Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

«Інформаційна система узгодження процесів електронної освіти (за зразком preply.com)»

на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки»

Виконавці проекту

№	П.І.Б., Група
1	Чернявський Дмитро Вікторович КН-П-202
2	Пересунько Ігор Вадимович КН-П-202
3	Легкодух Максим Віталійович КН-П-202

Автор звіту

_____Пересунько Ігор Вадимович

Одеса – 2024

АНОТАЦІЯ

Пересунько І. В. Інформаційна система узгодження процесів електронної освіти: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». - Одеса: ОТУШ, 2024.

Робота присвячена аналізу, розробці та тестуванню інформаційої системи узгодження процесів електронної освіти головною функцією якої є узгодження взаємодії надавачів та споживачів освітніх послуг. Основними функціями платформи являються: реєстрація, автентифікація, авторизація, придбання послуг репетиторів, отримання інформаційних листів на електронну пошту від нашого сервісу.

Актуальність проекту підтверджується впровадженням сучасних функцій, розроблених для того, щоб зробити процес пошуку та замовлення послуг репетитора максимально зручним та ефективним. Найбільшу увагу було приділено створенню мобільного застосунку.

Актуальність сервісу також грунтується на впровадженні надійної системи безпеки для захисту, спостережності та цілісності персональних даних. В цю систему входить шифрування вхідних та вихідних даних, розробка авторизаційних та інших фільтрів на серверній частині, перевірки на коректність передаваємих даних та багато інших технологій.

Об'єктом дослідження у рамках даної роботи виступає алгоритм підрахунку загальної оцінки послуг репетитора за кількома різними категоріями оцінювання. У якості задач дослідження було встановлено наступне:

- проаналізувати різні способи реалізації взаємодії клієнтської та серверної частини;
- реалізувати отримання вхідних даних від користувача на сервеній стороні сервісу;
- розробити програмний інтерфейс (API), реалізувавши безпечне отримання вхідних даних від клієнтської частини платформи за допомогою протокола HTTP;
- реалізувати продуктивний алгоритм з урахуванням всіх складностей, застосувавши передові технології для доступу до бази даних, такі як Entity Framework Core;
- впровадити у серверну та клієнтську частину програмні засоби захисту задля забезпечення сохранності та цілісності користувацьких

даних.

Методи дослідження, що використані для досягнення поставлених задач, утворені за допомогою методів тестування програмного забезпечення, гнучких методологій розробки, системного аналізу, моделювання архітектури за допомогою принципів SOLID та теорії баз даних.

Саме Agile допоміг нашій команді швидше реагувати на зміни різного характеру, ефективніше використовувати ресурси та створити продукт, які дійсно задовольнить потреби користувачів.

Завдяки Agile наша команда змогла:

- **Оптимізувати процес розробки:** Гнучкий підхід дозволив нам швидко адаптуватися до змін та ефективно розподіляти приоритетні завдання.
- **Покращити комунікацію:** Щільна співпраця з ментором та постійний обмін досвідом всередині команди забезпечили високу якість коду та своєчасне вирішення проблем.
- **Прискорити випуск продукту:** Завдяки безперервній інтеграції та ранньому виявленню дефектів ми змогли значно скоротити час випуску нових версій продукту.

ЗМІСТ

ВСТУП

Сучасне життя неможливо уявити без інформаційних технологій та цифрового розвитку суспільства. Кожного дня все більше аспектів нашого життя переходить до цифрового простору, включаючи освіту. Спираючись на актуальність створення програмного забезпечення у галузі е-освіти, зазначену в Стратегії стратегію розвитку інформаційних технологій Міністерства цифрової трансформації України, наша команда створила сервіс, який представляє собою інформаційну систему узгодження процесів електронної освіти.

Наша команда вірить, що інвестиції у технології, які полегшують та роблять життя людей комфортнішим, мають бути найважливішою ціллю сучасних розробників програмного забезпечення. Через це наш сервіс надає безліч можливостей для поліпшення навчального процесу без зайвих зусиль та пропонує популярні послуги з електронної освіти для здобуття знань у будь-якому місці та у зручний час.

Також платформа допомагає викладачам і навчальним закладам збільшувати аудиторію студентів, зокрема, у віддаленному режимі, застосовуючи максимально прості та вигідні умови для розвитку освіти в Україні та за її межами. В той же час, мільйони людей, котрі хочуть отримати якісну освіту, мають можливість скористатися найкращими інструментами для реалізації своїх навчальних планів. Які б цілі не мали користувачі нашої платформи, будучи викладачами або студентами, сервіс допоможе їм із вирішенням широкого спектру освітніх питань.

Далі у цьому документі розглянуто аналіз системи, проектування архітектури, саму розробку, тестування та майбутню підтримку цифрової платформи для електронної освіти. Також будуть розглянуті проблеми, спірні питання, вирішення та їх реалізація, а також проведений аналіз командної роботи.

Вичерпне технічне завдання до проекту представлено у наступному розділі під назвою “Технічне завдання до проекту”. У рамках даної роботи розглядається серверна частина платформи, а також елементи клієнтської сторони мобільного додатку. Висновки показують результати, які були отримані при тестуванні даної частини проекту, загальні практичні та логічні підсумки є сукупністю звітів усіх учасників проекту.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

ІНФОРМАЦІЙНА СИСТЕМА УЗГОДЖЕННЯ ПРОЦЕСІВ ЕЛЕКТРОННОЇ ОСВІТИ

(за прикладом сервісу Preply.com)

Загальний опис

Програмне забезпечення (ПЗ) призначене для створення розподіленої системи узгодження взаємодії користувачів у контексті навчального процесу. Головна функція ПЗ полягає в узгодженні інтересів користувача, який надає послуги навчання, та користувача, що споживає ці послуги. Додаткова функція ПЗ має дозволити залишати відгуки та оцінки щодо усіх видів користувачів.

Призначення:

ПЗ орієнтоване на різні види узгодження користувачів: за мовою викладання, за вартістю послуг, за рейтингом тощо. ПЗ має бути універсальним, щоб дозволити змінити цільове призначення для різних видів послуг.

Вимоги до функціоналу. Обов'язкова частина

Реєстрація користувачів

- *Користувач ПЗ має змогу зареєструватись за допомогою авторизаційних даних (логіну/паролю).*
- *Під час реєстрації обов'язково вказується електронна пошта, яка підлягає верифікації.*
- *Користувач обирає роль: споживач або надавач послуг. Одна особа може зареєструватись і як споживач, і як надавач послуг.*
- *Під час реєстрації користувач вказує своє ім'я, яке буде використовуватись у листуванні.*

Особистий кабінет надавача послуг

- *Можливість розміщення пропозицій з навчання.*
- *Встановлення обмежень щодо рейтингу споживачів.*
- *Управління розкладом занять.*

Особистий кабінет споживача послуг

- *Пошук пропозицій за різними критеріями.*
- *Перегляд відгуків та рейтингу надавачів послуг.*
- *Управління власним розкладом навчання.*

Авторизація та особистий кабінет користувача

- *За фактом успішної авторизації користувачу надається доступ до особистого кабінету. Це окремий інформаційний ресурс (сторінка), що являє собою основне робоче місце користувача.*

Зворотній зв'язок

- *Користувач має змогу встановити власну оцінку або відгук для спожитого ресурсу чи його власника (надавача).*

Засоби пошуку

Стартовий інтерфейс ПЗ (головна сторінка або активність) має за мету популяризацію системи. На ньому має бути забезпечено відображення популярних показників як-то загальна кількість користувачів та інформаційних одиниць, вибірка з найбільш популярного контенту та нових дописів, зовнішня реклама. Також головна сторінка має містити засоби авторизації та реєстрації або посилання на них.

При реєстрації обов'язково зазначається засіб оперативного контактування (ЗОК) - телефон та електронна пошта. На тестовому етапі, за умови ускладнення замовлення послуг телефонії чи СМС, можливе використання лише електронної пошти. Засіб оперативного контактування при реєстрації підлягає верифікації через відповідь на тестовий виклик чи надісланий лист, або введення коду, пересланого через повідомлення довільного типу. Рекомендується використовувати ЗОК у якості авторизаційного логіну користувача.

За умови що користувач забув пароль, ЗОК має бути використаний для його відновлення. Схема відновлення паролю жорстко не регламентується. Також передбачається наявність у системі користувачів з додатковими правами: модераторів та адміністраторів. Реєстрація кореневого адміністратора

системи має бути реалізоване при інсталяції системи, наступним користувачам змінюють права наявні адміністратори.

Засоби пошуку
фільтрування та групування на інших сторінках мають забезпечити врахування рейтингу контенту.

Вимоги до функціоналу. Опціональна частина

Модерація
Для забезпечення дотримання загальноприйнятих норм спілкування, а також з метою недопущення публікування контенту, який містить заборонені компоненти (ненормативні висловлювання, заклики до протиправних дії, приниження чи дискримінація, тощо), усі інформаційні одиниці, а також створені користувачами метадані (маркери) мають в обов'язковому порядку пройти погодження (модерацію). Право надати погодження належить кореневому адміністратору системи, а також модераторам. При погодженні інформаційна одиниця має зберігати ідентифікатор користувача, який надав погодження.

Архітектура
ПЗ має бути спроектоване за розподіленою клієнт-серверною архітектурою з чітко відокремленими розподіленими вузлами системи:

- **Backend** - серверне програмне забезпечення, що забезпечує збереження та оброблення поточних даних, взаємодію з базою даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами.
- **Web App** - додаток, доступний з веб-простору (сайт), що забезпечує візуальний інтерфейс користувачів усіх категорій
- **Mobile App** - додаток, що встановлюється на мобільних пристроях (смартфон, планшет). Дана частина виконується за наявності у команді проекту трьох і більше програмістів.
-

Усі модулі ПЗ повинні обов'язково включати до свого складу засоби

- тестування функціональності та модульні тести,
- розгортання на новому пристрої (інсталяції) із встановленням початкових таблиць у БД та кореневого користувача-адміністратора

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контроля версій (вибір конкретної системи узгоджується групою проекту). Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на зразок:

- Backend - .NET технології із хмарним розміщенням (Azure)
- Web - JS, HTML, CSS,або JS фреймворки, зокрема React, Angular
- Mobile - Java, або Kotlin, або Flutter

Команда проєкту

Для виконання поставлених завдань передбачається робота команди розробників окремо за кожним з модулів проекту.

Мінімальна команда проекту - два учасники, у такому складі один учасник виконує модуль Backend та програмну частину Web, другий - модель

Рекомендована команда проекту - три учасники, у такій команді кожен з учасників відповідає за одну з частин: Mobile, Backend, Web. Дозволяється включення до команди більшої кількості учасників, у такому разі модуль Mobile App набуває обов'язкової реалізації.

Звітність

За результатами виконання завдань із створення ПЗ кожен з учасників команди проекту складає власний (індивідуальний) звіт, що складається з практичної частини та автореферату. Практична частина є прямим результатом роботи і подається у вигляді посилання на репозиторій (чи інший URL) або у вигляді архіву на довільному носієві даних. Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.

Автореферат містить описову частину, у якій розкриваються основні концепти проекту, деталізуються завдання до виконавця, встановлюється актуальність роботи, її об'єкт та предмет, обґрунтовується вибір технологій та відзначається шлях виконання та ступінь досягнення поставлених задач, посилання на джерела інформації.

РОЗДІЛ 1. Загальна характеристика системи

Основа програмної архітектури складається з таких самовідокремлених частин:

- FrontMobile;
- Front Web;
- Backend;

Серверна частина “Backend” розроблена за допомогою технології:

- платформи Microsoft ASP.NET;
- та архітектурного стилю RESTful;

Основною мовою програмування обрана C#. Для перебійного зберігання персональних даних користувачів та даних платформи було використано Microsoft Azure Cosmos DB. Проаналізувавши та вивчивши предметну область роботи, було обрано саме цю СУБД (систему управління базами даних) через її можливості NoSQL і реляційних баз даних, забезпечуючи високу продуктивність, масштабованість і гнучкість. Це дозволило використовувати технологію ORM (Object Relational Mapping) Entity Framework Core від Microsoft, що дуже спрощує розробку та взаємодію з базою даних. У якості серверу реалізовано RESTful-API, який приймає мережеві запити від клієнтської частини за допомогою протоколу HTTP, проводить маніпуляції з даними, робить запити до бази даних (якщо це потребується) та відправляє відповідь з кінцевими та обробленими даними до клієнтської частини за допомогою текстового формату JSON. У якості сервісу з встановленням зовнішніх пакетів було використано систему NuGet від Microsoft. Для того, щоб мати масштабований та відповідавший до змін сервер, усі конфігураційні параметри були розподілені по спеціальним файлам у текстовому форматі JSON, з яких за потреби витягуються потрібні дані. Це також допомагає робити зміни у одному місці, що одразу ж відображається в усьому проекту. Для реалізації авторизації була використана технологія JWT-токенів, яка являється однією з найпоширеніших назараз. Спочатку клієнтська сторона відправляє запит до серверу з вимогою надати їй токен авторизації, після чого, при наступних запитах до серверу, сервер перевіряє цей токен та відповідає дозволом чи забороною до виконання будь-яких дій. Саме ця перевірка авторизації на стороні API реалізована в так званих фільтрах авторизації, де саме і відбувається перевірка коректності переданих від клієнтської частини даних. Головним 12 принципом проектування архітектури серверу було використання патернів проектування, а також принципів проектування класів SOLID. Під час розробки наша команда проаналізувала усі дії і ми прийшли до

висновку, що маємо застосувати методологію DevOps на базі Azure DevOps через те, що наша платформа здебільшого застосовує продукти Microsoft через повну сумісність із програмним забезпеченням. Для того, щоб це реалізувати, ми використали такі сервіси від Azure як: App Service, Azure CosmosDB. Увесь проект зараз зберігається та працює на хмарних технологіях від Microsoft Azure, що дозволяє отримувати максимальну продуктивність та безпечність системи, а також зручність, яка забезпечує нам, як розробникам - легке масштабування та швидкість оновлення програмного забезпечення, а користувачам безперебійність системи.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ

Серверна частина складається з: - RESTful-API з використанням платформи розробки ASP.NET від Microsoft; - Бази даних в основі Azure CosmosDB; - Мови програмування: C#; - Середовища розробки: Visual Studio; - Використанні технології: Entity Framework Core; - Встановлені додаткові пакети: Microsoft.Azure.Cosmos, Microsoft.Extensions.Configuration.Abstractions, Portable.BouncyCastle. З самого початку були створені необхідні сутності та моделі. (приклади нижче)

Користувач

```
1.  using Newtonsoft.Json;
2.  namespace Bachelor.Data.Entity
3.  {
4.      public class User
5.      {
6.          [JsonProperty("id")]
7.          public string? id { get; set; }
8.          public string partitionKey { get; set; } = "user";
9.          [JsonProperty("RealName")]
10.         public string RealName { get; set; } = string.Empty;
11.         [JsonProperty("Login")]
12.         public string Login { get; set; } = string.Empty;
13.         [JsonProperty("Email")]
14.         public string Email { get; set; } = string.Empty;
15.         [JsonProperty("PasswordHash")]
16.         public string PasswordHash { get; set; } = string.Empty;
17.         [JsonProperty("PasswordSalt")]
18.         public string PasswordSalt { get; set; } = string.Empty;
19.         [JsonProperty("Phone")]
20.         public string Phone { get; set; } = string.Empty;
21.         public string Role { get; set; } = "client";
22.     }
23. }
```

Репетитор

```
1.  using Newtonsoft.Json;
2.
3.  namespace Bachelor.Data.Entity
4.  {
5.      public class Tutor
6.      {
7.          [JsonProperty("name")]
8.          public string name { get; set; } = string.Empty;
9.          [JsonProperty("email")]
10.         public string email { get; set; } = string.Empty;
11.     }
12. }
```

Модель реєстрації

```
1.  namespace Bachelor.Models.UserModels
2.  {
3.      public class UserRegistrationModel
4.      {
5.          public string Login { get; set; } = null!;
6.          public string Password { get; set; } = null!;
7.          public string RepeatPassword { get; set; } = null!;
8.          public string Email { get; set; } = null!;
9.          public string Phone { get; set; } = null!;
10.         public string RealName { get; set; } = null!;
11.     }
12. }
```

Уся архітектура серверної частини була спроектована, використовуючи патерни та принципи проектування для того, щоб створити дійсно конкурентоспроможний продукт, який може масштабуватись та підтримуватись надалі без будь-яких проблем зі сторони розробників. Саме для цього наша команда також використовувала принципи SOLID для проектування класів, щоб забезпечити максимально сильне скріплення всередині класів та мінімізувати зв’язок між класами. Де це було можливо, віддано перевагу агрегації з композицією замість успадкуванню. Проектувавши архітектуру, ми мислили інтерфейсами, а не класами для того, щоб втілити принципи проектування вказані вище. При проектувані бази даних, аналізована та втілена її нормалізація, застосувавши так звані перші три її форми. Кожен файл відповідає за свою частину конфігурації системи. Усі персональні та системні дані зберігаються у базі даних (приклад нижче).

id	/partitionKey
a5924374-61b0-498b-a...	user
33340A25-E60F-4882-B...	lang
21140A25-E60F-4882-B...	lang
31440A25-E60F-4882-B...	lang
d37601d2-0d4f-4ad7-8...	user
f413ee8f-574c-4394-87...	user
31540A25-E60F-4882-B...	lang
31640A25-E60F-4882-B...	lang
31740A25-E60F-4882-B...	lang
31840A25-E60F-4882-B...	lang
31940A25-E60F-4882-B...	lang
524c6c24-4398-4fe6-9...	user
7e6d440c-e9f5-4dd7-8...	user
b14cd6c4-a09b-411d-b...	user
33981b5a-2e52-4d53-b...	user
9f722b63-0262-4d03-b...	user
a4f0709a-3ae3-47d5-a...	user
f6bc816c-2f77-4260-b1...	user
c4fd3857-e2a9-4be2-8...	user
db81809b-e834-4428-...	user
ae77e2e5-9d27-4061-a...	user

Для маніпулювання з даними використовується Entity.Framework.Core та Microsoft.Azure.Cosmos. На платформі реалізована надійна система реєстрації та авторизації, які забезпечують сохранність персональних даних користувача. На платформі реалізовано один види реєстрації і авторизації. Користувач має змогу увійти або зареєструватись за допомогою електронної пошті, паролю, вводимо свої дані у відповідні поля, після цих дій користувач буде зареєстрований. У разі авторизації треба вказати електронну пошту, та пароль. При реєстрації вказаний пароль буде зехешовано за допомгою бібліотеки BCrypt (див. лістинг коду, рядок 7).

```
1.  var user = new Data.Entity.User
2.  {
3.      id = Guid.NewGuid().ToString(),
4.      RealName = model.RealName,
5.      Email = model.Email,//добавить валидацию
6.      Phone = model.Phone
7.      PasswordHash
    =System.Text.Encoding.UTF8.GetString(BCrypt.PasswordToByteArray(model.Password.ToCharArray()))
8.  };
```

BCrypt генерує випадкову послідовність символів (сіль), яка є унікальною для кожного пароля. Сіль додається до паролю перед

хешуванням. Це дає нам змогу завжди отримувати унікальний хеш-код, що забезпечує надійність і сохранність персонального пароля користувача. Після цих дій усі зареєстровані дані вносяться у базу даних. Відбувається запит до серверу, після чого перевіряються дані. Якщо вони коректні, то запитана дія виконується. Як описано вище, наша команда проектувала усе через інтерфейсні посилання, а не конкретні реалізації, що дозволяє нам отримувати масштабування та легкість змін у випадку редагування коду. Щоб у коді не було ніяких магічних чисел та він був зрозумілий кожному розробнику. На головній сторінці сервісу можно побачити кількість блоків з мовами програмування де буде вказана кількість репетиторів за цією мовою, піля натисканя буде більше іноформації(нижче приведен приклад блока вілповдного мові C# де знаходитться статична інформвція).


```
{
    "id": "21140A25-E60F-4882-B521-7EF9B40818F9",
    "partitionKey": "lang",
    "title": "Tutors C#",
    "tutorsCNT": 514,
    "tutors": [
        {
            "name": "Walter White",
            "email": "ws@ukrnet.com"
        },
        {
            "name": "Megan Conor",
            "email": "mc@ukrnet.com"
        }
    ],
    "_rid": "tCNxALwOkewEAAAAAAAAAA==",
    "_self": "dbs/tCNxAA==/colls/tCNxALwOkew=/docs/tCNxALwOkewEAAAAAAAAAA==/",
    "_etag": "\"010077e9-0000-4d00-0000-66694a5f0000\"",
    "_attachments": "attachments/",
    "_ts": 1718176351
}
```

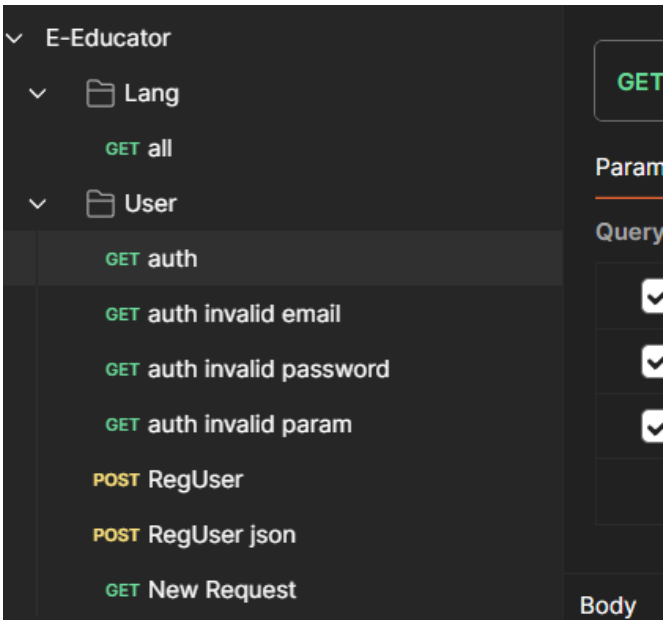

Не враховуючи цей запис у базі даних я під час розробки намагався мінімізувати присутність статичних даних. Це дозволяє отримувати максимально нове наповнення Web-сторінок, та застосунку. Основною частиною розробки був етап створення налагодженого механізму взаємодії репетиторів із нашим майданчиком. Верефікація їхніх документів, які підтверджують їхню кваліфікацію в тих галузях, які вони бажають викладати нашим клієнтам. Передбачалося, що вже зареєстрований користувач буде заповнювати відповідну форму, яку після цього перевірятиме модератор E-educator. Виходячи з потреб нашої платформи, ми створили з одного боку простий та швидкий, а з другого дуже місткий процесс реєстрації нового репетитора.

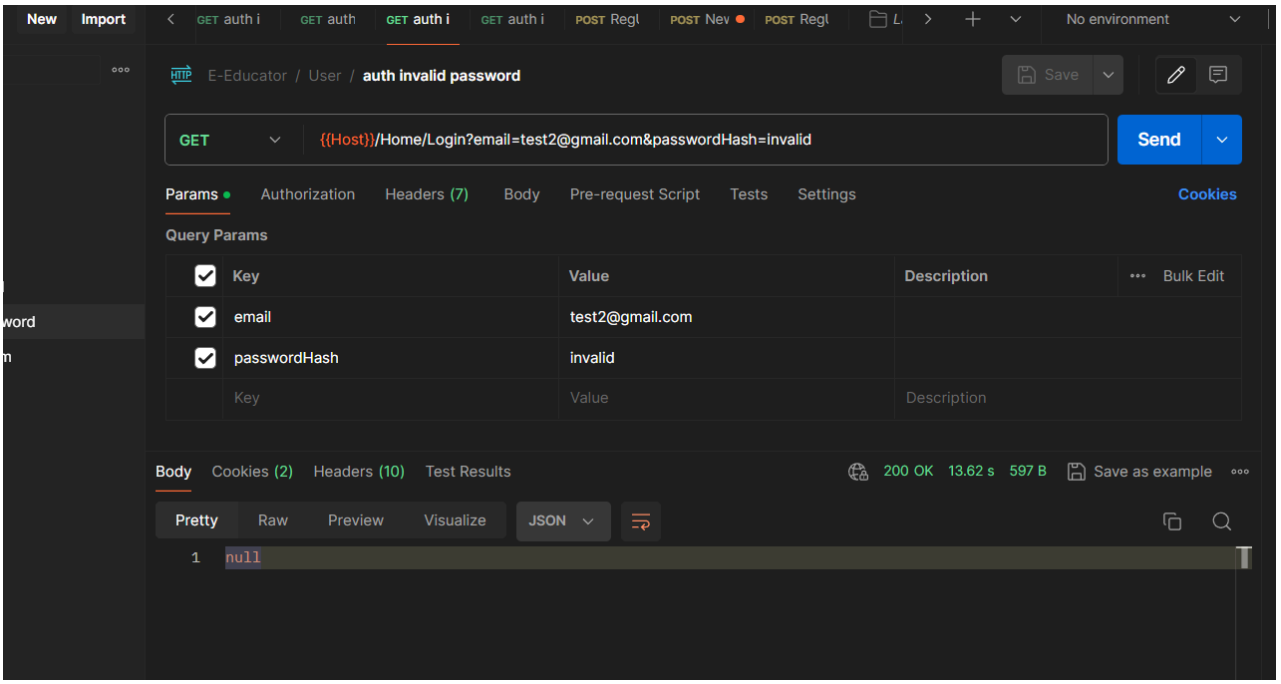
РОЗДІЛ 3
РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Результатом компіляції програмного коду сервеної частини є корректні запроси та відповіді між серверною та клієнтською частинами. В заголом для тестуваня був використан інструмент Postman, він дозволяє відправляти HTTP-запити до серверів, перевіряти відповіді

- інструменти для тестування: Postman;

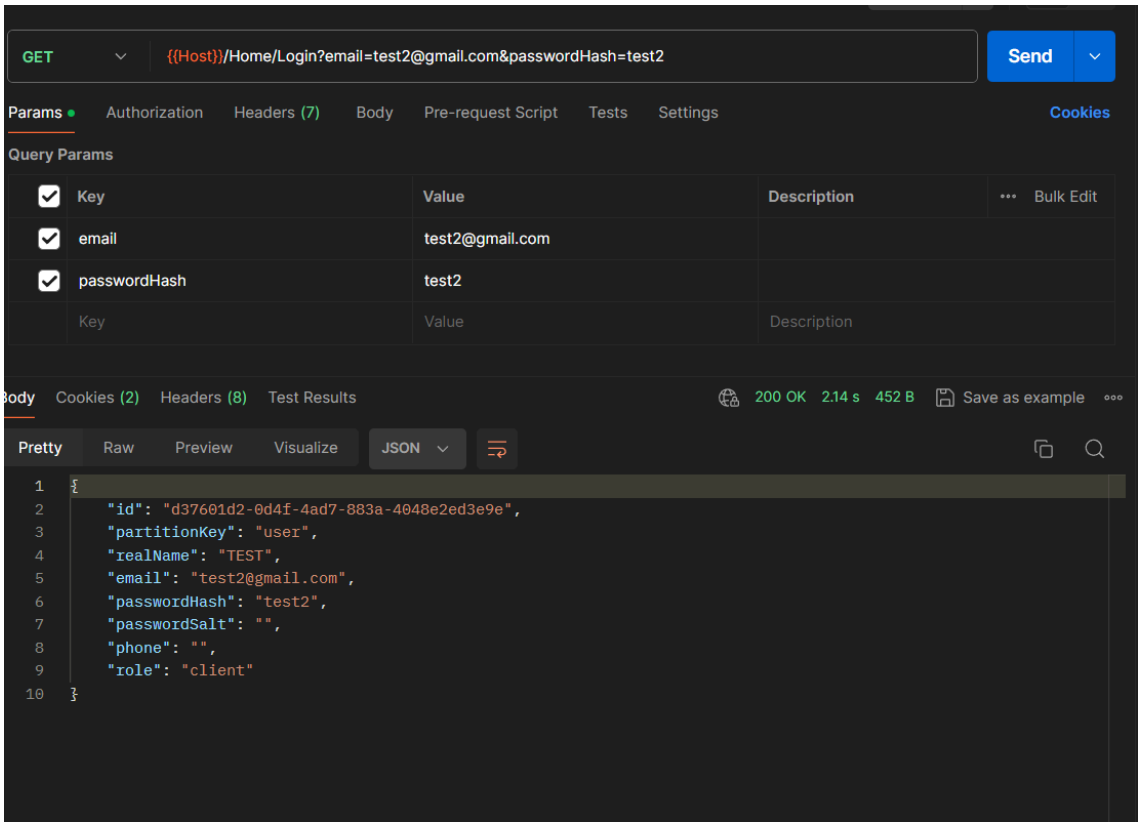
Під час створення проекту був використан інструмент Postman, він забеспечив зручне тестування шляхом надіслання запитів на сервер. Нижче преведені приклади запитів у постамн:





Приклад запиту на авторизацію з помилкою у паролі, у полі

passwordHash вказуэмо що завгодно але ні справжній пароль. Нижче бачимо відповіль у форматі json “null” що позначає що такого користувача за тким паролем у базі не знайдено.



Приклад запиту на авторизацію з зараєстрованою поштою та паролем. Нижче бачимо відповіль у форматі json в якому збережена інформація корситувача. Це позначає що корстивач існує та нас авторизовано

ВИСНОВКИ

Проведено огляд існуючих інформаційних систем, що забезпечують взаємодію викладачів та студентів у контексті електронної освіти. Виявлено, що важливу роль у таких системах складають функції реєстрації та авторизації користувачів, можливості редагування профілю, управління розкладом занять та обміну повідомленнями. Саме ці функції були визначені як ключові для реалізації у розробленій системі.

Для клієнтської частини було обрано платформу Android з використанням мови програмування Java та Kotlin. Серед основних бібліотек, що були використані, варто виділити Retrofit для роботи з мережевими запитами, OkHttp для обробки HTTP-запитів, Gson для серіалізації та десеріалізації даних, а також Glide для завантаження та відображення зображень.

Визначено та впроваджено сучасні технології, що дозволяють забезпечити безпечну та ефективну роботу системи. На серверній частині системи використано платформу ASP.NET Core для створення RESTful API та Entity Framework Core для доступу до бази даних. За безпеку передачі даних відповідав алгоритм шифрування AES-256. Для зберігання даних обрано Azure Cosmos DB, що забезпечує високу надійність та масштабованість системи. У якості хостингу серверної чатсини був обран Azure Web App, він надає місце для розгортання веб-додатку, великий набір інструментів для управління та розробки. Якщо коротко описати переваги варто відзначити його гнучкість, масштабованість і зручність використання, а також варто відзначити відносно невелику ціну в контесті нашого продукту, на відміну від багатьох інших хостингів тут оплата відбувається в міру використання інструментів Azure, а не стабільної виплати раз на місяць наприклад. В цілому цей аргумент підходить під опис причини того, чому я дотримувався екосистеми Azure. Основні бібліотеки які були використані під час розробки це: Microsoft.Azure.Cosmos, Microsoft.Extensions.Configuration.Abstractions, Portable.BouncyCastle.

Розроблено, реалізовано та випробувано інформаційну систему узгодження процесів електронної освіти. Система побудована за розподіленою клієнт-серверною архітектурою та складається з трьох основних компонентів:

- **Клієнтська частина (мобільний додаток)**, доступна на платформі Android.
- **Серверна частина**, розміщена на платформі Azure.
- **База даних**, реалізована на Azure Cosmos DB.

Система забезпечує можливість реєстрації та авторизації користувачів,

редагування профілю, управління розкладом занять, обмін повідомленнями між викладачами та студентами, а також пошук репетиторів. Впроваджено засоби підвищення інформаційної безпеки: шифрування даних при передачі та валідація введених даних.

Систему було випробувано шляхом інсталяції на фізичні пристрої та проведення тестових сеансів комунікації. Також був використан інструменту Postman завядки якому я зміг посилати запити на API та отримувати відповіді у у вигляді JSON файлів які вже можно було перевірита на корекнтість. Результати випробувань підтвердили загальну працездатність створеної системи та відповідність її заявленим вимогам. Система демонструє високу продуктивність та стабільність роботи, що дозволяє рекомендувати її для впровадження та подальшого використання в реальних умовах.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СЕРВЕРНОЇ ЧАСТИНИ

1. Microsoft Build. Overview of ASP.NET Core // Learn Microsoft. URL: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0 (дата звернення: 06.04.2024).
2. Microsoft Build. Entity Framework Core // Learn Microsoft. URL: https://learn.microsoft.com/en-us/ef (дата звернення: 06.04.2024).
3. Microsoft Build. Azure Cosmos DB // Learn Microsoft. URL: https://learn.microsoft.com/en-us/azure/cosmos-db (дата звернення: 06.04.2024).
4. Microsoft Build. C# language documentation // Learn Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/csharp (дата звернення: 06.04.2024).
5. Microsoft Build. Visual Studio // Learn Microsoft. URL:https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022 (дата звернення: 06.04.2024).
6. Microsoft Build Azure Web App // Learn Microsoft. URL: https://learn.microsoft.com/uk-ua/azure/app-service/ (дата звернення: 06.04.2024).
7. Postman // Learn Postman. URL: https://learning.postman.com/docs/introduction/overview/ (дата звернення: 21.04.2024)