Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Освітня платформа для надання та споживання послуг вивчення мов
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-9

Чернявський Д.В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Олійник О.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Чернявському Дмитру Вікторовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Освітня платформа для надання та споживання послуг вивчення мов

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2024

3. Вихідні дані до роботи Розробити освітню платформу, яка надаватиме можливість студентам притбати та запланувати урок на певний час, а також надавати можливість учителям розміщувати уроки на платформі, гнучко налаштовуючи графік, мовами C# та TypeScript.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	12.04.2024	*виконано*
3	Проектування ПЗ	30.04.2024	*виконано*
4	Розробка ПЗ	18.05.2024	*виконано*
5	Тестування ПЗ	25.05.2024	*виконано*
6	Оформлення пояснювальної записки	05.06.2024	*виконано*
7	Підготовка презентації та доповіді	06.06.2024	*виконано*
8	Попередній захист	13.06.2024	*виконано*
9	Нормоконтроль, рецензування	14.06.2024	*виконано*
10	Здача роботи у електронний архів	18.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	19.06.2024	*виконано*

Дата видачі завдання 21 травня 2024р.

Студент (ка)_____ Чернявський Д.В.
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Олійник О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 80 стор., 64 рис., 11 джерел., 2 додатки.

ВЧИТЕЛЬ, КУРС, ОСВІТНЯ ПЛАТФОРМА, ПЛАНУВАННЯ, РЕПЕТИТОР, РОЗКЛАД, СЕМІНАР, СТУДЕНТ, УРОК, ANGULAR, ASP.NET CORE, C#, TYPESCRIPT

Об’єкт розробки – освітня платформа для надання та споживання послуг вивчення мов

Мета розробки – створення освітньої платформи, яка забезпечить доступ до послуг з вивчення мов для користувачів у будь-який час та зручним способом.

Метод рішення – середовище розробки JetBrainds Rider та JetBrainds WebStorm, мови програмування C# та TypeScript.

У результаті розробки створено освітню платформу, яка надає студентам можливість придбати та запланувати урок на будь-який час та надає можливість вчителям розміщати свої уроки та гнучко керувати своїм графіком.

TEACHER, COURSE, EDUCATIONAL PLATFORM, PLANNING, TUTOR, SCHEDULE, SEMINAR, STUDENT, LESSON, ANGULAR, ASP.NET CORE, C#, TYPESCRIPT

Object of development – an educational platform for the provision and consumption of language learning services.

Development goal – to create an educational platform that will provide users with access to language learning services at any time and in a convenient way.

Solution method – JetBrains Rider and JetBrains WebStorm development environments, programming languages C# and TypeScript.

As a result of the development, an educational platform has been created that allows students to purchase and schedule lessons at any time and enables teachers to post their lessons and manage their schedules flexibly.

Я, Чернявський Дмитро Вікторович, студент гр. ПЗПІ-20-9, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Освітня платформа для надання та споживання послуг вивчення мов», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У нашому дедалі більш взаємопов'язаному світі здатність спілкуватися поза мовними кордонами ще ніколи не була такою важливою. Хоча традиційні моделі мовної освіти добре нам слугували, розвиток цифрових технологій відкриває безпрецедентні можливості для революційних змін у вивченні та викладанні мов[4][6]. У цій роботі досліджується та реалізується освітня платформа, спеціально розробленої для надання та споживання послуг з вивчення мов[3], вирішення ключових проблем та використання глобальних тенденцій для створення динамічного та доступного навчального середовища.

Наразі сфера мовної освіти стикається з низкою обмежень. Традиційні заняття в класі часто обмежені географічним розташуванням, жорстким розкладом і обмеженим доступом до різноманітних навчальних ресурсів[2]. Незважаючи на появу онлайн-платформ для вивчення мови, багато з них надають перевагу заучуванню та граматичним вправам, позбавленим захопливого та захоплюючого досвіду, необхідного для ефективного вивчення мови. Провідні інституції, такі як Британська Рада та Гете-Інститут, досліджують моделі змішаного навчання, інтегруючи технології з традиційними методами викладання. Однак ці ініціативи часто орієнтовані на конкретні мови та демографічні групи учнів, що залишає значний пробіл у доступності для мов з меншими ресурсами та різноманітних стилів навчання.

У всьому світі зростає тенденція до персоналізованої та орієнтованої на учня освіти з акцентом на інтерактивному навчанні, зворотному зв'язку в реальному часі та гейміфікованому навчальному середовищі. Ця тенденція проявляється в успіху таких платформ, як Duolingo і Babbel, які використовують гейміфікацію для мотивації учнів і відстеження прогресу. Однак цим платформам часто бракує глибини та комплексності, необхідних для досягнення вільного володіння мовою та розуміння нюансів різних мов і культур.

Дана робота має на меті усунути ці недоліки шляхом розробки освітньої платформи, яка використовує передові технології для створення динамічного, цікавого та доступного простору для вивчення мов. Платформа сприятиме

налагодженню зв'язків між тими, хто вивчає мову, та кваліфікованими викладачами, пропонуючи різноманітний спектр послуг з вивчення мови, адаптованих до індивідуальних потреб [3].

Застосовувати підхід до спільного навчання, заохочуючи взаємодію між учнями та сприяючи створенню дружньої онлайн-спільноти.

Надавати пріоритет доступності, пропонуючи послуги широким спектром мов і задовольняючи різні стилі навчання та рівні технологічної грамотності.

Це дослідження ґрунтується на існуючих напрацюваннях у сфері вивчення мов онлайн. Інтегруючи персоналізоване навчання та спільний підхід, ця платформа має на меті розширити можливості людей у подоланні мовних бар'єрів та налагодженні зв'язків із ширшим світом. Потенційні можливості застосування цієї платформи дуже широкі: від полегшення міжнародного спілкування до сприяння міжкультурному взаєморозумінню та особистісному зростанню через вивчення мови.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сучасний світ стає все більш глобалізованим, що збільшує попит на знання іноземних мов. Люди шукають зручні та ефективні способи вивчення мов, і освітня платформа, що надає послуги з вивчення іноземних мов, стає ключовим інструментом у цьому процесі. У цьому аналізі ми розглянемо ключові аспекти такої предметної галузі, оцінимо конкурентне середовище та визначимо потенційні можливості для успішного розвитку проекту.

Тенденції ринку. Зростаюча популярність онлайн-освіти сприяє розвитку платформ для вивчення мов. Потреба у міжнародній комунікації веде до збільшення числа людей, які бажають вивчити іноземні мови. Крім того, широкий доступ до Інтернету та технологічних засобів забезпечує зручність і доступність онлайн-освіти.

Аналіз конкурентного середовища. На сьогоднішній день існують декілька успішних платформ, що пропонують послуги з вивчення іноземних мов. Ці платформи надають можливість знаходити викладачів та студентів для індивідуальних уроків, а також матеріали для самостійного вивчення. Вони успішно поєднують технології з особистим підходом, що робить їх привабливими для користувачів.

Сильні сторони існуючих платформ:

- широкий вибір викладачів з різних країн та рівнів володіння мовою;
- можливість підібрати індивідуальний графік занять;
- різноманітність методик вивчення, включаючи онлайн-уроки, чати, аудіо- та відеоматеріали.

Слабкі сторони існуючих платформ:

- висока конкуренція та наявність великих гравців на ринку;
- певні обмеження у можливостях комунікації та виборі викладачів;
- потенційні проблеми з якістю викладання через різні методики та підходи викладачів.

Можливості для розвитку. Не зважаючи на конкуренцію, існують можливості для створення успішної освітньої платформи для вивчення іноземних мов:

- персоналізований підхід: Розвиток системи рекомендацій та персоналізованих уроків на основі вимог індивідуального студента;
- Інтерактивність та спільнота: Створення форумів, групових занять та можливості обміну знаннями між учасниками;
- нові технології: Використання штучного інтелекту для вдосконалення процесу вивчення мови та аналізу прогресу студентів.

Попит на вивчення іноземних мов зростає, що створює сприятливе середовище для розвитку освітніх платформ у цій сфері. Незважаючи на наявність конкурентів, новий проект може бути успішним шляхом за умови впровадження новаторських підходів та вдосконалення якості надання послуг.

1.2 Виявлення та вирішення проблем

Розглядаючи сучасний ландшафт освітніх платформ для вивчення іноземних мов, можна виокремити кілька ключових проблем, які стають викликом для користувачів та платформ самостійно. Перш за все, багато існуючих платформ, таких як iTalki та Preply, не завжди забезпечують достатньо індивідуалізованого підходу до навчання, що може призвести до неефективного вивчення мови. Більшість сервісів пропонують стандартизовані курси або репетиторів, не враховуючи особистих потреб та рівня знань кожного користувача.

Друга проблема полягає в обмеженій функціональності платформ. Хоча iTalki та Preply надають можливість спілкування з репетиторами та викладачами, а також проведення онлайн-уроків, вони часто обмежені у використанні додаткових інструментів для покращення навичок мови, таких як мовні ігри, вправи на вимову, аудіо-та відеоматеріали для самостійного вивчення тощо.

Крім того, існує проблема недостатньої взаємодії та зворотного зв'язку між користувачами та викладачами на існуючих платформах. Часто відсутній

механізм для ефективного відстеження прогресу, а також можливості для користувачів ділитися своїми досвідом та рекомендаціями з іншими учасниками спільноти.

Для вирішення цих проблем необхідно активно актуалізувати рішення. Одним із можливих напрямків є розробка платформи з акцентом на індивідуалізацію навчання, яка надає користувачам можливість підбору репетиторів та курсів відповідно до їхніх особистих потреб та рівня володіння мовою. Також важливо розширити функціонал платформи, додавши нові інструменти для підвищення ефективності навчання.

Крім того, необхідно забезпечити ефективну комунікацію між користувачами та викладачами, а також розробити механізми обміну досвідом у спільноті. Це може бути здійснено за допомогою впровадження соціальних функцій на платформі, таких як форуми, чати для спілкування, а також система відгуків та рейтингування.

Таким чином, активна робота над виявленням проблем та актуалізація рішень у сфері освітніх платформ для вивчення іноземних мов є ключовим завданням для покращення якості навчання та задоволення потреб сучасного користувача.

1.3 Постановка задачі

Постановка задачі полягає у розробці та впровадженні інноваційної освітньої платформи для навчання іноземних мов, яка відповідатиме сучасним вимогам ефективного та індивідуалізованого навчання. Задачі проекту включають аналіз потреб та очікувань користувачів, розробку платформи з урахуванням різних методик навчання, створення інтерактивних інструментів для практичного вивчення мови, тестування та вдосконалення функціоналу згідно з отриманими результатами, а також підготовку до впровадження та підтримку користувачів після запуску. Реалізація цих завдань спрямована на створення високоякісного та доступного інструменту для навчання іноземних мов, що відповідає сучасним стандартам освіти та вимогам ринку.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги до програмної системи

Реєстрація та авторизація:

- реалізація реєстрації користувачів з розділенням ролей (учень, репетитор) ;
- можливість входу через соціальні мережі (Google, Facebook).

Профілі користувачів:

- учні: можливість створення профілю з інформацією про себе, цілі вивчення мови, бажаний графік та бюджет;
- репетитори: створення розширеного профілю з інформацією про досвід, кваліфікацію, методики викладання, доступні мови та ціни за уроки.

Пошук та фільтрація репетиторів:

- можливість пошуку за мовою, спеціалізацією, ціною, рейтингом, доступністю;
- розширена фільтрація за різними критеріями (досвід, рівень репетитора, формат уроків).

Бронювання уроків:

- перегляд доступного графіку репетитора;
- бронювання уроку на певний час.

Навчальні матеріали:

- репозиторій навчальних матеріалів (аудіо, відео, тексти, вправи) з можливістю фільтрації за мовою та рівнем;
- можливість репетиторам завантажувати власні матеріали.

Підтвердження уроку:

- підтвердження про проведення уроку;
- вирішування конфліктів.

Оцінювання та відгуки:

- можливість учням оцінювати репетиторів та залишати відгуки;
- система рейтингу репетиторів.

Оплата:

– інтеграція з платіжними системами (Stripe, PayPal, тощо) ;
– безпечне проведення онлайн-платежів.

Адміністрування:

– панель адміністратора для управління користувачами, репетиторами, контентом, платежами, статистикою.

2.2 Нефункціональні вимоги до програмної системи

Безпека:

– захист персональних даних користувачів;
– безпека онлайн-платежів.

Надійність:

– стабільна робота платформи 24/7;
– резервне копіювання даних.

Масштабованість:

– можливість обробляти велику кількість користувачів та даних;
– гнучкість для розширення функціоналу в майбутньому.

Зручність:

– інтуїтивно зрозумілий інтерфейс та проста навігація;
– адаптивний дизайн для різних пристроїв.

Швидкодія:

– швидке завантаження сторінок;
– швидка обробка даних.

2.3 Технологічні вимоги

Клієнтська частина: Angular, TypeScript, HTML5, CSS3, Tailwind.

Серверна частина: ASP.NET Core C#, REST API.

База даних: PostgreSQL, Redis.

Платіжні системи: Stripe.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Один із найважливіших етапів розробки програмного забезпечення є концептуальний аналіз і одним із інструментів є UML моделювання. Одною з основних діаграм UML є діаграма прецедентів(Use Case Diagram), що використовується для моделювання функціональних вимог між акторами та системою.

У контексті освітньої платформи для надання та споживання послуг вивчення мов діаграма включає наступні типи користувачів:

а) адміністратор – відповідає за модерацію, контроль та управління системою. Адміністратор має наступний основний функціонал:
   1) блокування та розблокування користувачів;
   2) підтвердження заявок на становлення вчителем;
   3) модерація;
   4) привенція шахрайства;
   5) управління мовами, гео об’єктами, тощо;
   6) підтвердження виплат.

б) студенти або учні – люди, які використовують платформу для пошуку учителів та бронювання уроків на певний час. Учень має наступний основний функціонал:
   1) пошук репетитора;
   2) бронювання уроку;
   3) відміна бронювання;
   4) підтвердження про завершення уроку;
   5) оцінка репетитора;
   6) оплата уроку;
   7) рішення конфліктів.

в) вчителі або репетитори – люди, які розміщують свої уроки на платформі. Репетитор має наступний основний функціонал:
   1) управління уроками;

2) управління курсами;

3) управління календарем, в тому числі виділення вільного часу;

4) управління матеріалами.

Усі користувачі мають можливість авторизуватися у системі та управляти своїм профілем. На рисунку 3.1 можна побачити відображення даного опису функціоналу на діаграмі прецедентів.

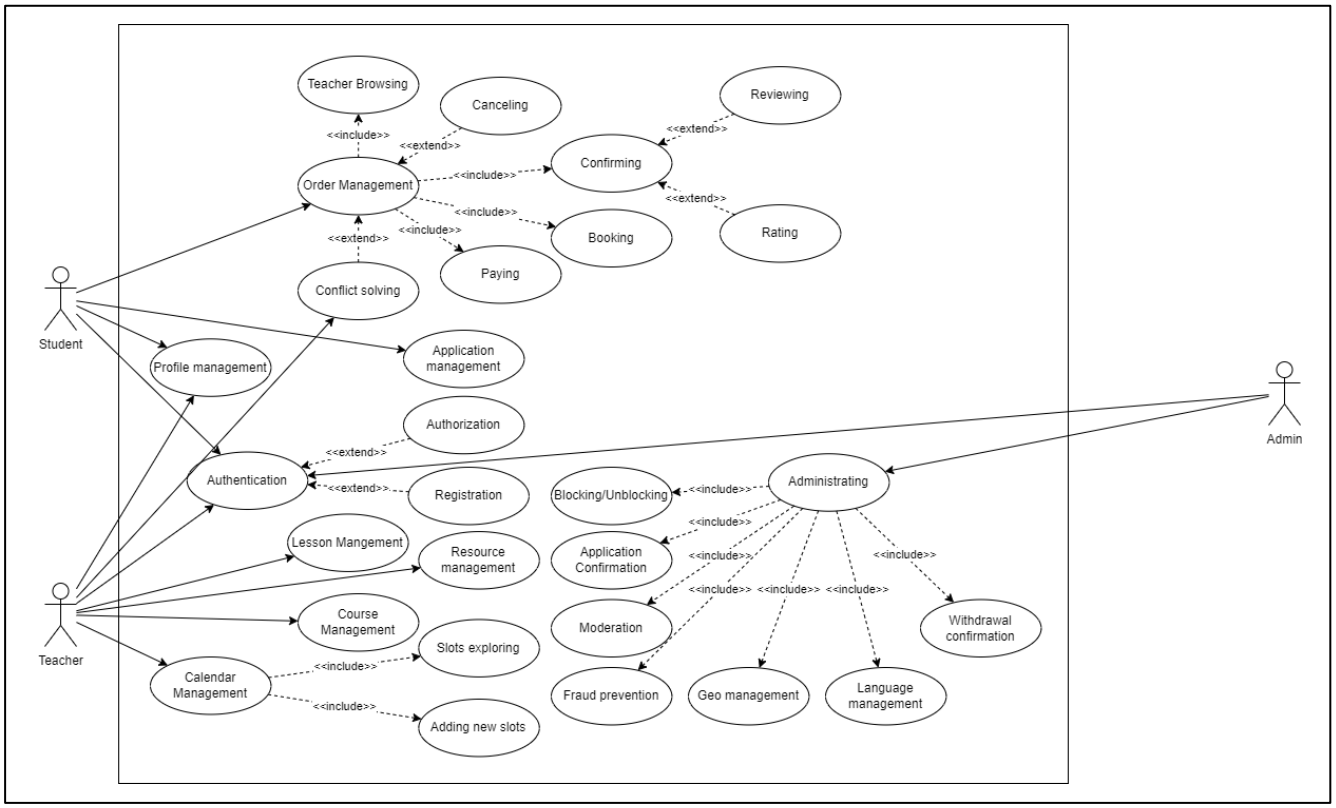


Рисунок 3.1 – Use Case діаграма(рисунок виконано самостійно)

3.2 Проєктування архітектури ПЗ

Серверна частина даної роботи використовує мікросервісну архітектуру у поєднані з чистою архітектурою та архітектурою вертикальних слайсів. Даний архітектурний підхід сприяє модульності та забезпечує масштабованість та гнучкість системи.

Система складається з двух основних мікросервісів, а саме з мікросервісу Identity та мікросервісу Learning, що зберігає всю інформацію про користувачів системи, уроки, курси, заняття, тощо.

Кожен мікросервіс побудований за принципами чистої архітектури, яка гарантує вімежування бізнес-логіки та даних, а також архітектури вертикальних слайсів з використання патерну CQRS(Command Query Responsibility Segregation), що надає гнучкість у розробці та роподіляє відповідальність між командами та запитами на рівні кожної моделі, що підвищує читабельність та зрозумілість системи.

Для комунікації між мікросервісами було обрано патерн Шина(Bus) за допомогою RabbitMQ – надійний та легкомасштабований з великою кількістю розширен та функціоналу брокер повідомлень, котрий надає великий асортимент управління повідомленнями та забезпечує відмовостійкість.

Серверний додаток було розроблено за допомогою кросплатформенного фреймворку ASP.NET Core для доступу до даних використовувалася ORM EntityFramework Core. Для авторизації використувався токен підхід.

Клієнтський додаток було розроблено на базі фреймворку Angular мовою TypeScript. На рисунку 3.2 можна побачити діаграму розгортання даної системи.

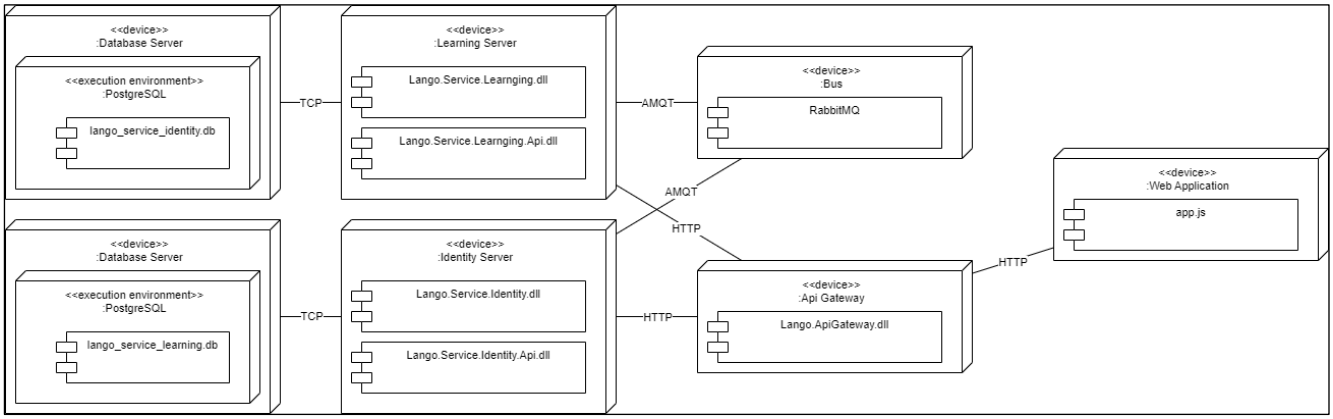


Рисунок 3.2 – Діаграма розгортання(рисунок виконано самостійно)

3.3 Проєктування структури зберігання даних

Під час аналізу структури зберігання було виділено наступні сутності: review, reaction, message, chat, topic, user, country, transaction, problem, schedule, group, lesson, application, plan, seminar, resource, language, order, objective, course; та були відображені на ER-діаграмі(див. рис. 3.3).

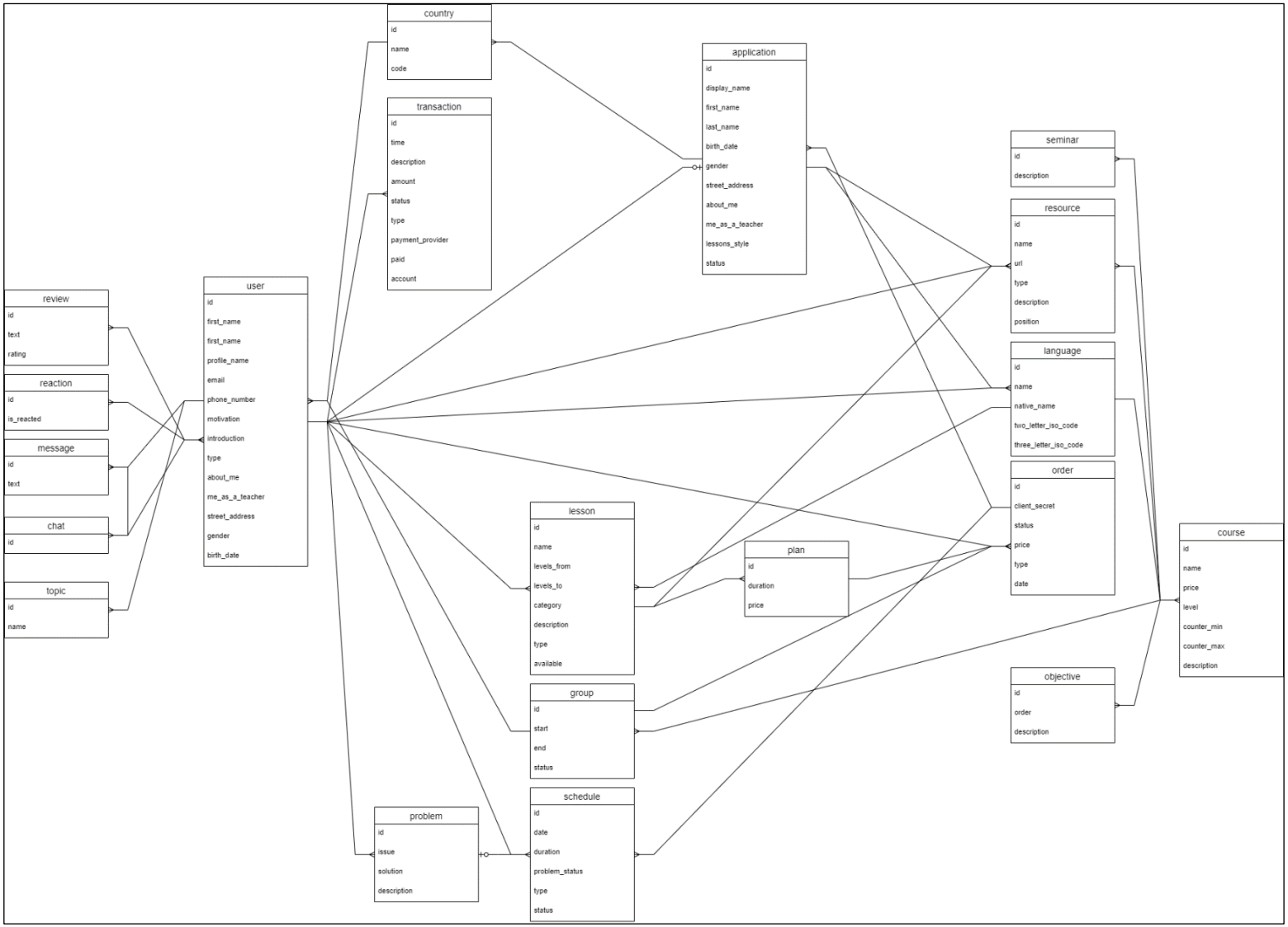


Рисунок 3.3 – ER-діаграма(рисунок виконано самостійно)

З діаграми видно, що

Сутність users:

- має зв’язок один до багатьох з сутністю lessons, оскільки кожен користувач з типом вчитель може створити багато уроків;
- має зв’язок один до багатьох з сутністю courses, оскільки кожен користувач з типом вчитель може створити багато курсів;
- має зв’язок один до багатьох з сутністю problems, оскільки кожен користувач може створити проблему на проблемне заняття;
- має зв’язок один до багатьох з сутністю schedules, оскільки кожен користувач має багато розкладів;
- має зв’язок один до одного з сутністю applications, оскільки кожен користувач може подати заявку на становлення вчителем лише один раз ;
- має зв’язок багато до багатьох з сутністю reactions, оскільки кожен користувач має залишити декілька реакцій на багатьох інших користувачів ;
- має зв’язок багато до багатьох з сутністю reviews, оскільки кожен користувач має залишити декілька відгуків на багатьох інших користувачів ;
- має зв’язок один до багатьох з сутністю topics, оскільки кожен користувач має декілька тем для вивчення мови ;
- має зв’язок один до багатьох з сутністю messages, оскільки повідомлення може бути написане лише одним користувачем ;
- має зв’язок багато до багатьох з сутністю chats, оскільки кожен користувач має бає багато чатів і в кожному чаті присутні декілька користувачів ;
- має зв’язок багато до багатьох з сутністю language, оскільки кожен користувач може знати декілька мов за різним призначенням на різний рівень.

Сутність lessons:

– має зв’язок один до багатьох з сутністю plans, оскільки кожен урок скалдається з багатьох планів з різною тривалістю.

Сутність plans:

– має зв’язок один до багатьох з сутністю orders, оскільки на кожен план може оформлюватися багато замовлень.

Сутність courses:

– має зв’язок один до багатьох з сутністю seminars, бо кожен курс складається з декількох семінарів;
– має зв’язок один до багатьох з сутністю objectives, бо кожен курс має якісь цілі ;
– має зв’язок один до багатьох з сутністю groups, бо на кожний курс відбувається по декілька наборів груп.

Сутність groups:

– має зв’язок один до багатьох з сутністю orders, бо для того щоб потрапити в групу кожен користувач у групі повинен оформити замовлення.

Сутність schedules:

– має зв’язок один до багатьох з сутністю problems, бо на кожний розклад вчитель та учень можуть поскаржитися на проблему.

Сутність chats:

– має зв’язок один до багатьох з сутністю messages, бо кожний чат має більше ніж одне повідомлення.

Сутність languages:

– має зв’язок один до багатьох з сутністю courses, бо мова може бути використана для багатьох курсів;
– має зв’язок один до багатьох з сутністю lessons, бо мова може бути використана для багатьох уроків.

Сутність countries:

– має зв’язок один до багатьох з сутністю users, бо в одній країні можуть проживати та перебувати безліч користувачів;

– має зв’язок один до багатьох з сутністю applications, за тією ж причиною.

Сутність orders:

– має зв’язок один до багатьох з сутністю schedules, бо під кожне замовлення підв’язується до одного або декількох розкладів в залежності від типу замовлення;
– має зв’язок один до багатьох з сутністю transactions, оскільки під замовлення може бути підв’язано декілька транзакцій, таких як оплата, повернення, тощо.

В рамках розробки освітньої платформи для вивчення іноземних мов, PostgreSQL обрана як система управління базами даних (СУБД) для зберігання та обробки даних.

PostgreSQL – це потужна, об'єктно-реляційна СУБД з відкритим кодом, що славиться своєю надійністю, гнучкістю та широким набором функцій.

PostgreSQL була обрана оскільки вона гарантує:

– надійність та стабільність: PostgreSQL відома своєю стабільністю та цілісністю даних. Вона суворо дотримується стандартів SQL, забезпечуючи точність та надійність даних;
– розширена функціональність: PostgreSQL підтримує різноманітні типи даних, включаючи JSON, геопросторові дані та інші, що забезпечує гнучкість у роботі з різною інформацією;
– масштабованість та продуктивність: PostgreSQL здатна обробляти великі обсяги даних та забезпечувати високу продуктивність навіть при значному навантаженні;
– безпека: PostgreSQL пропонує розширені механізми контролю доступу та шифрування для забезпечення безпеки даних;

відкритий код та активна спільнота: PostgreSQL - це безкоштовна СУБД з відкритим кодом, що має велику та активну спільноту розробників, яка забезпечує постійний розвиток та підтримку

На базі ER-діаграми та парадигми реляційних СУБД була побудована логічна схема даних для PostgreSQL(див. рис. 3.4).

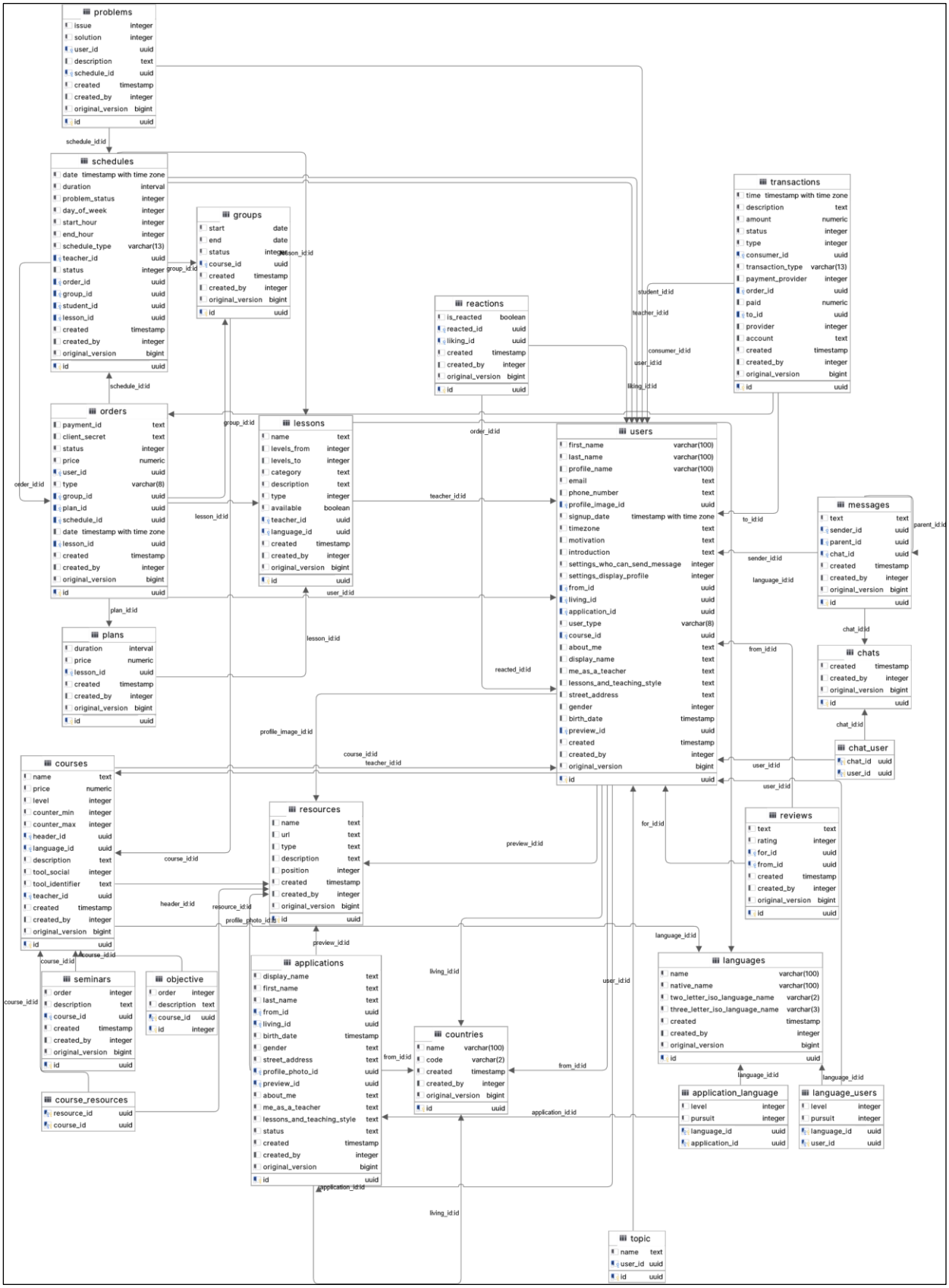


Рисунок 3.4 – Схема бази даних(рисунок виконано самостійно)

3.4 Приклади найцікавіших алгоритмів та методів

Одним із прикладів цікавих алгоритмів може бути пагінація на стороні серверу:

```
public static async Task<ListResultModel<T>> ApplyPagingAsync<T>(
this IQueryable<T> collection,
int page = 1,
int pageSize = 10,
CancellationToken cancellationToken = default
)
where T : notnull
{
if (page <= 0) page = 1;
if (pageSize <= 0) pageSize = 10;
var isEmpty = await collection.AnyAsync(cancellationToken: cancellationToken) == false;
if (isEmpty) return ListResultModel<T>.Empty;
var totalItems = await collection.CountAsync(cancellationToken: cancellationToken);
var totalPages = (int)Math.Ceiling((decimal)totalItems / pageSize);
var data = await collection.Limit(page, pageSize).ToListAsync(cancellationToken: cancellationToken);
return ListResultModel<T>.Create(data, totalItems, page, pageSize);
}
```

Цей код реалізує асинхронний метод ApplyPagingAsync, який застосовує пагінацію до колекції IQueryable<T>. Метод приймає параметри page і pageSize, що визначають поточну сторінку та розмір сторінки відповідно, а також CancellationToken для керування асинхронними операціями. Якщо значення page або pageSize недопустимі, вони встановлюються на 1 і 10 відповідно. Метод перевіряє, чи не є колекція порожньою, і якщо так, повертає порожню модель результату. Далі обчислюється загальна кількість елементів у колекції та загальна кількість сторінок. Після цього вибираються елементи для поточної сторінки з колекції і повертається модель результату, що містить вибрані дані, загальну кількість елементів, поточну сторінку та розмір сторінки.

Наступним цікавим прикладом може бути отримання вчителей

```
public async Task<GetTeachersResponse> Handle(GetTeachers request, CancellationToken cancellationToken)
{
var userId = _securityContextAccessor.ParseUserId();
```

```
var query = _context.Users.OfType<Teacher>()
.AsNoTracking()
.Where(x => x.Lessons.Any() || x.Courses.Any());
if (!string.IsNullOrWhiteSpace(request.SearchText))
{
query = query.Where(
x => (x.FirstName + x.LastName + x.ProfileName).Contains(
request.SearchText,
StringComparison.CurrentCultureIgnoreCase));
}
if (request.Languages != null && request.Languages.Any())
{
query = query
.Where(
x => x.Languages.Any(
y => request.Languages.Contains(y.Language.Name) &&
(y.Pursuit == Pursuit.Speaking || y.Pursuit == Pursuit.Teaching)));
}
if (request.Countries != null && request.Countries.Any())
{
query = query.Where(x => request.Countries.Contains(x.From.Name));
}
if (request.Liked)
{
query = query.Where(x => x.Liking.Any(y => y.IsReacted && y.ReactedId
== userId));
}
var entities = await query
.ProjectTo<TeacherBriefDto>(_mapper.ConfigurationProvider,        new
{userId})
.ApplyPagingAsync(request.Page, request.PageSize, cancellationToken:
cancellationToken);
return new GetTeachersResponse(entities);
}
}
```

У цьому коді описується процес обробки запиту на отримання списку викладачів, що відповідають певним критеріям. Спочатку визначаються структури даних для запиту та відповіді: GetTeachers, що містить параметри пошуку (текст, країни, мови, чи відмічені як улюблені) і відповідає типу ListQuery, та GetTeachersResponse, який містить результати у вигляді списку викладачів. Потім визначається обробник запиту GetTeachersHandler, що реалізує інтерфейс IQueryHandler. У класі обробника визначаються залежності (контекст бази даних, маппер, доступ до контексту безпеки), які ініціалізуються через конструктор. У методі Handle обробляється запит: отримується ідентифікатор користувача, формується базовий запит до бази даних для вибірки викладачів, які мають уроки або курси, а потім додаються додаткові фільтри відповідно до

параметрів запиту (пошук за текстом, мовами, країнами, та чи є вони улюбленими). Після застосування фільтрів запит проєктується до DTO TeacherBriefDto і результати з пагінацією повертаються у відповіді GetTeachersResponse.

Далі можна навести алгоритм отримання статистик учителя:

```
public record GetTeacherStatistics(Guid Id, ICollection<Statistics> Statistics)
: IQuery<GetTeacherStatisticsResponse>;
```

GetTeacherStatistics: Це рекорд, який представляє запит на отримання статистики для викладача, містить ідентифікатор викладача (Id) та колекцію статистик (Statistics).

```
public record GetTeacherStatisticsResponse(IDictionary<Statistics, double> Body);
```

GetTeacherStatisticsResponse: Це рекорд, який представляє відповідь на запит і містить словник статистик та їхніх значень.

```
public async Task<GetTeacherStatisticsResponse> Handle(
GetTeacherStatistics request,CancellationToken cancellationToken){
var dict = new Dictionary<Statistics, double>();
if (request.Statistics.Any(x => x == Statistics.Rating)){
var rating = await _context.Reviews.Where(x => x.ForId == request.Id)
.Select(x => x.Rating).DefaultIfEmpty()
.AverageAsync(x => x, cancellationToken: cancellationToken);
dict.Add(Statistics.Rating, rating);}
if (request.Statistics.Any(x => x == Statistics.StudentsCount))
{ar studentsCount = await _context.Orders.OfType<LessonOrder>()
.Where(x => x.Plan.Lesson.TeacherId == request.Id)
.Select(x => x.UserId).Union(_context.Orders.OfType<CourseOrder>()
.Where(x => x.Group.Course.TeacherId == request.Id)
.Select(x => x.UserId)).Distinct()
.CountAsync(cancellationToken: cancellationToken);
dict.Add(Statistics.StudentsCount, studentsCount);
}
if (request.Statistics.Any(x => x == Statistics.LessonsCount))
{
var lessonsCount = await _context.Schedules.OfType<LessonSchedule>()
.Where(
x => x.Order.Plan.Lesson.TeacherId == request.Id &&
x.Status == ScheduleStatus.Completed)
.CountAsync(cancellationToken: cancellationToken) +
```

```
await _context.Schedules.OfType<CourseSchedule>()
.Where(
x => x.Order.Group.Course.TeacherId == request.Id &&
x.Status == ScheduleStatus.Completed)
.CountAsync(cancellationToken: cancellationToken);
dict.Add(Statistics.LessonsCount, lessonsCount);
}
if (request.Statistics.Any(x => x == Statistics.Attendance))
{
var lessonQuery = _context.Orders.OfType<LessonOrder>().Where(x =>
x.Plan.Lesson.TeacherId == request.Id);
var countOfProblematicLessons = await lessonQuery.Where(
x => x.Schedule.Problems.Any(
y => y.Issue == Issue.TeacherDidNotAttendButGavePriorNotice ||
y.Issue ==
Issue.TeacherDidNotAttendAndDidNotGivePriorNotice))
.CountAsync(cancellationToken: cancellationToken);
var countOfLessons = await lessonQuery.CountAsync(cancellationToken:
cancellationToken);
var courseQuery = _context.Orders.OfType<CourseOrder>().Where(x =>
x.Group.Course.TeacherId == request.Id);
var countOfProblematicCourses = await courseQuery.SelectMany(
x => x.Schedules.Where(
y => y.Problems.Any(
z => z.Issue ==
Issue.TeacherDidNotAttendAndDidNotGivePriorNotice ||
z.Issue ==
Issue.TeacherDidNotAttendButGavePriorNotice)))
.CountAsync(cancellationToken: cancellationToken);
var countOfCourses = await courseQuery.SelectMany(x => x.Schedules)
.CountAsync(cancellationToken: cancellationToken);
var attendance = 0d;
if (countOfCourses + countOfLessons == 0){
attendance = 0d;
}
else{
}
```

У методі Handle обробляється запит: для кожного типу статистики, що запитується (наприклад, рейтинг, кількість студентів, кількість уроків, відвідуваність та відповідь на повідомлення), здійснюється відповідний запит до бази даних, обробляються результати та додаються до словника. Після обробки всіх запитуваних статистичних даних словник повертається у вигляді об'єкта GetTeacherStatisticsResponse.

Також можна продемонструвати алгоритм валідації запиту, який відбувається на базі бібліотеки FluentValidation:

```
public class CreateOrderValidator : AbstractValidator<CreateOrder>
{
```

```
public CreateOrderValidator(IDateTimeProvider dateTimeProvider)
{
var comparator = (ZonedDateTime a, ZonedDateTime b) =>
{
var result = ZonedDateTime.Comparer.Instant.Compare(a, b);
return result > 0;
};
When(
x => x.Type == OrderType.Lesson,
() =>
{
RuleFor(x => x.PlanId)
.Must(x => x != null && x != Guid.Empty)
.WithMessage("PlanId must not be null or empty.");
RuleFor(x => x.Date)
.Must(x => x != null)
.WithMessage("LessonId must not be null or empty.")
.Must(
x => comparator(
x!.Value,
dateTimeProvider.ZonedUtcNow))
.WithMessage("Date must be greater than current date.");
});
When(
x => x.Type == OrderType.Course,
() =>
{
RuleFor(x => x.GroupId)
.Must(x => x != null && x != Guid.Empty)
.WithMessage("GroupId must not be null or empty.");
});
}
}
```

Клас CreateOrderValidator успадковується від AbstractValidator<CreateOrder> і здійснює валідацію об'єкта CreateOrder. У конструкторі цей валідатор приймає IDateTimeProvider для отримання поточного часу. Всередині конструктора створюється компаратор для порівняння двох об'єктів ZonedDateTime. Валідація проводиться залежно від типу замовлення (OrderType). Якщо тип замовлення – Lesson, перевіряється, що PlanId не є порожнім або рівним Guid.Empty, і що Date не є порожнім та більше поточного часу. Якщо тип замовлення – Course, перевіряється, що GroupId не є порожнім або рівним Guid.Empty.

3.5 Створення UI/UX або іншого дизайну системи

Створення UI/UX дизайну є одним із найважливіших етапів розробки будь-якої сучасної цифрової платформи. Особливо це актуально для освітніх платформ,

спрямованих на надання послуг з вивчення іноземних мов. Основне завдання UI/UX дизайну полягає в тому, щоб зробити взаємодію користувача з системою інтуїтивно зрозумілою, ефективною та приємною.

UI (User Interface) та UX (User Experience) дизайни тісно пов'язані між собою, але мають різні акценти. UI-дизайн зосереджується на зовнішньому вигляді та відчуттях від використання інтерфейсу, включаючи кольори, шрифти, розташування елементів і візуальну ієрархію. UX-дизайн, натомість, охоплює ширший спектр питань, пов'язаних із загальним досвідом користувача, зокрема логіку навігації, зручність використання, задоволення від взаємодії з системою та ефективність виконання завдань.

В контексті освітньої платформи для вивчення іноземних мов, правильний підхід до UI/UX дизайну має враховувати різноманітність користувачів, серед яких можуть бути студенти різного віку, викладачі з різними методиками викладання, а також адміністратори платформи. Важливо забезпечити зручний доступ до всіх необхідних функцій, таких як пошук та вибір викладача, проведення онлайн-уроків, доступ до навчальних матеріалів, а також комунікацію між студентами та викладачами.

Далі було розроблено наступні сторінки. Головна сторінка де відображається головна та мотивуюча інформація про систему(див. рис. 3.5).

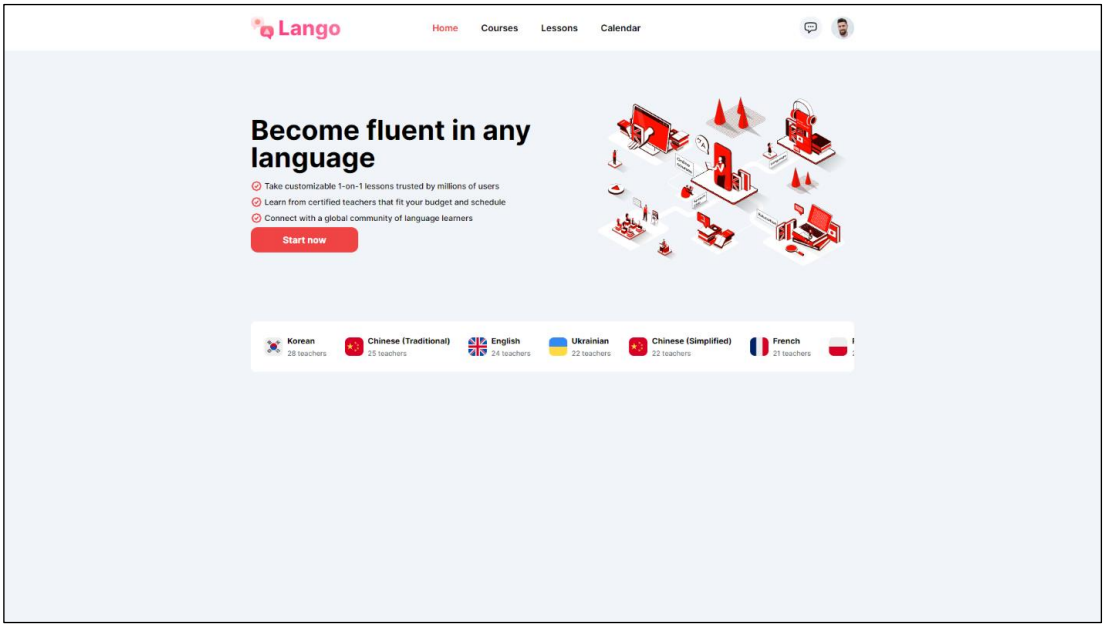


Рисунок 3.5 – Головна сторінка(рисунок виконано самостійно)

Наступним етапом було створення вікна входу(див. рис. 3.6).

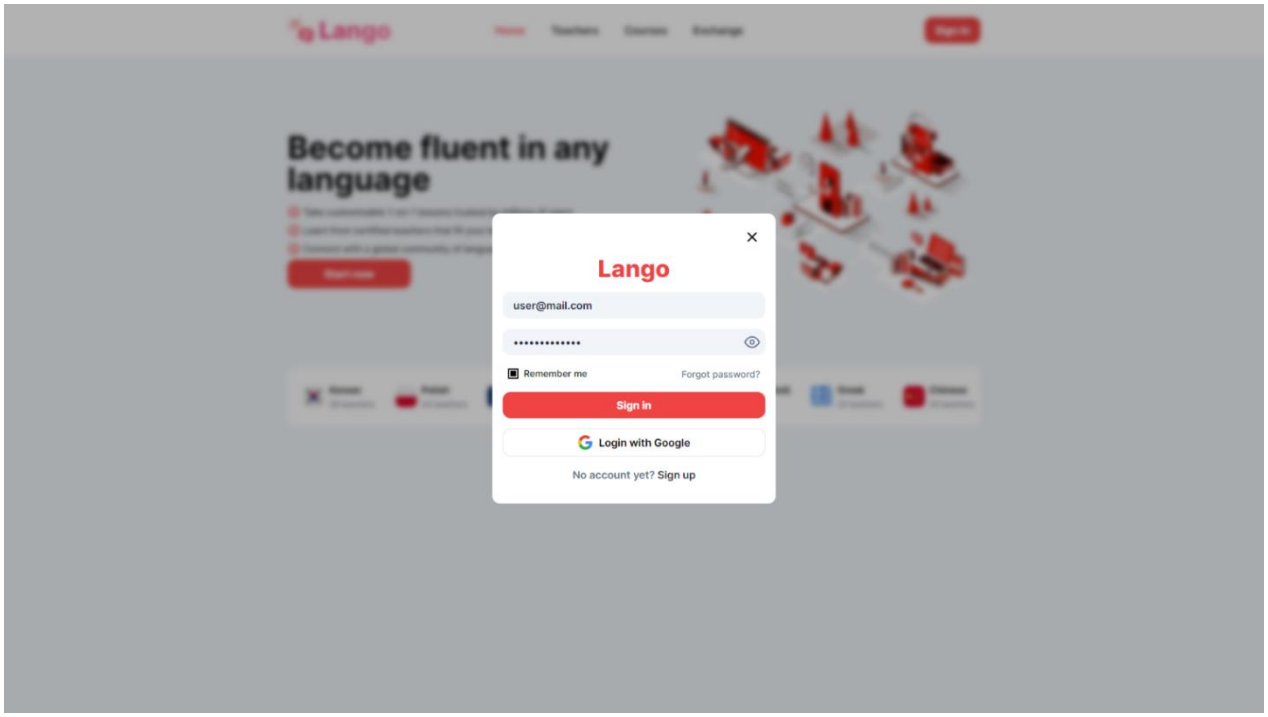


Рисунок 3.6 – Вікно входу(рисунок виконано самостійно)

Система повинна знати мову яку людина хоче вивчати, який рівень, намір, інтерес, де проживає та з якої країни, а також знати які ще додаткові мови людина знає або вивчає(див. рис. 3.7-3.11).

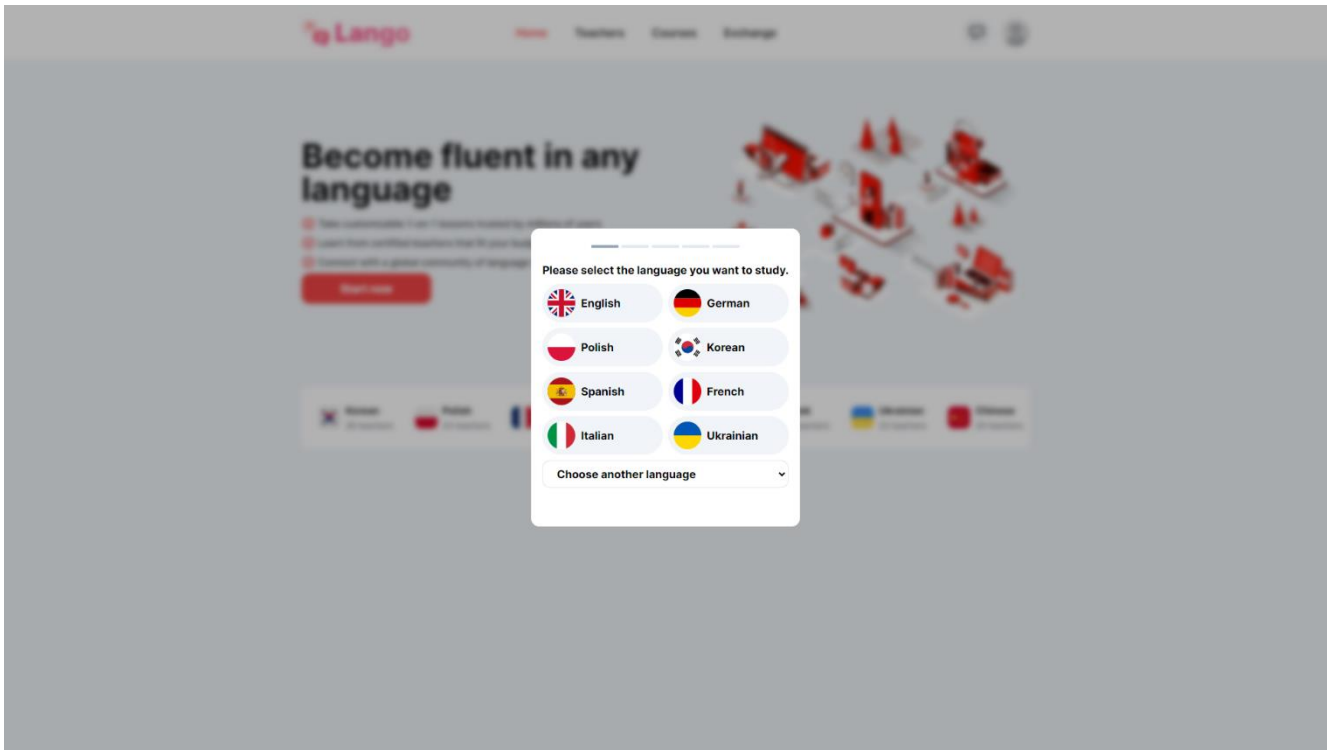


Рисунок 3.7 – Етап обирання мови(рисунок виконано самостійно)

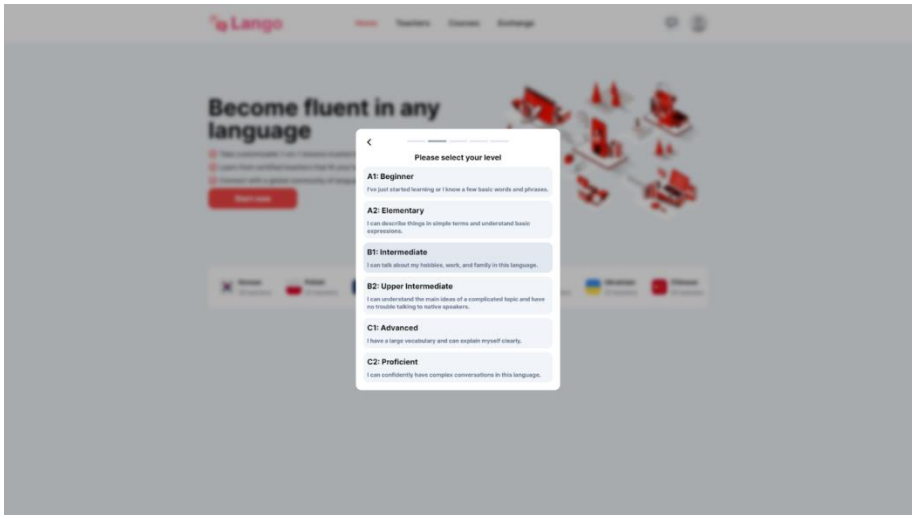


Рисунок 3.8 – Етап обирання рівня(рисунок виконано самостійно)

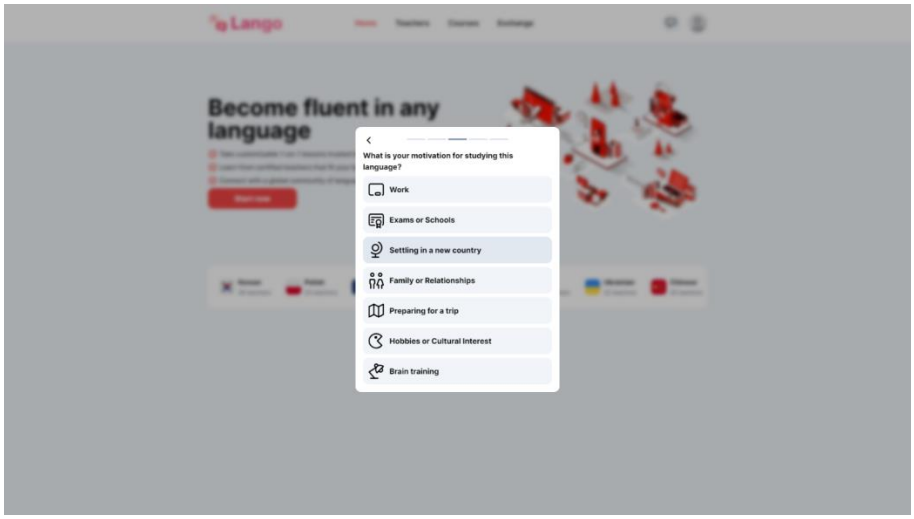


Рисунок 3.9 – Етап обирання мотивації/наміру(рисунок виконано самостійно)

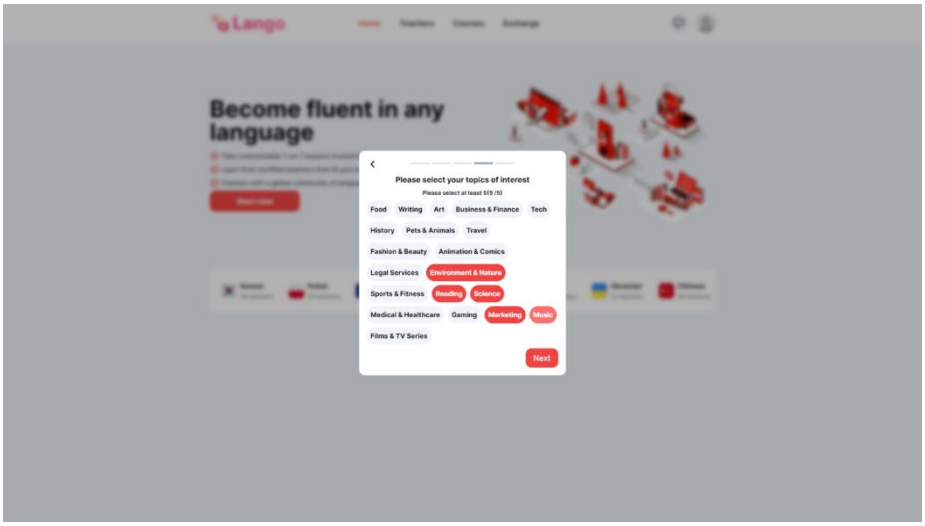


Рисунок 3.10 – Етап обирання інтересів(рисунок виконано самостійно)

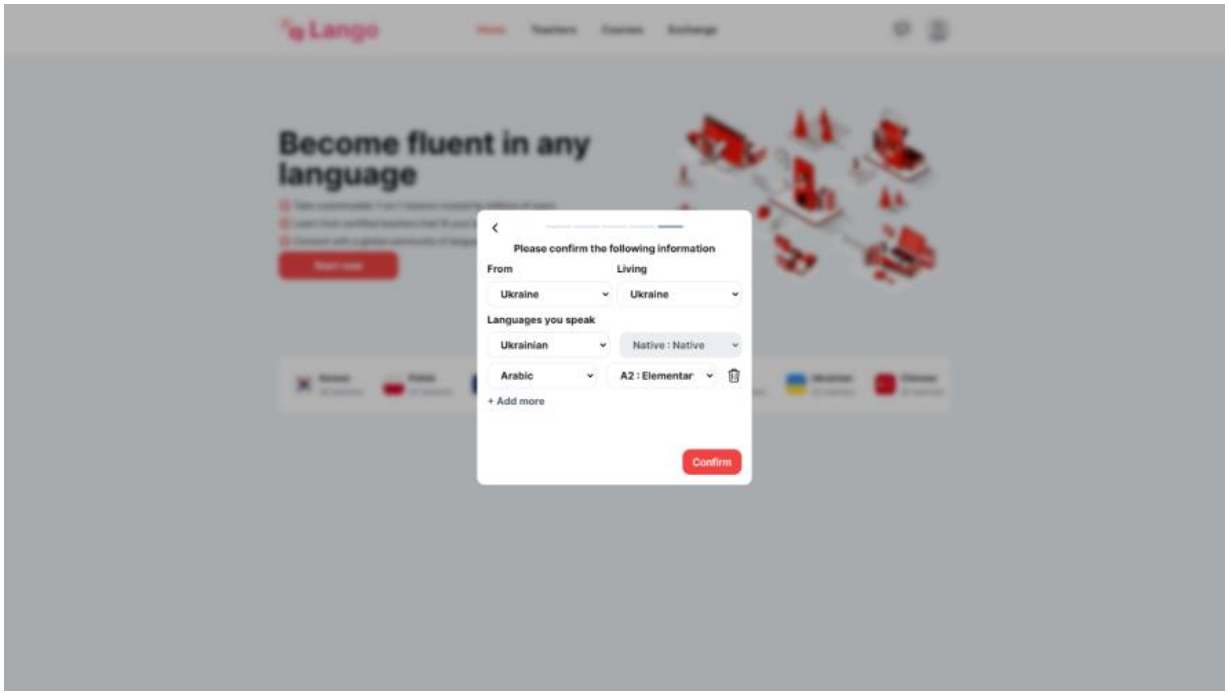


Рисунок 3.11 – Етап обирання країн та інших мов(рисунок виконано самостійно)

Також користувач повинен мати можливість керувати своїм профілем. На рисунку 3.12 відображена головна сторінка профілю з особистою інформацією.

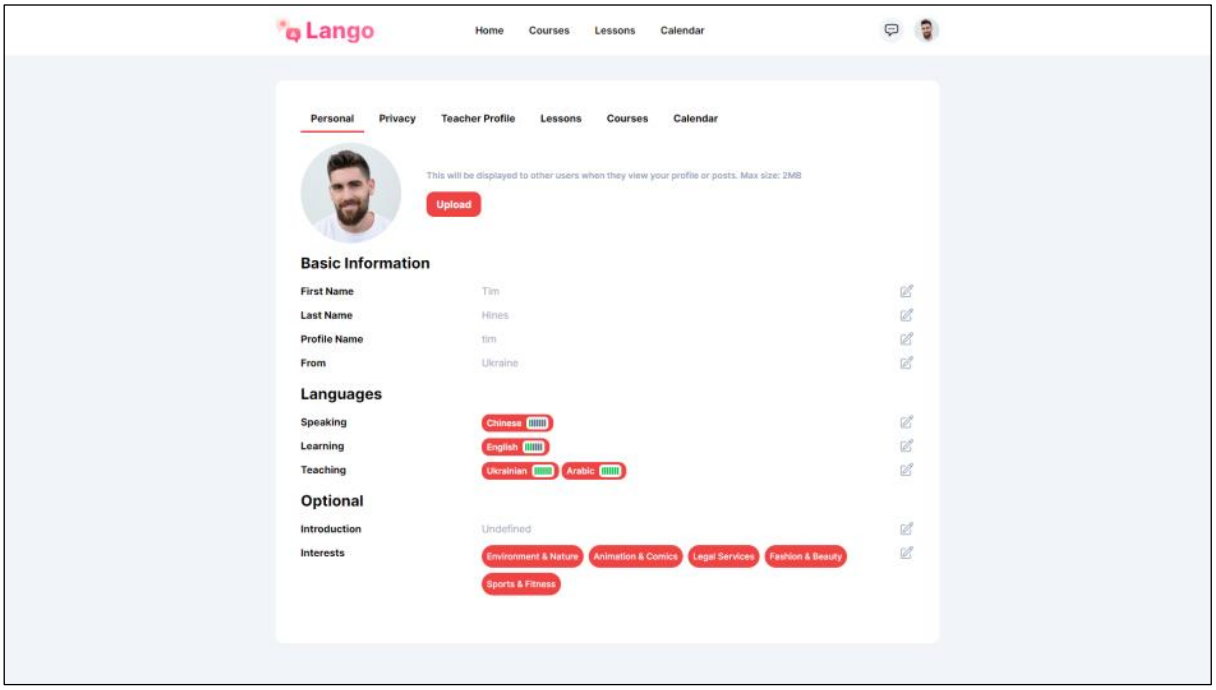


Рисунок 3.12 – Сторінка профілю з головною інформацією(рисунок виконано самостійно)

На рисунку 3.13 відображена приватна інформація та технічні функції.

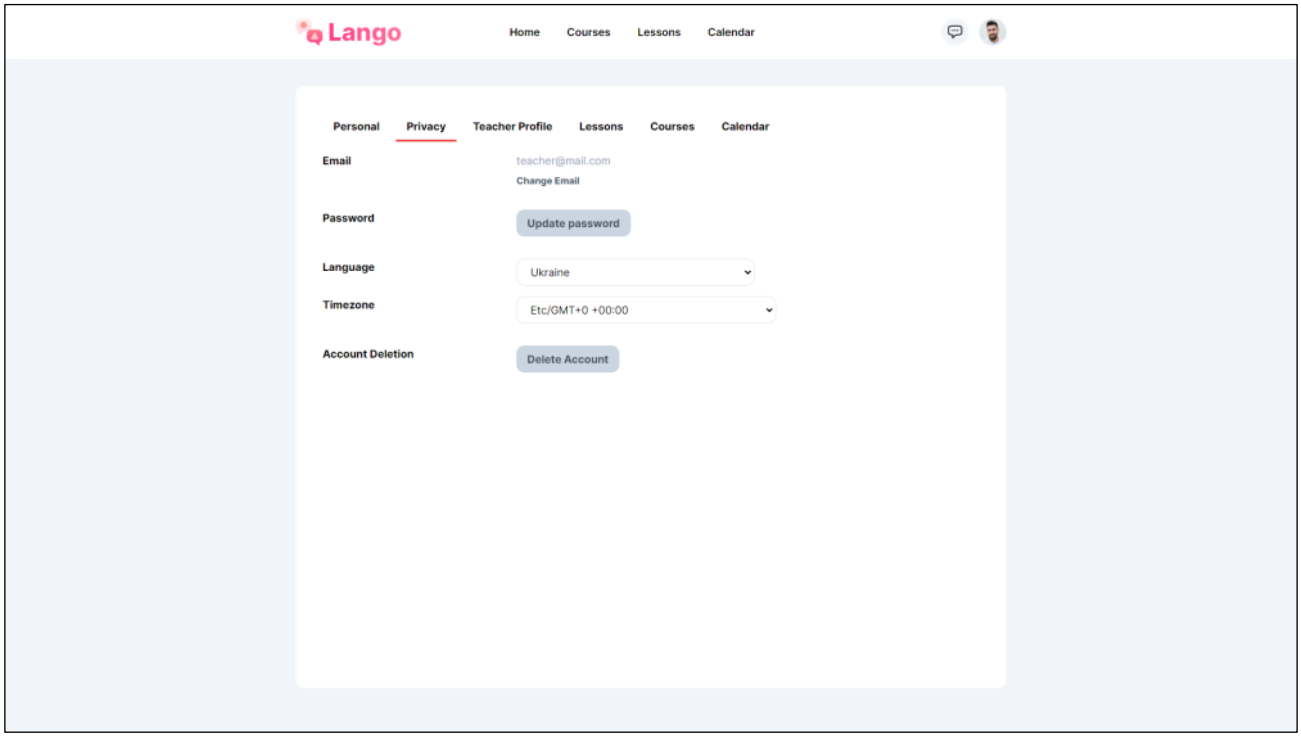


Рисунок 3.13 – Сторінка профілю з приватною та технічною інформацією

На рисунку 3.14 відображена інформація профілю вчителя.

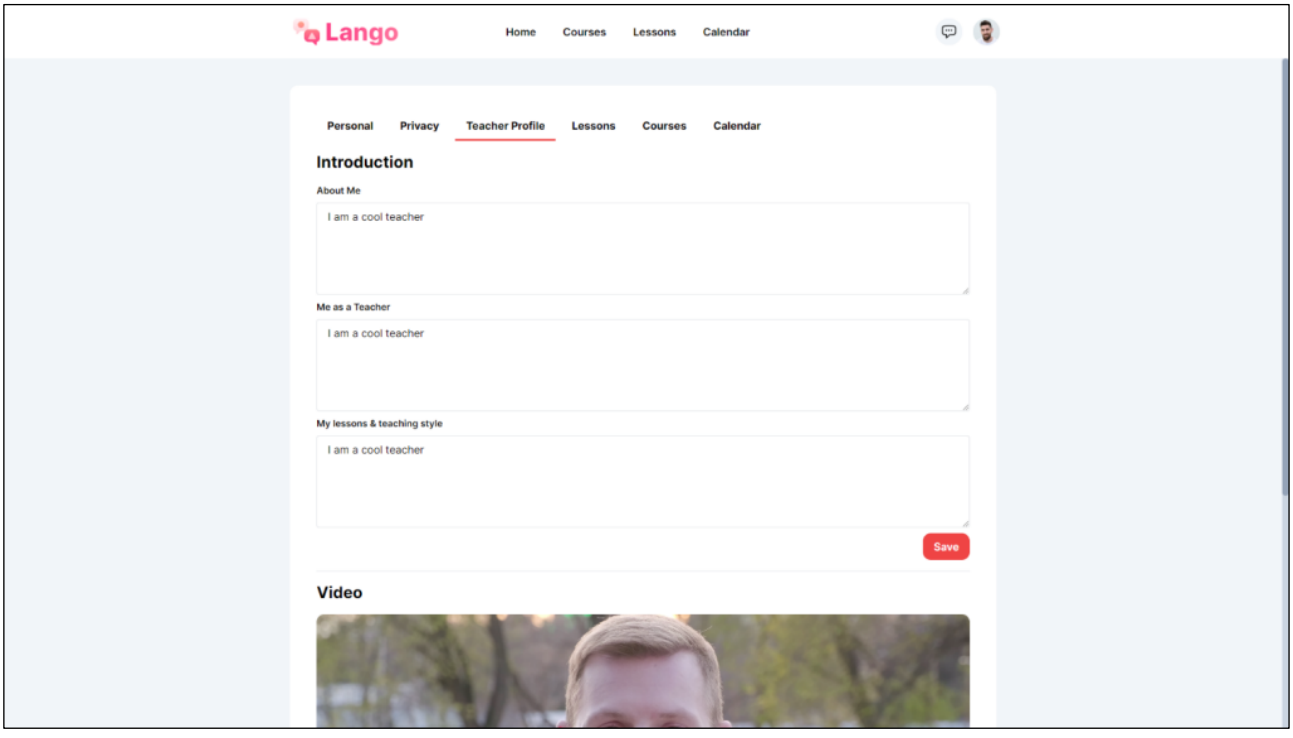


Рисунок 3.14 – Сторінка інформації вчителя(рисунок виконано самостійно)

На рисунку 3.15 відображена інформація з відображенням уроків, що були створені вчителем.

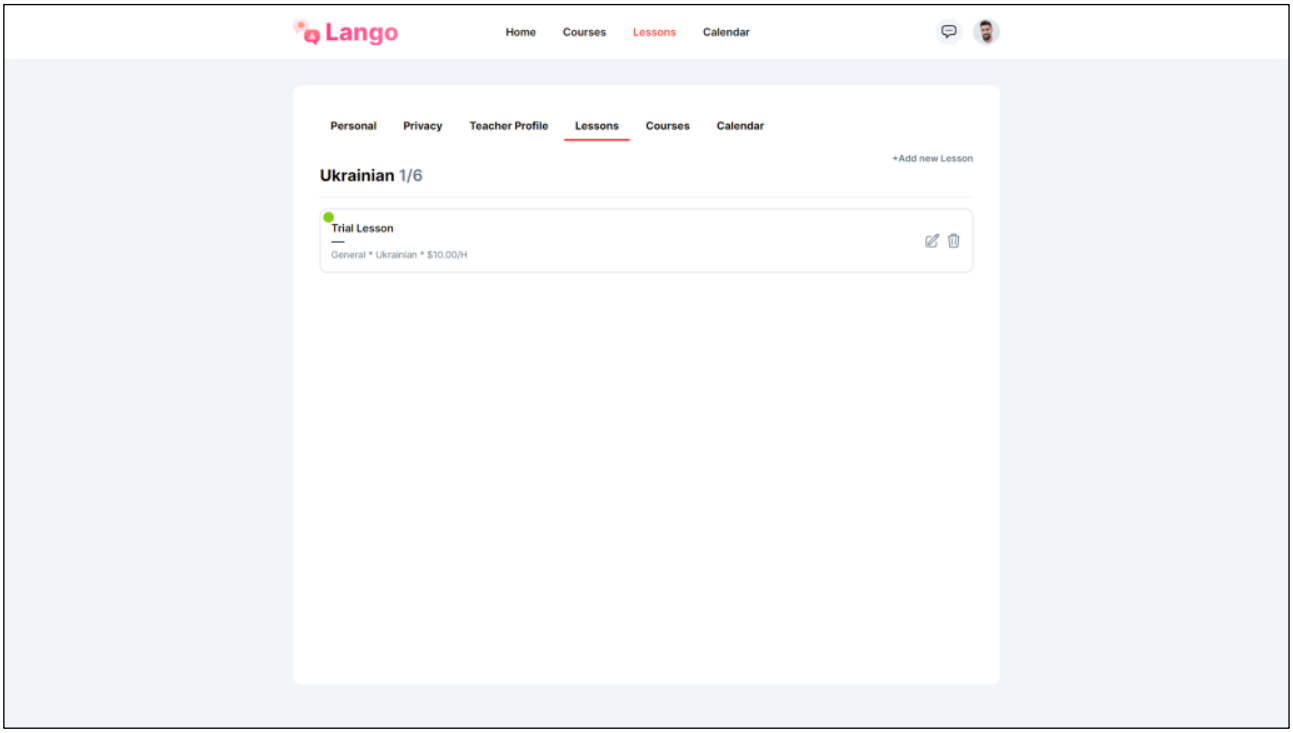


Рисунок 3.15 – Сторінка з уроками вчителя(рисунок виконано самостійно)

На рисунку 3.16 відображене вікно редагування або створення уроку.

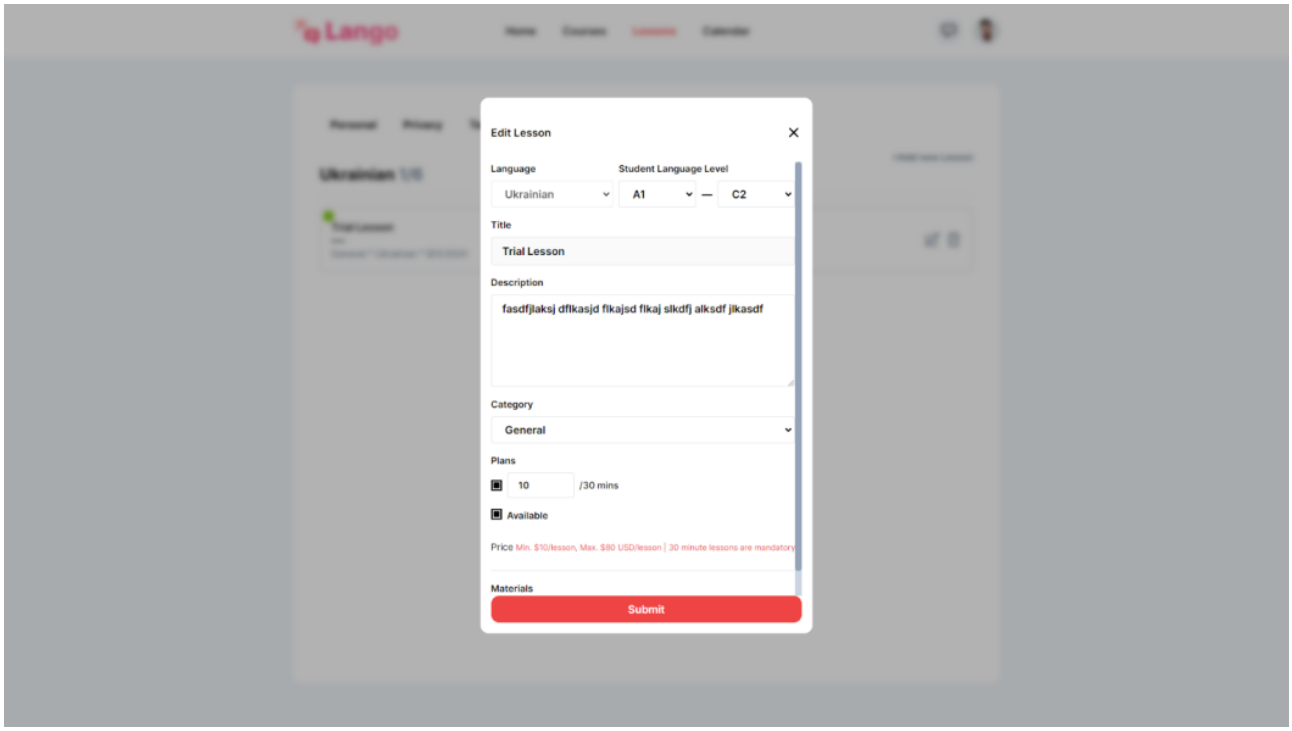


Рисунок 3.16 – Вікно створення/редагування уроку(рисунок виконано самостійно)

На рисунку 3.17 відображене вікно редагування або створення курсу.

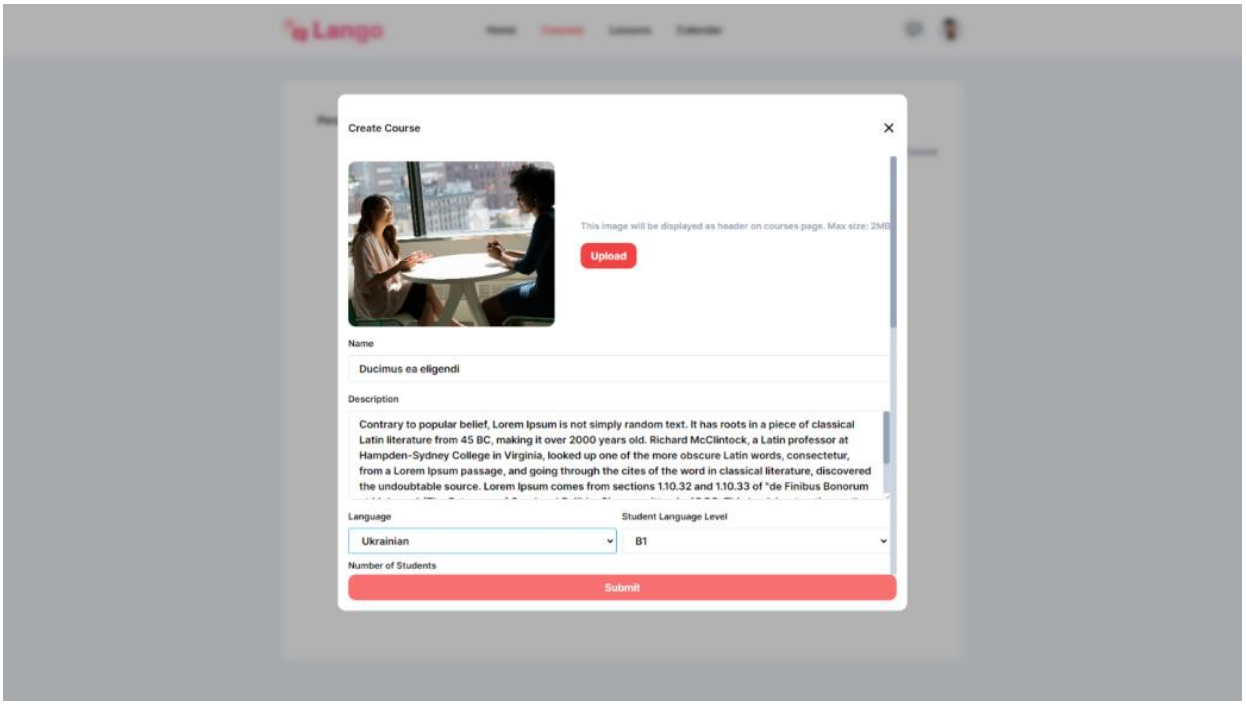


Рисунок 3.17 – Вікно створення/редагування курсу(рисунок виконано самостійно)

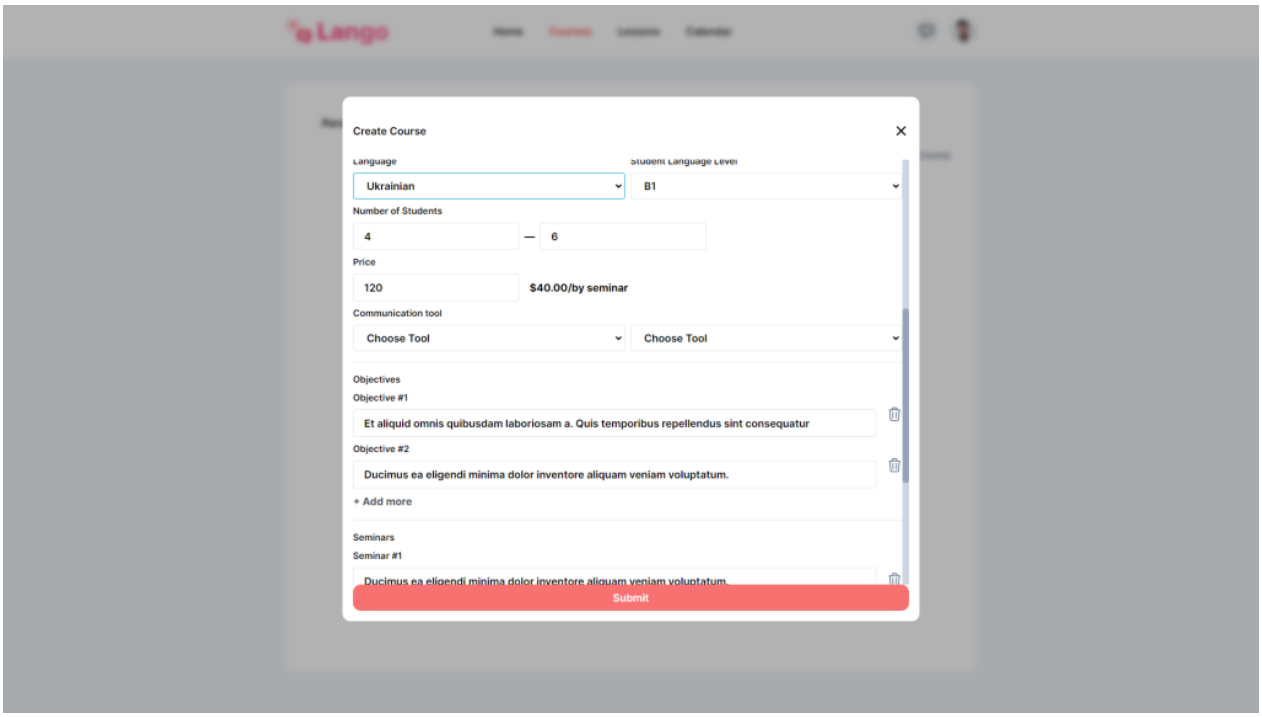


Рисунок 3.18 – Вікно створення/редагування курсу(рисунок виконано самостійно)

На рисунку 3.19 відображено вікно зі списком курсів.

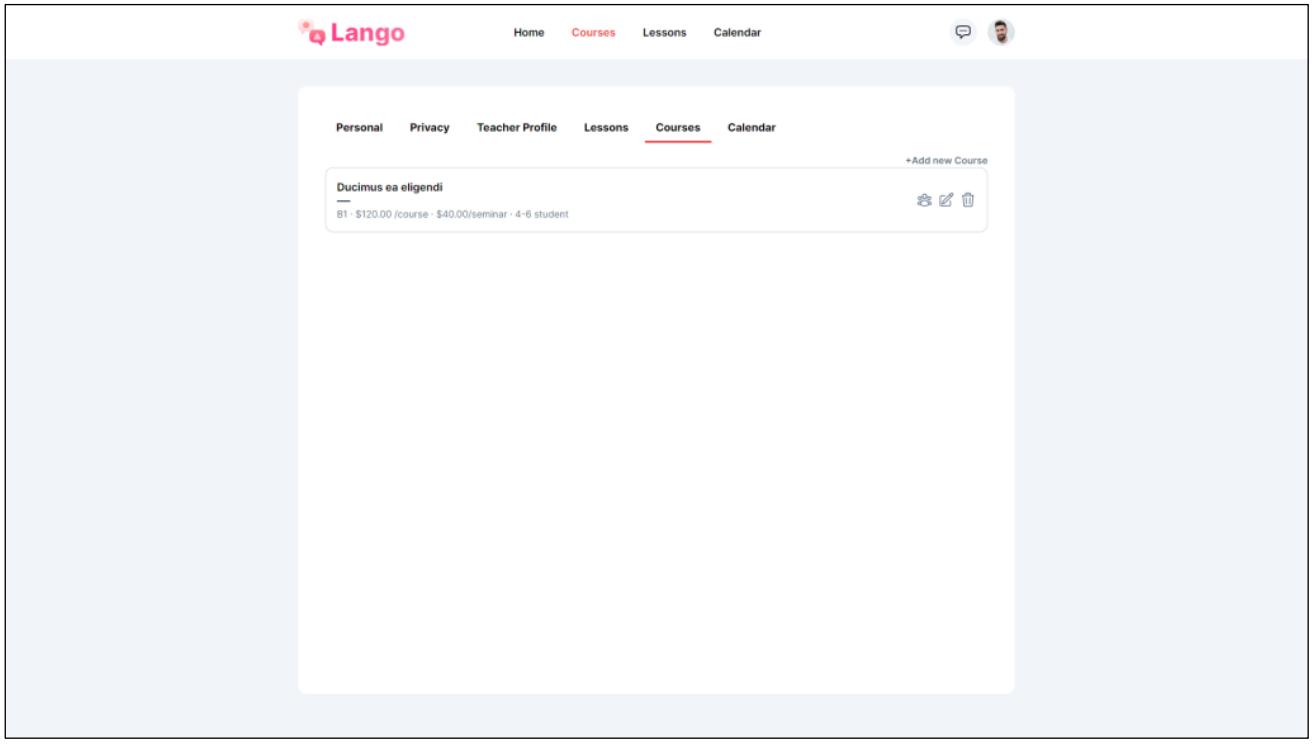


Рисунок 3.19 – Сторінка з курсами вчителя(рисунок виконано самостійно)

На рисунку 3.20 відображеноий календар вчителя, де він може редагувати свій вільний час.

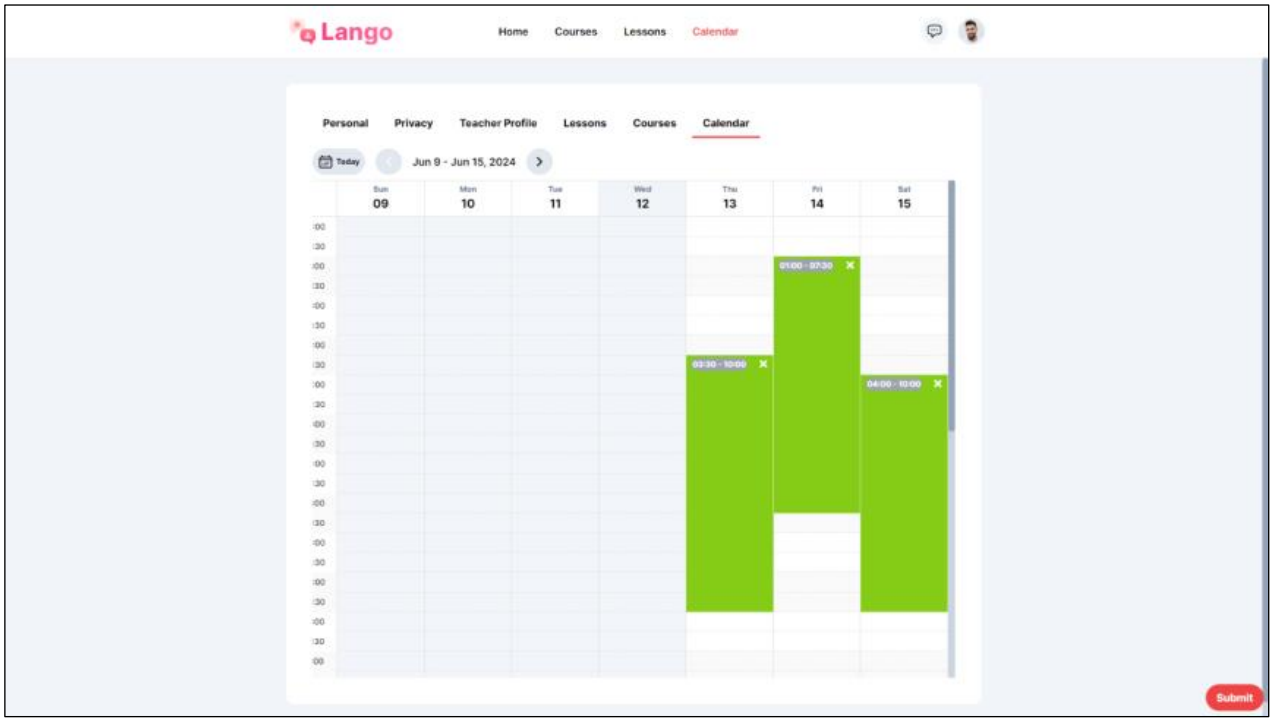


Рисунок 3.20 – Сторінка з календарем вчителя(рисунок виконано самостійно)

Також система повинна надавати можливість зв’язку між студентом та вчителем, тому необхідно розробити чат(див. рис. 3.21).

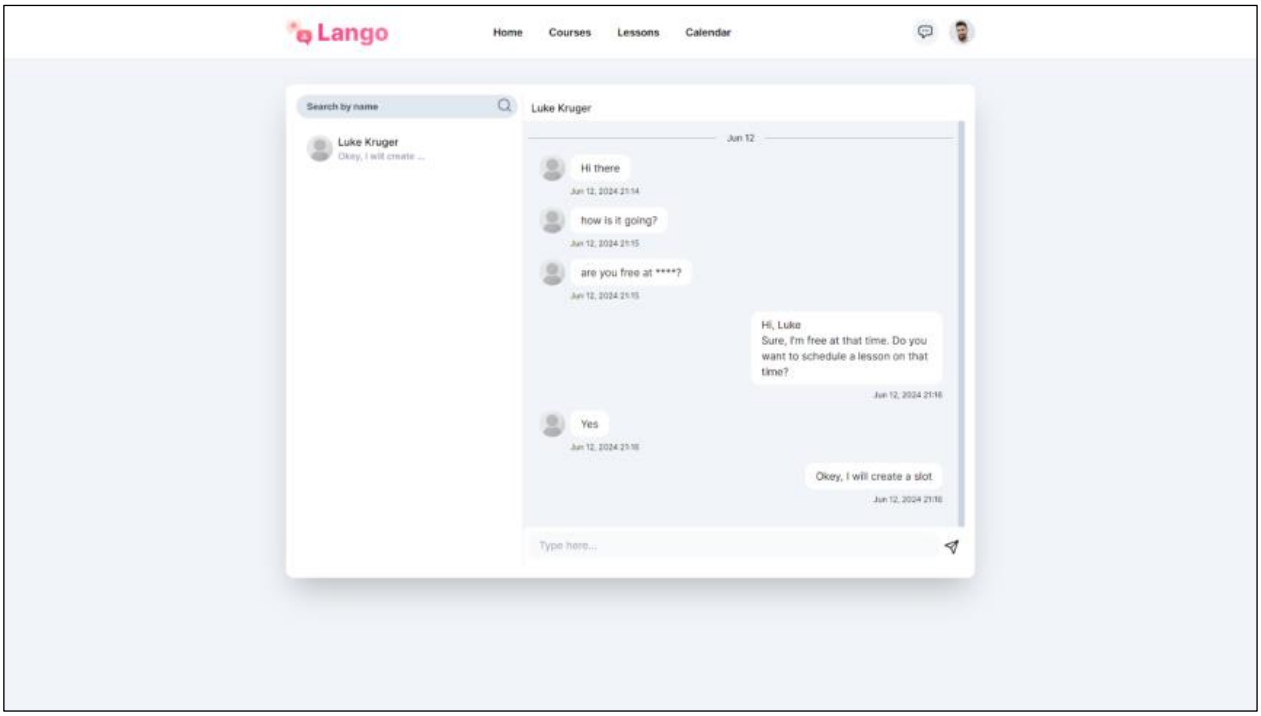


Рисунок 3.21 – Чат(рисунок виконано самостійно)

Перед тим як оформити замовлення користувач повинен знайти вчителя(див. рис. 3.22-3.24) або ж знайти необхідний йому курс(див. рис. 3.25-3.26).

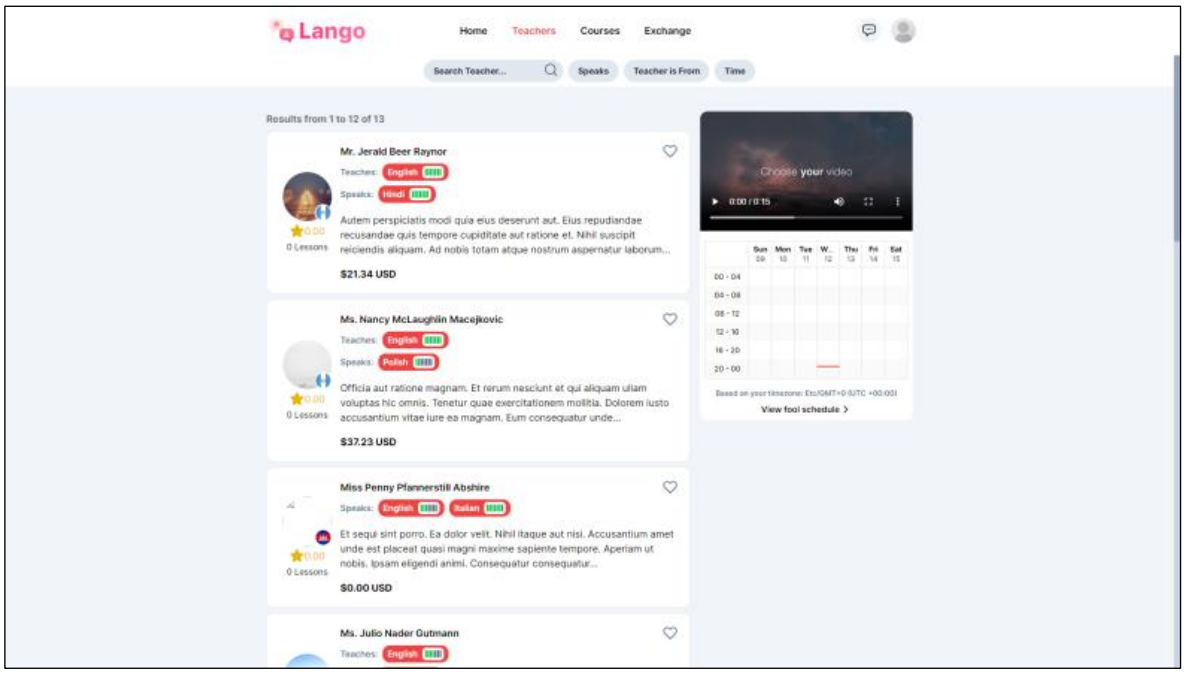


Рисунок 3.22 – Сторінка з відображенням вчителей(рисунок виконано самостійно)

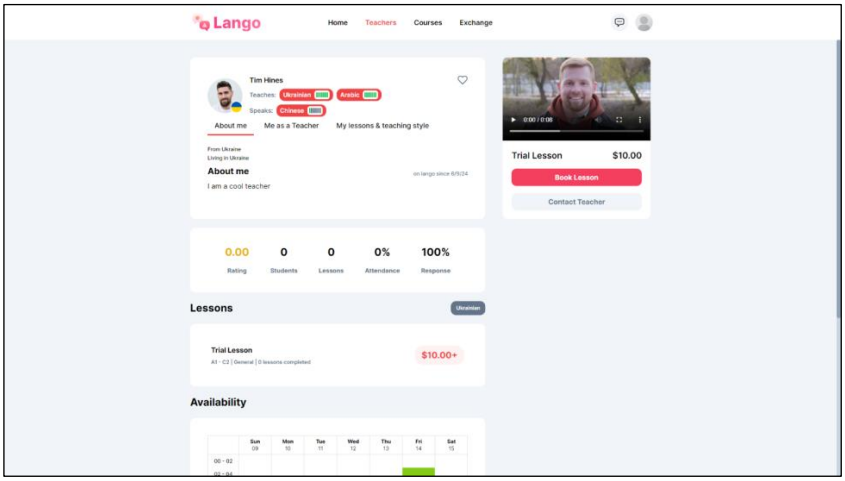


Рисунок 3.23 – Сторінка з відображенням інформації про вчителя(рисунок виконано самостійно)

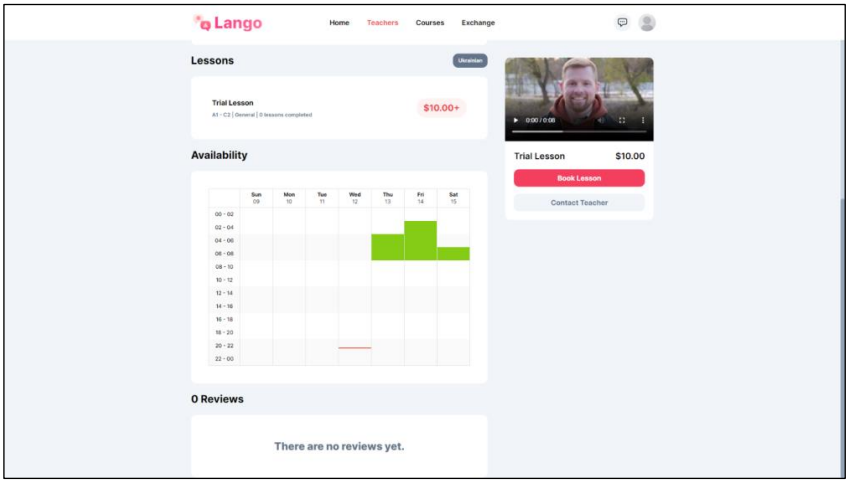


Рисунок 3.24 – Сторінка з відображенням інформації про вчителя(рисунок виконано самостійно)

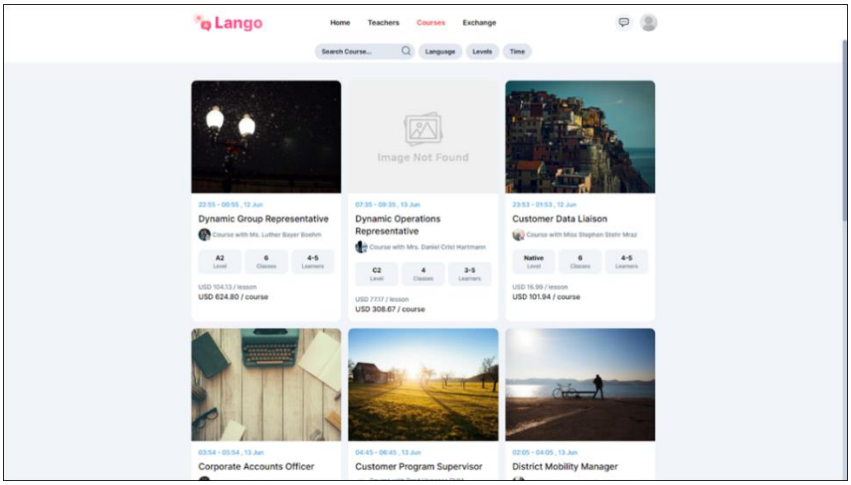


Рисунок 3.25 – Сторінка з відображенням курсів

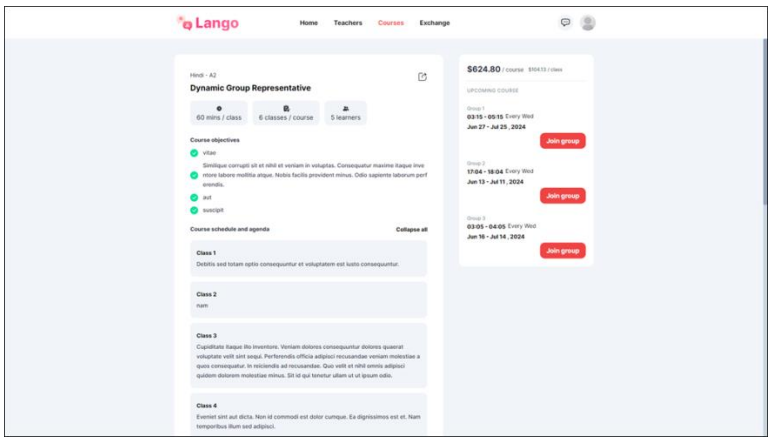


Рисунок 3.26 – Сторінка з відображенням інформації про курс(рисунок виконано самостійно)

Після того, як клієнт обрав вчителя – він може розпочати процес оформлення натиснувши на клавішу «Забронювати урок», після чого розпочнеться процес оформлення замовлення, що відображено на рисунках 3.27-3.30.

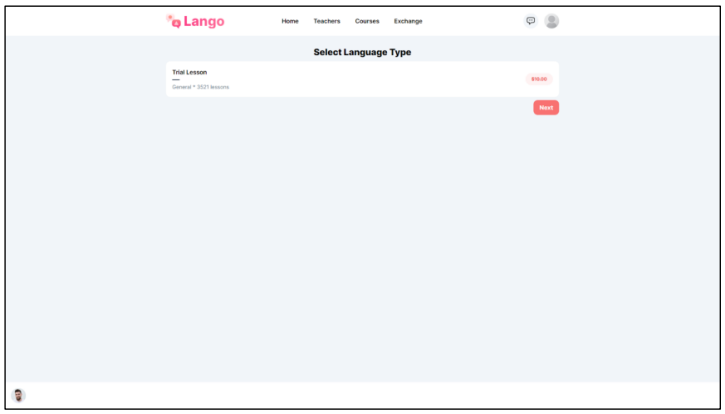


Рисунок 3.27 – Етап вибору уроку(рисунок виконано самостійно)

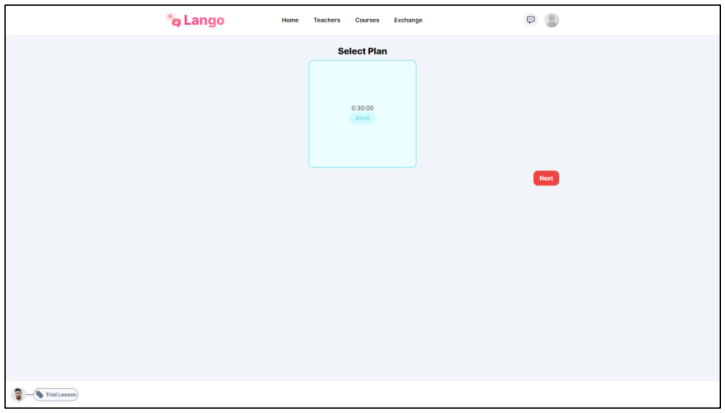


Рисунок 3.28 – Етап вибору плану(рисунок виконано самостійно)

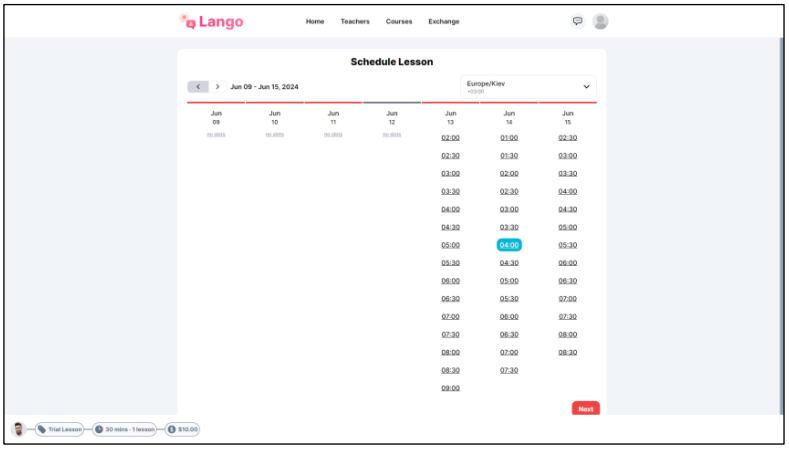


Рисунок 3.29 – Етап вибору дати(рисунок виконано самостійно)

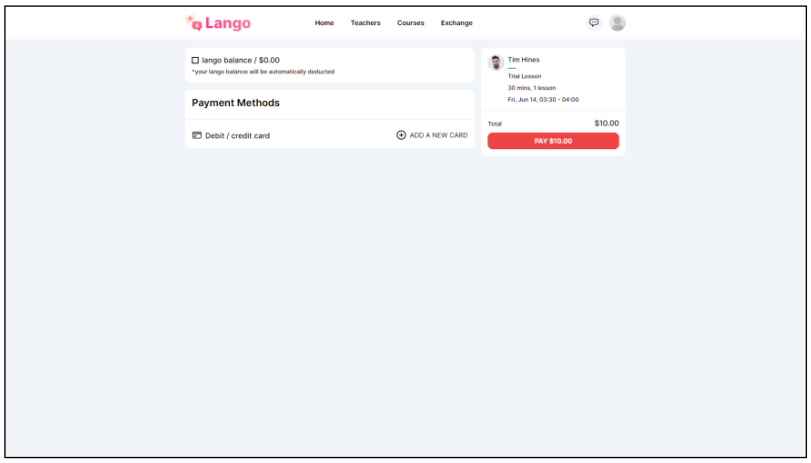


Рисунок 3.30 – Етап оплати(рисунок виконано самостійно)

Після завершення оплати користувачу необхідно побачити результат бронювання для цього він перенаправляється на сторінку з уроками(див. рис. 3.31) та потім він може побачити усі деталі та прикріпленні файли(див. рис. 3.32).

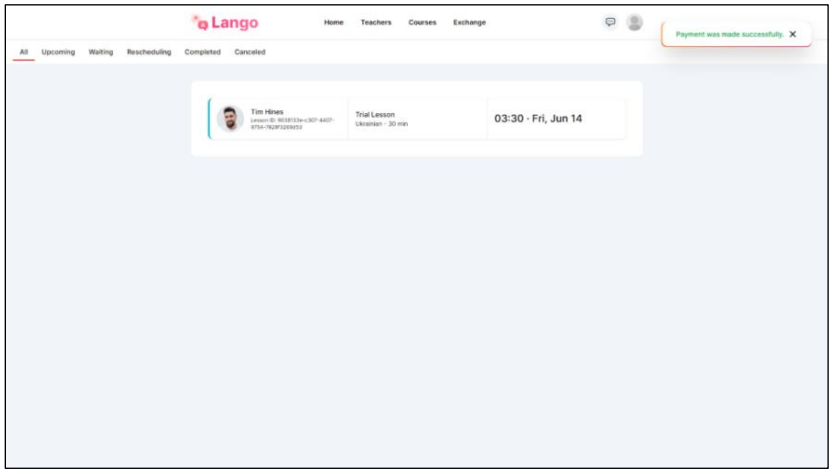


Рисунок 3.31 – Сторінка з запланованими уроками

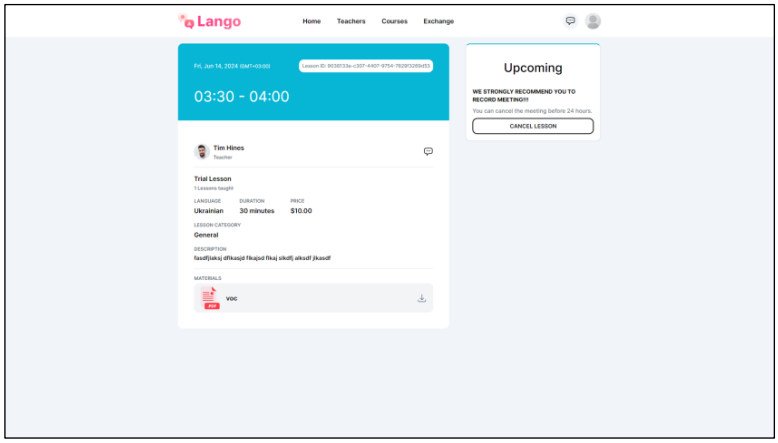


Рисунок 3.32 – Сторінка уроку(рисунок виконано самостійно)

Система має функціонал керування ресурсами, тому було створено дизайн для сторінки гаманцю(див. рис. 3.33), сторінку вкладу депозиту(див. рис. 3.35) та сторінку подання запиту на виплату(див. рис. 3.36).

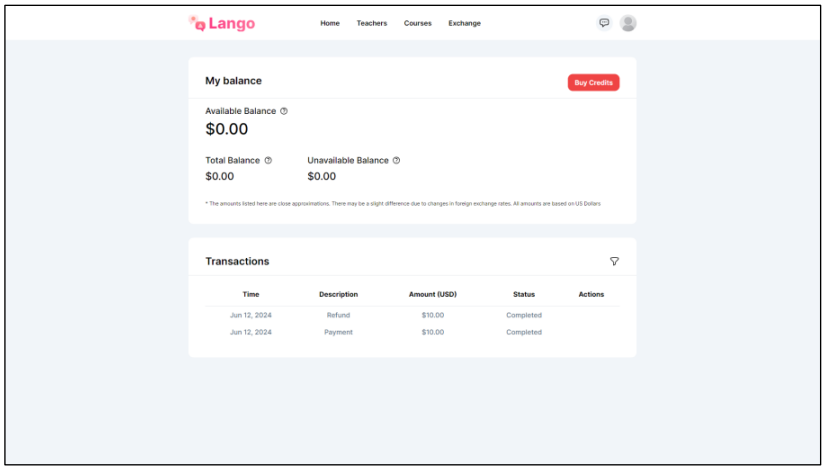


Рисунок 3.34 – Сторінка гаманцю(рисунок виконано самостійно)

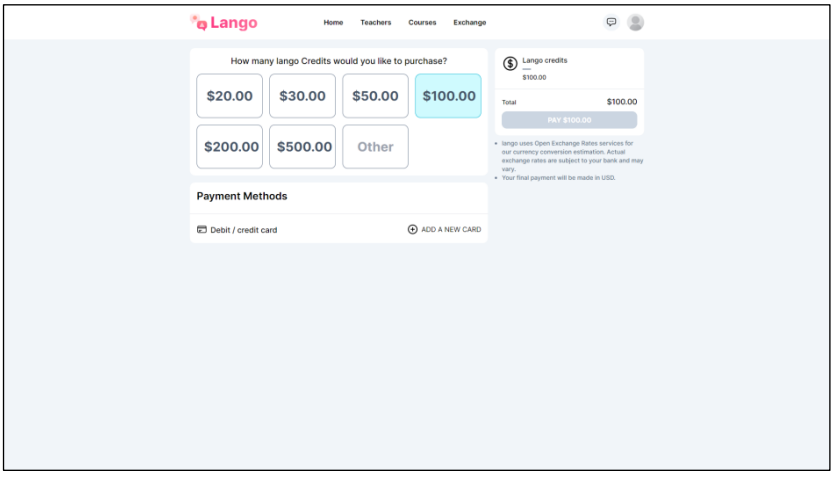


Рисунок 3.35 – Сторінка вносу депозиту(рисунок виконано самостійно)

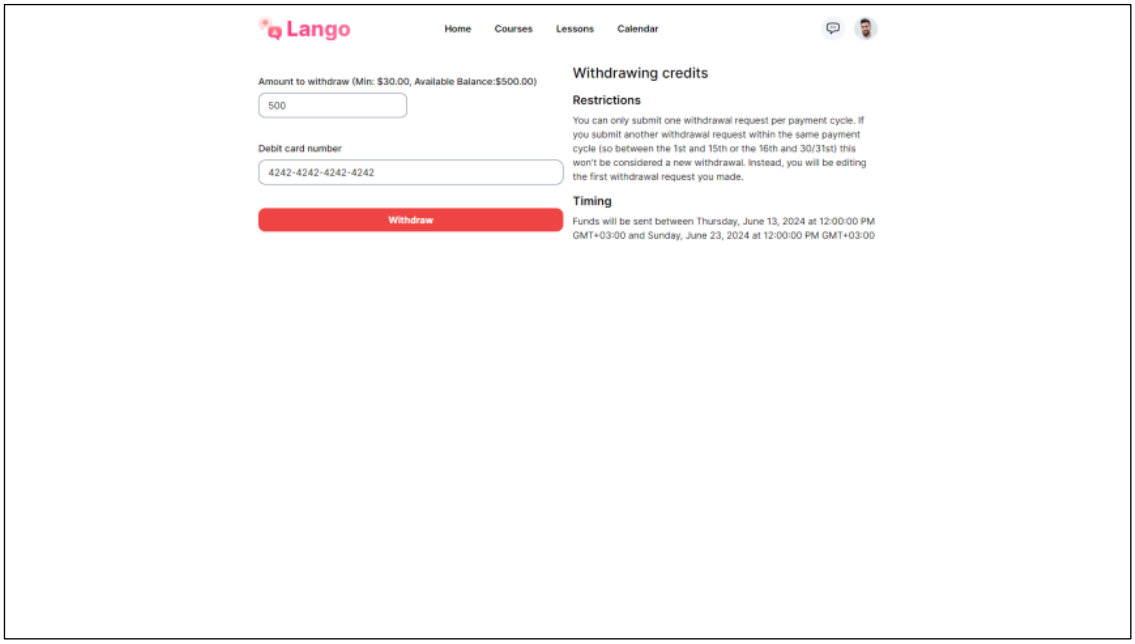


Рисунок 3.36 – Сторінка оформлення виплати(рисунок виконано самостійно)

Також у системі наявний функціонал «Обміну досвідом»(див. рис. 3.37).

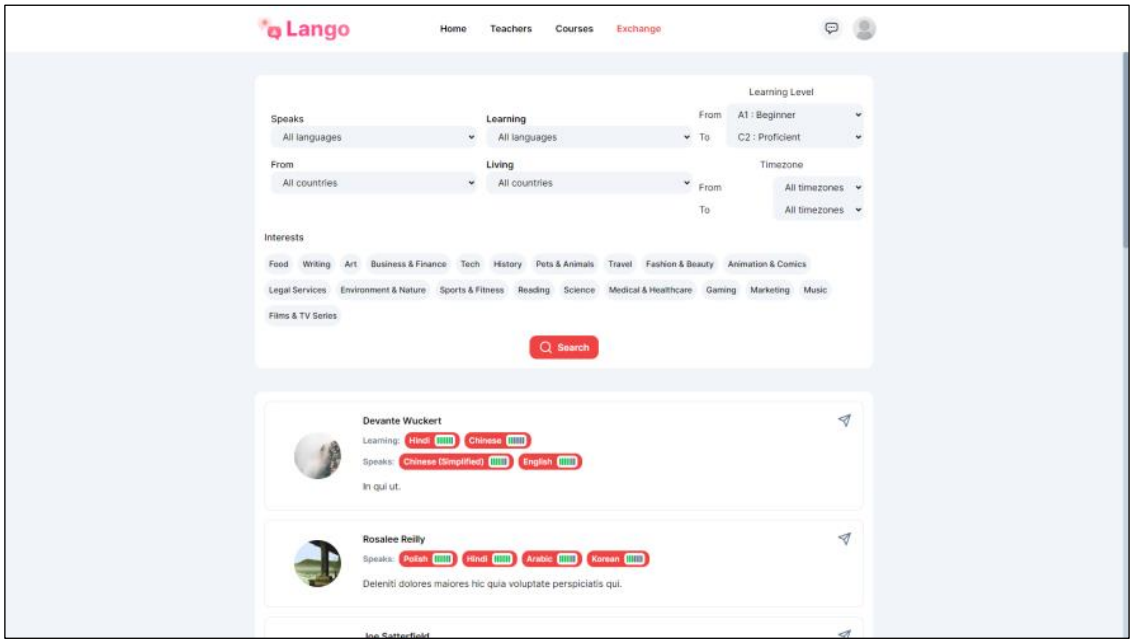


Рисунок 3.37 – Сторінка обміну досвідом(рисунок виконано самостійно)

Також необхідно надавати можливість студентам ставати вчителями та надавати свої послуги, тому необхідно змоделювати процес онбордингу вчителя(див. рис. 3.38-3.46). На рисунках 3.38-3.39 відображено стартову сторінку онбордингу.

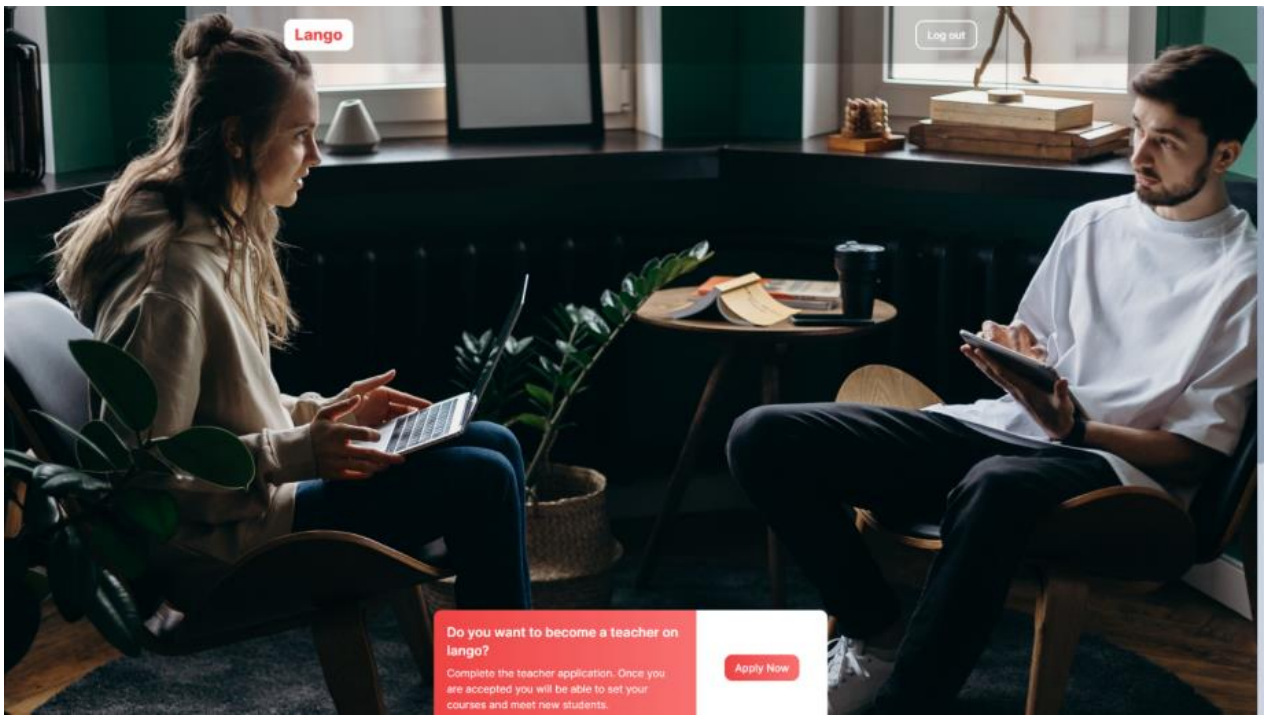


Рисунок 3.38 – Початкова сторінка онбордингу вчителя(рисунок виконано самостійно)

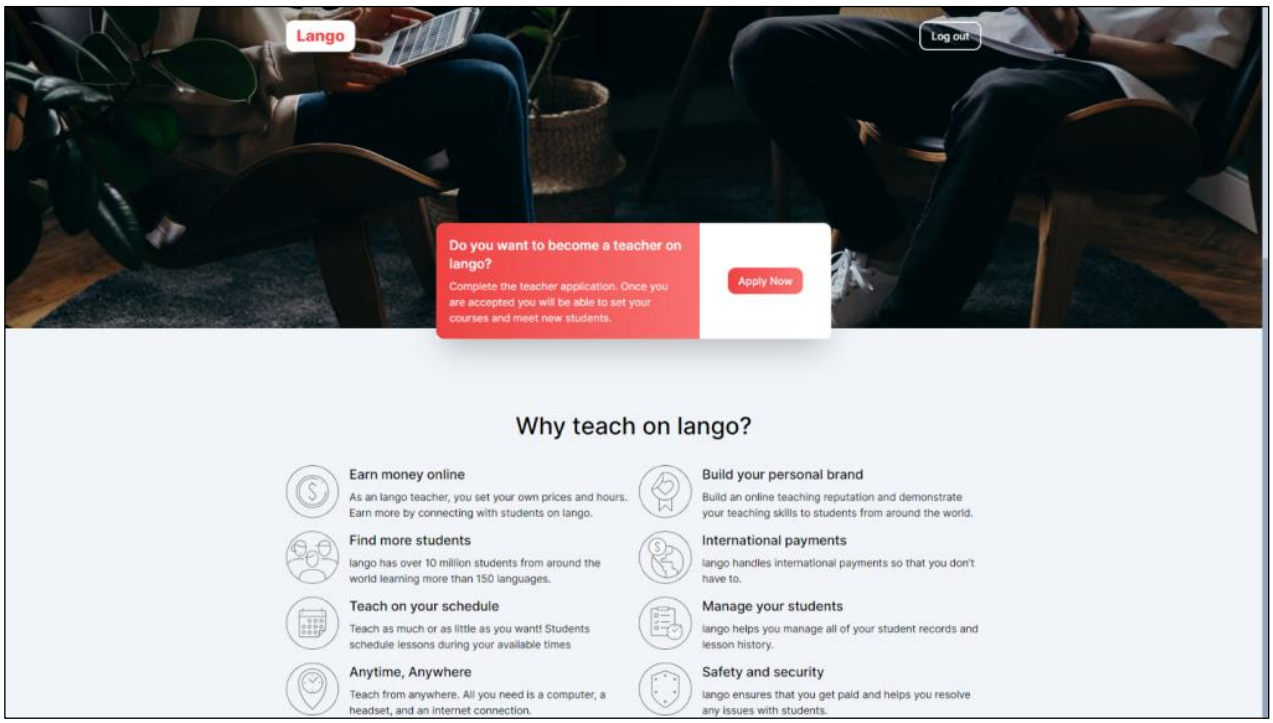


Рисунок 3.39 – Початкова сторінка онбордингу вчителя(продовження)(рисунок виконано самостійно)

На рисунку 3.40 відображено частину форми що відповідає за базову та приватну інформацію, а також мовні навички.

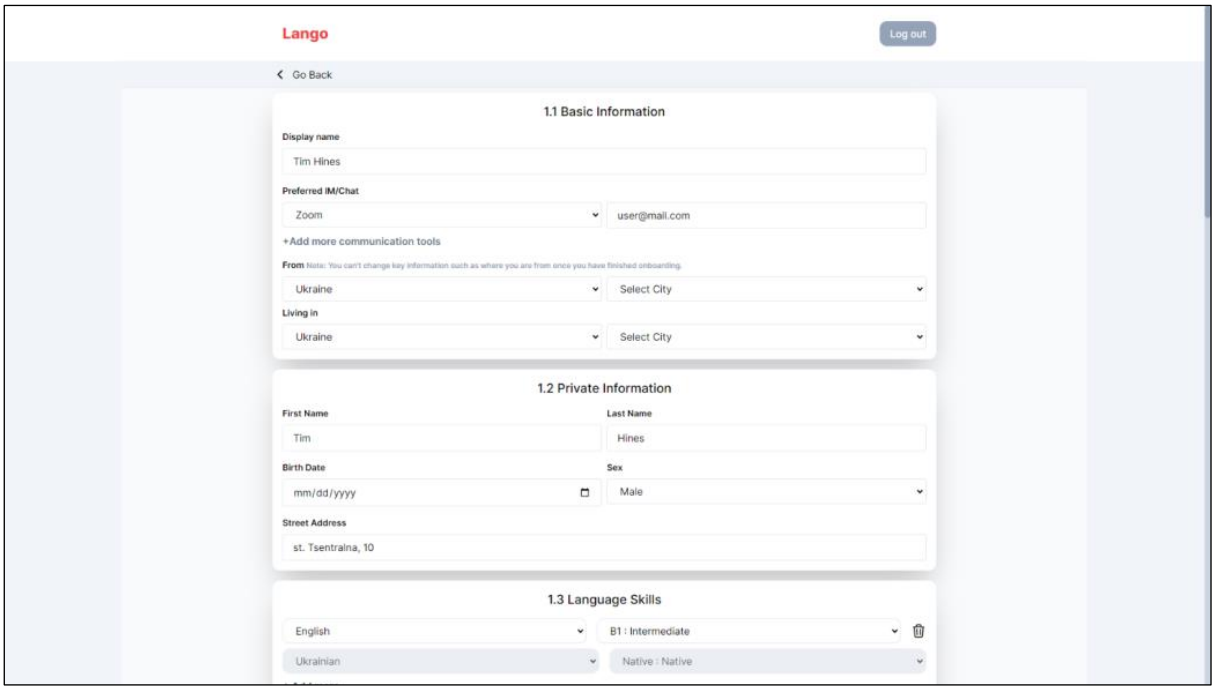


Рисунок 3.40 – Заповнення базової та приватної інформації в анкеті(рисунок виконано самостійно)

На рисунку 3.41 відображено частину форми з вказанням фотокартки профілю, та мов яким буде навчати майбутній вчитель

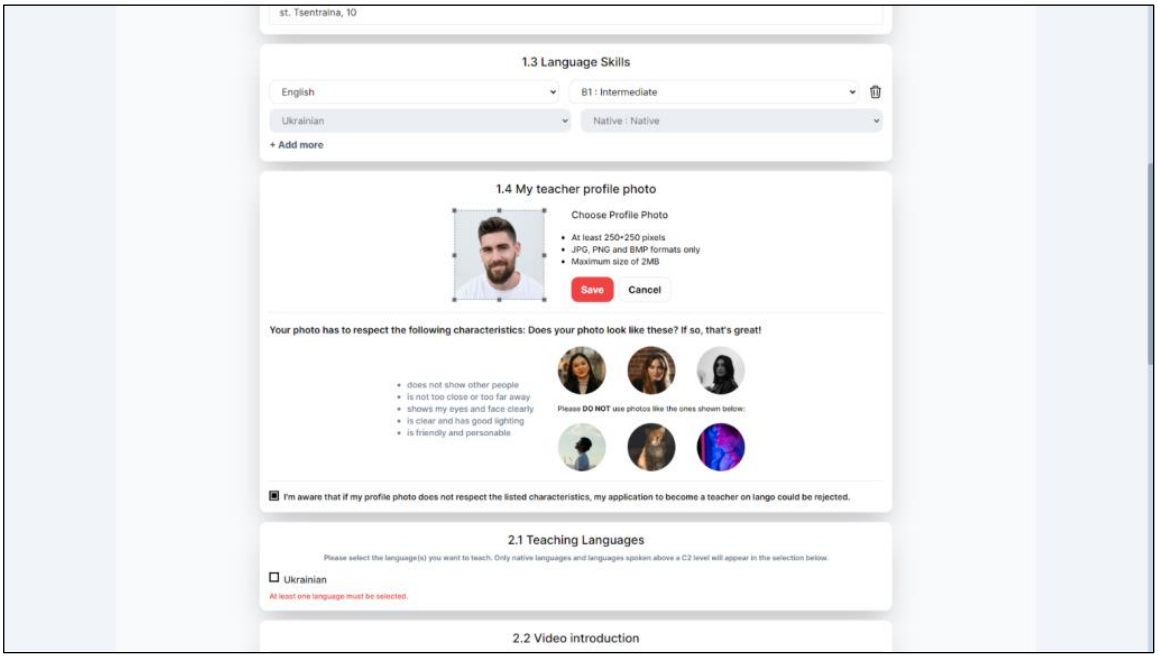


Рисунок 3.41 – Завантаження фото профілю та обирання мов вчителювання(рисунок виконано самостійно)

На рисунку 3.42 відображено частину форми для завантаження відео.

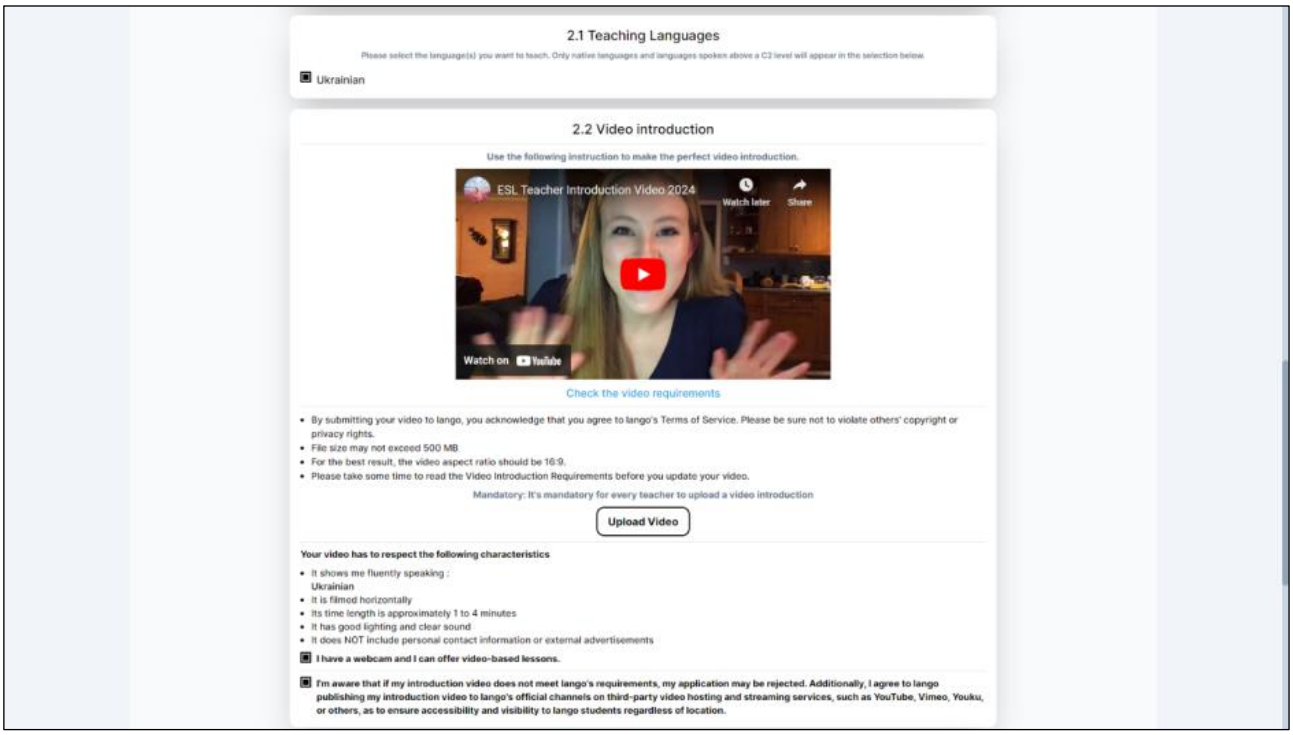


Рисунок 3.42 – Завантаження відео (рисунок виконано самостійно)

На рисунку 3.43 відображено частину форми для заповнення інфомації про вчителя та його стиль вчителювання.

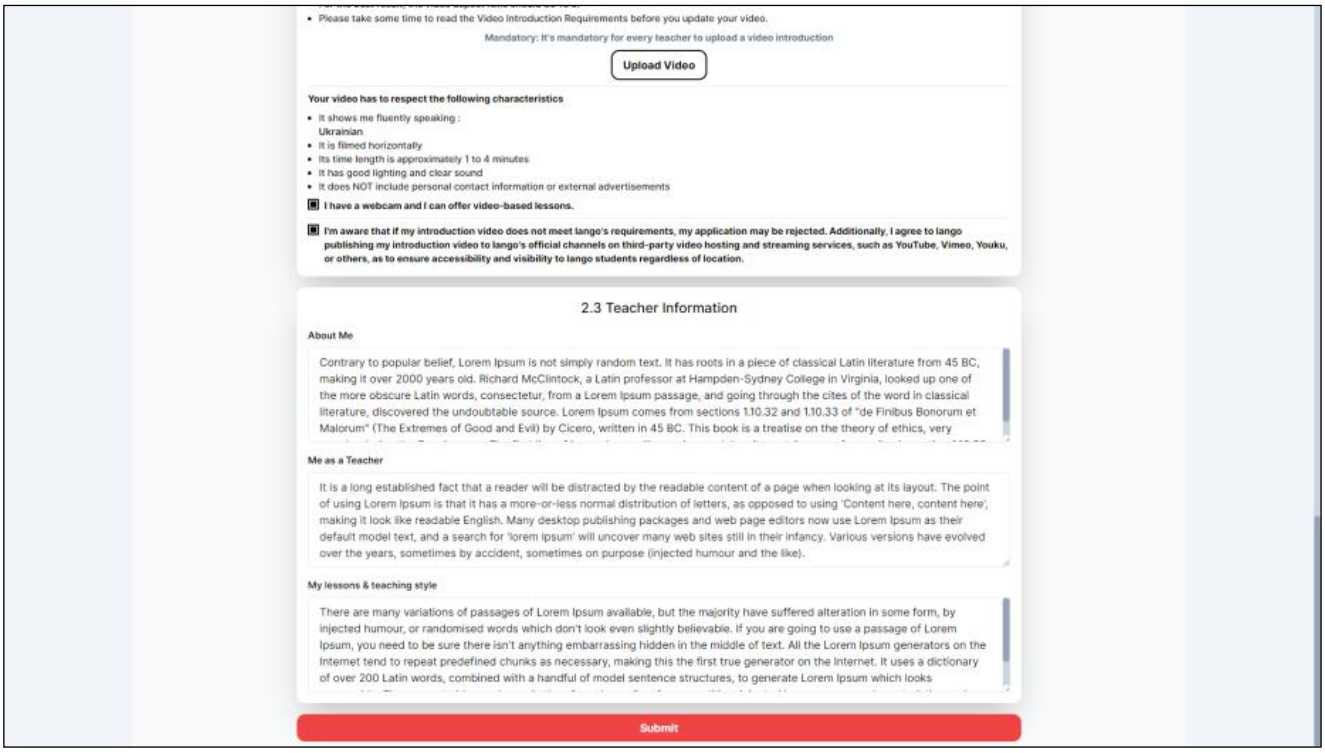


Рисунок 3.43 – Заповнення характеристики (рисунок виконано самостійно)

Для керуванням додатку та модерацію необхідно розробити панель адміністратору, тому було розроблено наступні сторінки(див. рис. 3.47-3.57). На рисунку 3.44 відображено форму авторизації.

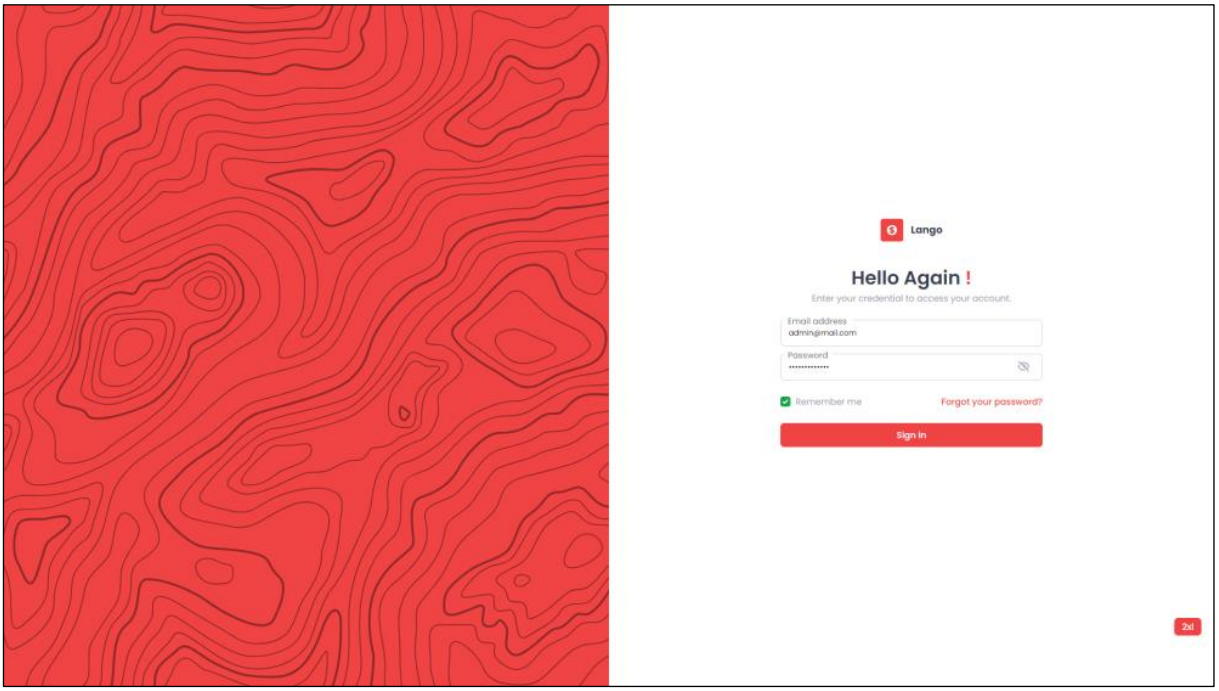


Рисунок 3.44 – Сторінка входу адміністратора(рисунок виконано самостійно)

На рисунку 3.45 відображено сторінку стаитстик.

Рисунок 3.45 – Сторінка з деякими статистиками(рисунок виконано самостійно)

На рисунку 3.46 відображено сторінку з учителями.

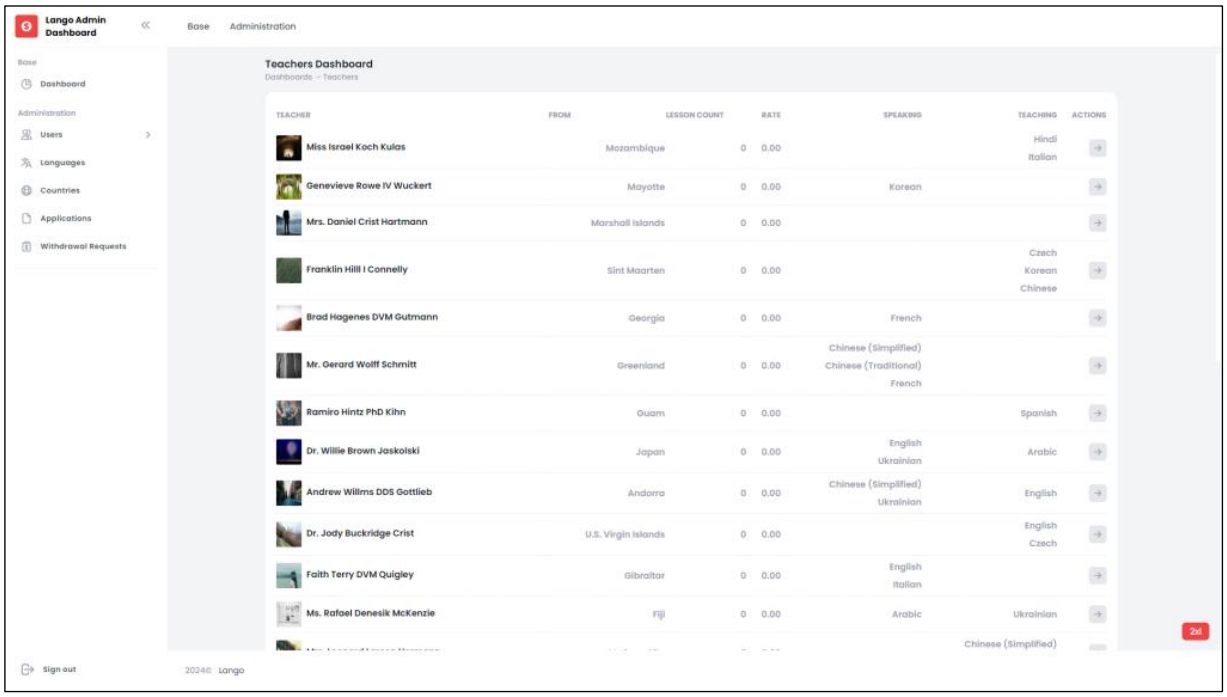


Рисунок 3.46 – Сторінка з учителями(рисунок виконано самостійно)

На рисунку 3.47 відображено сторінку керування мовами.

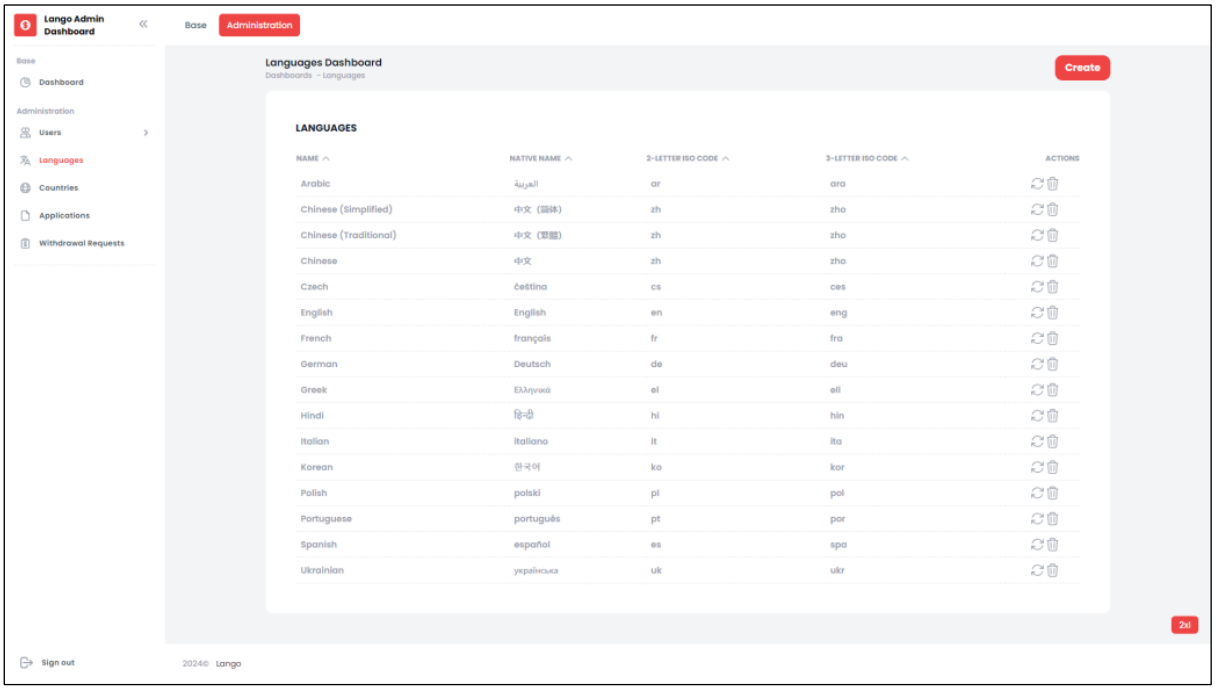


Рисунок 3.47 – Сторінка для керування мовами(рисунок виконано самостійно)

На рисунку 3.48 відображено модальне вікно для створення мови.

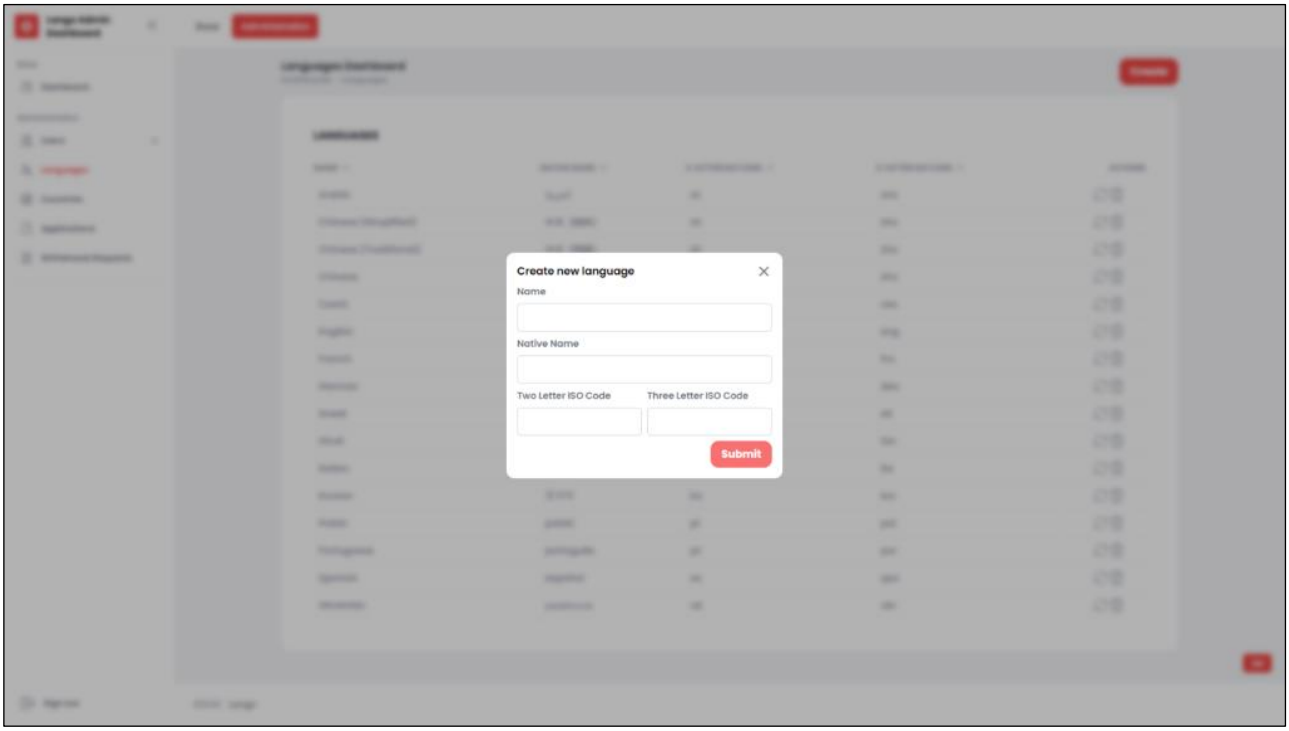


Рисунок 3.48 – Модальне вікно для створення/редагування мови(рисунок виконано самостійно)

На рисунку 3.49 відображено функціонал керування країнами.

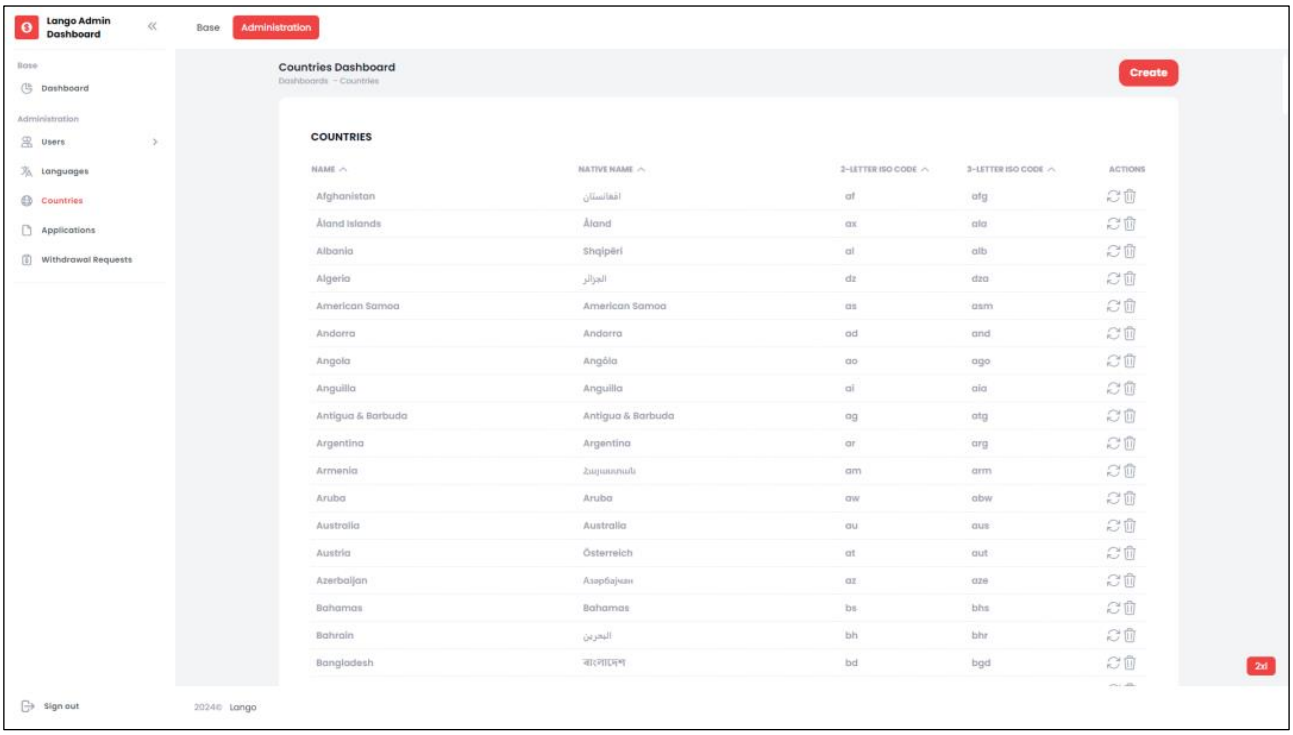


Рисунок 3.49 – Сторінка керування країнами(рисунок виконано самостійно)

На рисунку 3.50 відображено сторінку з запитами на вчителювання.

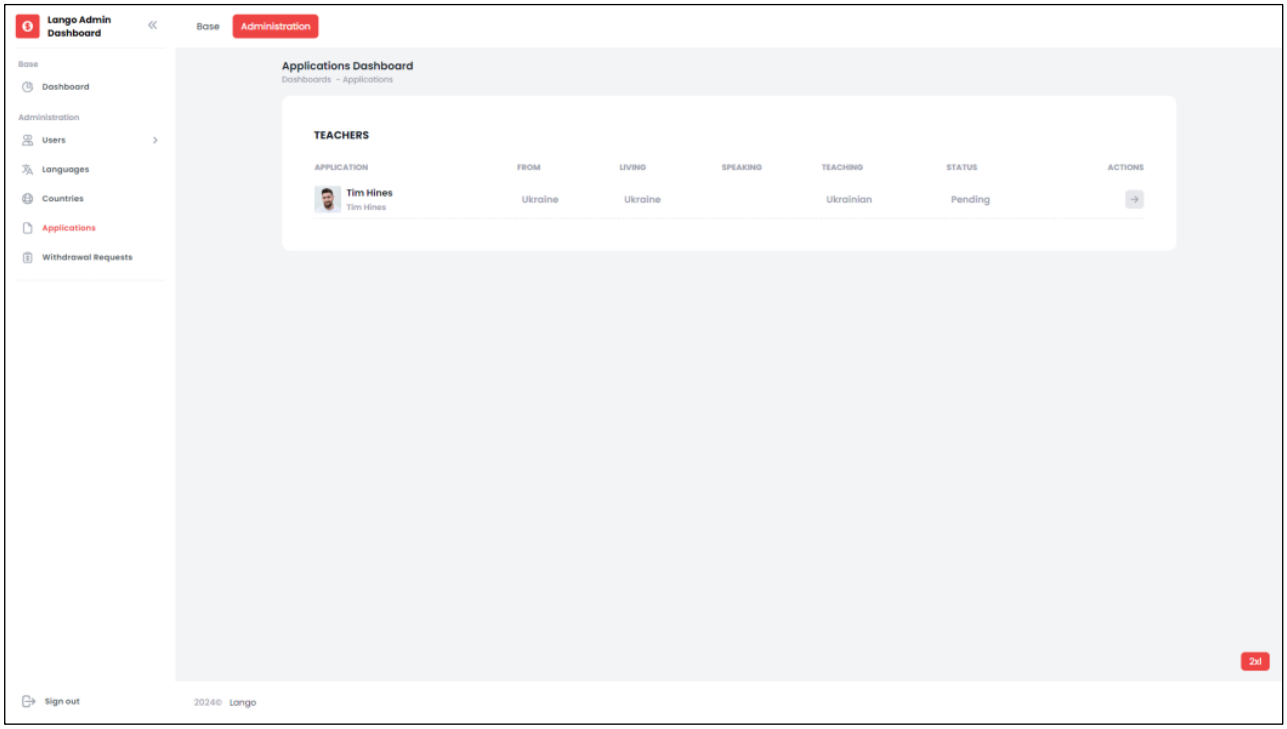


Рисунок 3.50 – Сторінка з запитами на становлення вчителем(рисунок виконано самостійно)

На рисунку 3.51-52 відображено сторінку окремої анкети.

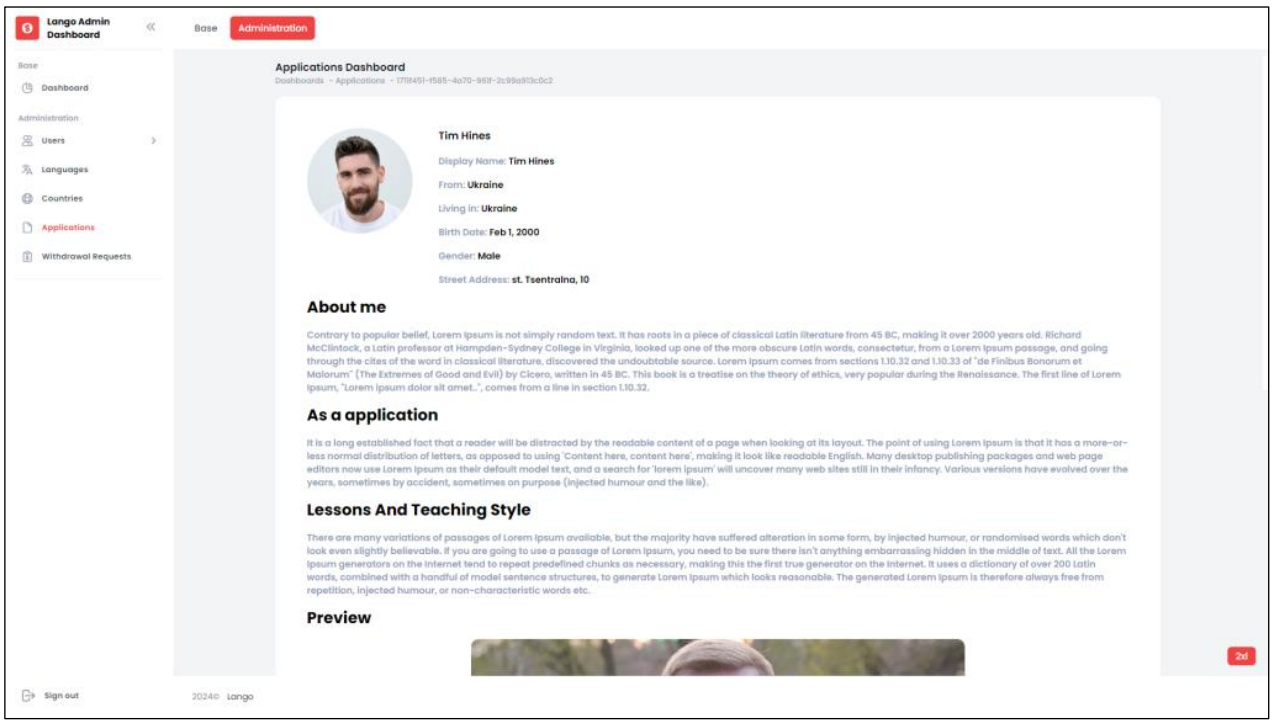


Рисунок 3.51 – Сторінка запиту з особистою інформацією(рисунок виконано самостійно)

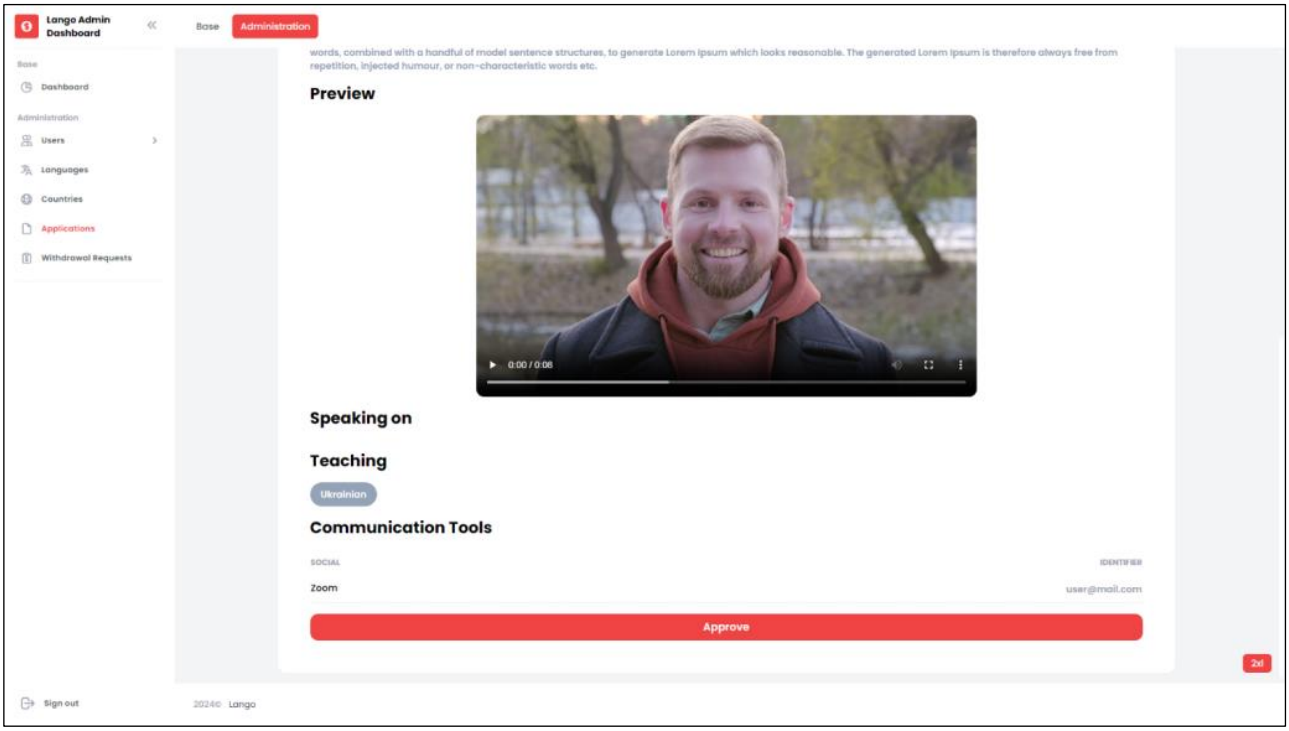


Рисунок 3.52 – Сторінка запиту з відобраенням відео(рисунок виконано самостійно)

На рисунка 3.53 та 3.54 відображено сторінку з запитами на виплату та сторінку окремого запиту, відповідно.

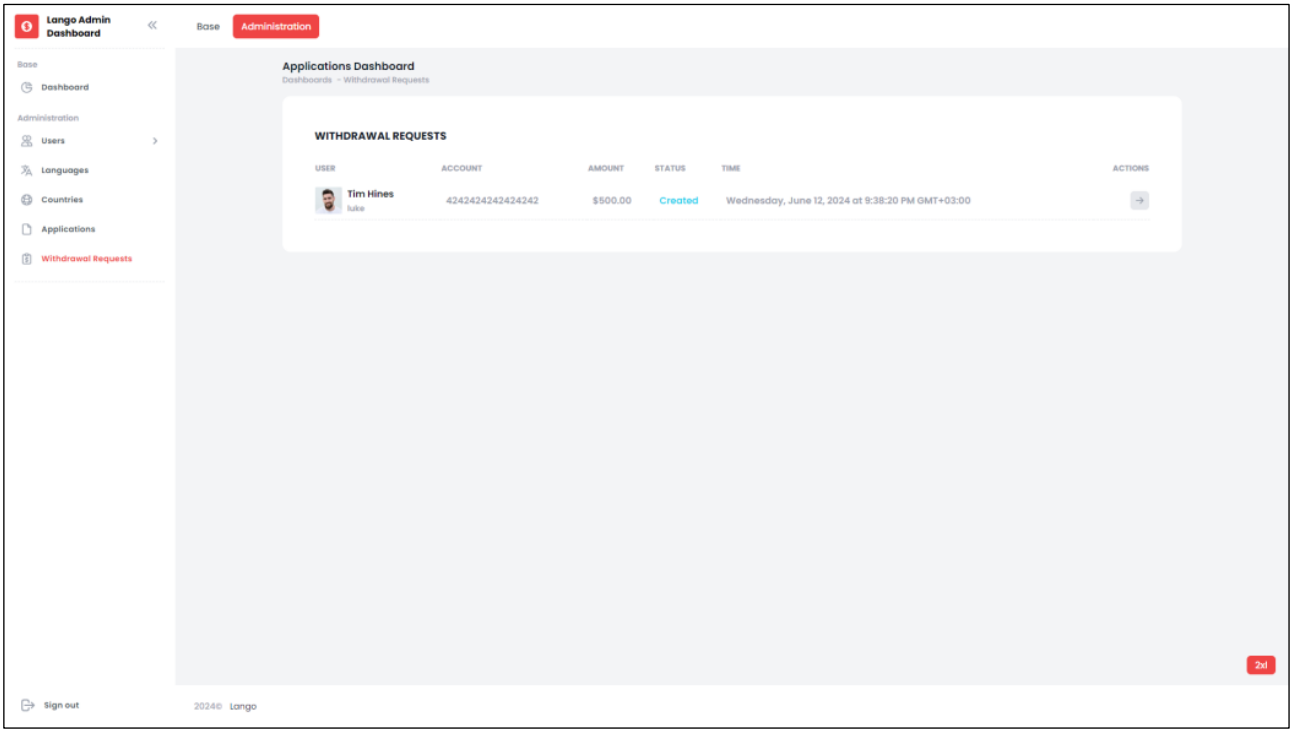


Рисунок 3.53 – Сторінка з запитами на виплату(рисунок виконано самостійно)

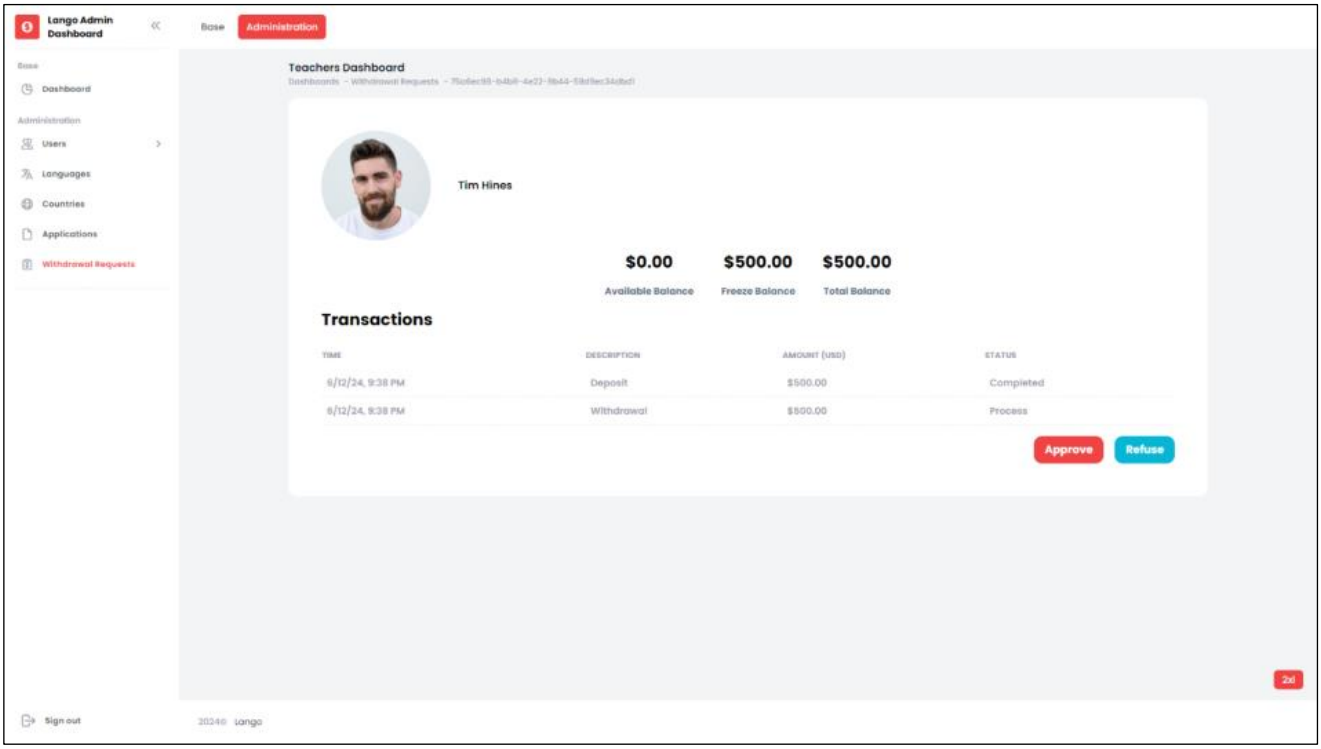


Рисунок 3.54 – Сторінка керування запиту на виплату(рисунок виконано самостійно)

Створення ефективного UI/UX дизайну для освітньої платформи з вивчення іноземних мов включає не лише привабливий візуальний стиль, але й інтуїтивно зрозумілі інтерфейси та зручну навігацію. Це сприяє покращенню користувацького досвіду та задоволеності користувачів, що є ключовими факторами для успішного розвитку платформи.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Програмне рішення для створення серверної частини

Для розробки серверної частини освітньої платформи з вивчення іноземних мов було обрано сучасні, надійні та високоефективні технології. Основним критерієм вибору технологій були їх продуктивність, масштабованість, зручність використання та підтримка спільнотою[7]. Нижче наведено детальний огляд вибраних технологій та бібліотек, а також обґрунтування їх використання.

Основні технології:

- ASP.NET Core: ASP.NET Core було обрано як основну платформу для створення веб-аплікацій через її високі показники продуктивності, крос-платформенність та модульність. ASP.NET Core дозволяє створювати високонавантажені додатки завдяки своїй оптимізованій архітектурі та підтримці хмарних сервісів[9];
- PostgreSQL: PostgreSQL було вибрано як основну базу даних завдяки її відкритому коду, надійності та потужним функціональним можливостям. PostgreSQL підтримує складні запити та забезпечує високий рівень безпеки даних, що робить її ідеальним вибором для освітньої платформи, яка потребує збереження великої кількості структурованих даних[8];
- Redis: Redis використовується для кешування даних, що дозволяє зменшити навантаження на базу даних і покращити швидкість відповіді додатка. Redis обрано через його високу швидкодію та простоту інтеграції з іншими компонентами системи;
- RabbitMQ: RabbitMQ використовується як шина даних між мікросервісами, забезпечуючи надійну та асинхронну передачу повідомлень. Це дозволяє мікросервісам взаємодіяти між собою без втрат продуктивності та з високою надійністю.

Для роботи було використано низку бібліотек, головні з низ перелічені нище:

- Serilog: Serilog обрано для логування через його гнучкість та потужні можливості для структурованого логування. Він дозволяє легко інтегрувати логування в додаток і надає широкий спектр синків для збереження логів;
- MassTransit та MassTransit.RabbitMQ: MassTransit є фреймворком для створення розподілених додатків, що підтримує RabbitMQ. Використання MassTransit дозволяє спростити інтеграцію з RabbitMQ та забезпечити високу надійність передачі повідомлень між мікросервісами;
- MediatR: MediatR використовується для реалізації патерну CQRS (Command Query Responsibility Segregation), що сприяє розділенню запитів та команд. Це покращує читабельність коду та спрощує підтримку додатка;
- AutoMapper: AutoMapper дозволяє автоматизувати процес мапінгу між об'єктами, що значно спрощує роботу з DTO (Data Transfer Objects) та зменшує кількість коду, необхідного для трансформації даних;
- Yarp.ReverseProxy: Yarp.ReverseProxy використовується для створення реверс-проксі, що дозволяє масштабувати додаток та забезпечувати балансування навантаження між мікросервісами;
- EasyCaching: EasyCaching надає зручний спосіб кешування даних, інтегруючись з Redis та іншими популярними механізмами кешування.
- NodaTime: NodaTime використовується для роботи з датами та часом, забезпечуючи точність та зручність у керуванні часовими зонами та іншими аспектами роботи з датами;
- Azure.Storage.Blobs: Azure.Storage.Blobs забезпечує зберігання файлів у хмарному середовищі, надаючи надійний та масштабований спосіб зберігання великих обсягів даних;
- Stripe.net: Stripe.net використовується для інтеграції платіжної системи, що дозволяє легко обробляти платежі та інтегрувати платіжні послуги у додаток;

- Quartz: Quartz використовується для керування та виконання завдань за розкладом, що є необхідним для автоматизації різних процесів на платформі;
- FluentValidation: FluentValidation обрано для валідації даних, що забезпечує високу якість та надійність перевірки вхідних даних;
- Ardalis та MinimalApi endpoints: Ardalis та MinimalApi endpoints дозволяють спростити створення API, забезпечуючи зручний та легковагий спосіб визначення кінцевих точок для взаємодії з клієнтами.

Для забезпечення модульності та повторного використання коду було розроблено низку бібліотек:

- BuildingBlocks.Abstractions;
- BuildingBlocks.Caching;
- BuildingBlocks.Core;
- BuildingBlocks.Email;
- BuildingBlocks.HealthCheck;
- BuildingBlocks.Integration.Mass Transit;
- BuildingBlocks.Logging;
- BuildingBlocks.Persistence.EfCore.Postgres;
- BuildingBlocks.Security;
- BuildingBlocks.Swagger;
- BuildingBlocks.Validation;
- BuildingBlocks.Web.

Ці бібліотеки забезпечують основні функціональні можливості для побудови мікросервісів, спрощуючи процес розробки та забезпечуючи єдині стандарти та підходи в межах проекту[10].

Були розроблені два основних мікросервіси:

- Identity: цей мікросервіс відповідає за авторизацію та аутентифікацію користувачів, забезпечуючи безпеку платформи та управління доступом до різних ресурсів;

– Learning: мікросервіс Learning займається керуванням даними, пов'язаними з навчанням, включаючи курси, уроки, викладачів та студентів.

Основні класи даних:

– User;
– Application;
– Chat;
– Message;
– Read;
– Teacher;
– Course;
– Group;
– Lesson;
– Plan;
– Seminar;
– Conference;
– ConferenceUser;
– LanguageUser;
– Language;
– Country;
– Review;
– Reaction;
– Transaction;
– Schedule;
– Resource;
– Order;
– Problem.

Ці класи даних представляють основні сутності платформи, забезпечуючи структуроване збереження та управління інформацією.

Також під час розробки було використано інструменти для документації коду за допомогою бібліотеки Swagger, яка збирає усю необхідну інформацію та генерує OpenAPI специфікацію, що може допомогти у генерації клієнтів під різноманітні платформи(див. рис. 4.1-4.3).

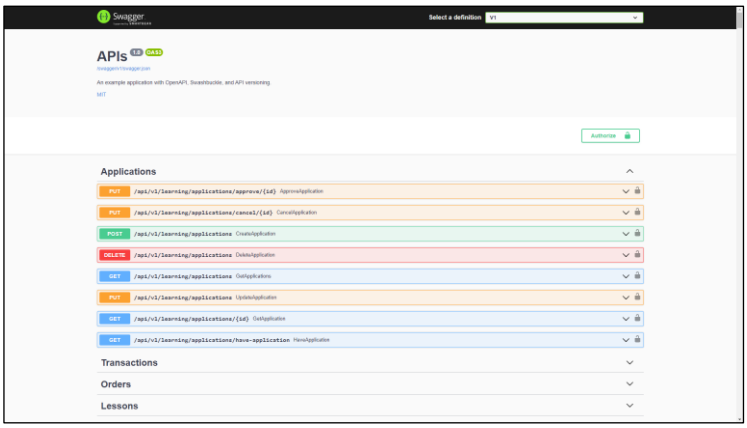


Рисунок 4.1 – Специфікація OpenAPI (рисунок виконано самостійно)

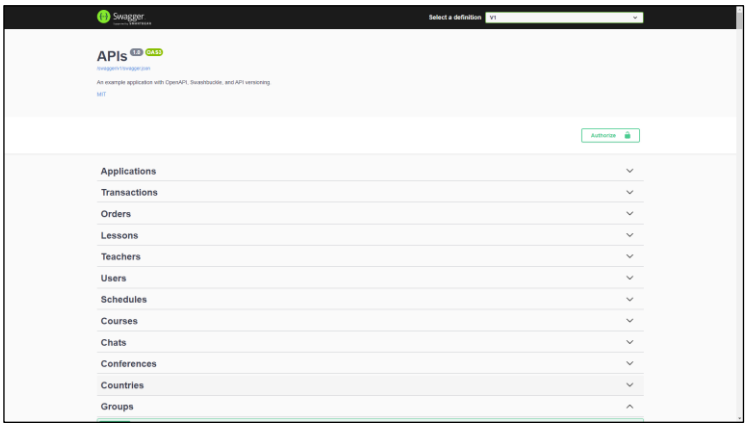


Рисунок 4.2 – Специфікація OpenAPI (рисунок виконано самостійно)

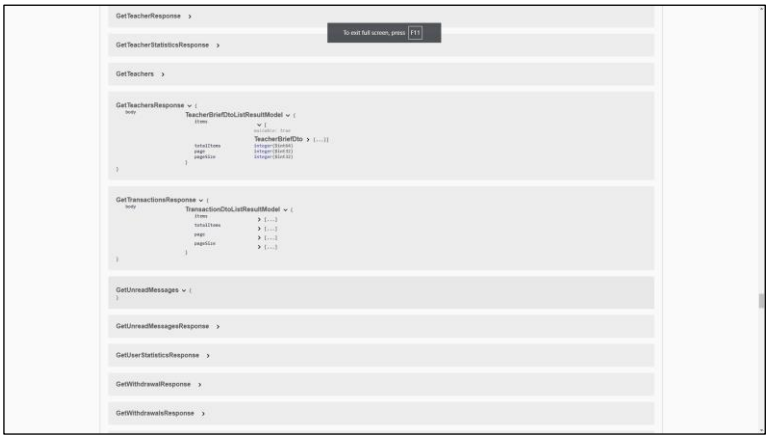


Рисунок 4.3 – Специфікація OpenAPI (рисунок виконано самостійно)

4.2 Програмне рішення для створення клієнтскьої частини

Для розробки клієнтської частини освітньої платформи з вивчення іноземних мов було обрано сучасні та перевірені технології, що забезпечують високу продуктивність, зручність у використанні та легкість у підтримці. Основні критерії вибору включали популярність технології, підтримку спільнотою, можливості для масштабування та інтеграції з іншими сервісами. Нижче надається детальний огляд вибраних технологій, бібліотек та фреймворків, а також обґрунтування їх використання.

Angular було обрано як основний фреймворк для створення клієнтської частини завдяки його потужним можливостям для побудови складних SPA (Single Page Applications). Angular забезпечує структурованість коду, модульність та високу продуктивність. Використання TypeScript дозволяє писати типізований код, що зменшує кількість помилок і покращує читабельність та підтримку коду.

Бібліотеки:

- @js-jodaбібліотека @js-joda була обрана для роботи з датами та часом завдяки її потужним можливостям та простоті використання. Вона забезпечує точність та зручність у керуванні часовими даними, що є критичним для платформи, де точність розкладу та терміни виконання завдань мають велике значення;
- @microsoft/signalr: @microsoft/signalR використовується для реалізації реального часу комунікацій між клієнтом та сервером. Це дозволяє реалізувати функції чату, нотифікацій та інших динамічних елементів додатка, забезпечуючи миттєве оновлення даних;
- @stripe/stripe-js: @stripe/stripe-js було обрано для інтеграції платіжної системи Stripe. Ця бібліотека забезпечує безпечний та зручний спосіб обробки платежів безпосередньо у браузері, що спрощує процес здійснення транзакцій для користувачів.

Tailwind CSS використовується для стилізації інтерфейсу додатка. Це утилітарний CSS-фреймворк, який дозволяє швидко та легко створювати

адаптивний дизайн. Використання Tailwind CSS значно спрощує процес розробки стилів та забезпечує високу гнучкість у створенні унікального інтерфейсу.

Розроблені додатки:

– Lango: додаток для вчителів та студентів, який забезпечує всі необхідні функції для проведення уроків, взаємодії між вчителями та студентами, керування курсами та розкладом;
– Lango.Dashboard: додаток для адміністратора системи, що дозволяє управляти користувачами, курсами, розкладом та іншими аспектами роботи платформи.

Авторизація та безпека.

Для авторизації використовується підхід JWT (JSON Web Tokens), що забезпечує безпечну передачу та зберігання токенів доступу[11]. Для автоматичного оновлення токену використовується HttpInterceptor. Нижче наведено короткий приклад використання HttpInterceptor для оновлення токену:

```
@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  constructor(private readonly authService: AuthService) {
  }

  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler,
  ): Observable<HttpEvent<unknown>> {
    request = this.addAuthHeader(request);
    return next.handle(request).pipe(
      catchError((err) => {
        if (err.status == 401) {
          return this.authService.refreshToken().pipe(
            switchMap((res) => {
              request = this.addAuthHeader(request);
              return next.handle(request);
            }),
          );
        }
        throw err;
      }),
    );
  }

  private addAuthHeader(request: HttpRequest<unknown>) {
    const token = this.authService.getToken();
    return request.clone({
      headers: request.headers
```

```
            .delete('Authorization')
            .append('Authorization', 'Bearer ' + token),
        });
    }
}
```

Для спрощення роботи з API було використано утиліту ng-openapi-gen, яка автоматично генерує клієнтський код на основі OpenApi специфікації. Це дозволяє швидко створювати клієнтські сервіси для взаємодії з сервером, забезпечуючи точність та відповідність API специфікації.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Юніт-тестування

При розробці освітньої платформи було використано декілька типів тестування для забезпечення якості, надійності та продуктивності системи. Нижче описані кожен з видів тестування, що використовувалися, разом з прикладами тестів для кожного з них.

Юніт-тестування (unit testing) – це метод тестування, при якому окремі модулі або компоненти системи тестуються ізольовано один від одного. Мета юніт-тестування - перевірка коректності роботи окремих функцій або методів без залежностей від інших частин системи. Для цього виду тестування використовувалися наступні інструменти:

- xUnit – один з найпопулярніших фреймворків для юніт-тестування у .NET середовищі;
- FluentAssertions – бібліотека для написання читабельних тверджень у тестах;
- NSubstitute – інструмент для створення заглушок та моків.
- Bogus – генератор фейкових даних для тестів.

Приклад юніт-тесту:

```
public class CancelOrderByUserIdTests : IDisposable
{
private readonly ApplicationDbContext _context;
private readonly RefundService _refundService;
private readonly IEmailNotificationService _emailNotificationService;
private readonly IDateTimeProvider _dateTimeProvider;
private readonly ITimeframeProvider _timeframeProvider;
public CancelOrderByUserIdTests()
{
_context = DbContextFactory.Create();
_emailNotificationService                                          =
Substitute.For<IEmailNotificationService>();
_dateTimeProvider = Substitute.For<IDateTimeProvider>();
_timeframeProvider = Substitute.For<ITimeframeProvider>();
_refundService = Substitute.For<RefundService>();
}
public void Dispose()
{
DbContextFactory.Destroy(_context).Wait();
}
```

```
    [Fact]
    public async Task CancelOrderByUserId_LessonOrder_StripeProvider_ShouldCancelOrderCreateTransactionSendEmail()
    {
    // Arrange
    var faker = new Faker();
    var user = new User()
    {
    FirstName = faker.Name.FirstName(),
    LastName = faker.Name.LastName(),
    Email = faker.Internet.Email()
    };
    var userId = user.Id;
    var paymentProvider = PaymentProvider.Stripe;
    var day = Duration.FromDays(1);
    var utcNow = ZonedDateTime.FromDateTimeOffset(DateTimeOffset.UtcNow);
    var date = utcNow + day;
    var order = new LessonOrder()
    {
    UserId = userId,
    User = user,
    Plan = new Plan()
    {
    Price = 12m,
    Lesson = new Lesson()
    {
    Name = "fasdfasf",
    Description = "fasdfasdf",
    Category = "fasdfasdf",
    Teacher = TestHelper.CreateTeacher()
    }
    },
    Price = 12m,
    Schedule = new LessonSchedule() {Date = date},
    Payment = TestHelper.CreatePayment(userId)
    };
    var orderId = order.Id;
    await _context.Orders.AddAsync(order);
    await _context.SaveChangesAsync();
    var request = new CancelOrder(userId, orderId, paymentProvider);
    var handler = new CancelOrderHandler(_context, _refundService, _emailNotificationService, _dateTimeProvider, _timeframeProvider);
    _timeframeProvider.CancelingFrame.Returns(day);
    _dateTimeProvider.ZonedUtcNow.Returns(utcNow);
    _refundService.CreateAsync(Arg.Any<RefundCreateOptions>()).Returns(new Refund() {Status = "succeeded"});
    // Act
    var response = await handler.Handle(request, CancellationToken.None);
    // Assert
    response.Should().Be(Unit.Value);
    order.Status.Should().Be(OrderStatus.Cancelled);
    order.User.Transactions.Any(x => x.Type == TransactionType.Refund).Should().Be(true);
    _emailNotificationService.Received(1)
    .SendCancelEmailAsync(Arg.Any<User>(),Arg.Any<User>(),Arg.Any<string>(),Arg.Any<string>());
    }
```

```
}
```

Результат юніт-тестування відображається в тест-менеджері де можна побачити результат виконання тесту, логи, час, тощо(див. рис. 5.1).

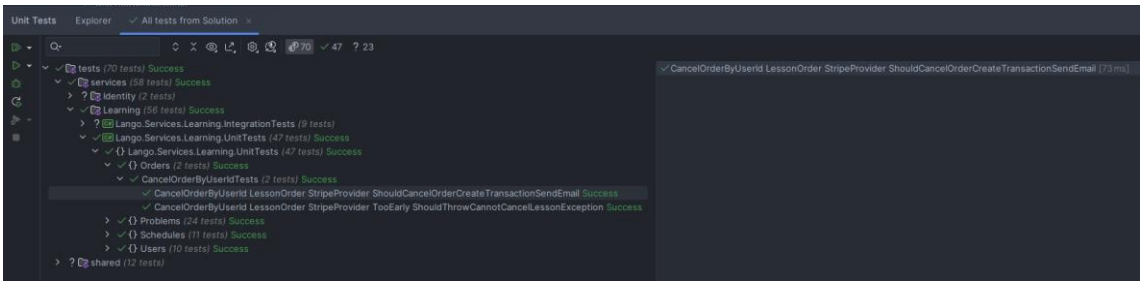


Рисунок 5.1 – Результат виконання юніт-тесту (рисунок виконано самостійно)

Юніт-тестування забезпечує високу покритість коду, дозволяє швидко знаходити та виправляти помилки, що знижує ризик збоїв на ранніх етапах розробки.

5.2 Інтеграційне тестування

Інтеграційне тестування (integration testing) - це процес тестування, при якому перевіряється взаємодія між різними модулями або компонентами системи. Мета інтеграційного тестування - переконатися, що окремі частини системи працюють разом належним чином. Для інтеграційного тестування використовувалися:

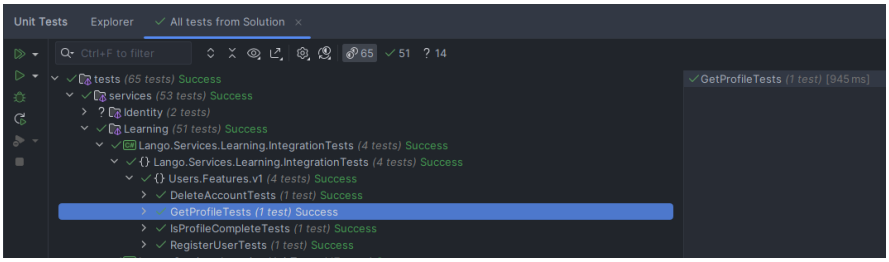
- xUnit - для написання тестів.
- Testcontainers – для створення ізольованих середовищ з використанням Docker-контейнерів.

Приклад інтеграційного тесту:

```
public class GetProfileTests : LearningServiceIntegrationTestBase
{
public GetProfileTests(
SharedFixtureWithEfCore<Program, ApplicationDbContext> sharedFixture,
ITestOutputHelper outputHelper
) : base(sharedFixture, outputHelper)
{
}
[Fact]
```

```
    [CategoryTrait(TestCategory.Integration)]
    public async Task ShouldGetProfileOfRegisteredUser()
    {
    var command = new RegisterUserFaker().Generate();
    var getProfile = new GetProfile();
    var                                provider                                =
SharedFixture.ServiceProvider.GetRequiredService<IFakeUserProvider>();
    provider.User = new FakeUser()
    {
    IsAuthenticated = true,
    JwtToken = "fasdfasdfasdf",
    Role = "student",
    UserId = command.Id.ToString()
    };
    await SharedFixture.SendAsync(command);
    var profile = await SharedFixture.SendAsync(getProfile);
    profile.Should().NotBeNull();
    profile.UserProfile.Id.Should().Be(command.Id);
    profile.UserProfile.FirstName.Should().Be(command.FirstName);
    profile.UserProfile.LastName.Should().Be(command.LastName);
    profile.UserProfile.ProfileName.Should().Be(command.UserName);
    }
    }
```

Результат виконання інтеграційного тесту відображається так само як і для юніт-тестів(див. рис. 5.2), але при цьому система тестувалася повністю та були створені декілька докер-контейнерів для ізоляції тестового середовища(див. рис. 5.3).



Риснок 5.2 – Результат виконання інтеграційного тесту (рисунок виконано самостійно)

Рисунок 5.3 – Створені допоміжні докер-контейнери для тестування (рисунок виконано самостійно)

Інтеграційне тестування допомагає виявити проблеми у взаємодії між компонентами системи, що забезпечує стабільну роботу всієї системи в цілому.

5.3 Навантажувальне тестування

Навантажувальне тестування (load testing) – це процес тестування системи під великим навантаженням для визначення її продуктивності та стійкості. Мета навантажувального тестування – оцінити, як система поводиться при великій кількості запитів або користувачів одночасно. Для цього використовувався інструмент NBomber.

Приклад навантажувального тесту:

```
var scenario6 = Scenario.Create(
"get_teachers_page_1_pageSize_12_searchText_mona_countries_Argentina_languages_Greek_days_5_intervals_start_12_end_16",
async ctx =>
{
var request = Http.CreateRequest("POST", "http://localhost:5004/api/v1/learning/teachers/search")
.WithHeader("accept", "text/plain")
.WithHeader("Content-Type", "application/json-patch+json")
.WithObjectBody(new
{
page = 1,
pageSize = 12,
searchText = "mona",
countries = new[]{"Argentina"},
languages = new[]{"Greek"},
days = new[]{4},
intervals = new[]{new {start = 12, end = 16}}
});
var response = await Http.Send(HttpClient, request);
await Task.Delay(1000);
return response;
})
.WithoutWarmUp()
.WithLoadSimulations(
Simulation.Inject(rate: 100, interval: TimeSpan.FromSeconds(20), during: TimeSpan.FromSeconds(2 * 60)));
NBomberRunner
.RegisterScenarios(scenario6)
.Run();
```

Під час виконання «бомбардування» серверу запитами у консолі відображається поточний стан виконання сценарію(див. рис. 5.4), після завершення виконання усіх сценаріїв у консолі відобразиться результат(див. рис.

5.5), а також результат можно подивитися у різних форматах, які генеруються бібліотекою, зокрема у зручному веб-форматі(див. рис. 5.6).

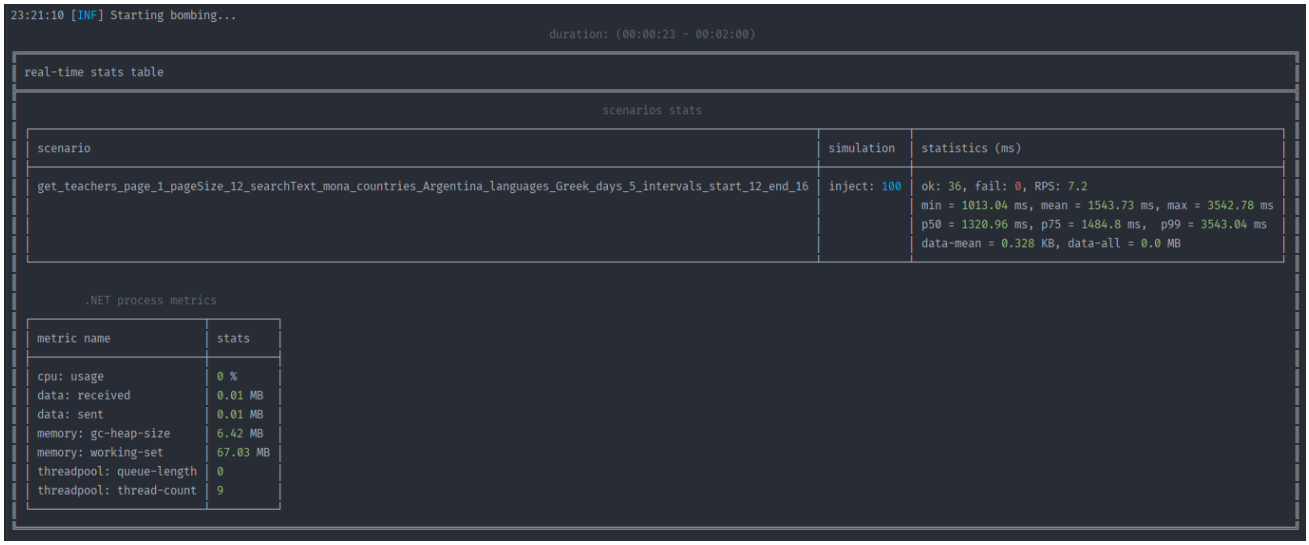
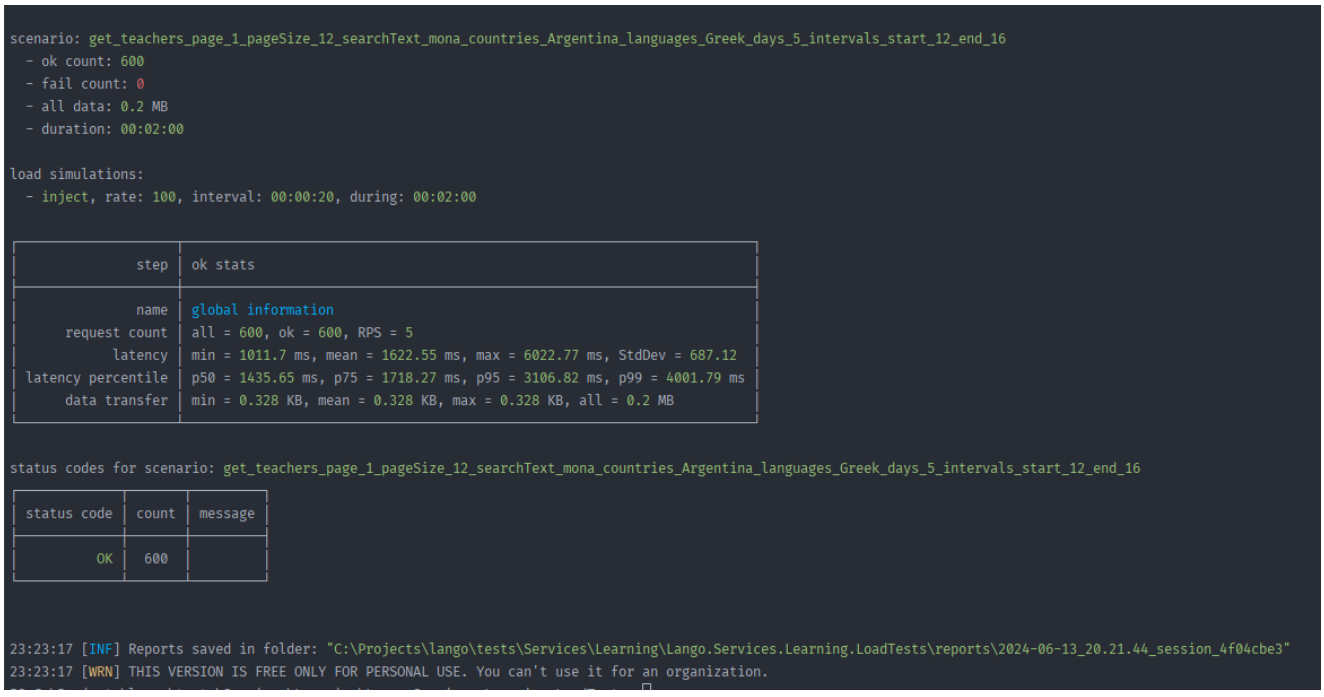
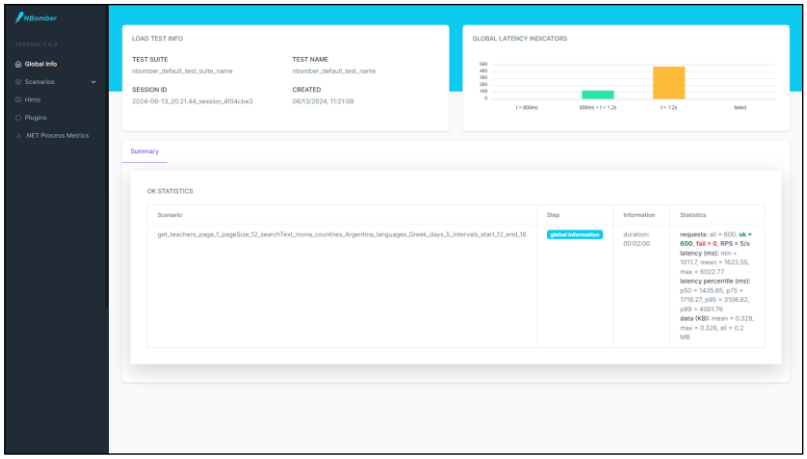


Рисунок 5.4 – Поточний стан виконання сценарію (рисунок виконано самостійно)



Риснок 5.5 – Результат виконання сценарію у консолі (рисунок виконано самостійно)

Навантажувальне тестування дозволяє виявити вузькі місця у продуктивності системи, що допомагає забезпечити її надійність та стійкість при великих навантаженнях.



Риснок 5.5 – Результат виконання сценарію у веб-форматі (рисунок виконано самостійно)

5.4 Ручне тестування через клієнт та Swagger

Ручне тестування (manual testing) – це процес тестування, при якому тестувальники вручну перевіряють функціональність системи через інтерфейс користувача. Мета ручного тестування – забезпечити, щоб система працювала відповідно до вимог та очікувань користувачів. Ручне тестування використовується для перевірки нових функцій, пошуку багів та оцінки зручності використання інтерфейсу.

Ручне тестування системи виковиконувалося на веб додатку(див. рис. 5.7) та за допомогою утиліти Swagger(див. рис. 5.8).

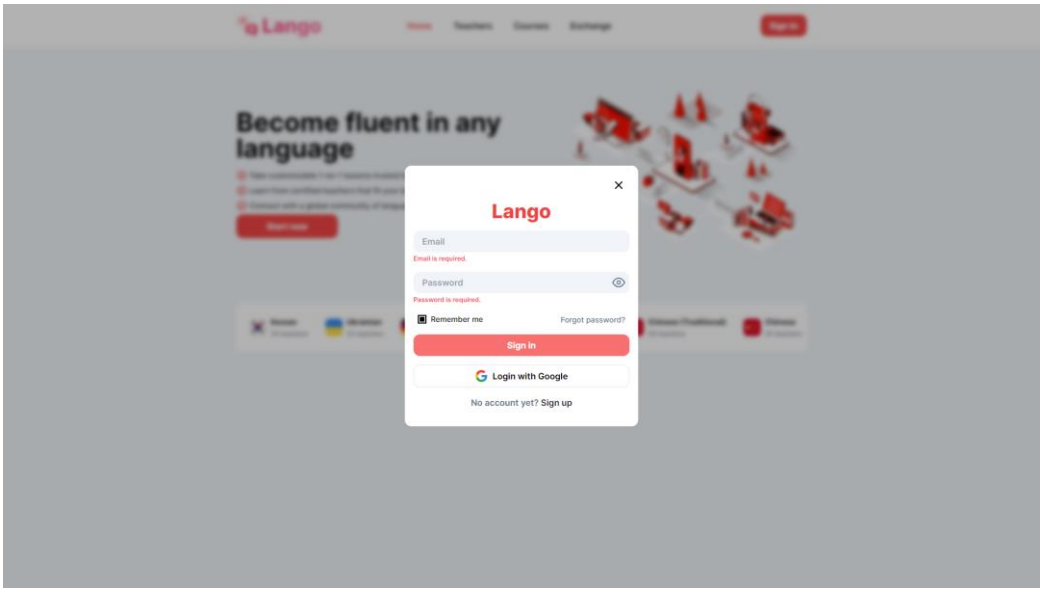


Рисунок 5.7 – Тестування форми у веб-додатку (рисунок виконано самостійно)

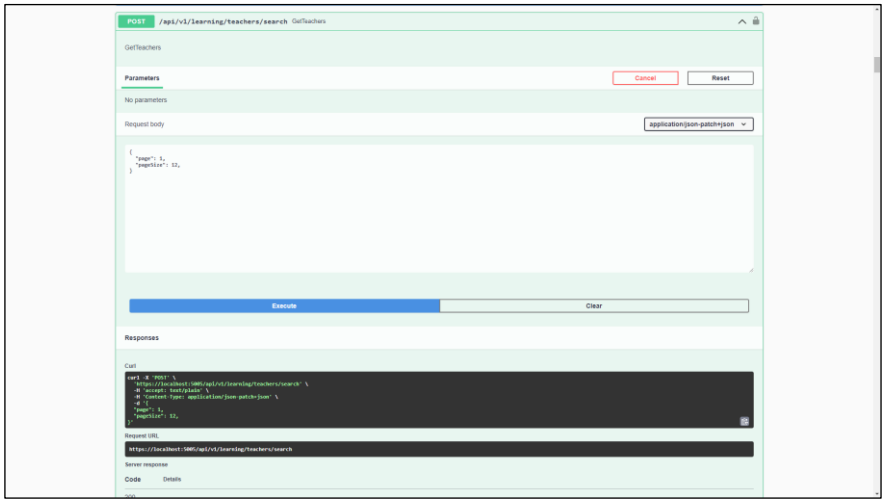


Рисунок 5.8 – Тестування ендпоінту за допомогою утиліти Swagger (рисунок виконано самостійно)

Ручне тестування дозволяє виявити проблеми, які можуть бути пропущені автоматичними тестами, особливо ті, що стосуються зручності використання та взаємодії з користувачем.

ВИСНОВКИ

Виконана робота з розробки освітньої платформи для навчання іноземних мов є значним кроком у напрямку забезпечення доступу до якісної та ефективної освіти в цій сфері. Теоретичний аспект дослідження відображає аналіз сучасних тенденцій у галузі онлайн-освіти та методів вивчення іноземних мов, що дозволило врахувати найкращі практики та інновації при розробці платформи.

Практичний аспект роботи передбачав створення функціональної, зручної у використанні та ефективної платформи, схожої на сервіси italki та preply. Розроблена платформа враховує індивідуальні потреби користувачів, надає широкий спектр сервісів, а також акцентує увагу на практичному вивченні мови, що робить процес навчання більш ефективним та захопливим[1].

Оцінка та аналіз використаних досліджень підтверджують необхідність подальшого розвитку та вдосконалення платформи. Виявлені проблеми та недоліки інших подібних сервісів стали основою для уникнення їх у розробці даної платформи, а також визначили напрямки подальших досліджень у цій галузі.

Загальним висновком є те, що розроблена освітня платформа має великий потенціал у поліпшенні процесу вивчення іноземних мов та задоволенні потреб користувачів у сфері освіти[5]. Пропозиції для вирішення розглянутих у роботі проблем включають постійне вдосконалення функціоналу платформи, впровадження нових методик навчання та розширення співпраці з викладачами та експертами у галузі мовної освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Developing Equitable Education Systems / M. Ainscow et al. London : Routledge, 2011. 192 p.

2. Application and Practice of VR Virtual Education Platform in Improving the Quality and Ability of College Students. IEEE Access. 2020. Vol. 8. P. 162830–162837.

3. THE LEARNING PLATFORM IN DISTANCE HIGHER EDUCATION: STUDENT'S PERCEPTIONS. Turkish Online Journal of Distance Education-TOJDE. 2019. Vol. 20, no. 1. 5.

4. Lai, C., Shum, M. and Tian, Y. (2014) 'Enhancing learners' self-directed use of technology for language learning: the effectiveness of an online training platform', Computer Assisted Language Learning, 29(1), pp. 40–60. doi: 10.1080/09588221.2014.889714.

5. E-Learning Environment: Blogging as a Platform for Language Learning. European Journal of Social Sciences. 2009. Vol. 9, no. 4. P. 594–604.

6. Leveraging Skype-based Webinars as an English Language Learning Platform. Al-Ishlah: Jurnal Pendidikan. 2021. Vol. 13, no. 1. P. 10–20.

7. Newman S. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, 2015. 278 p.

8. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison Wesley Professional, 2003. 560 p.

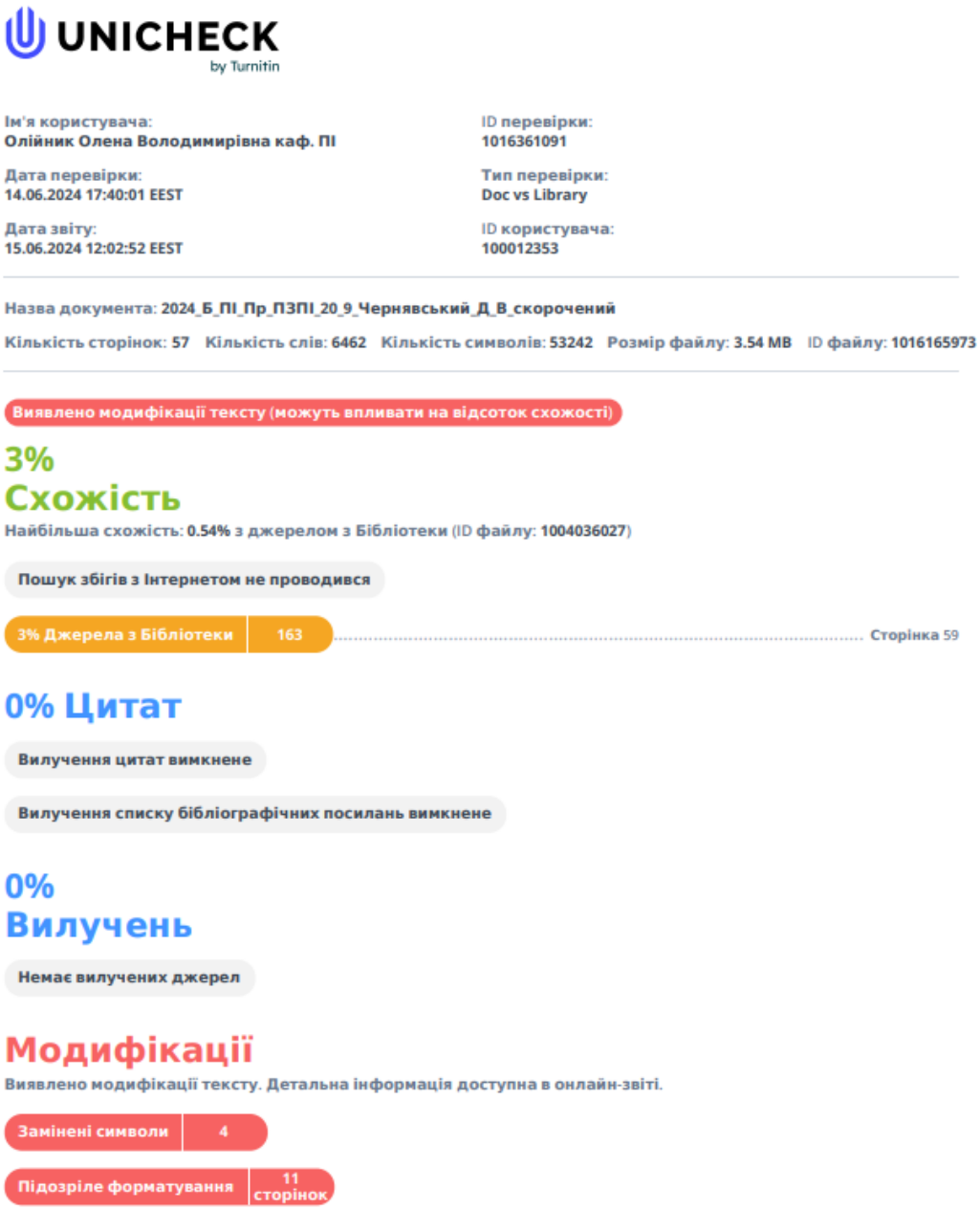
9. Wolff E. Microservices: A Practical Guide. 2nd ed. Scotts Valley: CreateSpace Independent Publishing Platform, 2018. 335 p.

10. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson Education Asia, 2017.

11. Exploring CQRS and Event Sourcing: A journey into high scalability, availability, and maintainability with Windows Azure. Microsoft patterns & practices, 2013. 376 p.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016361091

Дата перевірки:
14.06.2024 17:40:01 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
15.06.2024 12:02:52 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_Пр_ПЗПІ_20_9_Чернявський_Д_В_скорочений

Кількість сторінок: 57 Кількість слів: 6462 Кількість символів: 53242 Розмір файлу: 3.54 MB ID файлу: 1016165973

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3%
Схожість

Найбільша схожість: 0.54% з джерелом з Бібліотеки (ID файлу: 1004036027)

Пошук збігів з Інтернетом не проводився

3% Джерела з Бібліотеки 163 ..... Сторінка 59

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 4

Підозріле форматування 11 сторінок

ДОДАТОК Б

Слайди презентації

ХНУРЕ, КАФЕДРА ПІ
КВАЛІФІКАЦІЙНА РОБОТА

ОСВІТНЯ ПЛАТФОРМА ДЛЯ НАДАННЯ ТА СПОЖИВАННЯ МОВНИХ ПОСЛУГ

ВИКОНАВ:
СТ. ГР. ПЗПІ-20-9
ЧЕРНЯВСЬКИЙ ДМИТРО

КЕРІВНИК:
СТ. ВИКЛ. КАФ. ПІ ОЛІЙНИК О. В.

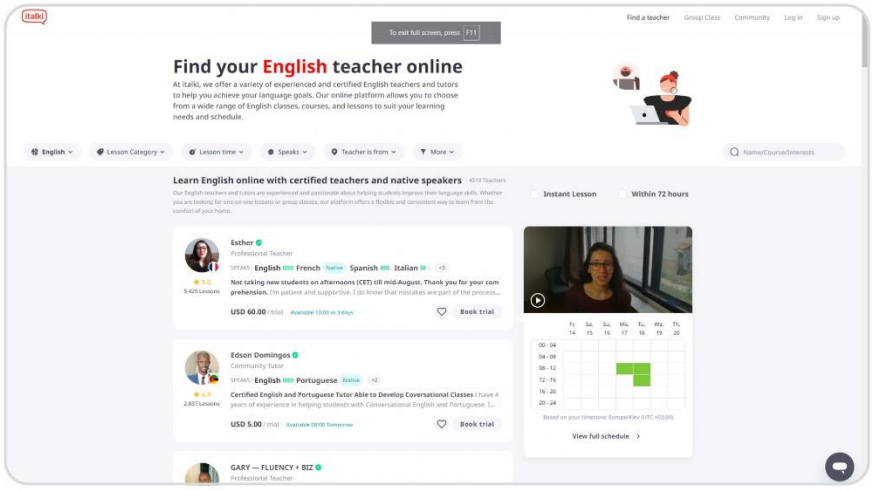
2

АКТУАЛЬНІСТЬ

- ЗРОСТАЮЧИЙ ПОПИТ НА ОНЛАЙН-ОСВІТУ
- ГНУЧКІСТЬ ТА АДАПТИВНІСТЬ
- ПІДТРИМКА ІНДИВІДУАЛЬНИХ ПОТРЕБ
- ІНТЕРАКТИВНЕ НАВЧАННЯ
- АВТОМАТИЗАЦІЯ АДМІНІСТРАТИВНИХ ПРОЦЕСІВ
- ЕФЕКТИВНІСТЬ
- ПІДТРИМКА БЕЗПЕРЕРВНОГО НАВЧАННЯ
- КОНКУРЕНТНІСТЬ

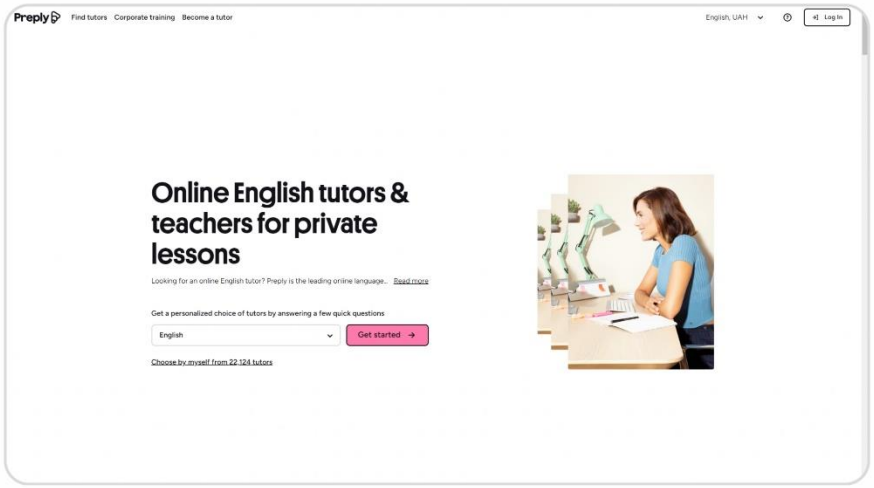
3

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ



4

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ



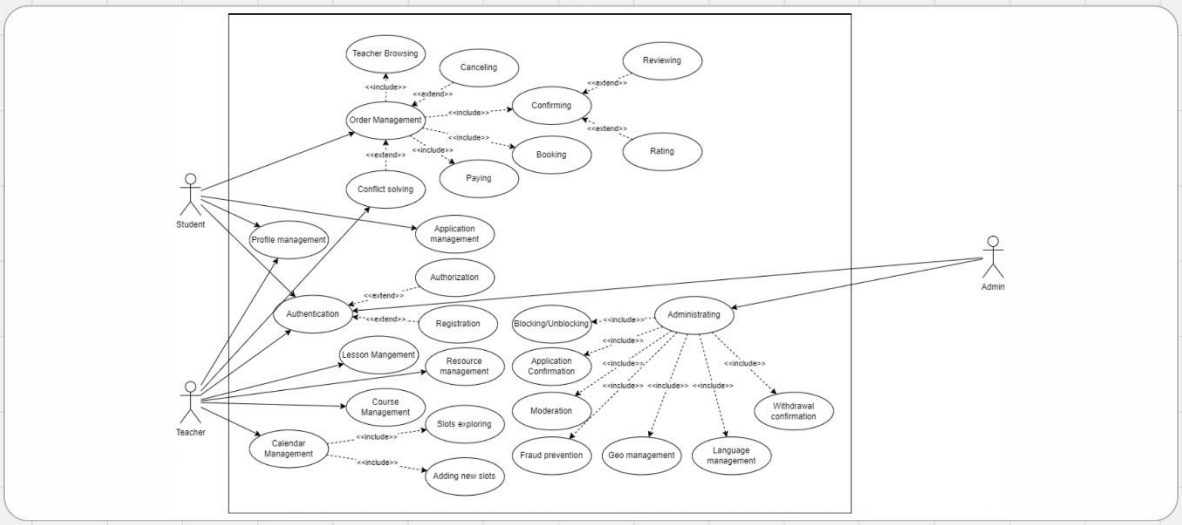
5

ПОСТАНОВКА ЗАДАЧІ

ОСНОВНОЮ МЕТОЮ ДАНОГО ДОСЛІДЖЕННЯ Є РОЗРОБКА ТА ВПРОВАДЖЕННЯ ОСВІТНЬОЇ ПЛАТФОРМИ ДЛЯ НАДАННЯ ТА СПОЖИВАННЯ ПОСЛУГ З ВИВЧЕННЯ ІНОЗЕМНИХ МОВ, ЯКА Б ВІДПОВІДАЛА СУЧАСНИМ ВИМОГАМ КОРИСТУВАЧІВ І СПРИЯЛА ПОКРАЩЕННЮ ПРОЦЕСУ НАВЧАННЯ.

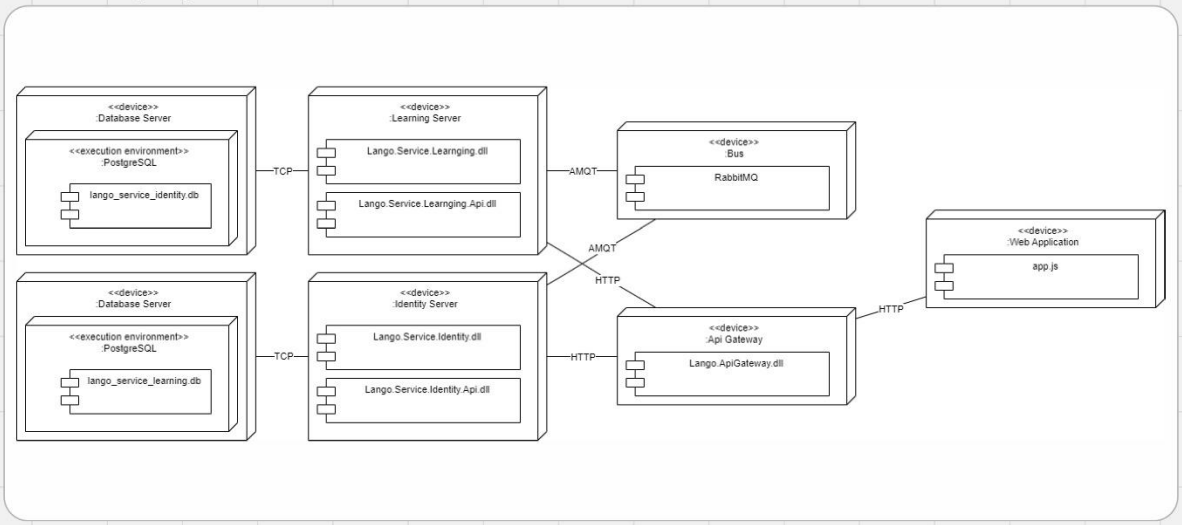
6

USE CASE ДІАГРАМА



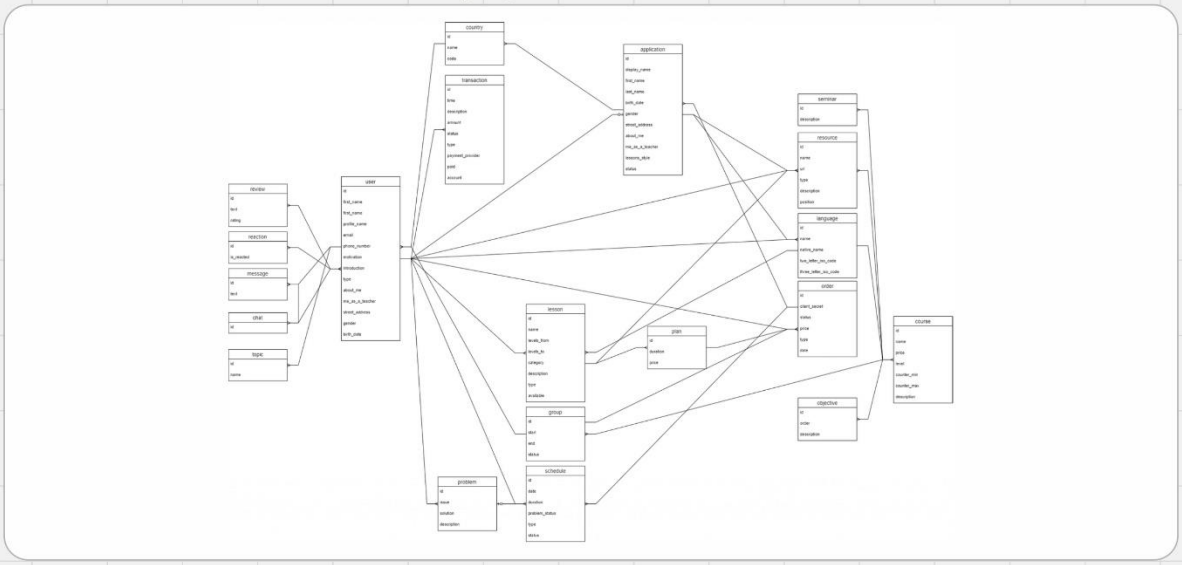
7

ДІАГРАМА РОЗГОРТАННЯ



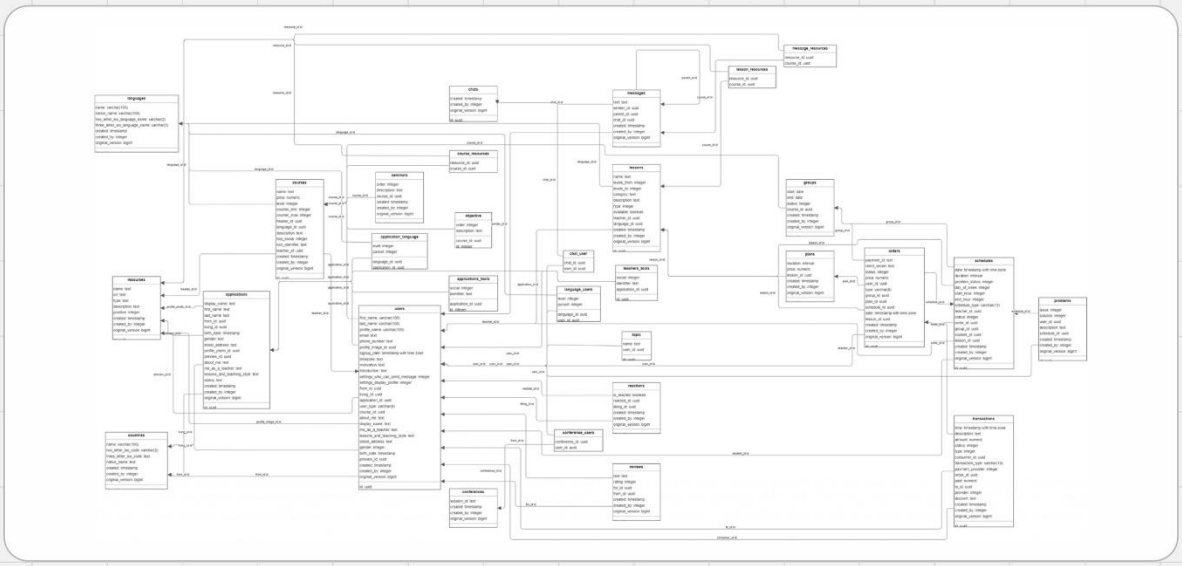
8

ER ДІАГРАМА



9

СХЕМА БАЗИ ДАНИХ



10

ТЕХНОЛОГІЇ

ASP.NET Core

PostgreSQL

redis

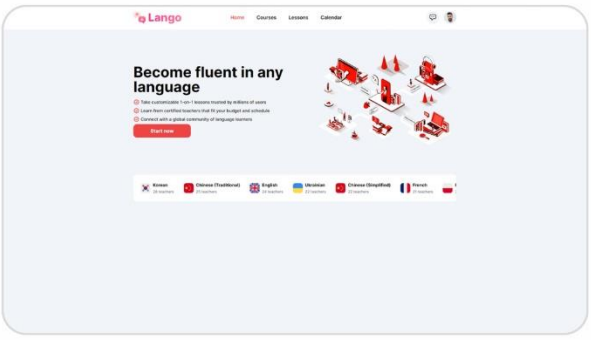
RabbitMQ

Angular

11

ГОЛОВНА СТОРІНКА

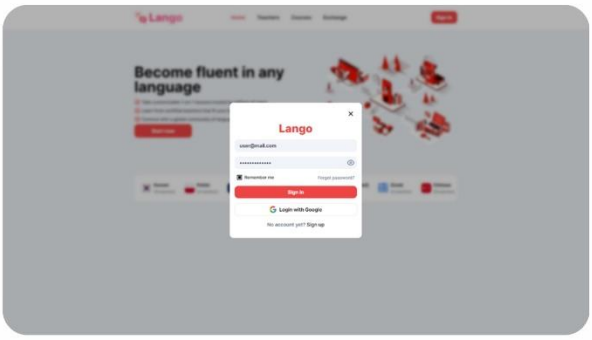
```
@for (l of languages; track l) {
    <a
      [routerLink]="['../teachers', l.name?.toLowerCase()]"
      class="flex flex-row gap-2 p-2 hover:bg-slate-100 rounded-lg transition-all duration-500
cursor-pointer">
      <div class="my-auto">
        <svg-icon src="assets/icons/flags/language/{{l.twoLetterISOLanguageName}}.svg"
                  svgClass="w-8 h-8 rounded-lg"></svg-icon>
      </div>
      <div class="flex flex-col gap-1 my-auto">
        <span class="font-bold text-sm whitespace-nowrap">{{ l.name }}</span>
        <span class="text-xs text-gray-500 font-medium whitespace-nowrap">{{ l.teacherCount }}
teachers</span>
      </div>
    </a>
  }
```



12

АВТОРИЗАЦІЯ

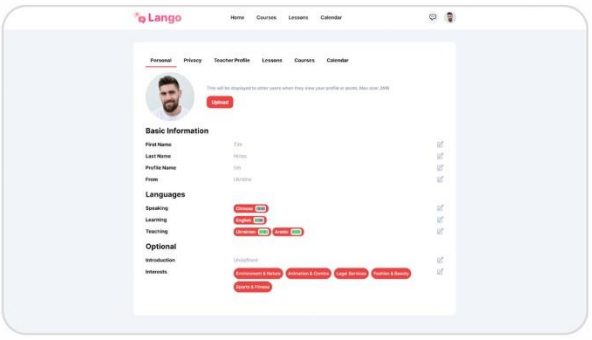
```
login(form: NgForm) {
  if (form.valid) {
    this.signing$.next(true);
    this.authService
      .login(this.loginRequest)
      .pipe(takeUntil(this.unsubscriber$), finalize(() => {
        this.signing$.next(false)
      }))
      .subscribe({
        next: (res) => {
          if (res) {
            this.modalWindowService.isAuthWindowHidden$.next(true);
            this.onlogin.emit();
          } else {
            this.invalid = true;
          }
        },
      });
  }
}
```



13

КАБІНЕТ

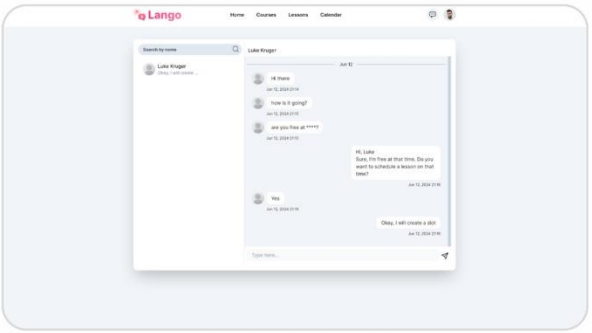
```
export let profileResolver: ResolveFn<any>;
profileResolver =
  (route: ActivatedRouteSnapshot, state: RouterStateSnapshot) => {
    return inject(UsersService)
      .getProfile()
      .pipe(map(res => res.userProfile!))
  };
```



14

ЧАТ

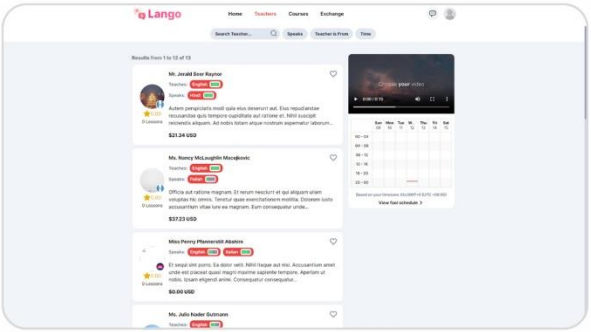
```
private hubConnection: HubConnection = new HubConnectionBuilder()
    .withUrl(environment.apiUrl + '/signalr/v1/learning/chat')
    .build();
  connect(id: string) {
    this.hubConnection.start().then((_) => {
      this.hubConnection.on('receiveMessage', (data: MessageDto) => {
        this.messages.push(data);
        this.chats.find(x => x.id === this.chatId)!.lastMessage = {
          text: data.text
        }
        this.group();
        this.scrollToBottom();
        this.readChat(this.chatId!).subscribe({
          next: res => {
            this.getUnreadMessages()
          }
        })
      });
    });
  }
```



15

ПОШУК ВЧИТЕЛЯ

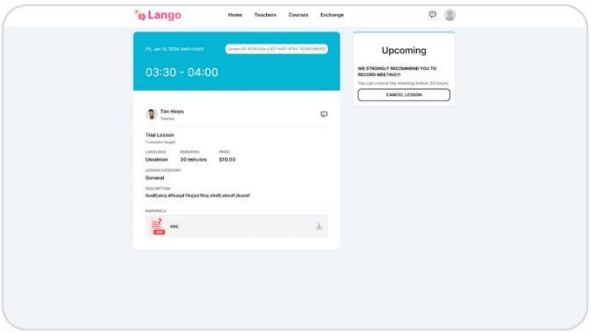
```
public record GetTeachersRequest(string? SearchText, ICollection<string>? Countries, bool Liked);
public record GetTeachers(string? SearchText, ICollection<string>? Countries, bool Liked) :
ListQuery<GetTeachersResponse>;
public record GetTeachersResponse(ListResultModel<TeacherBriefDto> Body);

public class GetTeachersHandler : IQueryHandler<GetTeachers, GetTeachersResponse>
{
    private readonly IApplicationDbContext _context; private readonly IMapper _mapper; private readonly
 ISecurityContextAccessor _securityContextAccessor;
    public GetTeachersHandler(IApplicationDbContext context, IMapper mapper, ISecurityContextAccessor
securityContextAccessor)
    { _context = context; _mapper = mapper; _securityContextAccessor = securityContextAccessor; }
    public async Task<GetTeachersResponse> Handle(GetTeachers request, CancellationToken
cancellationToken)
    {
        var userId = _securityContextAccessor.ParseUserId();
        var query = _context.Users.OfType<Teacher>().AsNoTracking().Where(x => x.Lessons.Any() ||
x.Courses.Any());
        // Add query conditions based on request properties here
        var entities = await query.ProjectTo<TeacherBriefDto>(_mapper.ConfigurationProvider, new {
userId })
                                  .ApplyPagingAsync(request.Page, request.PageSize,
cancellationToken: cancellationToken);
        // Additional processing on entities if needed
        return new GetTeachersResponse(entities);
    }
}
```



16

СТОРІНКА УРОКУ

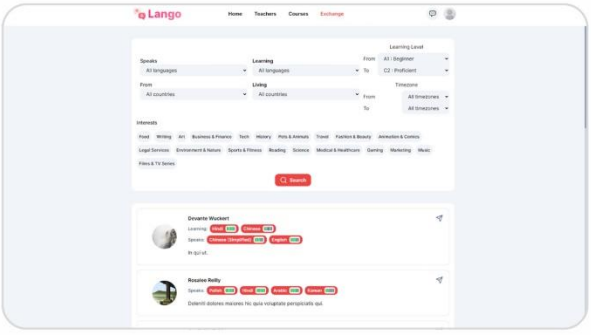
```
public class NotifyIn24HoursJob : IJob
{
    private readonly ISender _sender;

    public NotifyIn24HoursJob(ISender sender)
    {
        _sender = sender;
    }

    public async Task Execute(IJobExecutionContext context)
    {
        await _sender.Send(new SendNotification(SendIn.OneDay));
    }
}
```



17

ПОШУК ДЛЯ ОБМІНУ

```
    public async Task<GetStudentsResponse> Handle(GetStudents request, CancellationToken
cancellationToken)
    {
        var userId = _securityContextAccessor.ParseUserId();
        var query = _context.Users.OfType<Student>();
        if (request.SpeaksId is not null)
            query = query.Where(
                x => x.Languages.Any(
                    y => y.Pursuit == Pursuit.Speaking && y.LanguageId == request.SpeaksId &&
                         y.Level >= request.From && y.Level <= request.To));
        if (request.LearningId is not null)
            query = query.Where(
                x => x.Languages.Any(y => y.Pursuit == Pursuit.Learning && y.LanguageId ==
request.SpeaksId));
        if (request.FromId is not null)
            query = query.Where(x => x.FromId == request.FromId);
        if (request.LivingId is not null)
            query = query.Where(x => x.LivingId == request.LivingId);
        if (request.Topics is not null)
            query = query.Where(x => x.Topics.Any(y => request.Topics.Any(z => z == y.Name)));
        var entities = await query
                        .Where(x => x.Id != userId)
                        .AsNoTracking()
                        .Include(x => x.Languages)
                        .ProjectTo<StudentDto>(_mapper.ConfigurationProvider, new {userId})
                        .ApplyPagingAsync(request.Page, request.PageSize, cancellationToken:
cancellationToken);
        return new GetStudentsResponse(entities);
    }
```



18

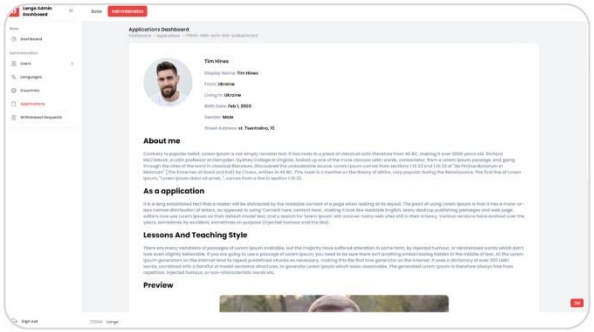
АДМІНІСТРАТИВНА ПАНЕЛЬ

```
<app-graph [chartOptions]="userChartOptions"></app-graph>

//*************************************

@if (chartOptions) {
  <apx-chart
    [series]="chartOptions.series!"
    [chart]="chartOptions.chart!"
    [legend]="chartOptions.legend!"
    [dataLabels]="chartOptions.dataLabels!"
    [fill]="chartOptions.fill!"
    [stroke]="chartOptions.stroke!"
    [xaxis]="chartOptions.xaxis!"
    [yaxis]="chartOptions.yaxis!"
    [states]="chartOptions.states!"
    [tooltip]="chartOptions.tooltip!"
    [colors]="chartOptions.colors!"
    [grid]="chartOptions.grid!"
    [title]="chartOptions.title!">
  </apx-chart>
}
```

19

ЗАПИТ НА ВЧИТЕЛЮВАННЯ

```
public class
    GetApplicationsEndpoint :
EndpointBaseAsync.WithRequest<GetApplicationsRequest>.WithActionResult<GetApplicationsResponse>
{
    private readonly IQueryProcessor _queryProcessor;

    public GetApplicationsEndpoint(IQueryProcessor queryProcessor)
    {
        _queryProcessor = queryProcessor;
    }

    [HttpGet(ApplicationConfigs.PrefixUri, Name = "GetApplications")]
    [ProducesResponseType(typeof(GetApplicationsResponse), StatusCodes.Status200OK)]
    [ProducesResponseType(typeof(StatusCodeProblemDetails), StatusCodes.Status401Unauthorized)]
    [ProducesResponseType(typeof(StatusCodeProblemDetails), StatusCodes.Status400BadRequest)]
    [SwaggerOperation(
        Summary = "Get Applications",
        Description = "Get Applications",
        OperationId = "GetApplications",
        Tags = new[]
                {
                    ApplicationConfigs.Tag
                })]
    [ApiVersion(1.0)]
    [Authorize(Roles = Roles.Admin)]
    public override async Task<ActionResult<GetApplicationsResponse>> HandleAsync(
        [FromQuery] GetApplicationsRequest request,
        CancellationToken cancellationToken = new CancellationToken()
    )
    {
        var query = new GetApplications()
                    {
                        Page = request.Page,
                        PageSize = request.PageSize,
                    };
        return await _queryProcessor.SendAsync(request, cancellationToken);
    }
}
```



20

ТЕСТУВАННЯ

- UNIT-ТЕСТУВАННЯ
- ІНТЕГРАЛЬНЕ ТЕСТУВАННЯ
- НАВАНТАЖУВАЛЬНЕ ТЕСТУВАННЯ
- МАНУАЛЬНЕ ТЕСТУВАННЯ

NBomber

fluent ASSERTIONS

21

ВИСНОВКИ

- ПРОВЕДЕНО АНАЛІЗ ГАЛУЗІ
- ДОСЛІДЖЕНО ІСНУЮЧИ РІШЕННЯ
- СФОРМУЛЬОВАНО ВИМОГИ
- РОЗРОБЛЕНО СЕРВЕРНИЙ ТА КЛІЄНТСЬКИЙ ДОДАТОК, ЩО ВІДПОВІДАЄ ВИМОГАМ
- ПРОТЕСТОВАНО СИСТЕМУ ВИКОРИСТОВУЮЧИ РІЗНІ ТИПИ ТЕСТУВАННЯ

ДЯКУЮ ЗА УВАГУ

"Освіта - це найпотужніша зброя, за допомогою якої можна змінити світ". - Нельсон Мандела