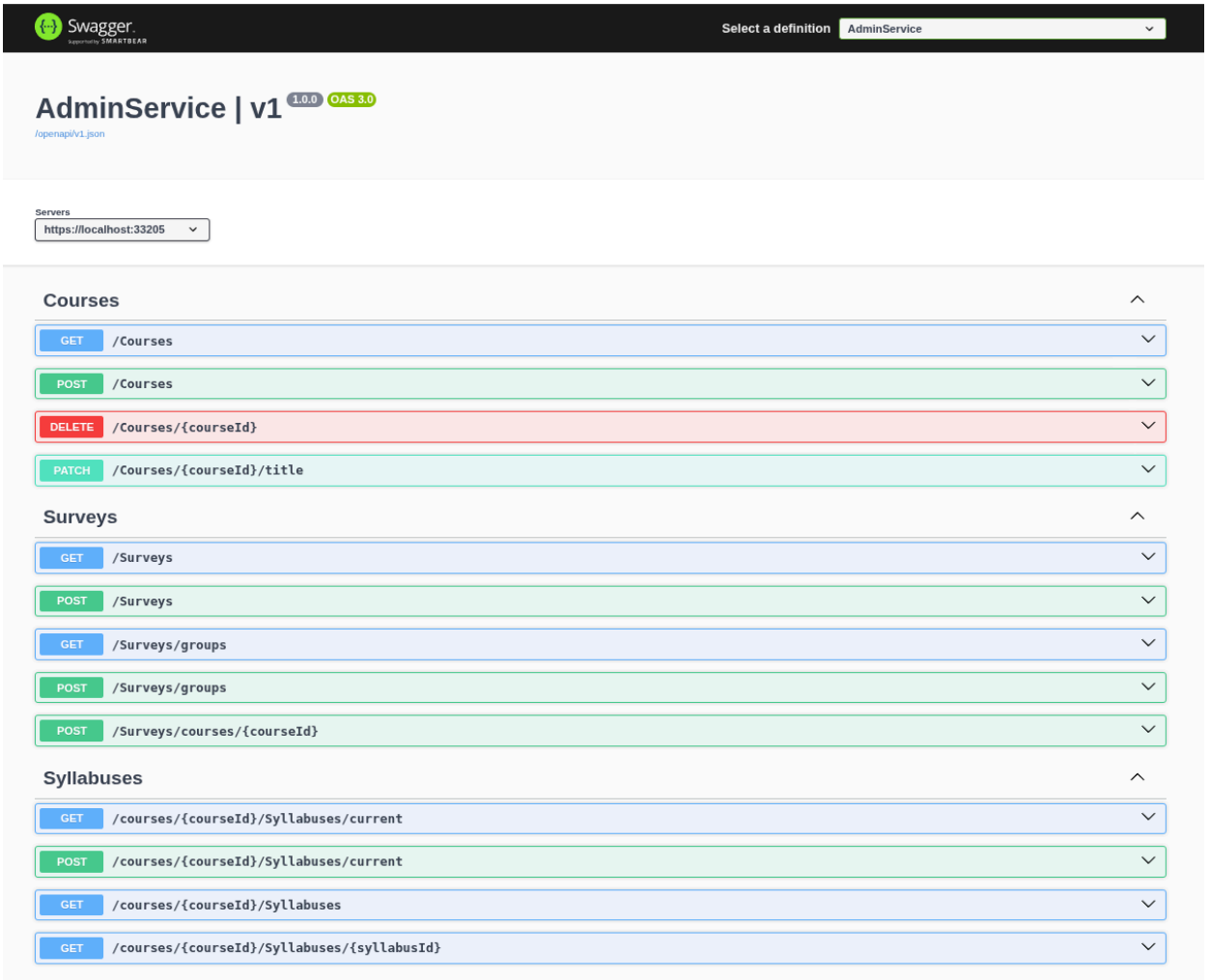


Рис. 4.3. Документація для керування створення та керування опитуваннями у SwaggerUI

4.3. Розробка сервісу для проходження оцінювання навчального навантаження

Сервіс для проходження оцінювання відповідає за взаємодію зі студентами під час процесу оцінювання навчального навантаження. Його основна функція – надання студентам доступу до активних опитувань, а також збереження та передача результатів для подальшого аналізу. Відповідний код сервісу зображено на рисунку 4.5.. Окрім цього, сервіс має гарантувати автентифікацію користувачів.

Сервіс для проходження оцінювання навчального навантаження визначає наступні ендпоінти:

- GET /Groups/{groupId} – завантажити групу опитувань;
- GET /Surveys/{surveyId} – завантажити опитування;
- POST /Surveys/{surveyId} – відправити оцінювання. Відповідний код можна побачити на рисунку 4.4.

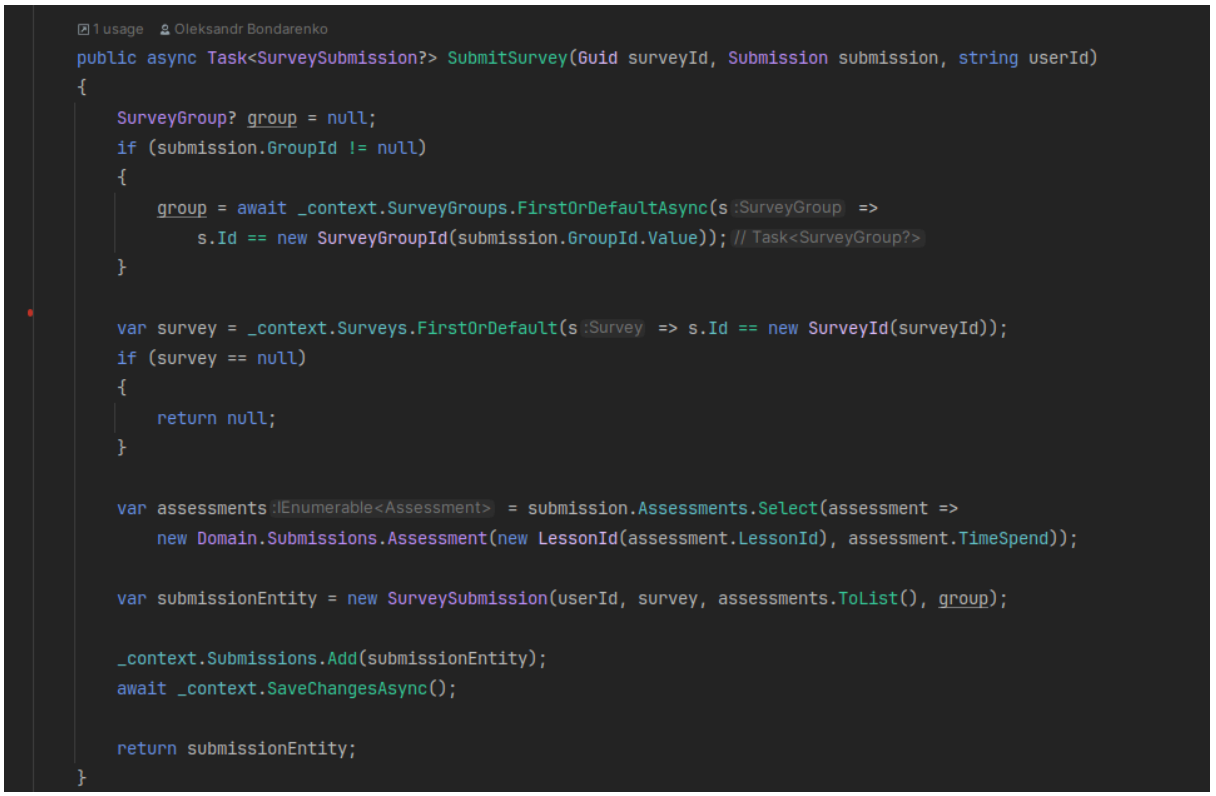
```
[HttpPost]
[Route( template: "")]
[ProducesResponseType( statusCode: StatusCodes.Status200OK)]
[ProducesResponseType( statusCode: StatusCodes.Status404NotFound)]
[ProducesResponseType( statusCode: StatusCodes.Status400BadRequest)]
Oleksandr Bondarenko
public async Task<IActionResult> SubmitSurvey(Guid surveyId, [FromBody] Submission submission)
{
    var userId :string? = await _auth.GetParticipantIdAsync();

    if (userId == null)
    {
        return Unauthorized();
    }

    var submissionResult :bool = await _surveys.SubmitSurvey(surveyId, submission);
    if (!submissionResult)
    {
        return NotFound();
    }

    return Ok();
}
```

Рис. 4.4. Код ендпоінта для відправлення результатів оцінювання



```
1 usage  Oleksandr Bondarenko
public async Task<SurveySubmission?> SubmitSurvey(Guid surveyId, Submission submission, string userId)
{
    SurveyGroup? group = null;
    if (submission.GroupId != null)
    {
        group = await _context.SurveyGroups.FirstOrDefaultAsync(s :SurveyGroup =>
            s.Id == new SurveyGroupId(submission.GroupId.Value)); // Task<SurveyGroup?>
    }

    var survey = _context.Surveys.FirstOrDefault(s :Survey => s.Id == new SurveyId(surveyId));
    if (survey == null)
    {
        return null;
    }

    var assessments :IEnumerable<Assessment> = submission.Assessments.Select(assessment =>
        new Domain.Submissions.Assessment(new LessonId(assessment.LessonId), assessment.TimeSpend));

    var submissionEntity = new SurveySubmission(userId, survey, assessments.ToList(), group);

    _context.Submissions.Add(submissionEntity);
    await _context.SaveChangesAsync();

    return submissionEntity;
}
```

Рис. 4.5. Код сервісу для збереження оцінювання студента

4.4. Розробка сервісу для моніторингу навчального навантаження

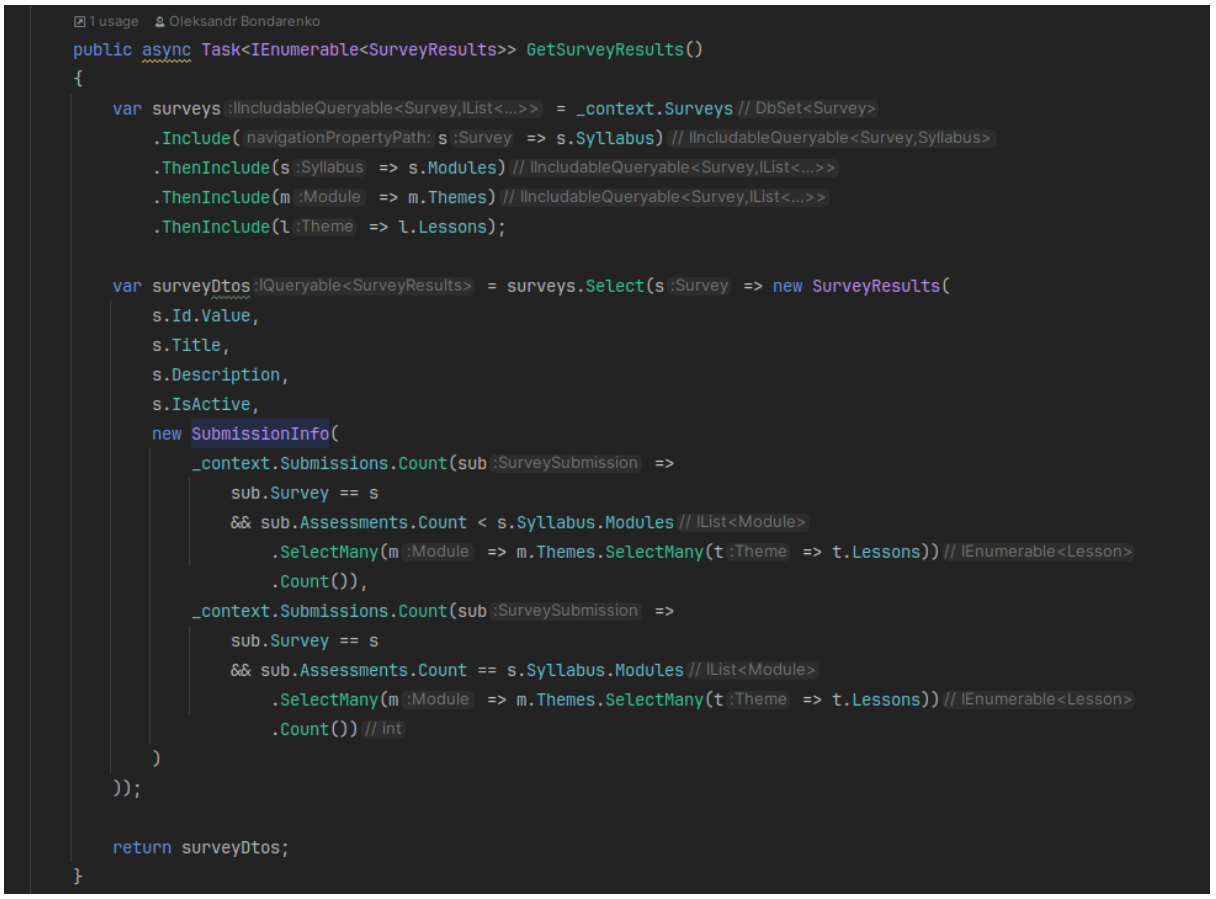
Сервіс для моніторингу навчального навантаження відповідає за обробку, аналіз і візуалізацію результатів оцінювання навчального навантаження студентів. Цей сервіс забезпечує адміністрації навчального закладу аналітичні інструменти для прийняття рішень, виявлення потенційних проблем і контролю якості освітнього процесу.

Він визначає наступні ендпоінти:

- GET /surveys/{surveyId}/statistics/timeline – отримати статистику оцінювання у форматі часової лінії як на рисунку. 4.6;
- GET /surveys/groups/{surveyGroupId}/statistics/timeline – отримати статистику групи оцінювань у форматі часової лінії;

- GET /surveys/{surveyId}/statistics/categorization – отримати розподіл студентів за навантаженнями за результатами оцінювання;
- GET /surveys/groups/{surveyGroupId}/statistics/categorization – отримати розподіл студентів за навантаженнями за результатами групи оцінювань;
- GET /surveys/results – отримати загальну інформацію про прогрес активних оцінювань. код відповідного ендпоінта зображено на рисунку. 4.7.;
- GET /surveys/groups/results – отримати загальну інформацію про прогрес активних груп оцінювань;

Рис. 4.6. Виконання ендпоінту отримання результати оцінювання у Swagger



```
1 usage  Oleksandr Bondarenko
public async Task<IEnumerable<SurveyResults>> GetSurveyResults()
{
    var surveys :IIncludableQueryable<Survey,IList<...>> = _context.Surveys // DbSet<Survey>
        .Include( navigationPropertyPath: s :Survey => s.Syllabus) // IIncludableQueryable<Survey,Syllabus>
        .ThenInclude(s :Syllabus => s.Modules) // IIncludableQueryable<Survey,IList<...>>
        .ThenInclude(m :Module => m.Themes) // IIncludableQueryable<Survey,IList<...>>
        .ThenInclude(l :Theme => l.Lessons);

    var surveyDtos :IQueryable<SurveyResults> = surveys.Select(s :Survey => new SurveyResults(
        s.Id.Value,
        s.Title,
        s.Description,
        s.IsActive,
        new SubmissionInfo(
            _context.Submissions.Count(sub :SurveySubmission =>
                sub.Survey == s
                && sub.Assessments.Count < s.Syllabus.Modules // IList<Module>
                    .SelectMany(m :Module => m.Themes.SelectMany(t :Theme => t.Lessons)) // IEnumerable<Lesson>
                    .Count()),
            _context.Submissions.Count(sub :SurveySubmission =>
                sub.Survey == s
                && sub.Assessments.Count == s.Syllabus.Modules // IList<Module>
                    .SelectMany(m :Module => m.Themes.SelectMany(t :Theme => t.Lessons)) // IEnumerable<Lesson>
                    .Count()) // int
        )
    ));

    return surveyDtos;
}
```

Рис. 4.7. Код для завантаження результатів оцінювання

4.5. Розробка сервісу для імпортування навчальної програми дисципліни з документів

Сервіс для імпортування навчальних дисциплін відповідає за автоматичне витягування структурованої інформації з навчальних документів (наприклад, PDF, DOCX) та перетворення її у формат, придатний для подальшого використання в системі. За допомогою ШІ сервіс розпізнає ключові елементи документа, та формує з них навчальний план. Ендпоінти сервісу розроблено за допомогою Python та FlaskApi, відповідний код можна побачити на рисунку 4.8.

Сервіс імпортування навчальних дисциплін визначає наступні ендпоінти:

- POST /documents – завантажити новий документ на обробку;

- GET /documents – отримати список всіх документів та їх статусів, як на рисунку 4.9;
- GET /documents/<doc_id>/status – отримати статус конкретного документа;
- GET /documents/<doc_id> – отримати оброблені дані конкретного документа.

```python
from flask import Blueprint, request, jsonify
from app.services.document_service import (
    upload_file,
    list_documents,
    get_status,
    get_processed_data
)

documents_bp = Blueprint('documents_bp', __name__)

@documents_bp.route('/documents', methods=['POST'])
def upload_document():
    if 'file' not in request.files or request.files['file'].filename == '':
        return jsonify({'error': 'No file selected'}), 400

    file = request.files['file']
    doc_info = upload_file(file)
    return jsonify(doc_info), 202

@documents_bp.route('/documents', methods=['GET'])
def list_all_documents():
    return jsonify(list_documents())

@documents_bp.route('/documents/<doc_id>/status', methods=['GET'])
def document_status(doc_id):
    result = get_status(doc_id)
    return (jsonify(result), 404) if 'error' in result else jsonify(result)

@documents_bp.route('/documents/<doc_id>', methods=['GET'])
def document_data(doc_id):
    result = get_processed_data(doc_id)
    if 'error' in result:
        return jsonify(result), 404 if result['error'] == 'Document not found' else 409
    return jsonify(result)
```

Рис. 4.8. Ендпоінти для імпортування даних з дисциплін

```
[
  {
    "documentId": "d1b97ca5-f49b-4c3a-8f71-785b4c980d74",
    "originalFilename": "AI.pdf",
    "status": "processing",
    "timestamp": "Sun, 25 May 2025 10:04:17 GMT"
  },
  {
    "documentId": "71f987c7-b995-4cfa-bb0f-ac095bb34028",
    "originalFilename": "English.pdf",
    "status": "processed",
    "timestamp": "Sun, 25 May 2025 10:04:05 GMT"
  },
  {
    "documentId": "7073af4b-c055-479b-91f6-91cf9e1a6106",
    "originalFilename": "Databases.pdf",
    "status": "processed",
    "timestamp": "Sun, 25 May 2025 10:03:06 GMT"
  }
]
```


Рис. 4.9. Отримання списку імпортованих документів

Цей сервіс обробляє інформацію з навчальних силабусів за допомогою великої мовної моделі Gemini, трансформуючи її у структурований формат JSON. Для здійснення цього процесу він надсилає запит до моделі Gemini 2.0-flash, яка аналізує текстовий вміст документа. Код звернення до моделі зображено на рисунку 4.10.

Модель аналізує документ і повертає структуровану відповідь у форматі, що відповідає схемі Syllabus. Далі результат автоматично перетворюється в об'єкт Python, що потім зберігається в окремому документі MongoDB. Згодом, результати обробки повертаються клієнту за запитом до відповідного ендпоінту.

```
system_prompt = """
    You are an intelligent parser specialized in analyzing academic course syllabi.
    Your task is to extract and structure key educational components from the provided syllabus text.
    Do not translate any text and use language used in the document.
    Identify and extract the main sections relevant to course organization, including (but not limited to):
        Course content or description
        Curriculum or learning plan
        Modular breakdown (if available)
    Analyze the extracted information and reconstruct a structured outline of the course. This outline should include:
        Content Modules (main sections or thematic blocks)
        Topics within each module
        Lessons or sessions under each topic (if detailed)
    Do not translate any text and use language used in the document.
    Output a clean, organized course structure in a hierarchical format.
    Use language of the document, do not translate anything.
    """

def parse_syllabus(document_text: str) -> Syllabus:
    user_request = document_text

    response = client.models.generate_content(
        model='gemini-2.0-flash',
        contents=[system_prompt, user_request],
        config={
            'response_mime_type': 'application/json',
            'response_schema': Syllabus,
        }
    )

    return Syllabus.model_validate_json(response.text)
```

Рис. 4.10. Код звернення до мовної моделі Gemini 2.0-flash

4.6. Оркестрування сервісів та інфраструктури

Оркестрування сервісів та інфраструктури за допомогою Aspire відповідає за централізоване керування запуском, налаштуванням і взаємодією між мікросервісами, які складають систему.

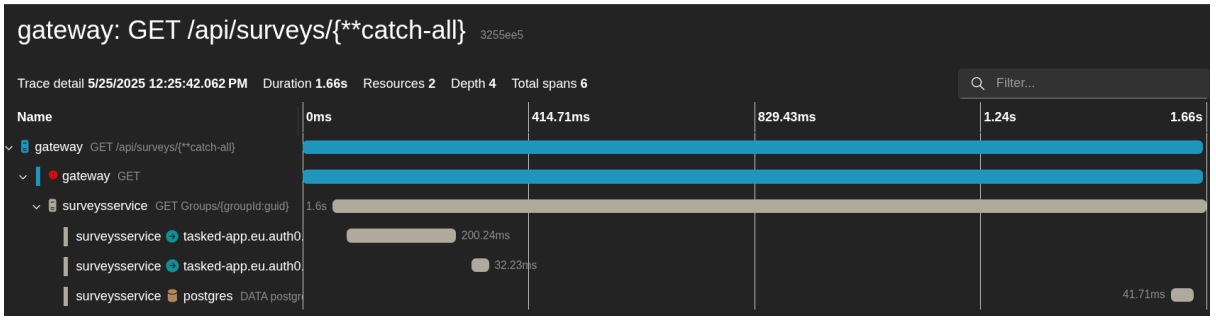


Рис. 4.11. Спостереження за запитом до сервісу за допомогою телеметрії та логів

Зокрема, Aspire автоматизує розгортання середовища розробки й тестування та забезпечує коректне підключення сервісів, як на рисунку 4.12., управління змінними середовища та секретами, а також надає спостережуваність – монітор здоров’я сервісів, телеметрію та логування, що видно на рисунку 4.11.

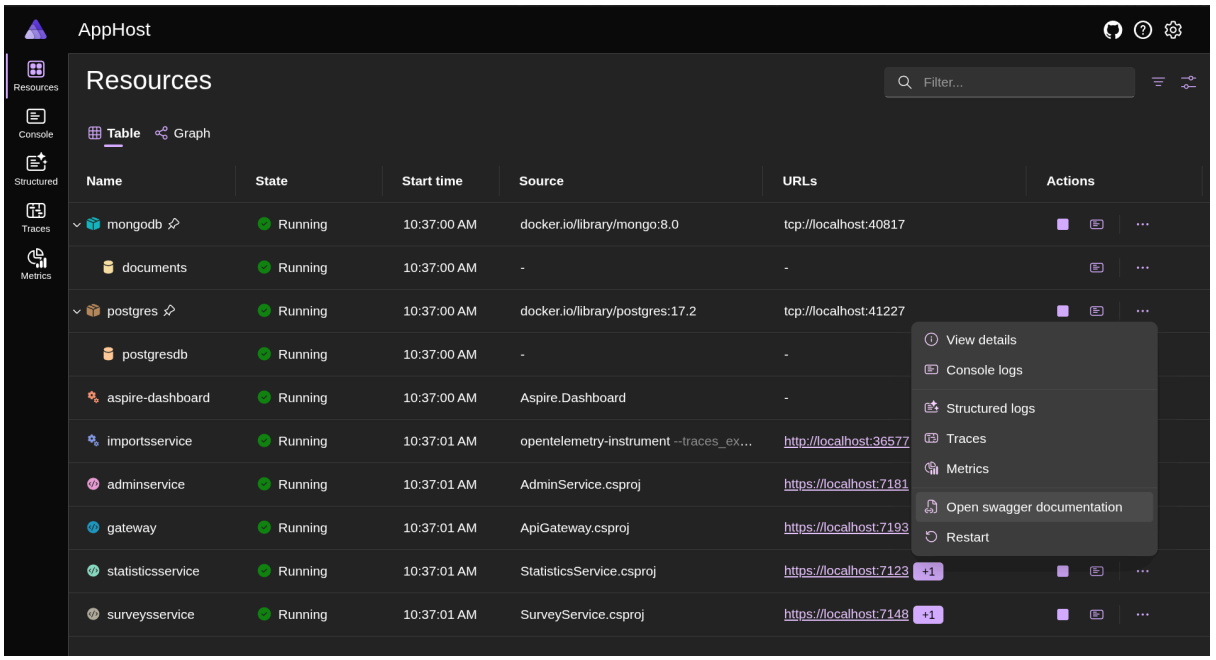


Рис. 4.12. Огляд розгорнутих ресурсів у Aspire Dashboard

4.7. Розробка бібліотек для клієнтів сервісів

Для спрощення взаємодії з сервісами були розроблені бібліотеки клієнт, створена з допомогою Refit[21] – бібліотеки для .NET, яка автоматично генерує HTTP-клієнт на основі визначених інтерфейсів. Приклад коду, що визначає ендпоінти та типи даних що приймають та повертаю ці ендпоінти, зображено на рисунку 4.13.

```
public interface ICoursesClient
{
    [Get( path: "/courses/")]
    1 usage  2 implementations  Oleksandr Bondarenko
    public Task<IList<Course>> GetCoursesAsync();

    [Post( path: "/courses/")]
    1 usage  2 implementations  Oleksandr Bondarenko
    public Task<Course> CreateCourseAsync(CreateCourseRequest request);

    [Delete( path: "/courses/{courseId}")]
    1 usage  2 implementations  Oleksandr Bondarenko
    public Task DeleteCourseAsync(Guid courseId);

    [Patch( path: "/courses/{courseId}/title")]
    1 usage  2 implementations  Oleksandr Bondarenko
    public Task<Course> UpdateCourse(Guid courseId, [Body] UpdateCourseTitleRequest request);
}
```

Рис. 4.13. Код розробленого клієнта до сервісу адміністрації для керування дисциплінами

Використання клієнтських бібліотек дозволяють розробникам клієнтських застосунків отримати суворо типізований клієнт, що визначатиме взаємодію з API, наприклад, як на рисунку 4.14 . Крім того, коли контракт змінюється, його можна централізовано оновити у бібліотеці-клієнті, і всі споживачі автоматично отримають актуальну реалізацію оновлювати клієнт вручну.

```
[RelayCommand]
2 usages  Ridcovets Sergo *
private async Task CreateCourse()
{
    var request = new CreateCourseRequest(CourseTitle);

    await _adminClient.CreateCourseAsync(request);
    await Shell.Current.GoToAsync( state: "..");
}
```

Рис. 4.14. Використання бібліотеки сервісу адміністрації в коді клієнта

В якості централізованого сховища для бібліотек клієнтів використовувався GitHub Packages. Це дозволило оперативно зберігати, оновлювати та завантажувати бібліотеки з єдиного репозиторію, завдяки інтеграції з менеджером пакетів C# nuget. Розроблені пакети можна побачити на малюнку. 4.15

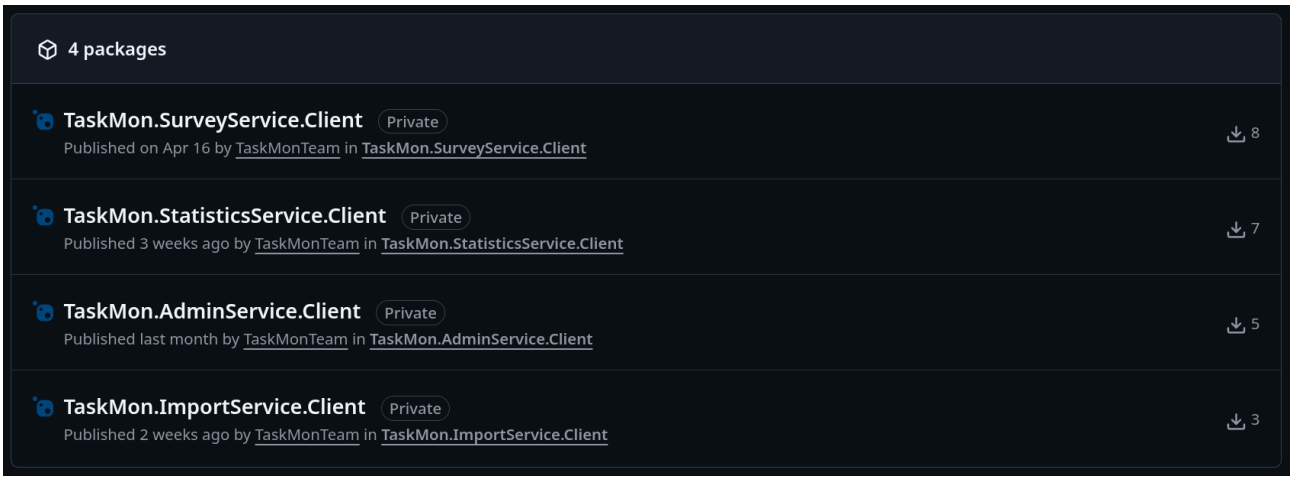


Рис. 4.15. Пакети клієнтських бібліотек в репозиторії GitHub

4.8. Тестування застосунку

Тестування застосунку проводилось переважно на рівні функціонального тестування клієнтського застосунку, що дозволило перевірити типові сценарії використання та взаємодію з API у реальному середовищі. Таке тестування допомогло виявити помилки в логіці, перевірити поведінку при помилках мережі або некоректних даних, а також переконатися, що усі сервіси працюють злагоджено через шлюз Api.

```
[Fact]
Oleksandr Bondarenko
async Task GetSurveyAsync_ReturnsValidResponse()
{
    var surveyId = Guid.NewGuid();

    var result :Survey  = await SurveyClient.GetSurveyAsync(surveyId);

    Assert.NotNull(result);
}
```

Рис. 4.16. Код інтеграційного тесту сервісу для проходження оцінювань

Крім того, для забезпечення стабільності інтеграції було розроблено димні тести для бібліотек-клієнтів. Вони дозволяють автоматично перевірити, що ключові запити до кожного API виконуються успішно після змін у коді – наприклад, після оновлення маршрутів, DTO чи логіки авторизації. Це допомагає швидко виявити критичні проблеми, які могли б зламати інтеграцію, і гарантує, що клієнтський застосунок залишиться працездатним після оновлень. Приклад коду та виконання інтеграційних тестів можна побачити на малюнках 4.16 та 4.17 відповідно.

Для перевірки коректності конфігурації бази даних та її сумісності зі схемою, визначеною в коді, був реалізований окремий тест CanApplyMigrations у проєкті Persistence.Test. Цей тест автоматично застосовує всі міграції до тестової бази даних, що дозволяє переконатися що схема бази даних налаштована коректно. Результат виконання тесту видно на рисунку 4.17

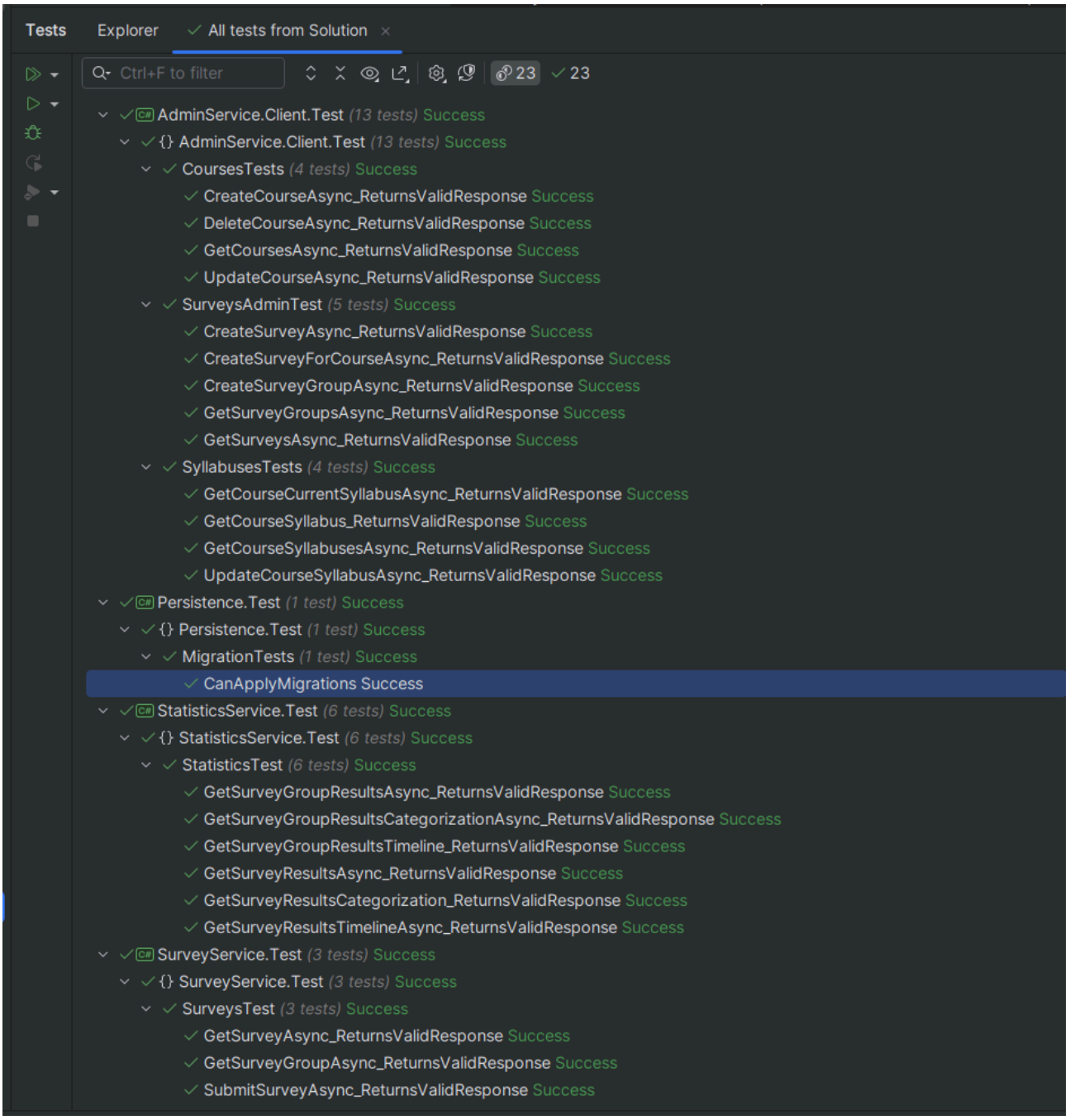


Рис. 4.17. Результат виконання автоматизованих тестів

ВИСНОВКИ

Під час роботи над кваліфікаційною роботою було проаналізовано методи проведення моніторингу навчального навантаження. Ознайомлено з сучасними методами проєктування серверних застосунків. Розроблено застосунок для проведення оцінки та моніторингу навчального навантаження студентів, що враховує недоліки існуючих засобів та особливості предметної області.

1. У ході виконання кваліфікаційної роботи було проаналізовано методи моніторингу навчального навантаження. Огляд існуючих програмних рішень показав, що на ринку відсутні прямі аналоги розроблюваного застосунку;

2. Сформульовано як функціональні та нефункціональні вимоги до застосунку. Особливу увагу приділено таким аспектам як, безпека та автоматизація процесу;

3. Спроєктовано архітектуру серверної частини застосунку, орієнтовану на модульність, розширюваність. Використано сучасні принципи розробки програмного забезпечення, зокрема принципи мікросервісної та чистої архітектури;

4. Реалізовано серверну частину застосунку, яка забезпечує основні функції: створення та редагування навчальних планів, створення та проведення оцінювань навчального навантаження з відповідних навчальних планів, генерація статистики з результатів оцінювань та можливість імпортування навчальних планів з документів силабусів;

5. Проведено розгортання застосунку у тестовому середовищі. У результаті тестування встановлено, що система функціонує коректно. Результати розробки мають потенціал для подальшого використання у навчальних закладах з метою підвищення ефективності організації навчального процесу.

Кваліфікаційна робота пройшла апробацію на двох конференціях та було опубліковано наступні дві тези:

1. Бондаренко О.О., Замрій І.В. Обробка документів засобами мови python та великої мовної моделі gemma 3. VI Міжнародна науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій (подано до друку).

2. Бондаренко О.О., Замрій І.В. Використання хмарної платформи для управління обліковими даними Auth0 для забезпечення безпечної аутентифікації користувачів. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С. 426-429.

ПЕРЕЛІК ПОСИЛАНЬ

1. Курінний О. В. Особливості застосування кредитно-модульної системи у ЗВО України. / Класичний приватний університет. – URL: https://scholar.google.com/scholar?cluster=7025332659070830183&hl=ukr&oi=scholarr.

2. Кириленко О. І. Визначення навчального навантаження студентів за допомогою план-форм. *Збірник наукових праць Міжнародної науково-практичної конференції «Сучасна освіта та наука: проблеми, перспективи, інновації»*.

3. Google Workspace. Google Forms. URL: https://workspace.google.com/products/forms.

4. SurveyMonkey. URL: https://www.surveymonkey.com.

5. Moodle. URL: https://moodle.com.

6. Instructure. Canvas. URL: https://www.instructure.com/canvas.

7. Learn Microsoft. A tour of the C# language. – URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview.

8. Python. URL: https://www.python.org/doc.

9. Auth0. URL: https://auth0.com.

10. Learn Microsoft. .NET Aspire overview. – URL: https://learn.microsoft.com/en-us/dotnet/aspire/get-started/aspire-overview.

11. Microservices from Theory to Practice. Creating Applications in IBM Bluemix Using the Microservices Approach / Shahir D. та ін.

12. Busi S. Understanding Microservices Architecture: A Comprehensive Guide. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*. T. 11.

13. Newman S. Building Microservices. Designing fine-grained system.

14. Martin R. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. ISBN 978‑0134494166.

15. Medium. Thoughts on Clean Architecture. URL: https://medium.com/android-news/thoughts-on-clean-architecture-b8449d9d02df.

16. Esposito D. Clean Architecture with .NET. Microsoft Corporation.

17. Siriwardena P. OAuth 2.0 Security. *Advanced API Security* / P. Siriwardena. URL: http://dx.doi.org/10.1007/978-1-4842-2050-4_14.

18. Teleport. How OAuth 2.0 Works. URL: https://goteleport.com/blog/how-oauth-authentication-works.

19. Watson M. Ollama in Action: Building Safe, Private AI.

20. Introducing Gemma 3: The most capable model you can run on a single GPU or TPU. URL: https://blog.google/technology/developers/gemma-3.

21. Kılıçarslan S. Using Refit in .NET. Medium. 01.01.2024. URL: https://medium.com/net-core/using-refit-in-net-0843bb199987.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка застосунку для моніторингу навчального навантаження студентів. Спец-частина: Розробка Backend-частини застосунку

Виконав студент 4 курсу
групи ПД-41
Бондаренко Олександр Олександрович
Керівник роботи
д.т.н., проф., завідувач кафедри ІПЗ Замрій Ірина Вікторівна

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи**: спрощення моніторингу навчального навантаження студентів на основі використання програмного забезпечення для вирішення проблеми збору даних про навантаженість студентів.
- **Об'єкт дослідження:** процес збору даних та моніторингу навчального навантаження студентів у закладах вищої освіти.
- **Предмет дослідження:** програмне забезпечення для оцінювання та моніторингу навчального навантаження студентів.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати існуючі рішення для моніторингу навчального навантаження студентів.
2. Визначити функціональні та нефункціональні вимоги до застосунку.
3. Спроєктувати архітектуру серверної частини застосунку
4. Розробити серверну частину застосунку.
5. Розгорнути застосунок у тестовому режимі.

3

АНАЛІЗ АНАЛОГІВ

Характеристика	**Google forms**	**SurveyMonkey**	**Moodle**	**Canvas**	**TaskMon**
Форма збору даних про навантаженість студентів	Опитування (оцінювання) студентів	Опитування (оцінювання) студентів	На основі взаємодії з курсом	На основі взаємодії з курсом та відвідуваності	Оцінювання студентів
Формування питань оцінювання	Вручну	Вручну, за допомогою ШІ	-	-	З навчального плану
Аналітика результатів	Прості діаграми	Прості діаграми	Гістограма та прості діаграми	Гістограма та прості діаграми	Динаміка навантаженості
Експорт аналітики	+	+	-	-	+
Обмеження	Необхідно мати електронну пошту Gmail	-	Використання Moodle в організації навчання	Використання Canvas в організації навчання	-

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Внесення даних про програму навчальної дисципліни.
2. Проведення оцінки реального навантаження студентів.
3. Спостереження за динамікою навчального навантаження у групі студентів та порівняння з очікуваним навантаженням.
4. Імпортування програми дисциплін з документів.

Нефункціональні вимоги:

1. Захист від несанкціонованого доступу до адміністративної частини застосунку.
2. Забезпечення анонімності відповідей студентів.

5

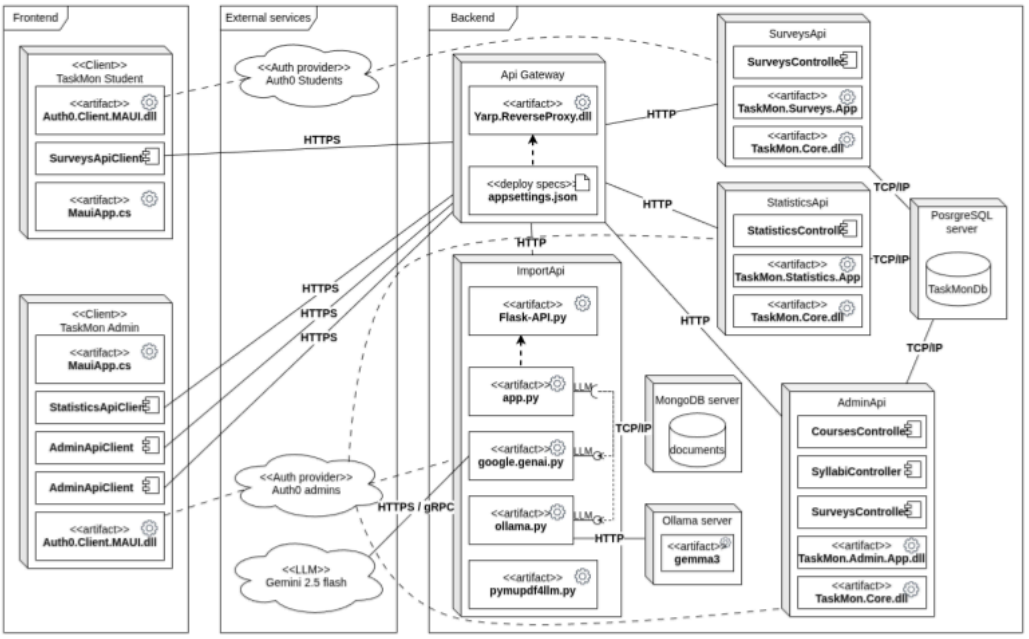
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

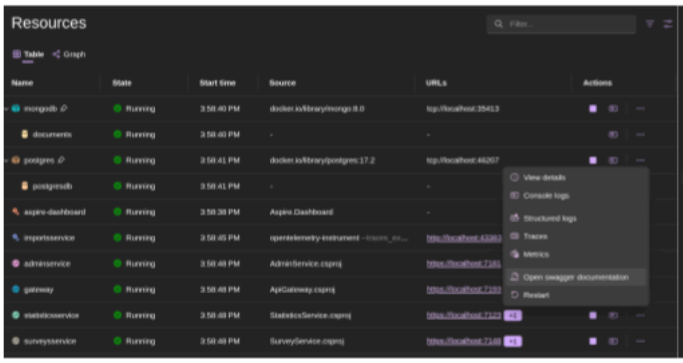
7

ДІАГРАМА РОЗГОРТАННЯ



8

ЕКРАННІ ФОРМИ

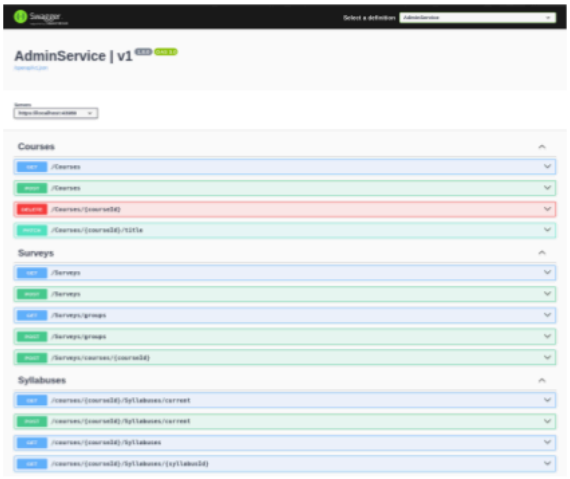


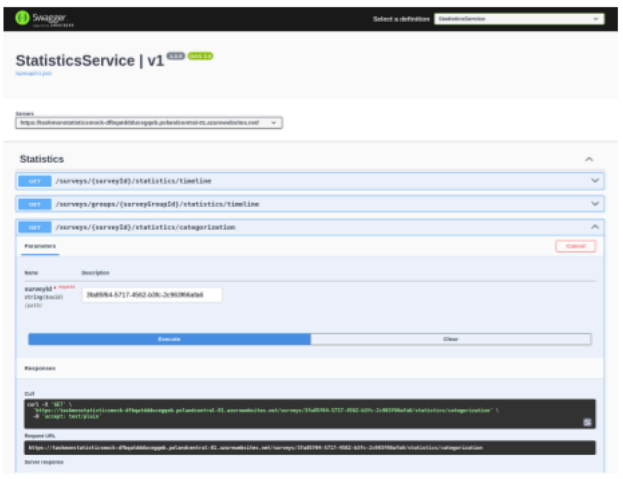
Розгорнуті сервіси в панелі ресурсів aspire

Зв’язок сервісів у панелі ресурсів aspire

9

ЕКРАННІ ФОРМИ





Приклад сторінок документації у Swagger UI

10

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Бондаренко О.О., Замрій І.В. Обробка документів засобами мови python та великої мовної моделі gemma 3. VI Міжнародна науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій (подано до друку).
2. Бондаренко О.О., Замрій І.В. Використання хмарної платформи для управління обліковими даними Auth0 для забезпечення безпечної аутентифікації користувачів. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С. 426-429.

ВИСНОВКИ

1. Проаналізовано існуючі рішення для моніторингу навчального навантаження. Було виявлено що додаток немає прямих аналогів, а існуючі рішення не враховують специфіку предметної області.
2. Розроблено функціональні та нефункціональні вимоги до застосунку, з урахуванням специфіки предметної галузі.
3. Спроєктовано архітектуру серверної частини застосунку, що дозволяє задовольнити всі визначені вимоги, та може бути легко розширена у майбутньому.
4. Розроблено серверну частину застосунку що дозволяє створювати та проводити оцінювання навчального навантаження студентів, та проводити моніторинг навчального навантаження.
5. Застосунок розгорнуто у тестовому режимі. Було визначено, що застосунок відповідає розробленим вимогам.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
using AppHost;

using Projects;

var builder = DistributedApplication.CreateBuilder(args);

var postgres = builder.AddPostgres("postgres")
   .WithLifetime(ContainerLifetime.Persistent);
var postgresdb = postgres.AddDatabase("postgresdb");

var mongo = builder.AddMongoDB("mongodb")
   .WithLifetime(ContainerLifetime.Persistent);
var mongodb = mongo.AddDatabase("documents");

var surveyService = builder.AddProject<SurveyService>("surveysservice")
   .WaitFor(postgresdb)
   .WithReference(postgresdb)
   .WithSwaggerDocumentation();

var adminService = builder.AddProject<AdminService>("adminservice")
   .WaitFor(postgresdb)
   .WithReference(postgresdb)
   .WithSwaggerDocumentation();

var statisticsService = builder.AddProject<StatisticsService>("statisticsservice")
   .WaitFor(postgresdb)
   .WithReference(postgresdb)
   .WithSwaggerDocumentation();

// Please, set up ImportsService:GeminiKey in user secrets to use importsService
var geminiKey = builder.Configuration
   .GetSection("ImportsService")
   .GetSection("GeminiKey").Value;

#pragma warning disable ASPIREHOSTINGPYTHON001
var importService = builder
                  .AddPythonApp("importsservice", "../../Services/ImportService/ImportService.FlaskApi", "run.py")
   .WithHttpEndpoint(env: "PORT")
   .WithReference(mongodb)
   .WaitFor(mongodb)
   .WithExternalHttpEndpoints()
   .WithOtlpExporter()
   .WithEnvironment("USE_TRACES", "True")
            .WithEnvironment("GEMINI_API_KEY", geminiKey);
#pragma warning restore ASPIREHOSTINGPYTHON001

var gateway = builder.AddProject<ApiGateway>("gateway")
   .WithReference(surveyService)
   .WithReference(adminService)
   .WithReference(statisticsService)
   .WithReference(importService)
   .WaitFor(surveyService)
   .WaitFor(adminService)
   .WaitFor(statisticsService)
   .WaitFor(importService)
   .WithExternalHttpEndpoints();

builder.Build().Run();

using Microsoft.AspNetCore.HttpOverrides;

var builder = WebApplication.CreateBuilder(args);


builder.Services.AddOpenApi();
builder.AddServiceDefaults();
builder.Services
   .AddReverseProxy()

.LoadFromConfig(builder.Configuration.GetSection("ReverseProxy"))
   .AddServiceDiscoveryDestinationResolver();

builder.Services.Configure<ForwardedHeadersOptions>(options =>
{
                  options.ForwardedHeaders = ForwardedHeaders.XForwardedFor
                                             | ForwardedHeaders.XForwardedProto
                                             | ForwardedHeaders.XForwardedHost;
});

var app = builder.Build();

app.UseForwardedHeaders(new ForwardedHeadersOptions
{
   RequireHeaderSymmetry = false,
                        ForwardedHeaders = ForwardedHeaders.XForwardedFor
                  | ForwardedHeaders.XForwardedProto
                  | ForwardedHeaders.XForwardedHost
});

app.MapReverseProxy();

app.Run();
 namespace Domain.Syllabus;

[StronglyTypedId]
```

```
public partial struct LessonId;

public class Lesson
{
   public LessonId Id { get; } = LessonId.New();
   public string Title { get; set; }
   public LessonType Type { get; set; }
   public float? StudyHours { get; set; }

    public Lesson(string title, LessonType type, float?
studyHours)
   {
      Title = title;
      Type = type;
      StudyHours = studyHours;
   }

   private Lesson() { }
}
namespace Domain.Syllabus;

public enum LessonType
{
   Lecture,
   Class,
   Practice,
   Laboratory,
   Seminar,
   Tutorial,
   Exam
}

namespace Domain.Syllabus;

[StronglyTypedId]
public partial struct ModuleId;

public class Module
{
   public ModuleId Id { get; } = ModuleId.New();
   public string Title { get; set; }
   public IList<Theme> Themes { get; set; }

   public Module(string title, IList<Theme> themes)
   {
      Title = title;
      Themes = themes;
   }

   private Module() { }
}
namespace Domain.Syllabus;

[StronglyTypedId]
public partial struct SyllabusId;

public class Syllabus
{
   public SyllabusId Id { get; } = SyllabusId.New();
   public string Title { get; set; }
        public DateTime PublishTime { get; } =
DateTime.Now;
   public DateTime? ArchiveTime { get; private set; }
   public IList<Module> Modules { get; }

   public Syllabus(string title, IList<Module> modules)
   {
      Title = title;
      Modules = modules;
   }

   private Syllabus() { }
}

namespace Domain.Syllabus;

[StronglyTypedId]
public partial struct ThemeId;

public class Theme
{
   public ThemeId Id { get; } = ThemeId.New();
   public string Title { get; set; }
   public IList<Lesson> Lessons { get; }

   public Theme(string title, IList<Lesson> lessons)
   {
      Title = title;
      Lessons = lessons;
   }

   private Theme() { }
}
namespace Domain.Survey;

[StronglyTypedId]
public partial struct SurveyId;

public class Survey
{
   public SurveyId Id { get; } = SurveyId.New();
   public string Title { get; set; }
   public string? Description { get; }
        public DateTime PublishDate { get; } =
DateTime.UtcNow;
   public bool IsActive { get; private set; }
   public Syllabus.Syllabus Syllabus { get; }

       public Survey(string title, string? description,
Syllabus.Syllabus syllabus)
   {
      Title = title;
      Description = description;
      Syllabus = syllabus;
   }

   private Survey() { }
}
using Domain.Syllabus;

namespace Domain.Submissions;

public class Assessment
{
   public LessonId LessonId { get; init; }
   public float Value { get; init; }
```

```
    public Assessment(LessonId lessonId, float value)
    {
        LessonId = lessonId;
        Value = value;
    }

    private Assessment() { }
}
using Domain.Abstractions.Helpers;
using Domain.Survey;

namespace Domain.Submissions;

[StronglyTypedId]
public partial struct SurveySubmissionId;

public class SurveySubmission
{
        public SurveySubmissionId Id { get; } =
SurveySubmissionId.New();
    public Survey.Survey Survey { get; }
    public string ParticipantHash { get; }
       public DateTime SubmissionDate { get; } =
DateTime.UtcNow;
    public IList<Assessment> Assessments { get; init; }
    public SurveyGroup? GroupContext { get; init; }

        public SurveySubmission(string userIdentifier,
Survey.Survey survey, IList<Assessment> assessments,
       SurveyGroup? groupContext = null)
    {
                                    ParticipantHash =
EncryptionHelper.Encrypt(userIdentifier);
        Survey = survey;
        Assessments = assessments;
        GroupContext = groupContext;
    }

    private SurveySubmission() { }
}
namespace Domain.Institution;

[StronglyTypedId]
public partial struct CourseId;

public class Course
{
    public CourseId Id { get; } = CourseId.New();
    public string Title { get; set; }
      public Syllabus.Syllabus? ActiveSyllabus { get; /*
protected */ set; }
    public IList<Syllabus.Syllabus> Syllabi { get; }

    public Course(string title)
    {
        Title = title;
        Syllabi = new List<Syllabus.Syllabus>();
    }

    private Course() { }
}
using Domain.Survey;

namespace Domain.Institution;

[StronglyTypedId]
public partial struct InstitutionId;

public class Institution
{
    public InstitutionId Id { get; } = new InstitutionId();
    public string Name { get; set; }
        public IList<Course> Courses { get; } = new
List<Course>();
     public IList<Survey.Survey> Surveys { get; } = new
List<Survey.Survey>();
    public IList<SurveyGroup> SurveyGroups { get; } =
new List<SurveyGroup>();

    public Institution(string name)
    {
        Name = name;
    }

    private Institution() { }
}
using System.Security.Cryptography;
using System.Text;

namespace Domain.Abstractions.Helpers;

public static class EncryptionHelper
{
    public static string Encrypt(string plainText)
    {
                                        byte[] bytes =
SHA256.HashData(Encoding.UTF8.GetBytes(plainTe
xt));
        StringBuilder builder = new StringBuilder();
        foreach (byte b in bytes)
        {
            builder.Append(b.ToString("x2"));
        }

        return builder.ToString();
    }
}
using AdminService.App;

using Application.Data;

using Microsoft.EntityFrameworkCore;

using Persistence;

using Scalar.AspNetCore;

var builder = WebApplication.CreateBuilder(args);

builder.AddServiceDefaults();

builder.Services.AddOpenApi();
builder.Services.AddControllers();
```

```
var connectionString = builder.Configuration.GetConnectionString("postgresdb");
builder.Services.AddDbContext<ApplicationContext>(o => o.UseNpgsql(connectionString));
builder.Services.AddTransient<IApplicationContext>(s => s.GetRequiredService<ApplicationContext>());

builder.Services.AddScoped<ICoursesService, CoursesService>();
builder.Services.AddScoped<ISurveysService, SurveysService>();
builder.Services.AddScoped<ISyllabiService, SyllabiService>();

var app = builder.Build();



app.MapOpenApi();
app.UseSwaggerUI(options => options.SwaggerEndpoint("/openapi/v1.json", "AdminService"));
app.MapScalarApiReference();

app.UseHttpsRedirection();

app.MapControllers();

app.Run();

public abstract partial class Program { }
using AdminService.App;
using AdminService.Client.Requests;
using Microsoft.AspNetCore.Mvc;
using Course = AdminService.Models.Course;

namespace AdminService.Controllers;

[ApiController]
[Route("/[controller]")]
public class CoursesController : ControllerBase
{
    private readonly ICoursesService _courses;

    public CoursesController(ICoursesService courses)
    {
        _courses = courses;
    }

    [HttpGet]
        [ProducesResponseType(typeof(IList<Course>), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]
    [Route("")]
    public IActionResult GetCoursesAsync()
    {
        var courses = _courses.GetCoursesAsync();
        return Ok(courses);
    }

    [HttpPost]
            [ProducesResponseType(typeof(Course), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]
    [Route("")]
            public async Task<IActionResult> CreateCourseAsync(CreateCourseRequest request)
    {
                var course = await _courses.CreateCourseAsync(request);
        return Ok(course);
    }

    [HttpDelete]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]
    [Route("{courseId:guid}")]
            public async Task<IActionResult> DeleteCourseAsync(Guid courseId)
    {
                var courseDeleted = await _courses.DeleteCourseAsync(courseId);
        if (!courseDeleted)
        {
            return NotFound();
        }
        return Ok();
    }

    [HttpPatch]
            [ProducesResponseType(typeof(Course), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]
    [Route("{courseId:guid}/title")]
            public async Task<IActionResult> UpdateCourse(Guid courseId, UpdateCourseTitleRequest request)
    {
                var course = await _courses.UpdateCourse(courseId, request);
        if (course == null)
        {
            return NotFound();
        }

        return Ok(course);
    }
}
using AdminService.App;
using AdminService.Client.Requests;
using Microsoft.AspNetCore.Mvc;

using Survey = AdminService.Models.Survey;
using SurveyGroup = AdminService.Models.SurveyGroup;
```

```
namespace AdminService.Controllers;

[ApiController]
[Route("/[controller]")]
public class SurveysController : ControllerBase
{
   private readonly ISurveysService _surveys;

   public SurveysController(ISurveysService surveys)
   {
      _surveys = surveys;
   }

   [HttpGet]
         [ProducesResponseType(typeof(IList<Survey>),
StatusCodes.Status200OK)]
   [Route("")]
   public IActionResult GetSurveysAsync()
   {
      var surveyDtos = _surveys.GetSurveysAsync();
      return Ok(surveyDtos);
   }

   [HttpGet]

[ProducesResponseType(typeof(IList<SurveyGroup>),
StatusCodes.Status200OK)]
   [Route("groups/")]
   public IActionResult GetSurveyGroupsAsync()
   {
      var dtos = _surveys.GetSurveyGroupsAsync();
      return Ok(dtos);
   }

   [HttpPost]
                [ProducesResponseType(typeof(Survey),
StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFo
und)]
   [Route("")]
                                public IActionResult
CreateSurveyGroupAsync([FromBody]
CreateSurveyRequest request)
   {
                                   var surveyDto =
_surveys.CreateSurveyGroupAsync(request);
      if (surveyDto == null)
      {
         return NotFound();
      }
      return Ok(surveyDto);
   }

   [HttpPost]
                [ProducesResponseType(typeof(Survey),
StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFo
und)]

[ProducesResponseType(StatusCodes.Status409Confli
ct)]
   [Route("courses/{courseId::guid}")]
                public async Task<IActionResult>
CreateSurveyForCourseAsync(Guid courseId,
         [FromBody] CreateSurveyForCourseRequest
request)
   {
                                var surveyDto = await
_surveys.CreateSurveyForCourseAsync(courseId,
request);
      if (surveyDto == null)
      {
         return NotFound();
      }
      return Ok(surveyDto);
   }

   [HttpPost]
         [ProducesResponseType(typeof(SurveyGroup),
StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFo
und)]
   [Route("groups/")]
                public async Task<IActionResult>
CreateSurveyGroupAsync([FromBody]
CreateSurveyGroupRequest request)
   {
                                var groupDto = await
_surveys.CreateSurveyGroupAsync(request);
      if (groupDto == null)
      {
         return NotFound();
      }
      return Ok(groupDto);
   }
}

using AdminService.App;
using AdminService.Client.Requests;
using AdminService.Models;
using Microsoft.AspNetCore.Mvc;
using Syllabus = AdminService.Models.Syllabus;

namespace AdminService.Controllers;

[ApiController]
[Route("/courses/{courseId:guid}/[controller]")]
public class SyllabusesController : ControllerBase
{
   private ISyllabiService _syllabi;

   public SyllabusesController(ISyllabiService syllabi)
   {
      _syllabi = syllabi;
   }

   [HttpGet]
               [ProducesResponseType(typeof(Syllabus),
StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFo
und)]
```

```
[ProducesResponseType(StatusCodes.Status409Conflict)]
   [Route("current")]
                public async Task<IActionResult> GetCourseCurrentSyllabusAsync(Guid courseId)
   {
                        var syllabusDto = await _syllabi.GetCourseCurrentSyllabusAsync(courseId);
      if (syllabusDto == null)
      {
         return NotFound();
      }
      return Ok(syllabusDto);
   }

   [HttpGet]

[ProducesResponseType(typeof(IList<SyllabusBrief>), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]
   [Route("")]
                        public IActionResult GetCourseSyllabusesAsync(Guid courseId)
   {
                            var syllabiBriefs = _syllabi.GetCourseSyllabusesAsync(courseId);
      if (syllabiBriefs == null)
      {
         return NotFound();
      }
      return Ok(syllabiBriefs);
   }

   [HttpGet]
               [ProducesResponseType(typeof(Syllabus), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]
   [Route("{syllabusId:guid}")]
                public async Task<IActionResult> GetCourseSyllabusAsync(Guid courseId, Guid syllabusId)
   {
                        var syllabusDto = await _syllabi.GetCourseSyllabusAsync(courseId, syllabusId);
      if (syllabusDto == null)
      {
         return NotFound();
      }
      return Ok(syllabusDto);
   }

   [HttpPost]
        [ProducesResponseType(typeof(SyllabusBrief), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]
   [Route("current")]
                public async Task<IActionResult> UpdateCourseSyllabus(Guid courseId, CreateSyllabusRequest request)
   {
                        var syllabusBrief = await _syllabi.UpdateCourseSyllabus(courseId, request);
      if (syllabusBrief == null)
      {
         return NotFound();
      }

      return Ok(syllabusBrief);
   }
}

using Application.Data;

using Microsoft.EntityFrameworkCore;

using Persistence;

using StatisticsService.App;

var builder = WebApplication.CreateBuilder(args);

builder.AddServiceDefaults();

builder.Services.AddOpenApi();
builder.Services.AddControllers();

var connectionString = builder.Configuration.GetConnectionString("postgresdb");
builder.Services.AddDbContext<ApplicationContext>(o => o.UseNpgsql(connectionString));
builder.Services.AddTransient<IApplicationContext>(s => s.GetRequiredService<ApplicationContext>());

builder.Services.AddScoped<IStatisticsService, StatisticsService.App.StatisticsService>();

var app = builder.Build();

app.MapOpenApi();
app.UseSwaggerUI(options => options.SwaggerEndpoint("/openapi/v1.json", "StatisticsService"));

app.UseHttpsRedirection();

app.MapControllers();

app.Run();

public abstract partial class Program
{
}

using Microsoft.AspNetCore.Mvc;

using StatisticsService.App;
using StatisticsService.Client.Models;
```

```
using
StatisticsService.Client.Models.StudentCategorization;

namespace StatisticsService.Controllers;

[ApiController]
public class StatisticsController : ControllerBase
{
   private readonly IStatisticsService _statisticsService;

            public StatisticsController(IStatisticsService
statisticsService)
   {
      _statisticsService = statisticsService;
   }

   [HttpGet]

[ProducesResponseType(typeof(SurveyResultsTimelin
e), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFo
und)]
   [Route("/surveys/{surveyId}/statistics/timeline")]
                public async Task<IActionResult>
GetSurveyResultsTimeline(Guid surveyId)
   {
                           var surveyResult = await
_statisticsService.GetSurveyResultsTimeline(surveyId)
;
      if (surveyResult == null)
      {
         return NotFound();
      }
      return Ok(surveyResult);
   }

   [HttpGet]

[ProducesResponseType(typeof(SurveyGroupResultsTi
meline), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFo
und)]

[Route("/surveys/groups/{surveyGroupId}/statistics/ti
meline")]
                public async Task<IActionResult>
GetSurveyGroupResultsTimeline(Guid
surveyGroupId)
   {
                                var result = await
_statisticsService.GetSurveyGroupResultsTimeline(sur
veyGroupId);
      if (result == null)
      {
         return NotFound();
      }

      return Ok(result);
   }

   [HttpGet]

[ProducesResponseType(typeof(SurveyResultsCategori
zation), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFo
und)]

[Route("/surveys/{surveyId:guid}/statistics/categorizati
on")]
                                public IActionResult
GetSurveyResultsCategorization(Guid surveyId)
   {
                                                   return
Ok(MockData.GetSurveyResultsCategorization());
   }

   [HttpGet]

[ProducesResponseType(typeof(SurveyGroupResultsC
ategorization), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFo
und)]

[Route("/surveys/groups/{surveyGroupId:guid}/statisti
cs/categorization")]
                                public IActionResult
GetSurveyGroupResultsCategorization(Guid
surveyGroupId)
   {
                                                   return
Ok(MockData.GetSurveyGroupResultsCategorization(
));
   }

   [HttpGet]

[ProducesResponseType(typeof(IList<SurveyResults>)
, StatusCodes.Status200OK)]
   [Route("/surveys/results")]
                public async Task<IActionResult>
GetSurveyResults()
   {
                                   var results = await
_statisticsService.GetSurveyResults();
      return Ok(results);
   }

   [HttpGet]

[ProducesResponseType(typeof(IList<SurveyGroupRe
sults>), StatusCodes.Status200OK)]
   [Route("/surveys/groups/results")]
                public async Task<IActionResult>
GetSurveyGroupResults()
   {
                                   var results = await
_statisticsService.GetSurveyGroupResults();
      return Ok(results);
   }
}

using Application.Data;
```

```
using Microsoft.EntityFrameworkCore;

using Persistence;

using Scalar.AspNetCore;

using SurveyService;
using SurveyService.App;
using SurveyService.App.Interfaces;

var builder = WebApplication.CreateBuilder(args);

builder.AddServiceDefaults();
builder.Services.AddOpenApi();

builder.Services.AddHttpContextAccessor();
builder.Services.AddSingleton<IParticipantAuthenticat
ionService, ParticipantAuthentication>();

builder.Services.AddAuthorization();
builder.Services
    .AddAuthentication("Auth0")
    .AddJwtBearer("Auth0");



var                connectionString                =
builder.Configuration.GetConnectionString("postgresd
b");
builder.Services.AddDbContext<ApplicationContext>(
o => o.UseNpgsql(connectionString));
builder.Services.AddTransient<IApplicationContext>(s
=> s.GetRequiredService<ApplicationContext>());

builder.Services.AddScoped<ISurveysService,
SurveysService>();
builder.Services.AddScoped<ISurveyGroupsService,
SurveyGroupsService>();

var app = builder.Build();

app.MapOpenApi();
app.UseSwaggerUI(options                           =>
options.SwaggerEndpoint("/openapi/v1.json",
"SurveyService"));
app.MapScalarApiReference();

app.UseAuthentication();
app.UseAuthorization();

app.UseHttpsRedirection();
app.MapControllers();


app.Run();


public abstract partial class Program
{
}

using Application.Data;
```

```
using Domain.Survey;

using Microsoft.EntityFrameworkCore;

using SurveyService.App.Interfaces;
using SurveyService.Models;

namespace SurveyService.App;

using SurveyGroup = Models.SurveyGroup;

public class SurveyGroupsService :
ISurveyGroupsService
{
   private readonly IApplicationContext _context;

      public SurveyGroupsService(IApplicationContext
context)
   {
      _context = context;
   }

         public SurveyGroup? GetSurveyGroup(Guid
groupId)
   {
      var group = _context.SurveyGroups
         .Include(sg => sg.Surveys)
                     .FirstOrDefault(g => g.Id == new
SurveyGroupId(groupId));

      if (group == null)
      {
         return null;
      }

      var groupDto = new SurveyGroup(
         group.Id.Value,
         group.Title,
         group.Surveys.Select(s =>
                              new SurveyInfo(s.Id.Value,
s.Title)).ToList());

      return groupDto;
   }

}

using Application.Data;

using Domain.Submissions;
using Domain.Survey;
using Domain.Syllabus;

using Microsoft.EntityFrameworkCore;

using SurveyService.App.Interfaces;
using SurveyService.Models;

using Lesson = SurveyService.Models.Lesson;
using Module = SurveyService.Models.Module;
using Survey = SurveyService.Models.Survey;
using SurveyGroup = Domain.Survey.SurveyGroup;
```

```
using Theme = SurveyService.Models.Theme;

namespace SurveyService.App;

public class SurveysService : ISurveysService
{
   private readonly IApplicationContext _context;
     private readonly IParticipantAuthenticationService
_authService;

    public SurveysService(IApplicationContext context,
IParticipantAuthenticationService authService)
   {
      _context = context;
      _authService = authService;
   }

        public async Task<Survey?> GetSurvey(Guid
surveyId)
   {
      var survey = await _context.Surveys
         .Include(s => s.Syllabus)
                  .FirstOrDefaultAsync(s => s.Id == new
SurveyId(surveyId));

      if (survey == null)
      {
         return null;
      }

      var syllabus = await _context.Syllabi
         .Include(s => s.Modules)
         .ThenInclude(m => m.Themes)
         .ThenInclude(t => t.Lessons)
                     .FirstOrDefaultAsync(c => c.Id ==
survey.Syllabus.Id);

      if (syllabus == null)
      {
         return null;
      }

      var surveyDto = new Survey(
         survey.Id.Value,
         survey.Title,
         syllabus.Modules.Select(m =>
                        new Module(m.Id.Value, m.Title,
m.Themes.Select(t =>
                           new Theme(t.Id.Value, t.Title,
t.Lessons.Select(l =>
                           new Lesson(l.Id.Value, l.Title,
(Models.LessonType)l.Type)).ToList())
            ).ToList())).ToList());



      return surveyDto;
   }

        public async Task<bool> SubmitSurvey(Guid
surveyId, Submission submission)
   {
                              var userId = await
_authService.GetParticipantIdAsync();
      if (userId == null)
         return false;

      SurveyGroup? group = null;
      if (submission.GroupId != null)
      {
                                          group = await
_context.SurveyGroups.FirstOrDefaultAsync(s =>
                                             s.Id == new
SurveyGroupId(submission.GroupId.Value));
      }

         var survey = _context.Surveys.FirstOrDefault(s
=> s.Id == new SurveyId(surveyId));
      if (survey == null)
      {
         return false;
      }

                                    var assessments =
submission.Assessments.Select(assessment =>
            new Domain.Submissions.Assessment(new
LessonId(assessment.LessonId),
assessment.TimeSpend));

                        var submissionEntity = new
SurveySubmission(userId,                    survey,
assessments.ToList(), group);

      _context.Submissions.Add(submissionEntity);
      await _context.SaveChangesAsync();

      return true;
   }

}
```

```
from dotenv import load_dotenv
load_dotenv()

from app import create_app
from app import config

app = create_app()

if __name__ == '__main__':
   print("run app")
    app.run(debug=config.DEBUG, host=config.HOST,
port=config.PORT)

import os

UPLOAD_FOLDER = 'uploads'
PROCESSED_FOLDER = 'processed'
PORT = int(os.environ.get('PORT', 5245))
DEBUG = bool(os.environ.get('DEBUG', False))
HOST = os.environ.get('HOST', '127.0.0.1')
USE_TRACES = bool(os.environ.get('USE_TRACES',
False))

MONGO_URI                                    =
os.getenv('ConnectionStrings__documents',
'mongodb://localhost:27017/')
```

```
MONGO_DB_NAME =
os.getenv('MONGO_DB_NAME', 'documents')
from flask import Blueprint, request, jsonify
from app.services.document_service import (
    upload_file,
    list_documents,
    get_status,
    get_processed_data
)

documents_bp = Blueprint('documents_bp', __name__)

@documents_bp.route('/documents',
methods=['POST'])
def upload_document():
        if 'file' not in request.files or
request.files['file'].filename == '':
        return jsonify({'error': 'No file selected'}), 400

    file = request.files['file']
    doc_info = upload_file(file)
    return jsonify(doc_info), 202

@documents_bp.route('/documents', methods=['GET'])
def list_all_documents():
    return jsonify(list_documents())

@documents_bp.route('/documents/<doc_id>/status',
methods=['GET'])
def document_status(doc_id):
    result = get_status(doc_id)
    return (jsonify(result), 404) if 'error' in result else
jsonify(result)

@documents_bp.route('/documents/<doc_id>',
methods=['GET'])
def document_data(doc_id):
    result = get_processed_data(doc_id)
    if 'error' in result:
        return jsonify(result), 404 if result['error'] ==
'Document not found' else 409
    return jsonify(result)
from enum import Enum
import os
from google import genai
from typing import Dict, Optional
from pydantic import BaseModel


class LesonType(str, Enum):
    lecture = 'Lecture'
    classWork = 'Class'
    practice = 'Practice'
    laboratory = 'Laboratory'
    seminar = 'Seminar'
    tutorial = 'Tutorial'
    exam = 'Exam'

class Lesson(BaseModel):
    title: str
    type: Optional[LesonType]
    studyHours: Optional[int]

class Theme(BaseModel):
    title: str
    lessons: list[Lesson]

class CourseModule(BaseModel):
    title: Optional[str]
    themes: list[Theme]

class Syllabus(BaseModel):
    modules: list[CourseModule]


api_key = os.environ.get('GEMINI_API_KEY')
client = genai.Client(api_key=api_key)

system_prompt = """
    You are an intelligent parser specialized in analyzing
academic course syllabi.
    Your task is to extract and structure key educational
components from the provided syllabus text.
    Do not translate any text and use language used in
the document.
    Identify and extract the main sections relevant to
course organization, including (but not limited to):
        Course content or description
        Curriculum or learning plan
        Modular breakdown (if available)
    Analyze the extracted information and reconstruct a
structured outline of the course. This outline should
include:
        Content Modules (main sections or thematic
blocks)
        Topics within each module
        Lessons or sessions under each topic (if detailed)
    Do not translate any text and use language used in
the document.
    Output a clean, organized course structure in a
hierarchical format.
    Use language of the document, do not translate
anything.
    """

def parse_sylabys(document_text: str) -> Syllabus:
    user_request = document_text

    response = client.models.generate_content(
        model='gemini-2.0-flash',
        contents=[system_prompt, user_request],
        config={
            'response_mime_type': 'application/json',
            'response_schema': Syllabus,
        }
    )

    return Syllabus.model_validate_json(response.text)
```