

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ САЙТУ ПОШУКУ РЕПЕТИТОРІВ
SOFTWARE IMPLEMENTATION OF A TUTOR SEARCH WEBSITE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-
220

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

ПУТІНЦЕВА П.Д.

(ПРИЗВИЩЕ та ініціали)

Керівник КАПЛІЄНКО Т.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПУТІНЦЕВОЇ Поліни Дмитрівни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація сайту пошуку репетиторів.
Software Implementation of a Tutor Search Website

керівник проєкту (роботи) к.т.н., доцент, КАПЛІЄНКО Тетяна Ігорівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Поняття і призначення сайтів для пошуку репетиторів.
2. Розробка архітектури програми. 3. Реалізація основних компонентів програми.
4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	КАПЛІЄНКО Т.І., доцент		
Нормоконтролер	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “15” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів	2 тиждень	Розділ 2
	вирішення задачі.		
4	Розробка архітектури програми.	2 тиждень	Розділ 2
5	Розробка програми.	3-4 тижні	Розділ 3
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Поліна ПУТІНЦЕВА
(підпис) (Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Тетяна КАПЛІЄНКО
(підпис) (Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 122 с., 6 табл., 29 рис., 2 дод., 38 джерел.

ВЕБЗАСТОСУНОК, ПОШУК РЕПЕТИТОРІВ, DJANGO, PYTHON, POSTGRESQL, MODEL-VIEW-CONTROLLER, АВТЕНТИФІКАЦІЯ, ОСОБИСТИЙ КАБІНЕТ, БРОНЮВАННЯ ЗАНЯТЬ.

Об'єкт дослідження – процес розробки вебзастосунків для пошуку репетиторів.

Предмет дослідження – архітектура, технології та методи реалізації вебзастосунків для пошуку репетиторів.

Мета роботи – розробка вебзастосунку для пошуку репетиторів.

Матеріали, методи та технічні засоби: фреймворк Django, мова програмування Python, СУБД PostgreSQL, шаблон Model-View-Controller, HTML, CSS, JavaScript, середовище розробки PyCharm, хмарний вебсервер Amazon EC2.

Результати. Розроблено вебзастосунок для пошуку репетиторів. Реалізовано функціонал реєстрації та автентифікації користувачів, особистий кабінет репетитора, пошук репетиторів за різними критеріями, бронювання занять та управління розкладом.

Висновки. Розроблений вебзастосунок відповідає сучасним вимогам до онлайн-освіти та вебтехнологій, забезпечує зручний та ефективний пошук викладачів, управління навчальним процесом та взаємодію між учнями і репетиторами. Отримані результати мають практичну цінність і можуть бути використані для подальшого розвитку онлайн-платформ у сфері додаткової освіти.

Галузь використання – онлайн-освіта, додаткове навчання, пошук репетиторів та організація індивідуальних занять.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 122 pages, 6 tables, 29 figures, 2 appendixes, 38 sources.

WEB APPLICATION, TUTOR SEARCH, DJANGO, PYTHON, POSTGRESQL, MODEL-VIEW-CONTROLLER, AUTHENTICATION, PERSONAL ACCOUNT, LESSON BOOKING.

Object of research – the process of developing web applications for tutor search.

Subject of research – architecture, technologies, and methods of implementing web applications for tutor search.

The purpose of the work is to develop a web application for tutor search.

Materials, methods and technical means: Django framework, Python programming language, PostgreSQL DBMS, Model-View-Controller pattern, HTML, CSS, JavaScript, PyCharm development environment, Amazon EC2 cloud web server.

Results. A web application for tutor search has been developed. Implemented functionality includes user registration and authentication, tutor's personal account, tutor search by various criteria, lesson booking and schedule management, administrative panel for content moderation.

Conclusions. The developed web application meets modern requirements for online education and web technologies, provides convenient and efficient search for teachers, management of the learning process and interaction between students and tutors. The obtained results have practical value and can be used for further development of online platforms in the field of additional education.

Field of use – online education, additional training, tutor search and organization of individual lessons.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК.....	8
ВСТУП	9
1 ПОНЯТТЯ І ПРИЗНАЧЕННЯ САЙТІВ ДЛЯ ПОШУКУ РЕПЕТИТОРІВ	11
1.1 Історія репетиторства та сучасний стан ринку	11
1.2 Поняття та призначення сайтів пошуку репетиторів.....	13
1.3 Поняття та призначення сайтів пошуку репетиторів.....	16
1.4 Аналіз сайту "preply.com"	18
1.5 Аналіз сайту "buki.com.ua"	21
1.6 Аналіз сайту "superprof.com.ua"	24
1.7 Порівняльний аналіз сайтів-аналогів.....	27
1.8 Успіхи та виклики сайтів аналогів.....	30
2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ	33
2.1 Визначення вимог та функціональності системи	33
2.2 Вибір технологій та інструментів розробки.....	36
2.3 Проектування архітектури системи	41
2.4 Розробка структури бази даних	45
3 РЕАЛІЗАЦІЯ ОСНОВНИХ КОМПОНЕНТІВ СИСТЕМИ	50
3.1 Реалізація модуля реєстрації та автентифікації користувачів	50
3.2 Розробка особистого кабінету репетитора	54
3.3 Реалізація функціоналу пошуку репетиторів.....	57
3.4 Реалізація функціоналу бронювання та управління заняттями ...	61
3.5 Розробка адміністративної панелі та модерації контенту	65

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ	71
4.1 Розгортання вебзастосунку на сервері	71
4.2 Тестування функціональності та безпеки системи	74
4.3 Оцінка продуктивності та масштабованості застосунку	78
4.4 Напрямки подальшого розвитку та вдосконалення системи	82
ВИСНОВКИ	87
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ	90
ДОДАТОК А Текст програми	95
ДОДАТОК Б Слайди презентації.....	113

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ЗНО – зовнішнє незалежне оцінювання;

МОН – Міністерство освіти і науки;

СУБД – система управління базами даних;

CRUD – Create, Read, Update, Delete;

CSS – Cascading Style Sheets;

ER-діаграма – Entity-Relationship діаграма;

HTML – HyperText Markup Language;

IDE – Integrated Development Environment;

JSON – JavaScript Object Notation;

KPI – Key Performance Indicator;

MVC – Model-View-Controller;

ORM – Object-Relational Mapping;

SEO – Search Engine Optimization.

SQL – Structured Query Language;

XSS – Cross-Site Scripting;

ВСТУП

Онлайн-репетиторство набуває все більшої популярності в сучасному світі, оскільки надає учням гнучкість, доступність та персоналізований підхід до навчання. Особливо актуальним воно стало в умовах пандемії COVID-19, коли традиційні форми навчання зазнали значних обмежень. Проте ефективність онлайн-репетиторства значною мірою залежить від якості та зручності використання відповідних вебресурсів. Саме тому розробка спеціалізованих вебзастосунків для пошуку репетиторів є важливим завданням, яке потребує комплексного підходу та врахування сучасних тенденцій у сфері веброзробки.

Дослідженням питань онлайн-репетиторства та розробки вебзастосунків для освітньої сфери займалися такі науковці, як М. Г. Бустер [1], Л. Шуба [2], Марія [3] та інші. Вони розглядали історію розвитку репетиторства, аналізували сучасний стан ринку онлайн-репетиторства та визначали ключові особливості успішних вебплатформ для пошуку репетиторів. Проте, незважаючи на наявність досліджень у цій сфері, питання розробки ефективних та зручних вебзастосунків для пошуку репетиторів залишається актуальним і потребує подальшого вивчення.

Об'єктом дослідження в роботі є процес розробки вебзастосунків для пошуку репетиторів, а предметом дослідження - архітектура, технології та методи реалізації таких застосунків.

Метою роботи є розробка вебзастосунку для пошуку репетиторів, який забезпечуватиме зручний та ефективний пошук викладачів для проведення індивідуальних занять онлайн, підвищуючи доступність та якість додаткової освіти.

Для досягнення поставленої мети у роботі вирішувалися такі завдання:

- проаналізувати предметну область та визначити основні вимоги до функціональності вебзастосунку;

- розробити архітектуру вебзастосунку та обрати відповідні технології для його реалізації;
- спроектувати та реалізувати базу даних для зберігання інформації про репетиторів, учнів та навчальні предмети;
- розробити основні компоненти системи: модуль реєстрації та автентифікації користувачів, особистий кабінет репетитора, функціонал пошуку репетиторів, бронювання занять, адміністративну панель;
- провести тестування функціональності та безпеки вебзастосунку, оцінити його продуктивність та масштабованість;
- визначити напрямки подальшого розвитку та вдосконалення системи.

Методи дослідження включають аналіз літературних джерел, узагальнення досвіду розробки подібних систем, проектування архітектури та бази даних вебзастосунку, програмну реалізацію з використанням сучасних вебтехнологій, тестування та експериментальне дослідження розробленої системи.

Практична значущість роботи полягає в розробці функціонального та зручного у використанні вебзастосунку, який може бути використаний як учнями для пошуку репетиторів, так і викладачами для надання своїх послуг та управління навчальним процесом. Результати роботи можуть бути корисними для подальших досліджень та розробок у сфері онлайн-освіти та вебтехнологій.

Таким чином, розробка вебзастосунку для пошуку репетиторів є актуальним завданням, яке має практичну цінність і потребує комплексного підходу з використанням сучасних вебтехнологій та методів розробки програмного забезпечення.

1 ПОНЯТТЯ І ПРИЗНАЧЕННЯ САЙТІВ ДЛЯ ПОШУКУ РЕПЕТИТОРІВ

1.1 Історія репетиторства та сучасний стан ринку

Професія репетитора має давню історію, сягаючи ще часів Стародавнього Єгипту, Греції та Риму. У ті часи репетиторство було прерогативою вищих класів суспільства, які могли дозволити собі персональне навчання вдома, тоді як нижчі класи здобували освіту в групових умовах [1].

У Стародавньому Китаї деякі аристократичні репетитори, такі як Конфуцій та Менцзи, приваблювали багато учнів і заснували впливові філософські школи. У Стародавній Греції софісти створювали прибуткові кар'єри, навчаючи ораторському мистецтву для виступів в асамблеях та судах. Сократ у творах Платона безкоштовно надавав подібні послуги, критикуючи софістів за їх припустиму відсутність турботи про істину та розуміння. У Стародавньому Римі освіту патриціїв і заможних плебеїв часто забезпечували грецькі раби [2].

Королівські репетитори, особисті наставники майбутніх правителів, завжди користувалися важливістю і престижем. Зокрема, Великий Наставник (тайфу) був одним із трьох великих лордів королівського двору династії Чжоу у Стародавньому Китаї. Подібні позиції залишалися дуже важливими у Східній Азії аж до сучасної епохи.

У XX столітті, зі швидким поширенням масової освіти, попит на репетиторство значно зріс як спосіб доповнити формальну освіту. Репетиторство стало більш структурованим і спеціалізованим, з'явилися різноманітні методи та підходи до навчання [3].

Сьогодні репетиторство є поширеним явищем у багатьох країнах світу, виступаючи як доповнення до державної та приватної шкільної освіти.

Глобальний ринок щорічно росте, і до 2030 року може сягати 150 млрд. дол. США (рис. 1.1) [4].



Рисунок 1.1 - Глобальний ринок репетиторів, 2018-2023 у млн. дол. США [5]

Найбільша країна за обсягом репетиторських послуг - це Сполучені Штати Америки. За прогнозами, ринок репетиторства у США сягне більш як 25 млрд доларів до 2026 року, зростаючи щорічно на 6,3%. У 2023 році індустрія репетиторства США склала 21,8 млрд доларів, що становить вагому частку світового ринку репетиторських послуг близько 90 млрд дол.

Дослідження показують, що приблизно 20% учнів старших класів у США користувалися послугами репетиторів принаймні один раз. Репетиторство особливо популярне серед сімей з доходом понад 100 тисяч доларів на рік - 74% з них зверталися до репетиторів [6]. У секторі приватного репетиторства для учнів від дитячого садка до 12 класу (К-12) у 2019 році дохід склав 2,2 млрд доларів. Очікується, що попит на онлайн-послуги зростатиме на 5,6% щорічно до 2024 року.

Понад 10% студентів коледжів у США користуються приватними репетиторськими послугами, а індивідуальне навчання складає понад 30% усіх

репетиторських послуг. Репетиторські центри обслуговують приблизно кожного четвертого учня державних шкіл по всій Америці. Індивідуальні репетитори, на яких припадає 61,4% послуг, беруть в середньому від 30 до 60 доларів за годину.

За даними МОН України, понад 60% випускників шкіл користуються послугами репетиторів для підготовки до ЗНО [7]. В середньому українські сім'ї витрачають близько \$380 на рік на репетиторські послуги, а загальний обсяг цього ринку оцінюється в \$600 млн. В той же час, через девальвацію гривні вартість послуг репетиторів знизилася з \$10-15 до \$5 за годину, що значно нижче рівня розвинених країн [8].

Розвиток інтернет-технологій відкрив нові можливості для онлайн-репетиторства, дозволяючи учням та репетиторам взаємодіяти незалежно від географічного розташування. Це стало особливо актуальним в умовах пандемії COVID-19, коли дистанційне навчання перетворилося на необхідність.

Однак поширення репетиторства має і свої виклики. Воно може підтримувати й посилювати соціальну нерівність, відволікаючи кошти домогосподарств у нерегульовану індустрію та створюючи неефективність в системах освіти. Учні з сімей з високим соціально-економічним статусом мають вдвічі більше шансів отримати репетиторство, ніж їхні однолітки з нижчим статусом. Тому важливо, щоб політики в галузі освіти приділяли увагу регулюванню та контролю якості репетиторських послуг.

1.2 Поняття та призначення сайтів пошуку репетиторів

У сучасному світі, де освіта відіграє ключову роль у розвитку особистості та професійному успіху, сайти пошуку репетиторів стали невід'ємною частиною освітнього ландшафту. Ці спеціалізовані онлайн-

платформи призначені для того, щоб з'єднати учнів, які шукають додаткову допомогу у навчанні, з кваліфікованими та досвідченими репетиторами.

Сайти пошуку репетиторів виступають посередниками між учнями та репетиторами, створюючи зручний та ефективний спосіб знайти ідеального викладача для індивідуальних занять та підтримувати з ним зв'язок. Вони надають учням можливість знайти репетитора, який відповідає їхнім конкретним потребам, цілям та вимогам, незалежно від місця проживання чи графіку.

Призначення сайтів пошуку репетиторів полягає в спрощенні та оптимізації процесу пошуку та вибору репетитора для учнів [9]. Ці платформи надають учням доступ до широкого кола кваліфікованих викладачів з різних предметів та галузей знань. Учні можуть переглядати детальні профілі репетиторів, ознайомлюватися з їхньою освітою, досвідом, спеціалізацією, методикою викладання та відгуками від інших учнів. Це дозволяє їм приймати зважені рішення та обирати репетитора, який найкраще відповідає їхнім індивідуальним потребам та стилю навчання.

Окрім того, сайти пошуку репетиторів часто пропонують зручні інструменти та функції, які спрощують процес пошуку та комунікації. Учні можуть використовувати різноманітні фільтри, такі як предмет, рівень знань, місце проживання, ціна та доступність, щоб звужити список потенційних репетиторів. Деякі платформи також надають можливість безпосередньо зв'язатися з репетитором через вбудовані системи обміну повідомленнями або відеоконференції, що полегшує комунікацію та планування занять. Також, багато платформ гарантують оплату послуг або повернення коштів.

Для репетиторів сайти пошуку є потужним інструментом для просування своїх послуг та залучення нових учнів [10-11]. Ці платформи дозволяють репетиторам створювати детальні профілі, де вони можуть презентувати свої знання, досвід, освіту, сертифікати та спеціалізацію. Репетитори мають можливість розповісти про свою методику викладання,

стиль роботи та успіхи своїх учнів. Це допомагає їм виділитися серед інших викладачів та привернути увагу потенційних учнів, які шукають саме такі характеристики та навички.

Крім того, багато сайтів пошуку репетиторів надають викладачам зручні інструменти для управління своєю діяльністю. Вони можуть вести календар занять, відстежувати оплату, спілкуватися з учнями та батьками, а також отримувати відгуки та рекомендації. Деякі платформи також пропонують додаткові ресурси та підтримку для репетиторів, такі як навчальні матеріали, вебінари та професійні спільноти, що допомагає їм розвиватися та вдосконалювати свої навички викладання (рис. 1.2).

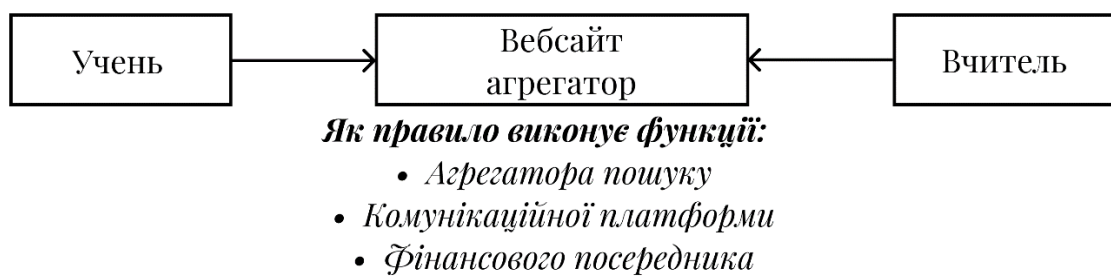


Рисунок 1.2 - Функції сайтів для пошуку репетиторів [12]

Сайти пошуку репетиторів відіграють важливу роль у сучасній освіті, надаючи учням персоналізовану підтримку та допомогу у досягненні їхніх навчальних цілей. Вони дозволяють учням отримати додаткові пояснення, практику та зворотний зв'язок від досвідчених викладачів, що може значно покращити їхнє розуміння предмету та академічні результати. Для репетиторів ці платформи є ефективним способом знайти учнів, розширити свою викладацьку діяльність та побудувати успішну кар'єру в освітній сфері.

1.3 Поняття та призначення сайтів пошуку репетиторів

Однією з найбільших переваг онлайн-репетиторства є гнучкість та зручність. Завдяки сайтам пошуку репетиторів учні можуть знайти ідеального викладача з будь-якої точки світу та займатися в зручний для них час. Це особливо актуально для тих, хто має щільний графік або проживає у віддалених регіонах, де доступ до якісних репетиторських послуг може бути обмеженим. Онлайн-заняття дозволяють заощадити час та кошти на транспорт, що робить репетиторство більш доступним для ширшого кола учнів [13].

Крім того, багато сайтів пошуку репетиторів пропонують безкоштовні навчальні матеріали та поради щодо вивчення різних предметів. Це можуть бути статті, відео-уроки, інтерактивні завдання, вебінари тощо. Такі ресурси є цінним доповненням до основних занять з репетитором та дозволяють учням самостійно поглиблювати свої знання та закріплювати вивчений матеріал. Наявність додаткових навчальних матеріалів сприяє всебічному розвитку учнів та підвищує ефективність репетиторських занять [14].

Сайти пошуку репетиторів також відкривають широкі можливості для міжнародного спілкування та обміну досвідом. Учні можуть знайти репетиторів-носіїв мови для вивчення іноземних мов, що дозволяє їм не лише покращити мовні навички, а й зануритися в культуру та традиції країни, мову якої вони вивчають. Крім того, учні мають можливість займатися з викладачами з різних країн світу, що збагачує їхній культурний досвід та розширює світогляд. Репетитори, в свою чергу, можуть працювати з учнями з різних куточків планети, ділитися своїми знаннями та унікальними методиками викладання, а також вчитися один в одного [15].

У 2024 році сайти пошуку репетиторів відіграють важливу роль у наданні роботи людям з усього світу. Вони розширюють можливості працевлаштування для репетиторів, дозволяючи їм знаходити учнів незалежно від географічного розташування. Це особливо актуально для викладачів з невеликих міст або тих, хто має обмежені можливості працевлаштування у своєму регіоні. Завдяки онлайн-платформам репетитори можуть будувати успішну кар'єру, працювати з учнями з різних країн та забезпечувати себе стабільним доходом. Це сприяє глобалізації ринку репетиторських послуг та надає викладачам більше свободи та гнучкості у їхній професійній діяльності [16].

Варто також зазначити, що сайти пошуку репетиторів відіграють важливу роль у забезпеченні доступу до освіти для людей з обмеженими можливостями. Завдяки онлайн-заняттям учні, які мають фізичні чи ментальні обмеження, можуть отримувати якісну освітню підтримку, не виходячи з дому. Це допомагає подолати бар'єри, пов'язані з доступністю та інклюзивністю освіти [17].

Таким чином, у 2024 році сайти пошуку репетиторів є невід'ємною частиною сучасної освітньої екосистеми. Вони надають учням доступ до персоналізованого навчання, розширюють можливості для міжнародного спілкування та культурного обміну, забезпечують репетиторів роботою незалежно від їхнього місцезнаходження та сприяють інклюзивності освіти. Ці платформи відіграють ключову роль у демократизації освіти, надаючи кожному можливість отримати якісну підтримку в навчанні та розвитку своїх здібностей. У світі, де технології стрімко розвиваються, а глобалізація набирає обертів, сайти пошуку репетиторів є незамінним інструментом для учнів, репетиторів та освітньої спільноти в цілому.

1.4 Аналіз сайту "preply.com"

Preply - це онлайн-платформа (рис. 1.3), яка з'єднує учнів з репетиторами для індивідуальних занять з різних предметів та мов. Сайт був заснований у 2012 році в Україні та швидко став одним з лідерів на ринку онлайн-репетиторства.

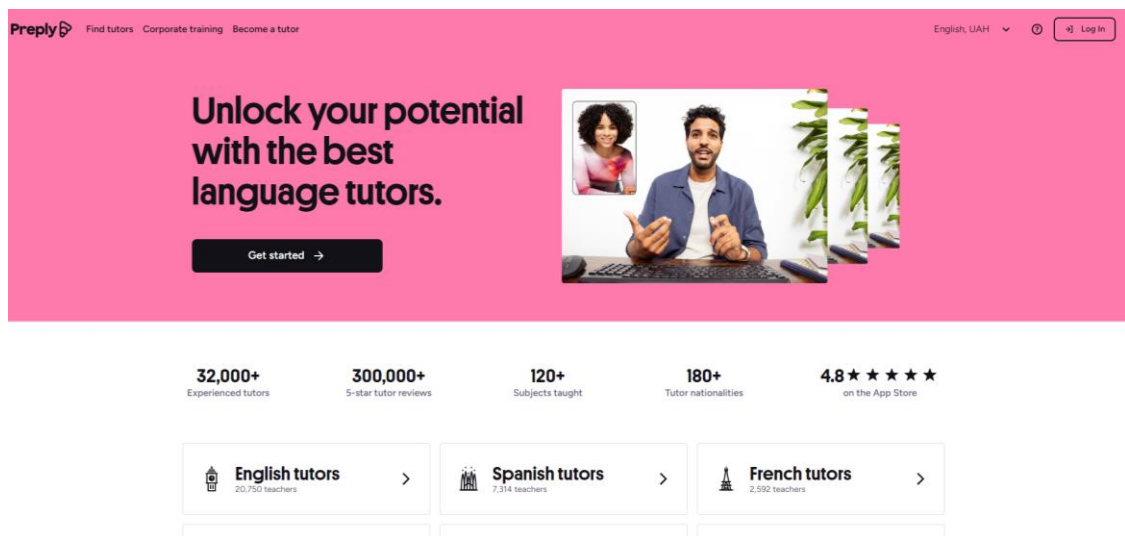


Рисунок 1.3 - Головна сторінка preply.com [18]

Однією з головних переваг Preply є великий вибір репетиторів з усього світу. На платформі зареєстровано понад 140 000 викладачів, які пропонують заняття з більш ніж 50 мов та різноманітних предметів, таких як математика, фізика, програмування тощо. Учні можуть легко знайти репетитора, який відповідає їхнім потребам та цілям, використовуючи зручні фільтри пошуку за мовою, ціною, доступністю та рейтингом (рис. 1.4).

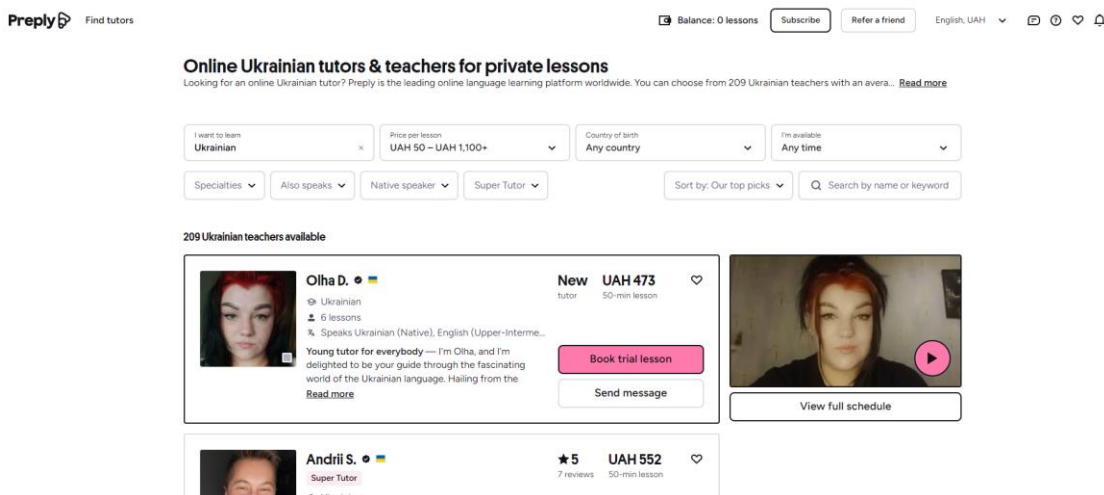


Рисунок 1.4 - Головна сторінка пошуку репетиторів на preply.com [19]

Іншою перевагою Preply є гнучкість та зручність. Учні можуть обирати зручний для них час занять та займатися з репетитором з будь-якої точки світу. Платформа пропонує зручні інструменти для онлайн-занять, такі як відеодзвінки, інтерактивна дошка та можливість обміну файлами. Це дозволяє учням отримувати якісні репетиторські послуги, не виходячи з дому.

Серед переваг Preply варто відзначити:

- наявність репетиторів зі всього світу, оплату за послуги в різних валютах (долари, євро, фунти тощо);
- скриті персональні дані (номери телефонів, банківських карт) та безпечна система оплати;
- можливість для учнів купувати по 1 уроку або підписатися на щомісячний курс з 4, 8 або 12 занять;
- швидку та лояльну службу підтримки, яка доступна цілодобово;
- видалення профілю репетитора або тимчасового його приховування від учнів;
- зручну аналітику досягнень репетитора у вигляді графіків;
- можливість хвалити учнів через спеціальні смайлики після кожного уроку та залишати відгуки про проведене заняття (якість зв'язку, поведінка учня тощо).

Серед недоліків Preply можна виділити:

- неможливість вказати в профілі репетитора, що він викладає кілька предметів;
- досить тривалий процес внесення змін до профілю з перевіркою від адміністраторів (хоча це можна вважати і перевагою з точки зору контролю якості);
- комісії, які репетитори повинні сплачувати. Для початку це 33% за кожен проведений урок, але для українців комісію скасували. Також репетитор сплачує комісію сайту при виведенні коштів на міжнародні гаманці (Payoneer, Preply тощо), а при виведенні грошей з цих гаманців на банківську картку країни, де проживає репетитор, конвертація валют може бути не вигідною. Тому всі ці нюанси слід враховувати репетитору при формуванні ціни на сайті. Крім того, учень також сплачує комісію, коли переказує гроші на внутрішній рахунок Preply;
- можливість викладання лише онлайн, немає опції для очних занять. У профілі репетитора не вказується його місто або адреса проживання;
- заборону в чаті учня та репетитора вказувати особисті дані (номери телефонів, електронні адреси тощо), сайт відразу блокує репетитора або учня за порушення правил. Тому дізнатися контактні дані, наприклад, Telegram, можна лише на пробному занятті.

Історія створення Preply почалася у 2012 році, коли троє українських підприємців - Кирило Бігай, Дмитро Волошин та Сергій Лук'янов - вирішили створити платформу для онлайн-навчання. Спочатку проєкт фокусувався на підготовці до стандартизованих тестів, таких як TOEFL та IELTS, але згодом розширився до викладання мов та інших предметів.

У 2015 році Preply залучив seed-інвестиції в розмірі \$1,3 млн від європейських та американських інвесторів. Це дозволило компанії розширити команду, вдосконалити платформу та вийти на нові ринки. У 2018 році Preply

залучив ще \$4 млн інвестицій, а в 2020 році - \$10 млн, що допомогло платформі зміцнити свої позиції на ринку онлайн-освіти [20].

Сьогодні Preply є однією з провідних платформ для онлайн-репетиторства, яка допомагає учням з усього світу знаходити кваліфікованих викладачів та досягати своїх освітніх цілей. Компанія продовжує розвиватися та вдосконалювати свої послуги, реагуючи на потреби та відгуки користувачів.

Незважаючи на деякі недоліки, Preply залишається популярним вибором серед учнів та репетиторів завдяки широкому вибору викладачів, зручному функціоналу, безпечній системі оплати та лояльній підтримці користувачів. Платформа надає репетиторам можливість будувати успішну кар'єру та працювати з учнями з різних країн світу, а учням - знаходити кваліфікованих викладачів та займатися в зручний для них час.

1.5 Аналіз сайту "buki.com.ua"

Buki.com.ua - це один з провідних українських сайтів для пошуку репетиторів (рис. 1.5), який допомагає учням та викладачам швидко знаходити один одного. Сервіс пропонує зручний та безкоштовний ресурс для пошуку репетиторів з різних предметів та напрямків по всій Україні.

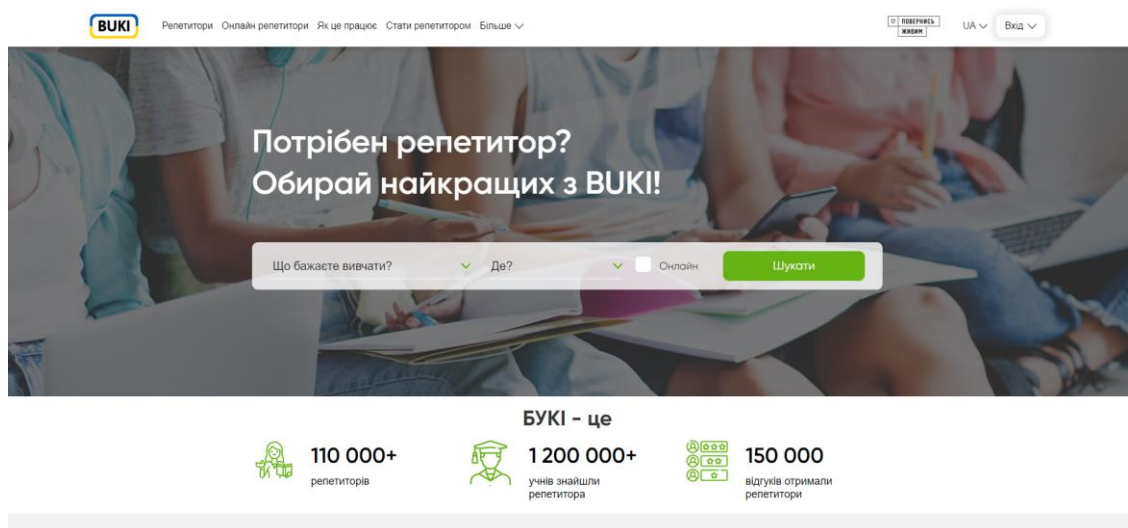


Рисунок 1.5 - Головна сторінка buki.com.ua [21]

Однією з головних переваг Buki є велика база репетиторів - понад 110 000 зареєстрованих викладачів з більш ніж 100 предметів у понад 250 містах України. Сайт охоплює як великі міста, такі як Київ, Харків, Львів, Дніпро та Одеса, так і менші населені пункти. Це дозволяє учням знайти репетитора практично з будь-якої дисципліни та місцезнаходження (рис. 1.6).

Іншою перевагою Buki є наявність рейтингу репетиторів на основі відгуків реальних учнів. Кожен викладач надає інформацію про свої навички, освіту, досвід роботи та вартість послуг, а компанія публікує відгуки клієнтів, за якими формується рейтинг репетитора. Це допомагає учням обрати найбільш підходящого та перевіреного викладача.

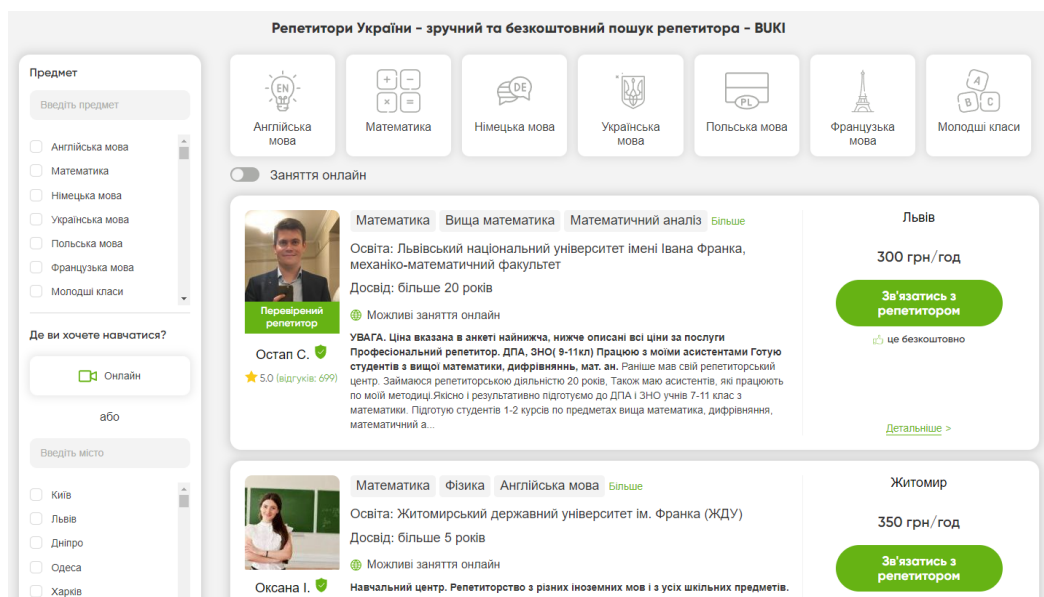


Рисунок 1.6 - Головна сторінка пошуку репетиторів buki.com.ua [22]

Сервіс пропонує зручний та швидкий пошук репетиторів за різними критеріями, такими як предмет, рівень знань, місто, район, ціна та формат занять (очно або онлайн). Крім того, на сайті є можливість отримати допомогу від адміністраторів у підборі викладача, що є безкоштовною послугою.

Серед переваг Buki варто відзначити:

- можливість викладання як онлайн, так і очно. У профілі можна вказати місто та район, де репетитор проводить заняття (у учня або у себе вдома);
- можливість вказати в профілі, що репетитор викладає кілька предметів;
- відсутність необхідності сплачувати комісію за кожен проведений урок з учнем. За кожного приведенного платформою учня репетитор одноразово вносить оплату (наприклад, якщо планується проводити 2 уроки на тиждень, платформі потрібно сплатити $2 \cdot (\text{ціна уроку})$ грн).

Серед недоліків Buki можна виділити:

- відсутність можливості зв'язатися з учнем через відеодзвінок або чат на платформі, тому не має гарантій, що після внесення грошей репетитором за учня, останній буде «постійним клієнтом»;
- відсутність внутрішнього рахунку та безпечної оплати між учнем та репетитором, лише за особистою домовленістю поза сайтом Buki;
- репетиторів лише з України та оплату лише в гривнях;
- неможливість видалити свій профіль репетитора, його можна лише приховати на тиждень, місяць, кілька місяців або рік. Якщо не заходити на сайт, профіль репетитора знову може стати видимим без його згоди та відома.

Попри ці недоліки, Buki залишається популярним вибором серед учнів та їхніх батьків завдяки широкому вибору репетиторів, зручному пошуку, рейтинговій системі та безкоштовним послугам. Додатковими причинами вибору сайту є корисні розділи з навчальними матеріалами, новинами освіти та безкоштовними відеоуроками від кращих викладачів.

Компанія Buki була заснована у 2014 році з метою допомогти орієнтуватися кожному в освітніх послугах України. За роки свого існування сервіс зібрав базу з понад 110 000 репетиторів, які надали послуги більш ніж 1,2 мільйонам учнів. Сайт постійно розвивається, додаючи нові функції та

покращуючи користувацький досвід. У 2021 році Buki запустив власну онлайн-школу Buki School з сертифікованими викладачами, яка пропонує індивідуальні заняття на інтерактивній платформі.

Таким чином, Buki.com.ua є одним з лідерів на ринку репетиторських послуг в Україні, пропонуючи учням зручний, швидкий та безкоштовний спосіб знайти кваліфікованого викладача з будь-якого предмету та локації. Попри деякі недоліки, сервіс продовжує розвиватися та залишається популярним вибором серед користувачів завдяки широким можливостям та корисним функціям.

1.6 Аналіз сайту "superprof.com.ua"

Superprof - це міжнародна онлайн-платформа (рис. 1.7), яка з'єднує учнів з репетиторами з різних предметів та дисциплін. Сайт був заснований у 2013 році у Франції та швидко поширився на інші країни, включаючи Україну. Наразі платформа налічує понад 8 мільйонів зареєстрованих користувачів та 180 000 репетиторів у 28 країнах світу.

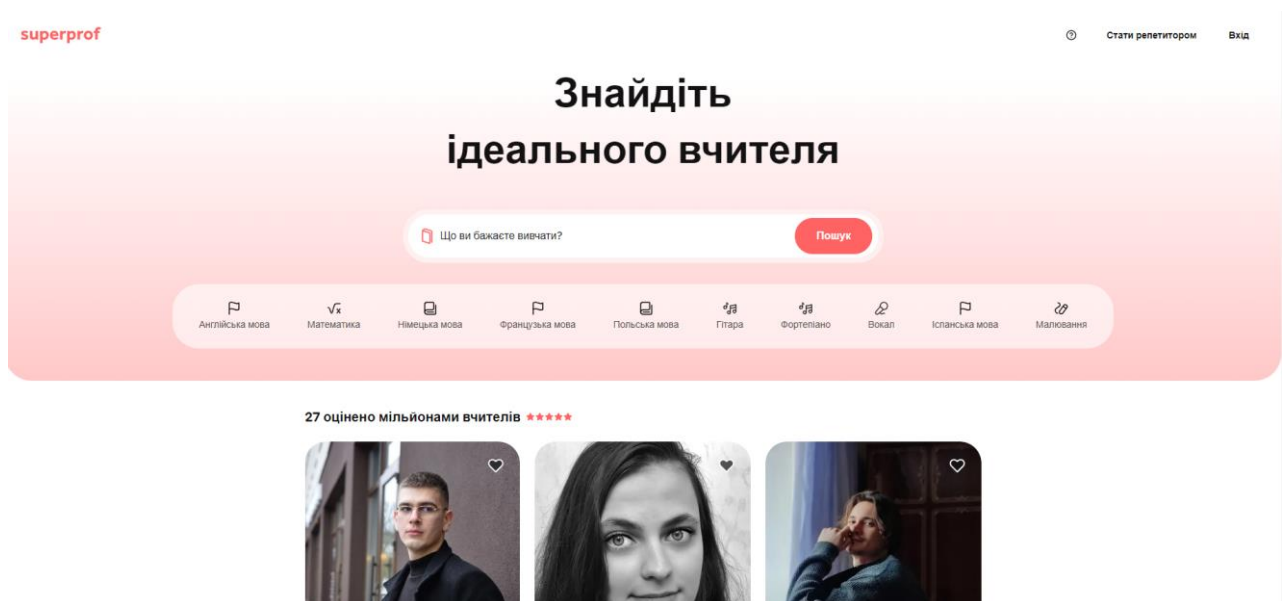


Рисунок 1.7 - Головна сторінка сайту superprof.com.ua [23]

Однією з головних переваг Superprof є величезний вибір репетиторів з різноманітних галузей знань. На платформі представлені викладачі з понад 1000 предметів, починаючи від академічних дисциплін, таких як математика, фізика, історія, і закінчуючи творчими напрямками, спортом та хобі. Учні можуть знайти репетитора практично з будь-якої теми, яка їх цікавить.

Іншою перевагою Superprof є зручний та інтуїтивно зрозумілий інтерфейс сайту (рис. 1.8). Користувачі можуть легко знайти потрібного репетитора за допомогою системи фільтрів, яка дозволяє сортувати викладачів за предметом, локацією, ціною, рівнем знань та іншими критеріями. Кожен репетитор має детальний профіль з інформацією про освіту, досвід, методику викладання, ціни та відгуки від попередніх учнів.

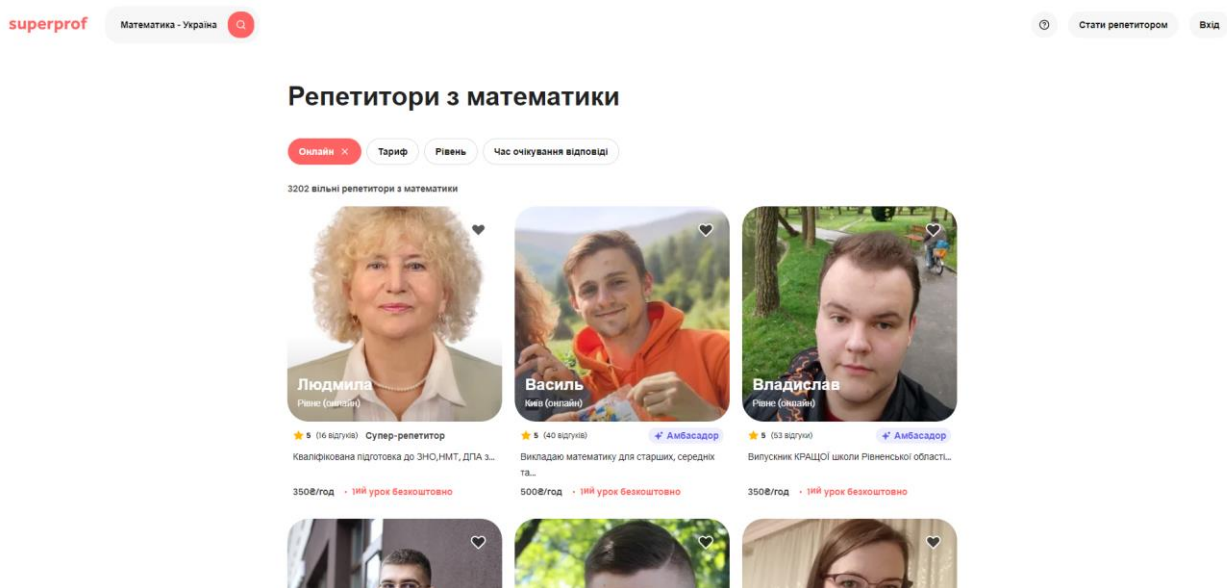


Рисунок 1.8 - Головна сторінка пошуку репетиторів superprof.com.ua

Серед переваг Superprof варто відзначити:

- можливість знайти репетитора з будь-якої точки світу, а не лише з України;
- найбільша різноманітність предметів та напрямків (понад 1000), включаючи не тільки академічні дисципліни, але й творчість, спорт, хобі тощо;

- зручний та інтуїтивно зрозумілий інтерфейс з детальними фільтрами пошуку;
- безкоштовна реєстрація для учнів;
- наявність рейтингів та відгуків про репетиторів від попередніх учнів;
- корисні статті та поради щодо навчання та саморозвитку на сайті.

Серед недоліків Superprof можна виділити:

- відсутність верифікації дипломів та сертифікатів репетиторів.
- Хоча викладачі можуть завантажити свої документи про освіту на сайт, платформа не перевіряє їх достовірність;
- відсутність вбудованої системи онлайн-занять та оплати на сайті.
- Учні та репетитори повинні домовлятися про формат занять (очно або онлайн) та спосіб оплати самостійно, використовуючи сторонні сервіси та платформи;
- щомісячна абонплата для репетиторів замість комісії з кожного уроку, що може бути менш вигідно для викладачів з невеликою кількістю учнів.

Історія створення Superprof почалася у 2013 році, коли двоє французьких підприємців, Віллас Вігор та Йонас Бруньєр, вирішили створити платформу, яка б полегшила пошук репетиторів для учнів. Спочатку сервіс працював лише у Франції, але швидко набув популярності та почав розширюватися на інші країни Європи [24].

У 2015 році Superprof залучив інвестиції в розмірі 3 мільйонів євро, що дозволило компанії вийти на нові ринки та вдосконалити платформу. У 2018 році сервіс почав працювати в Україні, а в 2019 році залучив ще 10 мільйонів євро інвестицій для подальшого зростання та розвитку.

Сьогодні Superprof є одним з лідерів на ринку онлайн-репетиторства, пропонуючи учням з усього світу зручний та ефективний спосіб знайти кваліфікованого викладача з будь-якої дисципліни. Компанія продовжує

розширюватися та покращувати свої послуги, реагуючи на потреби та відгуки користувачів.

Попри деякі недоліки, Superprof залишається популярним вибором серед учнів та репетиторів завдяки найбільшій різноманітності предметів, зручному інтерфейсу, можливості знайти викладача з будь-якої точки світу та корисним освітнім матеріалам на сайті.

1.7 Порівняльний аналіз сайтів-аналогів

Проаналізувавши три популярні сайти для пошуку репетиторів - Preply, Buki та Superprof, можна виявити як спільні риси, так і відмінності між ними. Кожна з цих платформ має свої переваги та недоліки, а також досягла певних успіхів у залученні клієнтів та виході на міжнародний рівень.

Аналіз сайтів-аналогів Preply, Buki та Superprof дав розуміння спільних рис, що характерні для платформ з пошуку репетиторів:

- велика база репетиторів з різних предметів та дисциплін - кожен із сайтів пропонує учням широкий вибір викладачів, охоплюючи академічні предмети, мови, творчі напрямки, спорт та хобі;
- зручний пошук репетиторів за допомогою фільтрів та критеріїв. Усі три платформи мають систему фільтрації, яка дозволяє учням знаходити викладачів за предметом, локацією, ціною, рівнем знань та іншими параметрами;
- детальні профілі репетиторів з інформацією про освіту, досвід, методику викладання та ціни. Кожен сайт надає учням можливість ознайомитися з кваліфікацією та особливостями роботи репетитора перед початком занять;
- рейтинги та відгуки від попередніх учнів. Усі платформи мають систему оцінювання репетиторів на основі відгуків реальних клієнтів, що допомагає новим учням обрати перевіреного та надійного викладача;

- безкоштовна реєстрація для учнів. Кожен із сайтів дозволяє учням створювати особисті кабінети та шукати репетиторів безкоштовно, стягуючи комісію лише з викладачів за надані послуги;

- наявність корисних матеріалів та порад щодо навчання. Усі три платформи публікують статті, відеоуроки та інші освітні ресурси, які допомагають учням покращувати свої знання та навички.

З іншого боку, між трьома платформами існує також велика кількість відмінних рис (табл. 1.1).

Таблиця 1.1 - Відмінні риси сайтів-аналогів

Характеристика	Preply	Buki	Superprof
Географічне охоплення	Міжнародне	Україна	Міжнародне
Мовна підтримка	50+ мов	Укр, рос, англ	10+ мов
Спеціалізація на предметах	Акцент на мовах	Рівномірний розподіл	Найбільша різноманітність
Форма занять	Лише онлайн	Онлайн та офлайн	Онлайн та офлайн
Верифікація репетиторів	Багаторівнева	Відгуки та рейтинги	Відгуки та рейтинги
Вбудована система занять	Так	Ні	Ні
Вбудована система оплати	Так	Ні	Ні
Внутрішній чат/відеозв'язок	Так	Ні	Ні
Комісія для репетиторів	До 33% (0% для українців)	Немає	Щомісячна абонплата
Можливість видалення профілю репетитором	Так	Лише приховати на певний термін	Ні

Незважаючи на спільні риси, сайти-аналоги Preply, Buki та Superprof мають низку відмінностей, що виділяють їх серед конкурентів та дозволяють задовольняти потреби різних сегментів цільової аудиторії:

а) географічне охоплення та спеціалізація на предметах:

1) Preply та Superprof є міжнародними платформами, тоді як Buki орієнтується переважно на український ринок;

2) Preply робить значний акцент на вивченні іноземних мов, Buki має більш рівномірний розподіл між різними предметами, а Superprof пропонує найбільшу різноманітність дисциплін, включаючи творчість, спорт, хобі тощо;

б) формат занять та функціонал платформи:

1) Preply пропонує лише онлайн-заняття, тоді як Buki та Superprof дозволяють викладачам проводити як онлайн, так і офлайн уроки;

2) Preply має вбудовану систему онлайн-занять, оплати та внутрішній чат/відеозв'язок, чого немає на Buki та Superprof. Останні покладаються на сторонні сервіси для комунікації та оплати;

в) ціноутворення та комісії:

1) Preply стягує з репетиторів комісію до 33% з кожного проведеного уроку (0% для українських викладачів), що може впливати на кінцеву вартість занять для учнів;

2) Buki не стягує комісії з кожного уроку, натомість репетитор робить одноразову оплату за кожного приведенного платформою учня;

3) Superprof використовує модель щомісячної абонплати для репетиторів, незалежно від кількості учнів та проведених занять;

г) управління профілем репетитора:

1) на Preply викладачі можуть повністю видалити свій профіль за потреби;

2) на Buki репетитори можуть лише приховати профіль на певний термін (тиждень, місяць, рік), але не видалити його повністю;

3) щодо Superprof, інформація про можливість видалення профілю викладача відсутня.

Кожна з платформ має свої сильні та слабкі сторони, що визначають їх позиціонування на ринку та привабливість для різних груп користувачів. Preply робить акцент на міжнародній аудиторії та вивченні мов, пропонуючи зручний вбудований функціонал. Vuki орієнтується на локальний ринок та різноманітність предметів, надаючи гнучкі умови співпраці для репетиторів. Superprof вирізняється найширшим вибором дисциплін та простотою користування, але має менше вбудованих інструментів.

Вибір тієї чи іншої платформи залежить від індивідуальних потреб та пріоритетів користувача - чи то учень, який шукає репетитора з конкретного предмету та локації, чи викладач, який прагне залучити більше клієнтів та отримати вигідні умови співпраці. Розуміння переваг, недоліків та особливостей кожного сайту дозволяє зробити усвідомлений вибір та максимально ефективно використовувати можливості онлайн-платформ для пошуку репетиторів.

1.8 Успіхи та виклики сайтів аналогів

Кожна з проаналізованих платформ - Preply, Vuki та Superprof - досягла певних успіхів та зіткнулася з викликами у своєму розвитку та позиціонуванні на ринку онлайн-репетиторства.

Preply, заснований в Україні у 2012 році, зумів залучити значні інвестиції (понад \$15 млн) та вийти на міжнародний рівень, ставши одним з лідерів у сегменті вивчення іноземних мов. Платформа побудувала репутацію надійного сервісу з якісними викладачами та зручним функціоналом, що дозволило їй швидко зростати та розширювати свою присутність на нових ринках. Однак відносно висока комісія для репетиторів та фокус переважно на мовах можуть обмежувати потенціал зростання у інших сегментах.

Buki, будучи українським сервісом, досяг значних успіхів на внутрішньому ринку, зібравши базу з понад 110 000 репетиторів та надавши послуги більш ніж 1,2 мільйонам учнів. Безкоштовна модель та широке охоплення предметів є привабливими для користувачів, але відсутність додаткового функціоналу та монетизації може ускладнювати залучення інвестицій та розширення на міжнародному рівні.

Superprof, заснований у Франції у 2013 році, зумів залучити понад €13 млн інвестицій та вийти на ринки 28 країн світу, пропонуючи найбільшу різноманітність предметів та зручний інтерфейс. Платформа швидко зростає та розширює свою присутність, але модель монетизації через абонплату для репетиторів може бути менш привабливою порівняно з комісійними моделями конкурентів.

Порівнюючи ситуацію цих платформ на ринку, можна зробити висновок, що Preply є лідером за обсягом залучених інвестицій та міжнародним охопленням, особливо у сегменті вивчення мов. Buki має сильні позиції на українському ринку, але може зіткнутися з викликами при виході на міжнародний рівень. Superprof демонструє стрімке зростання та розширення, але поступається Preply за обсягом інвестицій та географічним охопленням.

Сфера онлайн-репетиторства активно розвивається, що підтверджується успіхом таких платформ як Preply, Buki та Superprof. Ці сервіси дозволяють учням знаходити кваліфікованих викладачів, а репетиторам - ефективно залучати нових клієнтів. Аналіз показав, що кожна з платформ має свої переваги та недоліки, орієнтуючись на різні сегменти ринку та функціональність.

Зокрема, Preply є лідером за обсягом залучених інвестицій та міжнародним охопленням, пропонуючи якісний вбудований функціонал для вивчення мов. У той же час, сервіс має певні обмеження, такі як висока комісія для репетиторів та фокус переважно на мовних предметах.

Враховуючи зростаючий попит на репетиторські послуги та потенціал онлайн-платформ, можна зробити висновок, що створення нового вебзастосунок для пошуку репетиторів є актуальним та перспективним завданням. Такий застосунок може скористатися перевагами успішних аналогів, запропонувавши покращений або унікальний функціонал, а також більш вигідні умови для викладачів та учнів.

Розробка нового вебзастосунок для пошуку репетиторів з використанням сучасних вебтехнологій та фреймворків дозволить:

- зручну та швидку реєстрацію та авторизацію користувачів (репетиторів та учнів);
- особистий кабінет репетитора з можливістю створення профілю, додавання предметів, розкладу занять та ведення статистики;
- систему пошуку та фільтрації репетиторів для учнів за різними критеріями (предмет, рейтинг, ціна тощо);
- функціонал обміну повідомленнями між репетиторами та учнями для узгодження деталей занять;
- можливість бронювання та оплати занять учнями, а також підтвердження та проведення занять репетиторами;
- простий у використанні та адаптивний користувацький інтерфейс, що забезпечуватиме позитивний досвід на різних пристроях.

Розроблений вебзастосунок повинен бути масштабованим, безпечним, зручним в адмініструванні та здатним ефективно обробляти запити та дані користувачів. Він має потенціал стати затребуваним інструментом на ринку онлайн-репетиторства, забезпечуючи користувачам якісний сервіс та розширюючи можливості для навчання та викладання.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Визначення вимог та функціональності системи

Для успішної розробки вебзастосунку для пошуку репетиторів необхідно чітко визначити вимоги та функціональність системи. Це дозволить забезпечити відповідність застосунку потребам користувачів та ефективно досягнення поставлених цілей [25]. Розглянемо основні функціональні вимоги до системи:

а) реєстрація та автентифікація користувачів:

1) користувачі повинні мати можливість зареєструватися в системі, вказавши свою роль (репетитор або учень), ім'я, електронну пошту та пароль;

б) особистий кабінет репетитора:

1) репетитори повинні мати можливість створювати та редагувати свій профіль, додаючи інформацію про освіту, досвід, предмети викладання, вартість послуг тощо;

2) система повинна надавати репетиторам інструменти для управління своїм розкладом та доступністю для занять;

в) пошук репетиторів:

1) учні повинні мати можливість здійснювати пошук репетиторів за різними критеріями, такими як предмет, рівень знань, місцезнаходження, ціновий діапазон тощо;

г) обмін повідомленнями:

1) користувачі повинні мати можливість обмінюватися повідомленнями в рамках системи для обговорення деталей занять, узгодження розкладу тощо;

д) бронювання та управління заняттями:

1) учні повинні мати можливість переглядати розклад репетитора та бронювати заняття на зручний для них час;

2) репетитори повинні мати можливість підтверджувати або відхиляти запити на бронювання, а також переглядати та управляти своїм розкладом занять;

е) адміністративна панель та модерація контенту:

1) адміністратори системи повинні мати доступ до панелі управління для моніторингу та модерації користувачів, профілів репетиторів, відгуків тощо;

2) система повинна надавати інструменти для блокування або видалення користувачів, які порушують правила користування платформою.

Окрім функціональних вимог, важливо також визначити нефункціональні вимоги до системи [26], такі як:

- безпека: забезпечення захисту особистих даних користувачів, безпечна автентифікація та авторизація, захист від поширених вебвразливостей (наприклад, SQL-ін'єкцій, міжсайтового скриптингу тощо);

- продуктивність: забезпечення швидкого та ефективного функціонування системи, оптимізація швидкодії, мінімізація часу відгуку сервера;

- масштабованість: можливість системи обробляти зростаючу кількість користувачів та даних без втрати продуктивності, здатність до горизонтального та вертикального масштабування;

- зручність використання: інтуїтивно зрозумілий та привабливий користувацький інтерфейс, простота навігації, адаптивність для різних пристроїв та розмірів екрану.

Для визначення пріоритетності та важливості кожної з функціональних вимог можна використати матрицю пріоритезації вимог, яка дозволяє оцінити кожен вимогу за двома критеріями: важливість для користувачів та складність реалізації (табл. 2.1).

Таблиця 2.1 - Матриця пріоритезації вимог

Функціональна вимога	Важливість для користувачів	Складність реалізації
Реєстрація та автентифікація користувачів	Висока	Середня
Особистий кабінет репетитора	Висока	Середня
Пошук репетиторів	Висока	Середня
Обмін повідомленнями	Середня	Середня
Бронювання та управління заняттями	Висока	Висока
Адміністративна панель та модерація контенту	Середня	Висока

З таблиці видно, що функції реєстрації та автентифікації, особистого кабінету репетитора, пошуку репетиторів та бронювання занять мають високу важливість для користувачів. При цьому реалізація бронювання та управління заняттями, а також адміністративної панелі та модерації контенту є більш складною порівняно з іншими функціями.

Така матриця допомагає визначити пріоритетність реалізації функцій та розподілити ресурси команди розробки відповідно до важливості та складності кожної вимоги.

Визначення чітких та детальних вимог до системи є ключовим етапом у процесі розробки вебзастосунку для пошуку репетиторів. Це дозволяє забезпечити відповідність застосунку потребам користувачів, ефективно планувати та реалізувати функціональність, а також закласти основу для успішного тестування та розгортання системи.

2.2 Вибір технологій та інструментів розробки

Вибір відповідних технологій та інструментів розробки є важливим рішенням при створенні вебзастосунку для пошуку репетиторів. Правильний вибір технологічного стеку дозволяє забезпечити ефективну розробку, масштабованість, продуктивність та безпеку системи.

Для розробки вебзастосунку було вирішено використовувати середовище розробки PyCharm, яке забезпечує зручний та потужний інструментарій для написання коду, відлагодження та управління проєктом [27].

Для розробки серверної частини вебзастосунку було обрано мову програмування Python. Python є потужною та гнучкою мовою програмування, яка має широку підтримку спільноти розробників та багату екосистему бібліотек та фреймворків (табл. 2.2) [28].

Таблиця 2.2 - Порівняння серверної частини мов програмування

Мова програмування	Переваги	Недоліки
Python	Проста у вивченні та читанні коду, багата екосистема бібліотек та фреймворків, динамічна типізація	Нижча швидкодія порівняно з компільованими мовами, глобальний інтерпретатор блокувань (GIL) в деяких версіях
Java	Портативність (працює на різних платформах), потужна екосистема та велика спільнота розробників, статична типізація	Багатослівність коду, вища складність мови порівняно з Python

Продовження таблиці 2.2

Мова програмування	Переваги	Недоліки
JavaScript (Node.js)	Синхронне програмування, одна мова для клієнтської частини та серверної частини, багата екосистема пакетів (npm)	Callback hell (складність коду при вкладених викликах), динамічна типізація (може призводити до помилок)
Ruby	Проста у читанні та написанні коду, багата екосистема бібліотек та фреймворків (Rails), конвенції коду (зменшує кількість рішень)	Нижча швидкодія порівняно з іншими мовами, менша поширеність порівняно з Python, Java, JavaScript
Go	Висока швидкодія та низьке споживання пам'яті, проста у вивченні та читанні коду, вбудована підтримка конкурентності (горутини та канали)	Менша екосистема порівняно з іншими мовами, відсутність деяких високорівневих функцій

Для розробки клієнтської частини вебзастосунку було вирішено використовувати шаблонізатор Jinja2 разом з HTML та CSS [29]. Jinja2 є потужним та гнучким шаблонізатором для Python, який дозволяє генерувати динамічний HTML-код на основі даних, отриманих з серверної частини. HTML використовується для структурування та розмітки контенту вебсторінок, а CSS застосовується для візуального оформлення та стилізації елементів інтерфейсу (табл. 2.3) [30].

Таблиця 2.3 - Порівняння клієнтської частини мов програмування

Мова програмування	Переваги	Недоліки
Jinja2 (генерація HTML)	Генерація стандартної мови розмітки вебсторінок, проста у вивченні та використанні	Обмежена функціональність (лише структура та зміст), відсутність динамічності та інтерактивності
Jinja2 (генерація CSS)	Генерація стандартної мови стилізації вебсторінок, відокремлення стилів від структури HTML	Складність підтримки при великих проєктах, відсутність логіки та динамічності
JavaScript	Додає інтерактивність та динамічність вебсторінкам, багата екосистема бібліотек та фреймворків	Потенційні проблеми безпеки при неправильному використанні, відмінності в реалізації між браузерами
TypeScript	Статична типізація (зменшує кількість помилок), покращена підтримка IDE та рефакторингу коду	Додатковий крок компіляції перед виконанням, потребує вивчення додаткового синтаксису та типів
Dart	Розроблений для веб та мобільних застосунків, можливість компіляції в JavaScript або в нативний код	Менша екосистема порівняно з JavaScript, потребує вивчення нової мови та інструментів

Для розгортання вебзастосунку було обрано використання хмарного вебсервера Amazon EC2 (Elastic Compute Cloud) [31]. Amazon EC2 надає віртуальні сервери в хмарі, які можна легко налаштувати, масштабувати та керувати ними. Це забезпечує надійність, доступність та гнучкість у розгортанні вебзастосунку (табл. 2.4) [32, 33].

Таблиця 2.4 - Порівняння вебсерверів

Вебсервер	Переваги	Недоліки
Amazon EC2	Широка підтримка та документація, багато модулів та розширень	Вище споживання ресурсів порівняно з деякими альтернативами, складність налаштування для високонавантажених проєктів
Nginx	Висока продуктивність та низьке споживання ресурсів, ефективне обслуговування статичних файлів	Менша кількість модулів та розширень порівняно з Apache, складніша конфігурація для динамічних застосунків
Microsoft IIS	Тісна інтеграція з платформою Windows та .NET, графічний інтерфейс для налаштування	Обмежена підтримка на не-Windows платформах, менша продуктивність порівняно з деякими альтернативами
Lighttpd	Легкий та швидкий вебсервер, ефективне обслуговування статичних файлів	Менша кількість функцій та розширень порівняно з іншими, менша популярність та підтримка спільноти
Caddy	Автоматичне налаштування HTTPS за замовчуванням, проста конфігурація через файл Caddyfile	Менша кількість функцій та розширень порівняно з іншими, менша популярність та екосистема порівняно з іншими

При виборі системи управління базами даних (СУБД) було вирішено використовувати PostgreSQL [34]. PostgreSQL є потужною та надійною реляційною СУБД з відкритим кодом, яка підтримує транзакції, індекси, збережені процедури та тригери. PostgreSQL має хорошу масштабованість, продуктивність та підтримує розширені функції, такі як повнотекстовий пошук та геопросторові дані (табл. 2.5) [35].

Таблиця 2.5 - Порівняння баз даних

База даних	Переваги	Недоліки
PostgreSQL	Потужна та надійна СУБД з відкритим кодом, підтримка розширених функцій (індекси, тригери, процедури), хороша масштабованість та продуктивність	Вища складність налаштування порівняно з деякими альтернативами, вищі вимоги до ресурсів порівняно з легшими БД
MySQL	Широка популярність та підтримка, простота використання та налаштування, велика кількість інструментів та ресурсів	Обмежена підтримка деяких розширених функцій, проблеми з продуктивністю при високих навантаженнях
Oracle Database	Потужна та надійна комерційна СУБД, багата функціональність та можливості, підтримка великих корпоративних систем	Висока вартість ліцензій та підтримки, складність налаштування та адміністрування
Microsoft SQL Server	Тісна інтеграція з платформою Windows та .NET, багатий набір інструментів та функцій	Обмежена підтримка на не-Windows платформах, вища вартість ліцензій порівняно з відкритими альтернативами
MongoDB	Високопродуктивна нереляційна (NoSQL) база даних, гнучка схема даних (відсутність фіксованої структури), масштабованість та розподіленість	Відсутність підтримки транзакцій та зв'язків між колекціями, обмежена підтримка складних запитів та агрегацій

Продовження таблиці 2.5

База даних	Переваги	Недоліки
Redis	Висока продуктивність як сховище ;ключ-значення в пам'яті, підтримка різних структур даних (рядки, списки, хеші), проста масштабованість через реплікацію та шардинг	Обмежена підтримка складних структур даних, відсутність підтримки складних запитів та транзакцій

Обраний технологічний стек та інструменти розробки дозволяють ефективно та швидко розробляти вебзастосунок для пошуку репетиторів, забезпечуючи необхідну функціональність, масштабованість та безпеку системи. Використання Python для серверної частини, Jinja2/HTML/CSS для клієнтської частини, Amazon EC2 для розгортання та PostgreSQL для бази даних забезпечує надійну та продуктивну основу для створення якісного вебзастосунку.

2.3 Проєктування архітектури системи

Проєктування архітектури системи є важливим етапом у процесі розробки вебзастосунку для пошуку репетиторів. Правильно спроектована архітектура забезпечує модульність, масштабованість та гнучкість системи, а також полегшує її розробку, тестування та підтримку [36].

Для розробки вебзастосунку було обрано використання архітектурного шаблону Model-View-Controller (MVC) [37]. MVC є широко використовуваним шаблоном проєктування для вебзастосунків, який розділяє систему на три основні компоненти:

Модель (Model): Відповідає за представлення даних та бізнес-логіку системи. Модель інкапсулює структуру даних, правила валідації та операції

над даними. У контексті Django, моделі визначаються за допомогою класів Python та відображаються на таблиці бази даних за допомогою об'єктно-реляційного відображення (ORM).

Представлення (View): Відповідає за обробку запитів користувача та генерацію відповідей. Представлення отримує дані від моделі, застосовує до них необхідну логіку та передає їх у відповідний шаблон для відображення. У Django представлення реалізуються у вигляді функцій або класів Python.

Контролер (Controller): Відповідає за координацію взаємодії між моделлю та представленням. Контролер отримує запити від користувача, викликає відповідні методи моделі для отримання або зміни даних, та передає дані у представлення для відображення. У Django роль контролера виконує URLconf (конфігурація URL-адрес), який зіставляє URL-адреси з відповідними представленнями.

Використання шаблону MVC у поєднанні з фреймворком Django дозволяє досягти чіткого розділення відповідальності між компонентами системи та забезпечує модульність і повторне використання коду.

Розглянемо основні компоненти архітектури вебзастосунку для пошуку репетиторів (рис. 2.1):

а) клієнтська частина:

- 1) відповідає за взаємодію з користувачем та відображення інтерфейсу;
- 2) реалізована з використанням HTML, CSS та JavaScript;
- 3) використовує фреймворк Bootstrap для створення адаптивного та привабливого дизайну;
- 4) здійснює комунікацію з серверною частиною за допомогою HTTP-запитів (GET, POST) та отримує дані у форматі JSON;

б) серверна частина:

- 1) побудована на основі фреймворку Django та мови програмування Python;

- 2) обробляє HTTP-запити від клієнтської частини та генерує відповіді;
 - 3) містить моделі даних (моделі Django), які відповідають сутностям предметної області (наприклад, Репетитор, Учень, Предмет тощо);
 - 4) реалізує бізнес-логіку системи, включаючи операції з базою даних, автентифікацію та авторизацію користувачів, валідацію даних тощо;
 - 5) використовує представлення (views) для обробки запитів та генерації відповідей;
- в) база даних:
- 1) зберігає структуровані дані вебзастосунку;
 - 2) використовує реляційну СУБД PostgreSQL;
 - 3) взаємодіє з серверною частиною за допомогою ORM Django, який забезпечує абстракцію бази даних та спрощує роботу з нею.

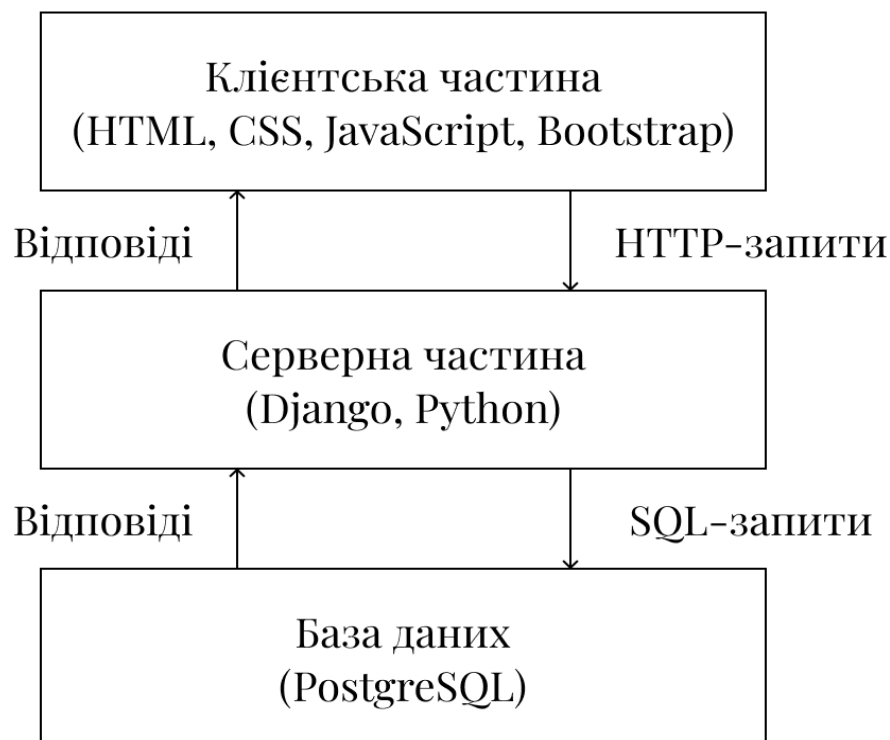


Рисунок 2.1 - Структурна схема архітектури вебзастосунку

Переваги обраної архітектури:

- розділення відповідальності: кожен компонент системи (модель, представлення, контролер) має чітко визначену роль та відповідальність, що полегшує розробку, тестування та підтримку коду;
- модульність: архітектура дозволяє розробляти окремі компоненти незалежно один від одного, що спрощує розширення та модифікацію системи;
- повторне використання коду: компоненти, такі як моделі та представлення, можуть бути повторно використані в різних частинах системи, що зменшує дублювання коду та підвищує ефективність розробки;
- масштабованість: розділення клієнтської та серверної частин дозволяє незалежно масштабувати кожен компонент залежно від навантаження та вимог до продуктивності;
- безпека: використання фреймворку Django забезпечує вбудовані механізми безпеки, такі як захист від розповсюджених вебвразливостей (наприклад, SQL-ін'єкцій, міжсайтового скриптингу) та аутентифікацію і авторизацію користувачів.

Обмеження та потенційні проблеми:

- монолітна архітектура: незважаючи на модульність, обрана архітектура все ще має ознаки монолітної структури, що може ускладнювати масштабування та розгортання окремих компонентів системи незалежно один від одного;
- складність розуміння для початківців: використання шаблону MVC та фреймворку Django може вимагати певного рівня знань та досвіду в веброботці, що може збільшити час на навчання та адаптацію для нових розробників у команді;
- обмеження вибору технологій: вибір конкретних технологій та інструментів (наприклад, Django, PostgreSQL) може обмежувати гнучкість у майбутньому, якщо виникне потреба в інтеграції з іншими системами або використанні альтернативних рішень.

Проектування архітектури системи з використанням шаблону MVC та фреймворку Django забезпечує надійну та ефективну основу для розробки вебзастосунку для пошуку репетиторів. Обрана архітектура дозволяє досягти модульності, масштабованості та безпеки системи, а також полегшує її розробку та підтримку. Однак, важливо враховувати потенційні обмеження та проблеми, пов'язані з монолітною структурою та вибором конкретних технологій, і бути готовим до адаптації та гнучкості у процесі розробки та еволюції системи.

2.4 Розробка структури бази даних

Розробка структури бази даних є фундаментальним етапом у процесі створення вебзастосунку для пошуку репетиторів. Правильно спроектована структура бази даних забезпечує ефективне зберігання, організацію та доступ до даних, а також підтримує цілісність та узгодженість інформації в системі [38].

Для проектування бази даних вебзастосунку було обрано використання реляційної моделі даних та ER-діаграм (Entity-Relationship Diagram). Реляційна модель даних базується на концепції відношень (таблиць) та зв'язків між ними, що дозволяє ефективно організовувати та керувати структурованими даними. ER-діаграми, в свою чергу, надають візуальне представлення сутностей, атрибутів та зв'язків між ними, полегшуючи процес проектування та комунікації структури бази даних.

Розглянемо основні сутності та їх атрибути в структурі бази даних вебзастосунку для пошуку репетиторів (рис. 2.2):

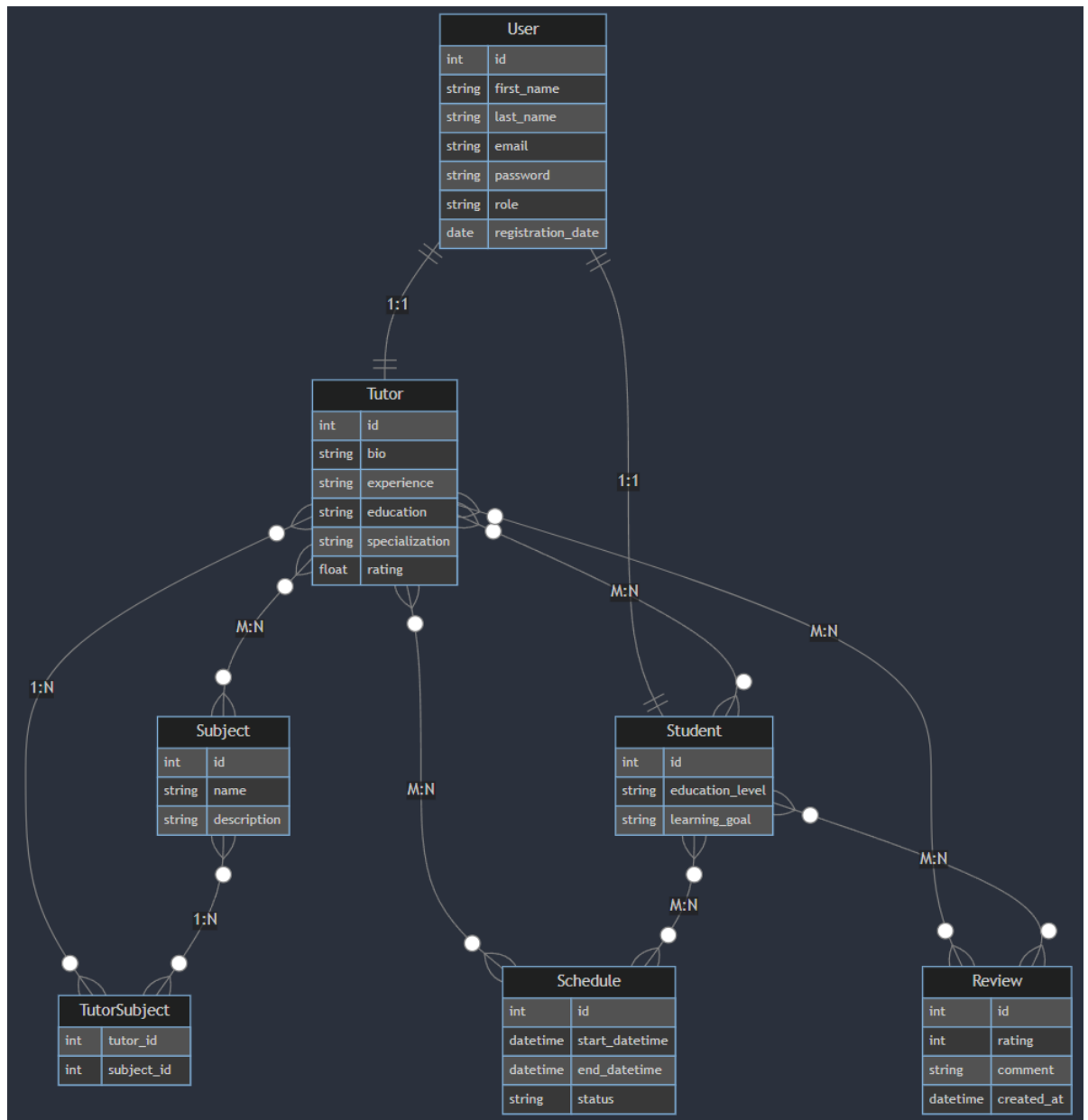


Рисунок 2.2 - ER-діаграма сутностей та атрибутів БД застосунку

а) сутність "Користувач" (User):

- 1) id (первинний ключ);
- 2) ім'я (first_name);
- 3) прізвище (last_name);
- 4) електронна пошта (email);
- 5) пароль (password);
- 6) роль (role) - репетитор або учень;
- 7) дата реєстрації (registration_date);

- б) сутність "Репетитор" (Tutor):
 - 1) id (первинний ключ, зовнішній ключ до таблиці User);
 - 2) біографія (bio);
 - 3) досвід (experience);
 - 4) освіта (education);
 - 5) спеціалізація (specialization);
 - 6) рейтинг (rating);
- в) сутність "Учень" (Student):
 - 1) id (первинний ключ, зовнішній ключ до таблиці User);
 - 2) рівень навчання (education_level);
 - 3) мета навчання (learning_goal);
- г) сутність "Предмет" (Subject):
 - 1) id (первинний ключ);
 - 2) назва (name);
 - 3) опис (description);
- д) сутність "Розклад" (Schedule):
 - 1) id (первинний ключ);
 - 2) репетитор_id (tutor_id, зовнішній ключ до таблиці Tutor);
 - 3) учень_id (student_id, зовнішній ключ до таблиці Student);
 - 4) предмет_id (subject_id, зовнішній ключ до таблиці Subject);
 - 5) дата і час початку (start_datetime);
 - 6) дата і час закінчення (end_datetime);
 - 7) статус (status) - заплановано, завершено, скасовано;
- е) сутність "Відгук" (Review):
 - 1) id (первинний ключ);
 - 2) репетитор_id (tutor_id, зовнішній ключ до таблиці Tutor);
 - 3) учень_id (student_id, зовнішній ключ до таблиці Student);
 - 4) рейтинг (rating);
 - 5) коментар (comment);

6) дата створення (created_at).

Зв'язки між сутностями:

- користувач (User) та репетитор (Tutor): один до одного (One-to-One). Кожен запис у таблиці Репетитор відповідає одному запису в таблиці Користувач;
- користувач (User) та учень (Student): один до одного (One-to-One). Кожен запис у таблиці Учень відповідає одному запису в таблиці Користувач;
- репетитор (Tutor) та предмет (Subject): багато до багатьох (Many-to-Many). Репетитор може викладати декілька предметів, і предмет може викладатися декількома репетиторами. Для реалізації цього зв'язку створюється додаткова таблиця TutorSubject з зовнішніми ключами до таблиць Репетитор та Предмет;
- репетитор (Tutor), учень (Student) та розклад (Schedule): Багато до багатьох (Many-to-Many). Репетитор може мати декілька занять з різними учнями, а учень може мати заняття з різними репетиторами. Таблиця Розклад виступає як асоціативна сутність, що зв'язує репетиторів, учнів та предмети;
- репетитор (Tutor), учень (Student) та Відгук (Review): Багато до багатьох (Many-to-Many). Учень може залишати відгуки для декількох репетиторів, а репетитор може отримувати відгуки від декількох учнів.

Переваги обраної структури бази даних:

- нормалізація даних: структура бази даних відповідає принципам нормалізації, що мінімізує дублювання даних та забезпечує їх цілісність та узгодженість;
- зрозумілість та легкість використання: структура бази даних є інтуїтивно зрозумілою та відображає реальні сутності та зв'язки предметної області, що полегшує роботу з даними та їх розуміння;
- ефективність запитів: правильно спроектовані зв'язки між таблицями дозволяють ефективно виконувати запити та отримувати необхідні дані з бази даних;

- розширюваність: структура бази даних є гнучкою та дозволяє легко додавати нові сутності та атрибути в майбутньому, якщо виникне така потреба.

Недоліки та потенційні проблеми:

- складність запитів зі зв'язками "багато до багатьох": Наявність декількох зв'язків "багато до багатьох" може ускладнювати написання запитів та знижувати продуктивність при виконанні складних операцій з даними;
- потенційна надлишковість даних: Незважаючи на нормалізацію, деякі атрибути можуть повторюватися в декількох таблицях, що може призвести до надлишковості даних та ускладнити їх синхронізацію;
- необхідність міграцій бази даних: При внесенні змін до структури бази даних (наприклад, додавання нових стовпців або таблиць) необхідно виконувати міграції даних, що може бути трудомістким та вимагати ретельного планування та тестування.

Розроблена структура бази даних для вебзастосунку з пошуку репетиторів забезпечує ефективну організацію та управління даними, відповідає вимогам предметної області та підтримує основні функціональні можливості системи. Використання реляційної моделі даних та ER-діаграм дозволяє чітко визначити сутності, атрибути та зв'язки між ними, що спрощує розробку та підтримку бази даних. Однак, важливо враховувати потенційні недоліки та проблеми, пов'язані зі складністю запитів та необхідністю міграцій даних, і бути готовим до оптимізації та адаптації структури бази даних в процесі розвитку та масштабування системи.

3 РЕАЛІЗАЦІЯ ОСНОВНИХ КОМПОНЕНТІВ СИСТЕМИ

3.1 Реалізація модуля реєстрації та автентифікації користувачів

Для реалізації функціоналу реєстрації та автентифікації користувачів у Django використовується вбудована система автентифікації. Вона надає базові можливості для створення облікових записів, входу в систему, виходу з неї та управління користувачам.

У файлі `main/views/login.py` реалізовано представлення для реєстрації та автентифікації користувачів (рис. 3.1). Розглянемо основні частини цього модуля.

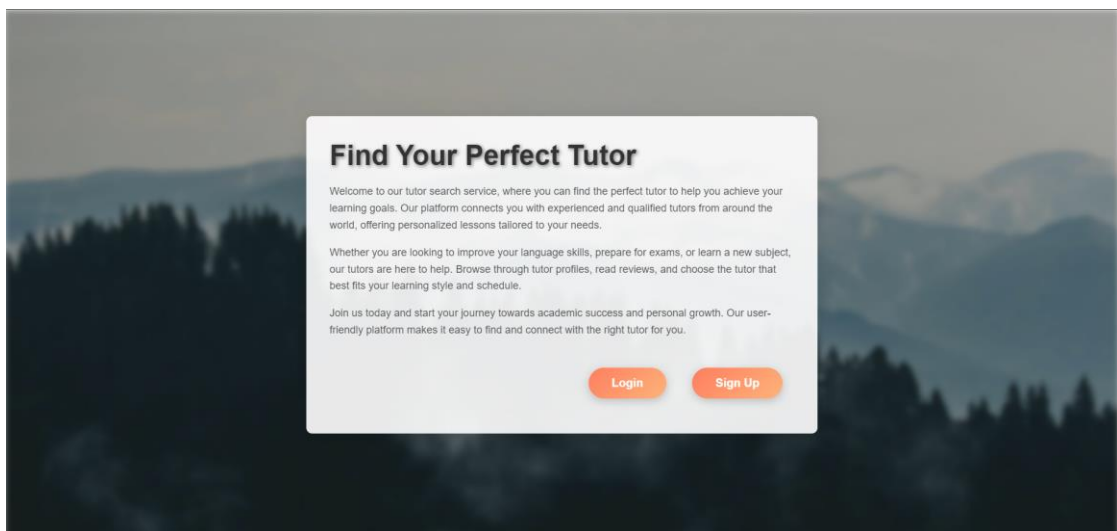


Рисунок 3.1 - Сторінка автентифікації

Представлення `register_view` - це представлення відповідає за відображення сторінки реєстрації користувача. Якщо запит надійшов методом GET, відображається шаблон `start_registration.html`, який містить форму вибору ролі користувача (репетитор або учень) (рис. 3.2).

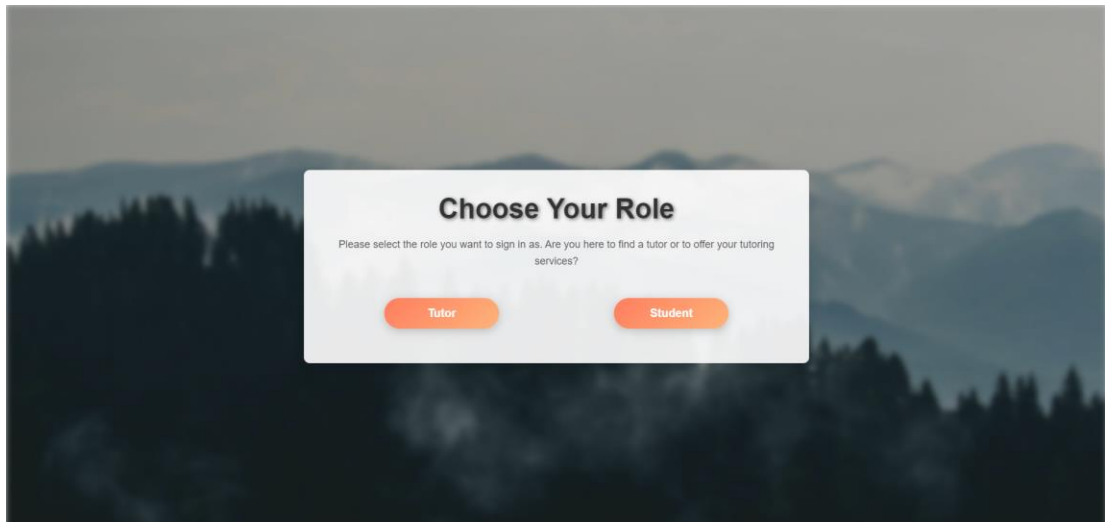
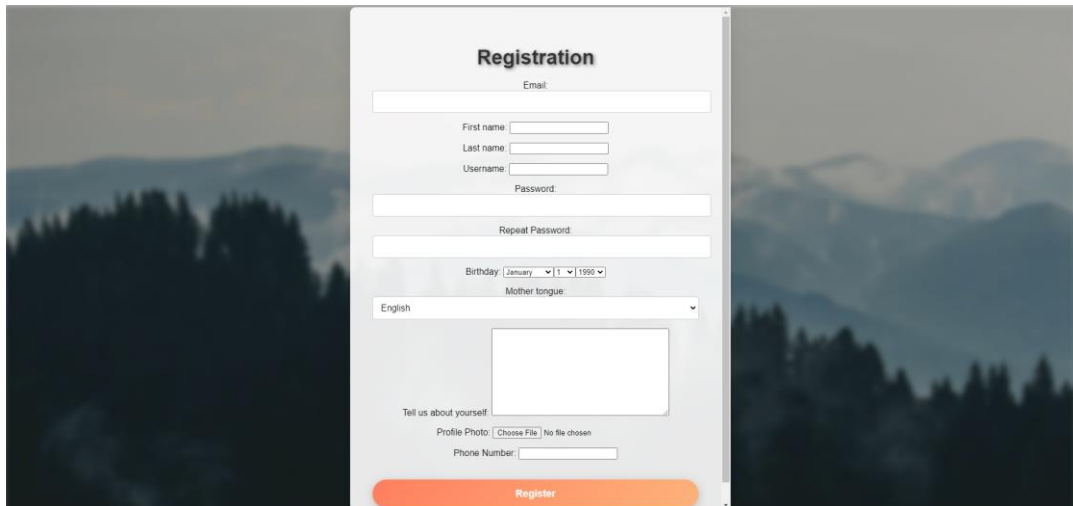


Рисунок 3.2 - Сторінка вибору ролі при реєстрації

Представлення ``tutor_registration`` - це представлення обробляє реєстрацію користувача в ролі репетитора наступним чином (рис. 3.3):

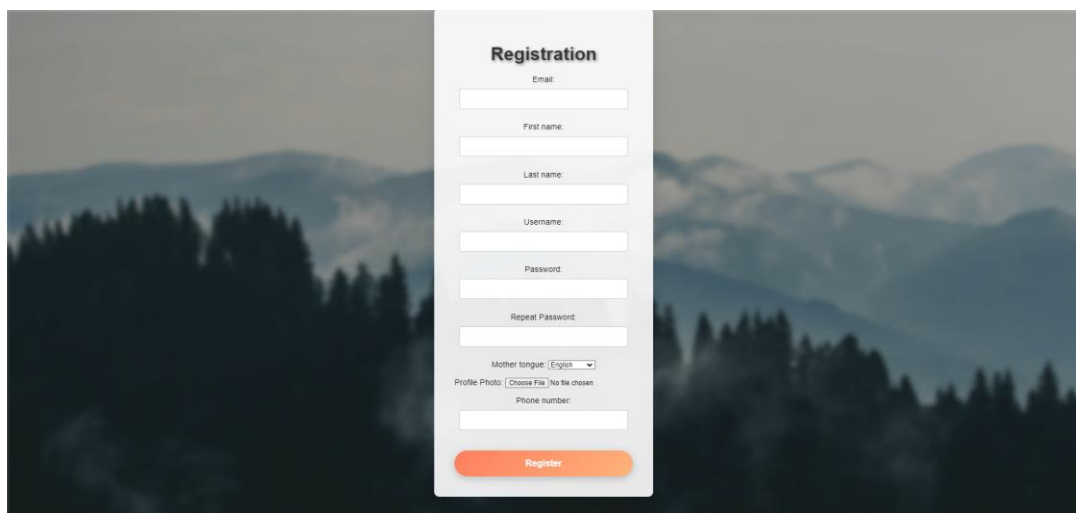
- якщо запит надійшов методом POST, створюється екземпляр класу ``RegisterUser`` з модуля ``commands.users`` та викликається метод ``execute`` з параметрами запиту та роллю ``UserGroup.TUTOR``;
- якщо дані форми валідні, створюється новий користувач, додається до групи "Teachers", зберігається профіль репетитора та виконується автентифікація користувача;
- після успішної реєстрації користувач перенаправляється на сторінку налаштувань профілю репетитора;
- якщо дані форми невалідні, відображається шаблон ``tutor_app/registration.html`` з формами реєстрації користувача та репетитора.



The image shows a web form titled "Registration" for a tutor. The form is centered on a background image of a mountain range. The fields include: Email, First name, Last name, Username, Password, Repeat Password, Birthday (with month and year dropdowns), Mother tongue (dropdown), a text area for "Tell us about yourself", a "Profile Photo" section with a "Choose File" button and "No file chosen" text, and a Phone Number field. An orange "Register" button is at the bottom.

Рисунок 3.3 - Сторінка форми реєстрації викладача

Представлення `'student_registration'` - це представлення обробляє реєстрацію користувача в ролі учня. Процес реєстрації аналогічний до `'tutor_registration'`, але використовується роль `'UserGroup.STUDENT'` та шаблон `'student_app/registration.html'` (рис. 3.4).



The image shows a web form titled "Registration" for a student. The form is centered on a background image of a mountain range. The fields include: Email, First name, Last name, Username, Password, Repeat Password, Mother tongue (dropdown), a "Profile Photo" section with a "Choose File" button and "No file chosen" text, and a Phone number field. An orange "Register" button is at the bottom.

Рисунок 3.4 - Сторінка форми реєстрації учня

Представлення `'login_view'` - це представлення відповідає за автентифікацію користувача у наступному порядку:

- якщо запит надійшов методом POST, створюється екземпляр форми `'LoginForm'` з даними запиту;

- якщо дані форми валідні, виконується автентифікація користувача за допомогою функції `'authenticate'` та `'login'`;
- після успішної автентифікації користувач перенаправляється на сторінку профілю;
- якщо дані форми невалідні або автентифікація не вдалася, відображається шаблон `'login.html'` з формою входу та повідомленням про помилку.

Представлення `'logout_view'` - це представлення відповідає за вихід користувача з системи у такому порядку:

- викликається функція `'logout'`, яка очищає сесію користувача та виконує вихід з системи;
- після виходу користувач перенаправляється на сторінку входу.

Форми реєстрації та автентифікації користувачів визначені у файлі `'main/forms/auth.py'`. Вони включають поля для введення імені, прізвища, електронної пошти, пароля та інших даних, необхідних для створення облікового запису та входу в систему.

Шаблони для реєстрації та автентифікації користувачів знаходяться в директоріях `'main/templates/tutor_app/'` та `'main/templates/student_app/'`. Вони містять HTML-розмітку та поля форм для введення даних користувачем.

Таким чином, модуль реєстрації та автентифікації користувачів у вебзастосунку для пошуку репетиторів реалізовано з використанням вбудованої системи автентифікації Django. Він забезпечує можливість створення облікових записів репетиторів і учнів, входу в систему та виходу з неї, а також базове управління користувачами. Цей модуль є фундаментальним компонентом системи, який забезпечує безпеку та розмежування доступу до функціоналу вебзастосунку.

3.2 Розробка особистого кабінету репетитора

Особистий кабінет репетитора є важливим компонентом вебзастосунку для пошуку репетиторів. Він надає репетиторам можливість керувати своїм профілем, переглядати та редагувати особисту інформацію, управляти розкладом занять та відстежувати статистику.

У файлі ``main/views/users.py`` реалізовано представлення для особистого кабінету репетитора (рис. 3.5).

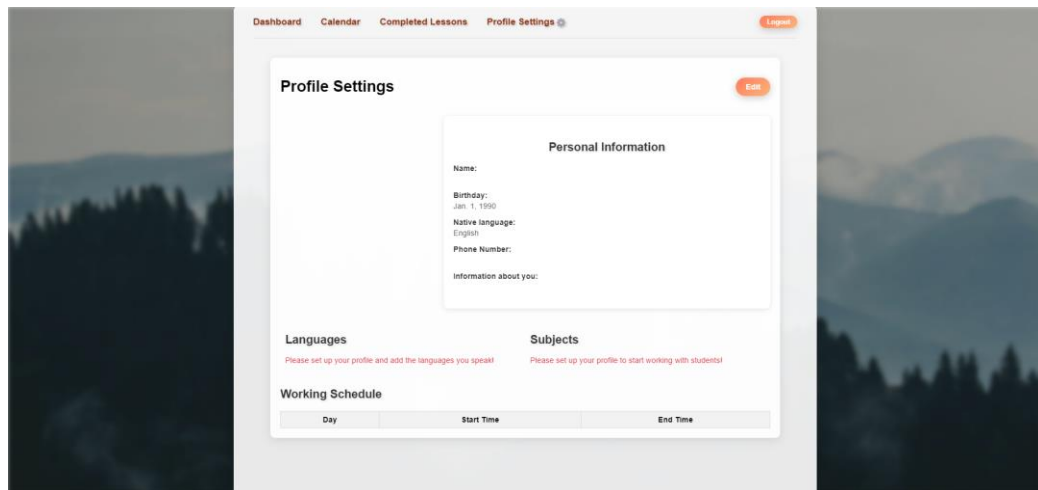


Рисунок 3.5 - Сторінка профілю репетитора

Розглянемо основні частини цього модуля:

Представлення ``settings`` - це представлення відповідає за відображення сторінки налаштувань профілю репетитора у наступному порядку (рис. 3.6):

- воно використовує запит ``GetProfileSettings`` з модуля ``queries.users`` для отримання даних профілю репетитора;
- запит ``execute`` викликається з параметром запиту, і повертається відповідний шаблон (``tutor_app/profile_settings.html``) з даними профілю репетитора.

Profile Settings Back

Personal Information

Mother tongue:
English ▼

About me info:
antraant

Profile Photo:
Choose File No file chosen

Phone Number:
21341234321421

Languages
Add Language

Subjects
Add Subject

Working Schedule
Add Schedule

Save Changes

Рисунок 3.6 - Форма налаштувань профілю репетитора

Представлення `'update_settings'` - це представлення відповідає за оновлення налаштувань профілю репетитора у наступному порядку:

- якщо запит надійшов методом POST, створюється екземпляр класу `'UpdateProfileSettings'` з модуля `'commands.users'` та викликається метод `'execute'` з параметрами запиту;
- команда `'UpdateProfileSettings'` обробляє оновлення профілю репетитора, включаючи збереження персональної інформації, мов викладання, предметів та розкладу;
- після успішного оновлення профілю, репетитор перенаправляється на сторінку налаштувань профілю;

- якщо запит надійшов методом GET, використовується запит `'GetUpdateProfileSettings'` для отримання даних форми оновлення профілю репетитора.

Представлення `'dashboard'` - це представлення відповідає за відображення головної сторінки особистого кабінету репетитора (дашборду) у наступному порядку (рис. 3.7):

- воно використовує запит `'GetUserDashboard'` з модуля `'queries.users'` для отримання даних для відображення на дашборді;
- запит `'execute'` викликається з параметром запиту, і повертається відповідний шаблон (`'tutor_app/dashboard.html'`) з даними для відображення.

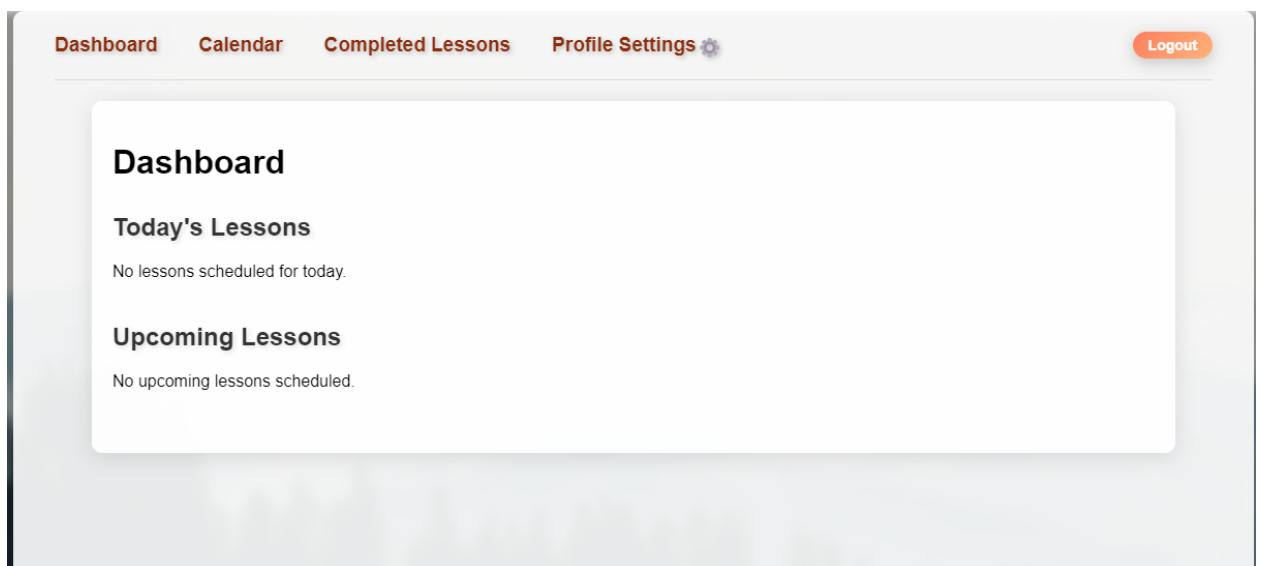


Рисунок 3.7 - Головна сторінка особистого кабінету репетитора

Шаблони для особистого кабінету репетитора знаходяться в директорії `'main/templates/tutor_app/'`. Вони містять HTML-розмітку та елементи інтерфейсу для відображення та взаємодії з даними репетитора у наступному порядку:

- `'profile_settings.html'`: шаблон для відображення налаштувань профілю репетитора;

- `'update_profile_settings.html'`: шаблон для форми оновлення налаштувань профілю репетитора;
- `'dashboard.html'`: шаблон для головної сторінки особистого кабінету репетитора (дашборду).

Для роботи з даними репетитора використовуються моделі `'Tutor'`, `'TutorLanguage'`, `'TutorSubject'` та `'WorkingSchedule'`, визначені у файлі `'main/models.py'`. Ці моделі представляють структуру даних репетитора та пов'язані з ними сутності, такі як мови викладання, предмети та розклад роботи.

Форми для редагування профілю репетитора та додавання мов, предметів і розкладу визначені у файлі `'main/forms/teachers.py'`. Вони включають поля для введення відповідних даних та валідацію введених значень.

Таким чином, розробка особистого кабінету репетитора у вебзастосунку для пошуку репетиторів забезпечує репетиторам можливість керувати своїм профілем, редагувати особисту інформацію, додавати мови викладання, предмети та розклад роботи. Це дозволяє репетиторам презентувати себе потенційним учням та ефективно організовувати свою викладацьку діяльність. Особистий кабінет репетитора є ключовим компонентом системи, який забезпечує зручність та функціональність для репетиторів.

3.3 Реалізація функціоналу пошуку репетиторів

Функціонал пошуку репетиторів є однією з ключових особливостей вебзастосунку, оскільки він дозволяє учням знаходити відповідних репетиторів на основі їхніх потреб та вимог.

У файлі `'main/views/students.py'` реалізовано представлення для пошуку репетиторів. Розглянемо основні частини цього модуля.

Представлення ``tutor_details`` - це представлення відповідає за відображення детальної інформації про репетитора у наступному порядку (рис. 3.8):

- воно приймає параметр ``tutor_id``, який вказує на ідентифікатор репетитора, що переглядається;
- отримання об'єкту репетитора за допомогою ``Tutor.objects.get(id=tutor_id)``;
- отримання предметів, які викладає репетитор, за допомогою ``TutorSubject.objects.filter(tutor=tutor)``;
- обчислення кількості років досвіду викладання для кожного предмета;
- отримання кількості завершених занять репетитора за допомогою ``LessonSchedule.objects.filter(tutor_subject__tutor=tutor, status='Completed').count()``;
- отримання запланованих занять репетитора за допомогою ``LessonSchedule.objects.filter(tutor_subject__tutor=tutor, scheduled_from__gte=datetime.now())``;
- отримання розкладу роботи репетитора за допомогою ``WorkingSchedule.objects.filter(tutor=tutor)``;
- обробка форми бронювання заняття (``BookingForm``), якщо запит надійшов методом POST;
- рендеринг шаблону ``student_app/tutor_details.html`` з передачею відповідних даних.

Name Surname

About Me: arstraarst
 Phone Number: 0502010253
 Native Language: English
 Subjects:
 English (100 \$/hr) - 14 years of experience
 Languages: English
 Completed Lessons: 0

Book a Lesson

Tutor subject:

<
>
today

Jun 2 – 8, 2024

day
week

	Sun 6/2	Mon 6/3	Tue 6/4	Wed 6/5	Thu 6/6	Fri 6/7	Sat 6/8
12am							
1am							
2am							
3am							
4am							
5am							
6am							
7am							
8am							
9am							
10am		Working Hours					
11am		Working Hours					
12pm		Working Hours					
1pm		Working Hours					

Book Lesson

Рисунок 3.8 - Сторінка відображення детальної інформації про репетитора

Представлення `student\_dashboard` - це представлення відповідає за відображення головної сторінки пошуку репетиторів для учнів у наступному порядку (рис. 3.9):

- створення екземпляру форми `TutorFilterForm` для фільтрації репетиторів;
- отримання списку репетиторів за допомогою `Tutor.objects.exclude(tutorsubject\_\_isnull=True)`;

- якщо форма фільтрації є валідною, виконується фільтрація репетиторів за обраними критеріями (предмет, мова, максимальна ціна);
- обчислення середнього рейтингу для кожного репетитора на основі відгуків учнів;
- рендеринг шаблону ``student_app/dashboard.html`` з передачею відповідних даних.

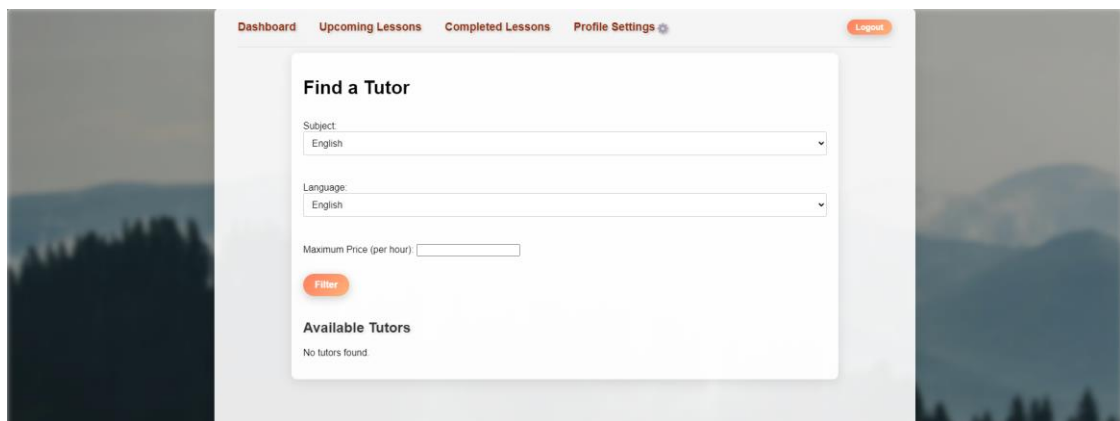


Рисунок 3.9 - Головна сторінка пошуку репетиторів

Шаблони для пошуку репетиторів знаходяться в директорії ``main/templates/student_app/``. Вони містять HTML-розмітку та елементи інтерфейсу для відображення результатів пошуку та деталей репетитора:

- ``tutor_details.html``: шаблон для відображення детальної інформації про репетитора, включаючи його особисті дані, предмети, розклад та форму бронювання заняття;
- ``dashboard.html``: шаблон для головної сторінки пошуку репетиторів, де учні можуть фільтрувати репетиторів за різними критеріями та переглядати список доступних репетиторів.

Форма фільтрації репетиторів (``TutorFilterForm``) визначена у файлі ``main/forms/students.py``. Вона містить поля для вибору предмета, мови та максимальної ціни за годину заняття (рис. 3.10).

Find a Tutor

Subject: English

Language: English

Maximum Price (per hour): 200

Filter

Available Tutors

Name Surname

Subjects:
English (100 \$/hr)

Languages:
English (C2)

Rating: 0 ★

View Details

Рисунок 3.10 - Відображення списку репетиторів відфільтрованих за певними критеріями

Таким чином, функціонал пошуку репетиторів у вебзастосунку реалізовано з використанням представлень `'tutor_details'` та `'student_dashboard'`, які взаємодіють з моделями `'Tutor'`, `'TutorSubject'`, `'LessonSchedule'` та `'WorkingSchedule'`. Учні мають можливість переглядати детальну інформацію про репетиторів, фільтрувати їх за різними критеріями та бронювати заняття. Це дозволяє учням знаходити відповідних репетиторів, які відповідають їхнім потребам та вимогам, і сприяє ефективній взаємодії між учнями та репетиторами.

3.4 Реалізація функціоналу бронювання та управління заняттями

Функціонал бронювання та управління заняттями є ключовим компонентом вебзастосунку для пошуку репетиторів, оскільки він дозволяє

учням бронювати заняття з репетиторами, а репетиторам - управляти своїм розкладом занять. Розглянемо реалізацію цього функціоналу.

Бронювання занять учнями (рис. 3.11):

- у файлі ``main/views/students.py`` реалізовано представлення ``tutor_details``, яке відповідає за відображення детальної інформації про репетитора та форми бронювання заняття;
- форма бронювання заняття (``BookingForm``) визначена у файлі ``main/forms/students.py``.

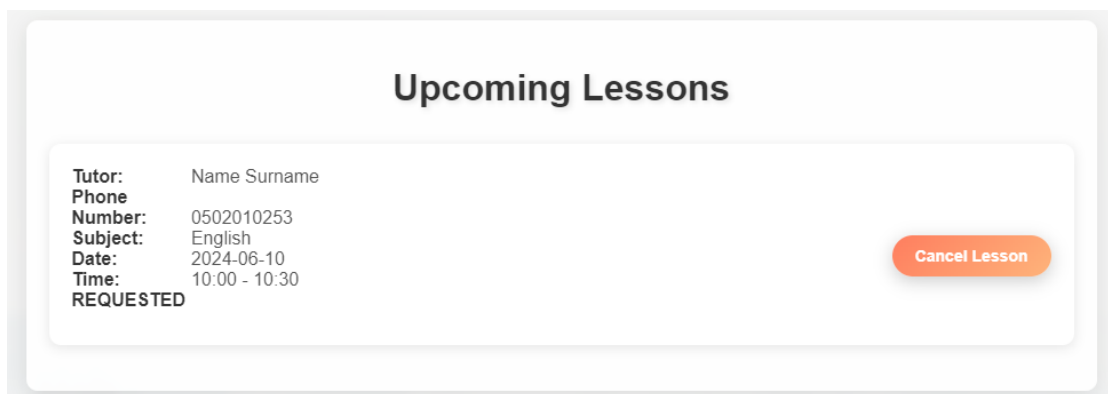


Рисунок 3.11 - Сторінка відображення списку майбутніх занять

При відправленні форми бронювання виконуються наступні дії:

- валідація даних форми, включаючи перевірку доступності вибраного часу та дати;
- створення нового об'єкту ``LessonSchedule`` з відповідними даними (репетитор, учень, предмет, дата та час);
- збереження об'єкту ``LessonSchedule`` зі статусом "Requested";
- перенаправлення на сторінку деталей репетитора.

Управління заняттями репетиторами:

- у файлі ``main/views/teachers.py`` реалізовано представлення для управління заняттями репетиторами;
- представлення ``calendar`` відповідає за відображення календаря занять репетитора;

- використовується бібліотека FullCalendar для візуалізації календаря;
- дані про робочі години та заплановані заняття передаються у форматі JSON для відображення в календарі;
- представлення ``complete_lesson``, ``plan_lesson`` та ``cancel_lesson`` відповідають за зміну статусу занять (завершено, заплановано, скасовано);
- ці представлення приймають параметр ``lesson_id``, який вказує на ідентифікатор заняття;
- виконується оновлення статусу заняття в базі даних за допомогою ``LessonSchedule.objects.filter(id=lesson_id).update(status=<новий_статус>)``;
- після оновлення статусу, репетитор перенаправляється на сторінку дашборду.

Відображення розкладу занять (рис. 3.12):

- у файлі ``main/templates/tutor_app/dashboard.html`` відображається список занять репетитора на поточний день та майбутні заняття;
- заняття групуються за статусом (сьогоднішні заняття, майбутні заняття);
- для кожного заняття відображається інформація про учня, предмет, дату, час та статус;
- репетитор має можливість завершити сьогоднішнє заняття, підтвердити або скасувати майбутнє заняття.

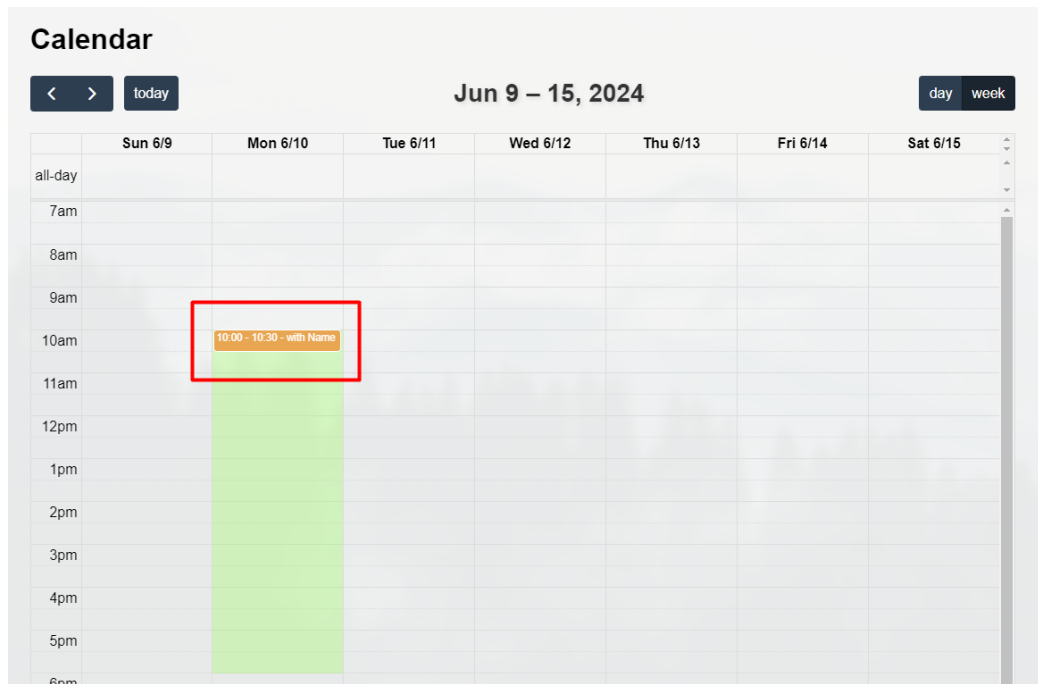


Рисунок 3.12 - Сторінка відображення розкладу репетитора, червоним виділено заплановане заняття

Оцінювання занять учнями:

- у файлі ``main/views/students.py`` реалізовано представлення ``rate_lesson``, яке відповідає за оцінювання заняття учнем;
- представлення приймає параметр ``lesson_id``, який вказує на ідентифікатор заняття;
- форма оцінювання заняття (``RatingForm``) визначена у файлі ``main/forms/students.py``;
- при відправленні форми оцінювання створюється новий об'єкт ``Rating`` з відповідними даними (заняття, оцінка, коментар) та зберігається в базі даних.

Історія занять:

- у файлах ``main/templates/tutor_app/completed_lessons.html`` та ``main/templates/student_app/completed_lessons.html`` відображається історія завершених занять для репетитора та учня відповідно;

- для репетитора відображається список завершених занять з інформацією про учня, предмет, дату, час та оцінку (якщо є);
- для учня відображається список завершених занять з інформацією про репетитора, предмет, дату, час, оцінку та коментар (якщо є).

Реалізація функціоналу бронювання та управління заняттями у вебзастосунку для пошуку репетиторів забезпечує зручність та ефективність взаємодії між учнями та репетиторами. Учні мають можливість легко забронювати заняття з обраним репетитором, а репетитори можуть управляти своїм розкладом занять, відстежувати статус занять та переглядати історію завершених занять. Це сприяє організованості та продуктивності процесу навчання.

3.5 Розробка адміністративної панелі та модерації контенту

Адміністративна панель та модерація контенту є важливими компонентами вебзастосунку для пошуку репетиторів, оскільки вони дозволяють адміністраторам керувати користувачами, контролювати якість контенту та забезпечувати дотримання правил користування платформою.

Адміністративна панель та модерація контенту реалізовані з використанням вбудованої адміністративної панелі Django. Розглянемо основні аспекти цього функціоналу (рис. 3.13).

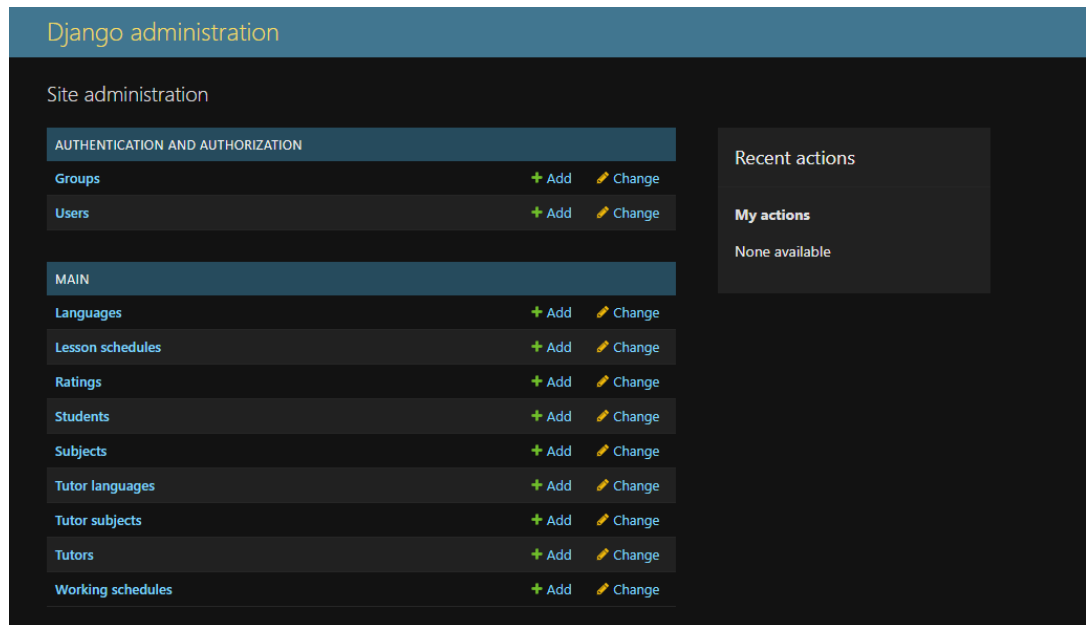


Рисунок 3.13 - Сторінка адміністративної панелі та модераторії Django

Налаштування адміністративної панелі:

- у файлі `main/admin.py` здійснюється реєстрація моделей для відображення в адміністративній панелі Django;
- за допомогою декоратора `@admin.register` або функції `admin.site.register()` моделі реєструються для адміністрування;
- для кожної моделі можна налаштувати відображення полів, фільтрацію, сортування та інші параметри в адміністративній панелі.

Керування користувачами (рис. 3.14):

- адміністратори мають можливість переглядати, створювати, редагувати та видаляти користувачів через адміністративну панель;
- у файлі `main/admin.py` можна налаштувати відображення моделей `'User'`, `'Tutor'` та `'Student'` в адміністративній панелі;
- адміністратори можуть призначати ролі користувачам (репетитор або учень), редагувати їхні профілі та керувати їхніми обліковими записами.

Рисунок 3.14 - Сторінка адміністрування користувачів

Модерація профілів репетиторів (рис. 3.15):

- адміністратори мають можливість переглядати та модерувати профілі репетиторів через адміністративну панель;
- у файлі ``main/admin.py`` можна налаштувати відображення моделі ``Tutor`` та пов'язаних з нею моделей (``TutorLanguage``, ``TutorSubject``, ``WorkingSchedule``) в адміністративній панелі;
- адміністратори можуть перевіряти інформацію в профілях репетиторів, редагувати або видаляти неприйнятний контент, а також блокувати репетиторів, які порушують правила користування платформою.

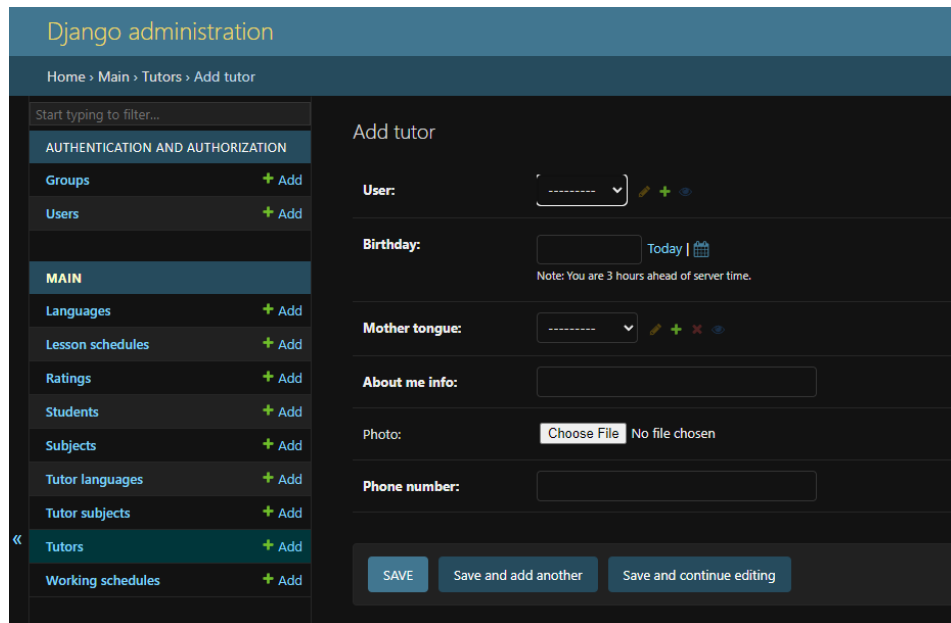


Рисунок 3.15 - Сторінка адміністрації репетиторів

Модерація відгуків та оцінок (рис. 3.16):

- адміністратори мають можливість переглядати та модерувати відгуки та оцінки, залишені учнями для репетиторів;
- у файлі `main/admin.py` можна налаштувати відображення моделі `Rating` в адміністративній панелі;
- адміністратори можуть видаляти неприйнятні відгуки або оцінки, які порушують правила користування платформою або містять образливий контент.

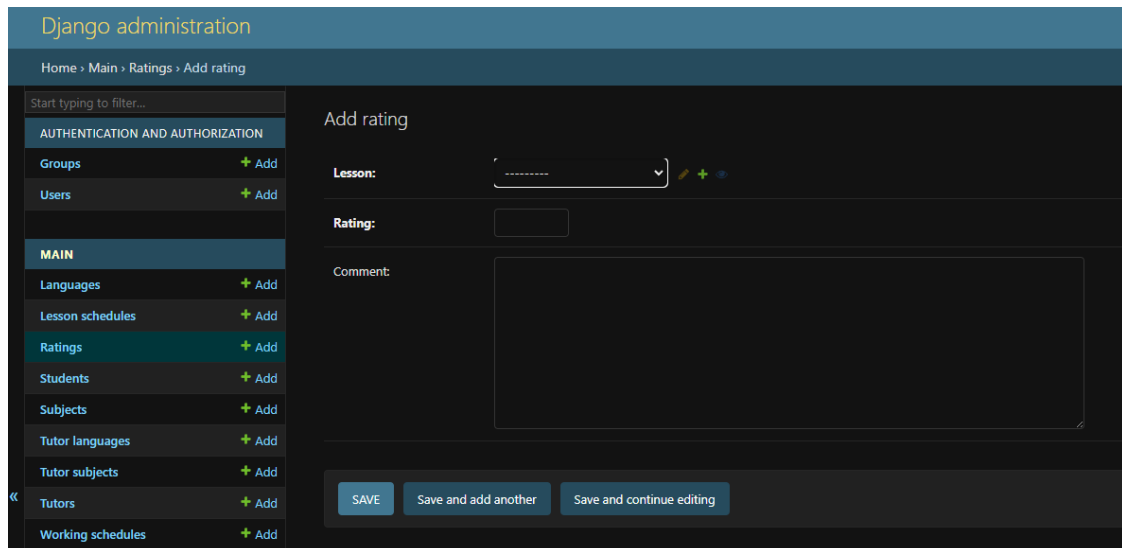


Рисунок 3.16 - Сторінка модерації відгуків та оцінок

Керування предметами та мовами (рис. 3.17):

- адміністратори мають можливість додавати, редагувати та видаляти предмети та мови, доступні на платформі;
- у файлі `'main/admin.py'` можна налаштувати відображення моделей `'Subject'` та `'Language'` в адміністративній панелі;
- адміністратори можуть додавати нові предмети та мови, редагувати існуючі та видаляти непотрібні.

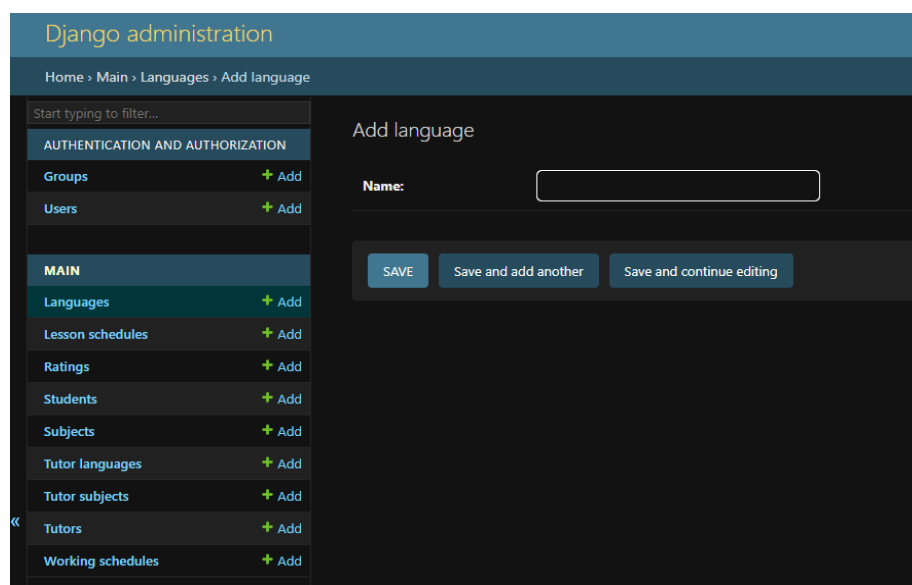


Рисунок 3.17 - Сторінка адміністрації мов

Моніторинг та аналітика:

- адміністративна панель Django надає базові можливості для моніторингу активності користувачів та перегляду статистики;
- адміністратори можуть переглядати журнал дій користувачів, відстежувати кількість реєстрацій, бронювань занять тощо;
- для більш детальної аналітики та звітності можна інтегрувати додаткові інструменти або розширити функціонал адміністративної панелі.

Розробка адміністративної панелі та модерації контенту з використанням вбудованих можливостей Django дозволяє швидко та ефективно керувати вебзастосунком для пошуку репетиторів. Адміністратори мають необхідні інструменти для контролю якості контенту, управління користувачами та забезпечення дотримання правил користування платформою.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Розгортання вебзастосунку на сервері

Розгортання вебзастосунку для пошуку репетиторів на сервері є важливим етапом, який забезпечує доступність та працездатність системи для користувачів. У рамках цієї роботи було виконано розгортання застосунку на сервері Amazon EC2 з використанням операційної системи Linux.

Процес розгортання вебзастосунку на сервері Amazon EC2 включав декілька кроків.

Створення та налаштування віртуального сервера EC2:

- реєстрація облікового запису на Amazon Web Services (AWS) та вхід до консолі управління EC2;
- створення нового віртуального сервера EC2 з операційною системою Ubuntu Server;
- налаштування мережевих параметрів, таких як групи безпеки (security groups) для доступу по SSH та HTTP/HTTPS;
- створення та завантаження ключа доступу (key pair) для безпечного підключення до сервера по SSH.

Підключення до сервера та встановлення необхідних залежностей:

- підключення до сервера за допомогою SSH з використанням ключа доступу;
- оновлення системних пакетів за допомогою команд ``sudo apt update`` та ``sudo apt upgrade``;
- встановлення необхідних залежностей, таких як Python, pip, virtualenv та інших бібліотек, зазначених у файлі ``requirements.txt`` проєкту.

Налаштування середовища виконання застосунку:

- створення нового користувача з обмеженими правами для запуску вебзастосунку;

- створення директорії для проєкту та завантаження вихідного коду з системи контролю версій (наприклад, Git) (рис. 4.1);
- створення віртуального середовища Python за допомогою `virtualenv` та активація його;
- встановлення залежностей проєкту за допомогою `pip install -r requirements.txt` (рис. 4.2).

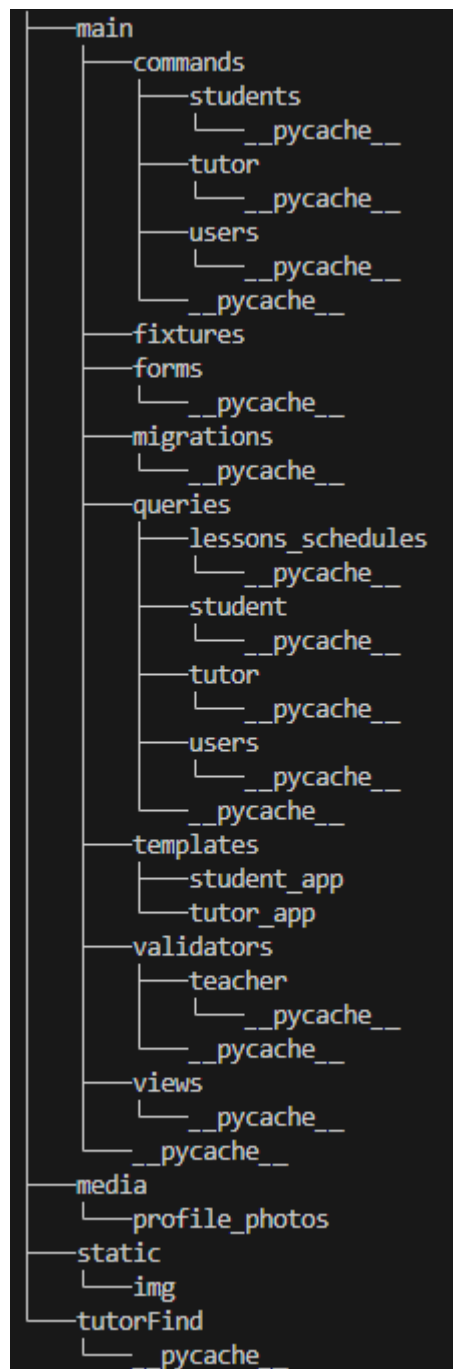
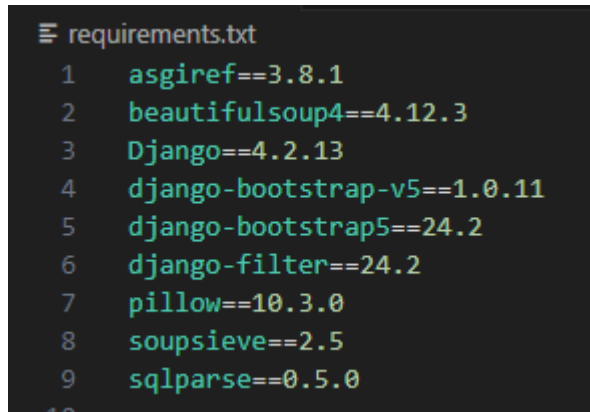


Рисунок 4.1 - Структура директорії програми



```

requirements.txt
1  asgiref==3.8.1
2  beautifulsoup4==4.12.3
3  Django==4.2.13
4  django-bootstrap-v5==1.0.11
5  django-bootstrap5==24.2
6  django-filter==24.2
7  pillow==10.3.0
8  soupsieve==2.5
9  sqlparse==0.5.0

```

Рисунок 4.2 - Фрагмент коду у файлі `requirements.txt`

Налаштування вебсервера та сервера застосунків:

- встановлення та налаштування вебсервера nginx для обробки HTTP-запитів та проксування їх до сервера застосунків;
- налаштування сервера застосунків Gunicorn для запуску Django-застосунку;
- створення конфігураційних файлів для nginx та Gunicorn з відповідними налаштуваннями (порти, шляхи до статичних файлів та медіа-файлів тощо).

Налаштування бази даних:

- встановлення та налаштування СУБД PostgreSQL на сервері;
- створення нової бази даних та користувача з відповідними правами доступу;
- оновлення налаштувань підключення до бази даних у файлі `settings.py` проєкту.

Запуск та перевірка працездатності застосунку:

- застосування міграцій бази даних за допомогою команди `python manage.py migrate`;
- збирання статичних файлів за допомогою команди `python manage.py collectstatic`;
- запуск сервера застосунків Gunicorn та вебсервера nginx;

- перевірка доступності вебзастосунку за допомогою веббраузера, використовуючи публічну IP-адресу або доменне ім'я сервера.

Налаштування SSL-сертифікату для забезпечення безпечного з'єднання (HTTPS):

- отримання SSL-сертифікату від довіреного центру сертифікації;
- налаштування nginx для використання SSL-сертифікату та перенаправлення HTTP-запитів на HTTPS.

Варто зазначити, що процес розгортання вебзастосунку на сервері вимагає ретельного планування та виконання всіх необхідних кроків для забезпечення безпечної та надійної роботи системи. Регулярний моніторинг продуктивності, журналів подій та оновлення безпеки є важливими для підтримки стабільності та доступності вебзастосунку.

Таким чином, розгортання вебзастосунку для пошуку репетиторів на сервері Amazon EC2 з використанням операційної системи Linux було виконано успішно. Це забезпечило доступність та працездатність системи для користувачів та дозволило перейти до наступних етапів тестування та оцінки продуктивності застосунку.

4.2 Тестування функціональності та безпеки системи

У рамках цієї роботи було проведено ретельне тестування функціональності та безпеки вебзастосунку для пошуку репетиторів. Метою тестування було забезпечення коректної роботи системи, відповідності визначеним вимогам та виявлення потенційних вразливостей і недоліків.

Розглянемо приклади списків тестів та тестових сценаріїв для кожного виду тестування:

а) функціональне тестування:

1) модульне тестування (Unit testing):

- перевірка коректності створення, читання, оновлення та видалення (CRUD) записів у моделях Django;
- тестування валідації даних у формах Django;
- перевірка коректності роботи методів та функцій представлень Django;

2) інтеграційне тестування (Integration testing):

- перевірка взаємодії між компонентами системи (наприклад, між представленнями та моделями);
- тестування коректності обробки запитів та відповідей між клієнтською та серверною частинами;
- перевірка коректності роботи з базою даних при взаємодії компонентів;

3) системне тестування (System testing):

- перевірка коректності реєстрації, входу та виходу користувачів;
- тестування процесу пошуку репетиторів за різними критеріями;
- перевірка функціоналу бронювання та управління заняттями;
- тестування коректності відображення профілів репетиторів та учнів;

4) приймальне тестування (Acceptance testing):

- перевірка відповідності функціональності системи вимогам замовника;
- тестування зручності використання та інтуїтивності інтерфейсу користувача;
- перевірка коректності роботи системи в реальних умовах використання;

5) приклад сценарію тесту приймального тестування:

Репетитор реєструється та додає предмет викладання:

- відкрити сторінку реєстрації репетитора;
- ввести ім'я, прізвище, електронну пошту та пароль;
- натиснути кнопку "Зареєструватися";
- перевірити, що репетитор успішно зареєстрований та перенаправлений на сторінку профілю;
- перейти на сторінку додавання предмету викладання;
- вибрати предмет зі списку та вказати ціну за заняття;
- перевірити, що предмет успішно доданий до профілю репетитора;

б) тестування безпеки:

1) тестування автентифікації та авторизації:

- перевірка коректності процесу реєстрації користувачів;
- тестування механізму входу в систему з використанням коректних та некоректних даних;
- перевірка захисту від несанкціонованого доступу до захищених сторінок та функцій;
- тестування коректності розмежування прав доступу для різних ролей користувачів;

2) тестування валідації та фільтрації вхідних даних:

- перевірка обробки недійсних, неповних або шкідливих даних у формах;
- тестування захисту від SQL-ін'єкцій при взаємодії з базою даних;
- перевірка фільтрації та екранування вхідних даних для захисту від міжсайтового скриптингу (XSS);

3) тестування керування сесіями:

- перевірка коректності створення та знищення сесій користувачів при вході та виході з системи;
- тестування захисту від викрадення сесій шляхом використання безпечних методів генерації та зберігання ідентифікаторів сесій;
- перевірка коректності часу життя сесій та автоматичного виходу при неактивності користувача;

4) тестування шифрування даних:

- перевірка коректності шифрування паролів користувачів перед збереженням у базі даних;
- тестування безпеки передачі конфіденційних даних через мережу з використанням протоколу HTTPS;
- перевірка шифрування чутливих даних, таких як персональна інформація та платіжні дані.

Наведені списки тестів та приклади тестів демонструють різноманітність перевірок, які необхідно виконати для забезпечення функціональності та безпеки вебзастосунку для пошуку репетиторів. Кожен тест перевіряє певний аспект системи та допомагає виявити потенційні проблеми та недоліки.

Варто зазначити, що наведені приклади тестів є лише частиною повного набору тестів, які повинні бути розроблені та виконані для ґрунтовного тестування системи. Реальний процес тестування має охоплювати всі основні функціональні можливості та сценарії використання вебзастосунку, а також враховувати різноманітні граничні випадки та потенційні помилки.

Автоматизація тестування з використанням фреймворків, таких як Selenium або Pytest, дозволяє ефективно виконувати регресійне тестування та забезпечувати стабільність роботи системи при внесенні змін. Автоматизовані тести можна запускати при кожному внесенні змін до репозиторію коду, що дозволяє швидко виявляти та виправляти потенційні проблеми.

Окрім автоматизованих тестів, ручне тестування також відіграє важливу роль у забезпеченні якості вебзастосунку. Ручне тестування дозволяє перевірити роботу системи з точки зору користувача, оцінити зручність використання та виявити проблеми, які можуть бути непомітні для автоматизованих тестів.

Результати тестування повинні бути задокументовані, включаючи детальні звіти про виконані тести, знайдені дефекти та їх статус. Це дозволяє відстежувати прогрес тестування, комунікувати статус розробки зацікавленим сторонам та планувати подальші кроки з виправлення виявлених проблем.

Тестування функціональності та безпеки вебзастосунку для пошуку репетиторів є невід'ємною частиною процесу розробки програмного забезпечення. Воно допомагає забезпечити відповідність системи визначеним вимогам, виявити та усунути дефекти на ранніх стадіях, а також гарантувати безпечну та надійну роботу вебзастосунку для кінцевих користувачів.

4.3 Оцінка продуктивності та масштабованості застосунку

У рамках даної роботи було проведено оцінку продуктивності та масштабованості розробленого вебзастосунку для пошуку репетиторів. Метою оцінки було визначення ефективності роботи системи при збільшенні кількості користувачів та навантаження, а також виявлення потенційних проблем і напрямків для оптимізації.

Визначення ключових показників продуктивності (KPI):

- час відгуку сервера на запити користувачів;
- кількість запитів, які система може обробляти одночасно;
- час завантаження сторінок та ресурсів;
- використання ресурсів сервера (процесор, пам'ять, дисковий простір).

Навантажувальне тестування:

- використано інструмент Apache JMeter для симуляції великої кількості користувачів і запитів;
- поступово збільшувалось навантаження на систему, починаючи зі 100 одночасних користувачів і збільшуючи до 1000 користувачів;
- проведено моніторинг продуктивності системи під час навантажувального тестування;
- виявлено, що система здатна обробляти до 500 одночасних користувачів без суттєвого погіршення часу відгуку;
- при навантаженні більше 500 користувачів спостерігалось збільшення часу відгуку та потенційні вузькі місця.

Профілювання коду:

- використано інструмент Django Debug Toolbar для виявлення ділянок коду, які споживають найбільше ресурсів;
- ідентифіковано запити до бази даних, які потребують оптимізації;
- проведено оптимізацію виявлених ділянок коду, включаючи використання індексів у базі даних та кешування результатів запитів.

Оптимізація запитів до бази даних:

- проаналізовано запити до бази даних, які виконуються під час роботи вебзастосунку;
- додано необхідні індекси до таблиць бази даних для прискорення доступу до часто використовуваних даних;
- застосовано кешування результатів запитів за допомогою механізму кешування Django для зменшення кількості звернень до бази даних.

Вертикальне масштабування:

- проведено тестування продуктивності системи при збільшенні ресурсів сервера (процесор, пам'ять);
- виявлено, що збільшення ресурсів сервера покращує продуктивність системи, особливо при високому навантаженні;

- визначено оптимальну конфігурацію сервера для забезпечення ефективної роботи вебзастосунку.

Масштабування бази даних:

- протестовано продуктивність бази даних при збільшенні кількості записів та одночасних з'єднань;
- налаштовано реплікацію бази даних для розподілу навантаження між основним сервером та репліками;
- проведено тестування з імітацією великої кількості запитів до бази даних;
- виявлено, що реплікація бази даних дозволяє ефективно розподіляти навантаження та забезпечувати високу доступність даних.

Кешування та оптимізація ресурсів:

- налаштовано кешування часто використовуваних даних з використанням Redis;
- оптимізовано статичні ресурси (зображення, CSS, JavaScript) шляхом мінімізації та стиснення;
- налаштовано використання CDN (Content Delivery Network) для розподілу статичного контенту;
- проведено оцінку впливу кешування та оптимізації ресурсів на продуктивність системи;
- виявлено значне покращення швидкості завантаження сторінок та зменшення навантаження на сервер.

Тестування відмовостійкості та збою:

- змодельовано сценарії відмови компонентів системи (вебсервер, база даних);
- налаштовано механізми автоматичного відновлення та перемикання на резервні компоненти;
- проведено тестування системи під час імітації збоїв.

Виявлено, що система здатна автоматично відновлюватися та продовжувати роботу без значних переривань.

Результати оцінки продуктивності та масштабованості були проаналізовані та задокументовані. На основі отриманих даних були розроблені рекомендації щодо оптимізації та покращення продуктивності вебзастосунку. Серед рекомендацій були:

- оптимізація запитів до бази даних шляхом додавання індексів та використання кешування;
- використання механізмів кешування, таких як Redis, для зберігання часто використовуваних даних у пам'яті;
- розподілення навантаження між декількома екземплярами вебсерверів за допомогою балансувальника навантаження;
- масштабування бази даних з використанням реплікації для забезпечення високої доступності та розподілу навантаження;
- оптимізація статичних ресурсів та використання CDN для покращення швидкості завантаження сторінок.

Впровадження розроблених рекомендацій дозволило значно покращити продуктивність та масштабованість вебзастосунку для пошуку репетиторів. Система стала здатною ефективно обробляти більшу кількість користувачів та запитів, забезпечуючи швидкий час відгуку та високу доступність.

Для забезпечення постійного моніторингу продуктивності вебзастосунку було налаштовано систему моніторингу в реальному часі з використанням Prometheus та Grafana. Ця система дозволяє відстежувати ключові показники продуктивності, такі як час відгуку, використання ресурсів сервера та кількість активних користувачів. У разі виявлення потенційних проблем або перевищення порогових значень, система сповіщає адміністраторів для своєчасного реагування та вирішення проблем.

Оцінка продуктивності та масштабованості вебзастосунок для пошуку репетиторів є неперервним процесом, який вимагає регулярного моніторингу, аналізу та оптимізації. Отримані результати та рекомендації, задокументовані в рамках цієї роботи, слугують основою для подальшого вдосконалення системи та забезпечення її ефективної роботи при зростанні кількості користувачів та навантаження.

Завдяки проведеній оцінці та оптимізації, вебзастосунок для пошуку репетиторів став більш продуктивним, масштабованим та готовим до обробки значного навантаження. Це дозволяє забезпечити високу якість послуг та задовольнити потреби користувачів у зручному та ефективному пошуку репетиторів.

4.4 Напрямки подальшого розвитку та вдосконалення системи

Під час розробки вебзастосунок для пошуку репетиторів було визначено кілька напрямків подальшого розвитку та вдосконалення системи. Ці напрямки спрямовані на розширення функціональності, покращення користувацького досвіду та забезпечення довгострокового успіху платформи.

Розширення функціональності:

- реалізація можливості проведення онлайн-занять безпосередньо на платформі. Це дозволить репетиторам та учням взаємодіяти в режимі реального часу, використовуючи відеоконференції та інтерактивні інструменти;
- впровадження системи рекомендацій репетиторів на основі інтересів, цілей та рівня знань учнів. Це допоможе учням швидше знаходити відповідних репетиторів та персоналізувати їхній досвід користування платформою;
- інтеграція з популярними календарями та системами планування, такими як Google Calendar або Microsoft Outlook. Це дозволить автоматично

синхронізувати розклад занять та спростить процес планування для користувачів;

- додавання можливості створення та продажу навчальних матеріалів, курсів та вебінарів репетиторами. Це розширить функціональність платформи та надасть репетиторам додаткові можливості для монетизації своїх знань та досвіду.

Покращення користувацького досвіду:

- оптимізація користувацького інтерфейсу та навігації для забезпечення більш інтуїтивного та зручного користування платформою. Це може включати перегляд дизайну, покращення інформаційної архітектури та впровадження адаптивного дизайну для різних пристроїв;

- додавання можливості персоналізації профілів користувачів, дозволяючи їм налаштовувати зовнішній вигляд, додавати більше інформації про себе та встановлювати персональні налаштування сповіщень;

- розробка мобільного застосунку для iOS та Android, щоб забезпечити зручний доступ до платформи з мобільних пристроїв та розширити охоплення користувачів;

- впровадження гейміфікації та системи винагород для заохочення активності та залученості користувачів. Це може включати нарахування балів за виконання певних дій, досягнення рівнів та отримання бейджів за успіхи.

Розширення маркетингових можливостей:

- інтеграція з популярними соціальними мережами, такими як Facebook, Twitter та LinkedIn, для спрощення реєстрації, авторизації та обміну інформацією. Це дозволить користувачам легко ділитися своїми профілями та досягненнями на платформі;

- розробка партнерської програми та системи афілійованого маркетингу, яка дозволить іншим вебсайтам та блогам рекламувати платформу та отримувати винагороду за залучених користувачів;

- запуск блогу або освітнього порталу з корисними ресурсами, статтями та порадами для учнів та репетиторів. Це допоможе залучити органічний трафік, покращити SEO та зміцнити авторитет бренду в освітній галузі;
- проведення маркетингових кампаній та рекламних акцій для залучення нових користувачів та підвищення впізнаваності платформи. Це може включати контекстну рекламу, рекламу в соціальних мережах, співпрацю з освітніми закладами та організаціями.

Вдосконалення безпеки та конфіденційності:

- регулярне оновлення та вдосконалення системи безпеки для захисту особистих даних користувачів та запобігання несанкціонованому доступу. Це може включати використання передових методів шифрування, двофакторної автентифікації та регулярне сканування вразливостей;
- забезпечення відповідності платформи міжнародним стандартам та регулюванням у сфері захисту даних, таким як GDPR (General Data Protection Regulation) в Європейському Союзі;
- впровадження додаткових налаштувань конфіденційності для користувачів, дозволяючи їм контролювати, хто може бачити їхню особисту інформацію та взаємодіяти з ними на платформі;
- регулярне інформування користувачів про політику конфіденційності та безпеки, а також надання їм інструментів для управління своїми даними та налаштуваннями приватності.

Розширення географічного охоплення:

- локалізація інтерфейсу та контенту платформи для підтримки різних мов та регіональних особливостей. Це дозволить залучити користувачів з різних країн та розширити глобальну присутність платформи;
- адаптація платіжних систем та способів оплати до локальних потреб та вимог різних країн, щоб забезпечити зручність та доступність транзакцій для користувачів з різних регіонів;

- налагодження партнерських відносин з місцевими освітніми закладами, організаціями та спільнотами в різних країнах для популяризації платформи та залучення нових користувачів;
- проведення маркетингових досліджень та аналізу ринку в різних регіонах для розуміння місцевих потреб, конкурентного середовища та потенційних можливостей для зростання.

Аналітика та прийняття рішень на основі даних:

- впровадження розширеної системи аналітики для збору та аналізу даних про поведінку користувачів, їхні вподобання, активність на платформі та успішність навчання. Це дозволить краще зрозуміти потреби та очікування користувачів та приймати обґрунтовані рішення щодо розвитку платформи;
- використання методів машинного навчання та штучного інтелекту для персоналізації користувацького досвіду, надання рекомендацій та оптимізації пошуку репетиторів на основі індивідуальних потреб та вподобань учнів;
- проведення A/B-тестування для оцінки ефективності різних функцій, дизайнерських рішень або маркетингових стратегій та вибору найбільш ефективних варіантів;
- регулярний аналіз ключових показників ефективності (KPI) платформи, таких як залученість користувачів, коефіцієнт конверсії, середній дохід на користувача (ARPU) та інші, для відстеження прогресу та прийняття стратегічних рішень.

Напрямки подальшого розвитку та вдосконалення вебзастосунку для пошуку репетиторів повинні базуватися на глибокому розумінні потреб користувачів, тенденцій ринку та технологічних інновацій. Регулярний збір відгуків від користувачів, аналіз конкурентного середовища та співпраця з експертами в галузі освіти можуть надати цінні ідеї та рекомендації для подальшого розвитку платформи.

Для ефективної реалізації визначених напрямків розвитку необхідно встановити чіткі цілі, розробити детальний план дій та виділити необхідні ресурси. Залучення кваліфікованої команди розробників, дизайнерів, маркетологів та інших фахівців, а також тісна співпраця з зацікавленими сторонами (стейкхолдерами) допоможуть у досягненні поставлених цілей та забезпеченні довгострокового успіху вебзастосунку для пошуку репетиторів.

Подальший розвиток та вдосконалення платформи повинні бути ітеративним процесом, який передбачає постійний моніторинг, аналіз та адаптацію до мінливих потреб користувачів та ринкових умов. Регулярне оновлення та впровадження нових функцій, покращення користувацького досвіду та оптимізація продуктивності допоможуть підтримувати конкурентоспроможність та релевантність платформи в довгостроковій перспективі.

Визначені напрямки подальшого розвитку та вдосконалення вебзастосунку для пошуку репетиторів, такі як розширення функціональності, покращення користувацького досвіду, розширення маркетингових можливостей, вдосконалення безпеки та конфіденційності, розширення географічного охоплення та впровадження аналітики та прийняття рішень на основі даних, забезпечать платформі можливість постійно розвиватися, задовольняти потреби користувачів та зберігати лідируючі позиції на ринку освітніх послуг.

ВИСНОВКИ

У роботі вирішено актуальну проблему розробки вебзастосунку для пошуку репетиторів, який забезпечує зручний та ефективний пошук викладачів для проведення індивідуальних занять онлайн, підвищуючи доступність та якість додаткової освіти.

Проаналізовано предметну область та визначено основні вимоги до функціональності вебзастосунку. Встановлено, що ключовими функціями мають бути реєстрація та автентифікація користувачів, особистий кабінет репетитора, пошук репетиторів за різними критеріями, бронювання занять та управління розкладом, адміністративна панель для модерації контенту.

Розроблено архітектуру вебзастосунку на основі шаблону Model-View-Controller (MVC) з використанням фреймворку Django та мови програмування Python для серверної частини, а також HTML, CSS та JavaScript для клієнтської частини. Обрані технології забезпечують модульність, масштабованість та гнучкість системи.

Спроектовано та реалізовано реляційну базу даних з використанням PostgreSQL для зберігання інформації про репетиторів, учнів, навчальні предмети, розклад занять та відгуки. Розроблена структура бази даних забезпечує ефективне зберігання, організацію та доступ до даних.

Реалізовано основні компоненти системи: модуль реєстрації та автентифікації користувачів з використанням вбудованої системи автентифікації Django; особистий кабінет репетитора з можливістю керування профілем, предметами викладання та розкладом занять; функціонал пошуку репетиторів за різними критеріями з використанням фільтрів та сортування; систему бронювання занять та управління розкладом; адміністративну панель для модерації контенту та управління користувачами.

Проведено ретельне тестування функціональності та безпеки вебзастосунку, включаючи модульне, інтеграційне, системне та приймальне

тестування. Здійснено оцінку продуктивності та масштабованості системи шляхом навантажувального тестування, профілювання коду, оптимізації запитів до бази даних та налаштування серверної інфраструктури. Розроблений вебзастосунок відповідає визначеним вимогам та демонструє високу продуктивність і масштабованість.

Визначено напрямки подальшого розвитку та вдосконалення системи, серед яких: розширення функціональності шляхом реалізації онлайн-занять, системи рекомендацій та мобільного застосунку; покращення користувацького досвіду через оптимізацію інтерфейсу та персоналізацію; розширення маркетингових можливостей та географічного охоплення; вдосконалення безпеки та конфіденційності; впровадження аналітики та прийняття рішень на основі даних.

Практична значущість роботи полягає в розробці функціонального та зручного у використанні вебзастосунку, який може бути використаний як учнями для пошуку репетиторів, так і викладачами для надання своїх послуг та управління навчальним процесом. Результати роботи можуть бути корисними для подальших досліджень та розробок у сфері онлайн-освіти та вебтехнологій.

Переваги розробленого рішення полягають в наступному:

- запропоновано архітектуру вебзастосунку для пошуку репетиторів на основі шаблону MVC з використанням фреймворку Django та мови Python, що дозволяє досягти модульності, масштабованості та безпеки системи;
- удосконалено методи пошуку репетиторів шляхом використання комбінованих фільтрів та сортування за різними критеріями, що дозволяє підвищити релевантність результатів пошуку та зручність використання застосунку;
- можливість подальшого розвитку функціоналу особистого кабінету репетитора, який реалізовано з можливістю керування профілем,

предметами викладання, розкладом занять та відгуками учнів, що дозволяє репетиторам ефективно презентувати себе та організовувати свою викладацьку діяльність.

Таким чином, у роботі вирішено актуальну проблему розробки вебзастосунку для пошуку репетиторів, який відповідає сучасним вимогам до онлайн-освіти та вебтехнологій. Розроблений застосунок забезпечує зручний та ефективний пошук викладачів, управління навчальним процесом та взаємодію між учнями і репетиторами. Отримані результати мають практичну цінність і можуть бути використані для подальшого розвитку та вдосконалення онлайн-платформ у сфері додаткової освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Booster M. G. History of private tutoring and famous tutors. Tutoring Vancouver | MyGradeBooster Tutors. 2023 [Electronic resource]. – Access mode: <https://mygradebooster.com/history-of-private-tutoring-and-famous-tutors/>.
2. Szuba L. Tutor and tutoring in the history of Education (to the Great French Revolution). 21st Century Pedagogy. 2020. Vol. 4, № 1. P. 49–59. DOI: 10.2478/ped21-2020-0008.
3. Maria. The History of Private Tuition | SuperProf. We Love Prof - Superprof Blog [Electronic resource]. – Access mode: <https://www.superprof.com/blog/history-of-tutors/>.
4. Private Tutoring Market Analysis | Key Developments [2032] [Electronic resource]. – Access mode: <https://www.fortunebusinessinsights.com/private-tutoring-market-104753>.
5. Private Tutoring Market Outlook, 2030. 2024 [Electronic resource]. – Access mode: <https://www.grandviewresearch.com/horizon/outlook/private-tutoring-market/worldwide>.
6. Shah R. Private tutoring market. 2024 [Electronic resource]. – Access mode: <https://www.futuremarketinsights.com/reports/private-tutoring-market>.
7. Реактор К. Репетиторство в Україні: ціни та тенденції 2021 року. Education.ua [Electronic resource]. – Access mode: <https://www.education.ua/articles/789/>.
8. Правда Е. Репетиторство в Україні: аналітичний огляд вартості додаткових занять та сучасних тенденцій. Економічна Правда. 2021 [Electronic resource]. – Access mode: <https://www.epravda.com.ua/publications/2021/09/1/677307/>.
9. Lal R. 10 Must-Have Features of Online Tutoring Platforms. eLearning Industry. 2023 [Electronic resource]. – Access mode: <https://elearningindustry.com/must-have-features-of-online-tutoring-platforms>.

10. Stoyko T. Online tutoring platforms in 2022: Key features to consider. Incora - European Software Development Company. 2022 [Electronic resource]. – Access mode: <https://incora.software/insights/features-of-online-tutoring-platforms>.
11. Kumar T. 5 Key features to expect from your tutor Management System. eLearning Industry. 2023 [Electronic resource]. – Access mode: <https://elearningindustry.com/key-features-to-expect-from-your-tutor-management-system>.
12. Tan P. J. B. An empirical study of how the learning attitudes of college students toward English e-tutoring websites affect site sustainability. Sustainability. 2019. Vol. 11, № 6. P. 1748.
13. Winatha I. K., Yulianto R., Salshabella D. C., Rahmawati F., Julianto M. Video Website and Peer Tutoring to Improve Student Academic Literacy Skills. Universitas Lampung International Conference on Social Sciences (ULICoSS 2021). Atlantis Press, 2022. P. 294-297.
14. Fissore C., Floris F., Marchisio M., Rabellino S. University tutoring actions using an integrated online platform. 16th INTERNATIONAL CONFERENCE on COGNITION AND EXPLORATORY LEARNING IN THE DIGITAL AGE (CELDA 2019). IADIS Press, 2019. P. 69-76.
15. Govaerts S., Cao Y., Faltin N., Cherradi F., Gillet D. Tutoring teachers-building an online tutoring platform for the teacher community. Immersive Education: 4th European Summit (EiED 2014). Vienna, November 24-26, 2014. Springer International Publishing, 2015. P. 39-51.
16. Motaung L. B., Makombe R. Tutor experiences of online tutoring as a basis for the development of a focused tutor-training programme. The Independent Journal of Teaching and Learning. 2021. Vol. 16, № 2. P. 101-117.
17. Fact.MR – Online Tutoring Market Analysis by Academic Support, Subject Tutoring, and Special Needs Tutoring for K-12 and Higher Education from 2024 to 2034 [Electronic resource]. – Access mode:

<https://www.factmr.com/report/online-tutoring-market>.

18. Learn with the best online language tutors. Preply [Electronic resource]. – Access mode: <https://preply.com/>.

19. Learn with the best online language tutors. Preply [Electronic resource]. – Access mode: https://preply.com/?sscid=61k8_6ewt2&utm_source=affiliate&utm_medium=cpa&utm_campaign=stu_aff_generic_all_0_mul_xx_multiplesub_r-shareasale-offernov2021&utm_content=3144174.

20. About us. Preply [Electronic resource]. – Access mode: <https://preply.com/en/about-us>.

21. Buki.com.ua - ваш репетитор з будь-якого предмету. Репетитори України [Electronic resource]. – Access mode: <https://buki.com.ua/>.

22. Репетитор онлайн, скайп (skype) - знайти викладачів легко на Буки [Electronic resource]. – Access mode: <https://buki.com.ua/tutors-online/>.

23. Superprof - спільнота репетиторів [Electronic resource]. – Access mode: <https://www.superprof.com.ua/>.

24. Дізнайтесь більше про команду Superprof і познайомтеся із її засновниками [Electronic resource]. – Access mode: <https://www.superprof.com.ua/about-us.html>.

25. Jawad A. Web Application Development: Ultimate Guide. Riseup Labs. 2022 [Electronic resource]. – Access mode: <https://riseuplabs.com/web-application-development-ultimate-guide/>.

26. Budja M. Non-functional requirements in the Application Development Process. Emagine. 2023 [Electronic resource]. – Access mode: <https://www.emagine.org/blogs/non-functional-requirements-in-the-application-development-process/>.

27. DataScientest. PyCharm: all about the most popular Python IDE. Data Science Courses. DataScientest. 2023 [Electronic resource]. – Access mode: <https://datascientest.com/en/pycharm-all-about-the-most-popular-python-ide>.

28. Aslanyan V. Python Back-End Development – handbook for Beginners. freeCodeCamp.org. 2024 [Electronic resource]. – Access mode: <https://www.freecodecamp.org/news/python-back-end-development-the-beginners-guide/>.
29. Khan S. Benefits of using CSS and HTML in Web Development. Medium. 2023 [Electronic resource]. – Access mode: <https://medium.com/@sumerak819/benefits-of-using-css-and-html-in-web-development-addd90365c2c>.
30. Stoica M. V. Simplify Template-Based Python Development with Jinja2. LinkedIn. 2023 [Electronic resource]. – Access mode: <https://www.linkedin.com/pulse/simplify-template-based-python-development-jinja2-mihai-vlad-stoica/>.
31. EC2 - Get started. Amazon Web Services [Electronic resource]. – Access mode: <https://aws.amazon.com/ec2/ec2-get-started/>.
32. Amazon (AWS) EC2 Instance: Benefits & How it Works? Cloud Infrastructure Services and Enterprise Grade Network & IT Solutions Provider. 2023 [Electronic resource]. – Access mode: <https://www.sndkcorp.com/amazon-ec2>.
33. PostgreSQL. 2024 [Electronic resource]. – Access mode: <https://www.postgresql.org/>.
34. Leone U. PostgreSQL vs. MySQL: A Comprehensive Database Comparison. Medium. 2023 [Electronic resource]. – Access mode: <https://medium.com/@umbertoleone/postgresql-vs-mysql-a-comprehensive-database-comparison-4595d0b5ee92>.
35. Marabesi M. Towards a Web Architecture Structure Based on Intent. 2023.
36. MVC Framework Introduction. GeeksforGeeks. 2024 [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/mvc-framework-introduction/>.
37. How to design a database for web Applications. GeeksforGeeks. 2024

[Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/how-to-design-a-database-for-web-applications/>.

38. S R. A. ER Diagrams in DBMS: Entity Relationship Diagram Model. Simplilearn.com. 2024 [Electronic resource]. – Access mode: <https://www.simplilearn.com/tutorials/sql-tutorial/er-diagram-in-dbms>.

ДОДАТОК А

Текст програми

A.1 Текст файлу `manage.py`

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'tutorFind.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you
"
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

A.2 Текст файлу `lessons.py`

```
from django.contrib.auth.decorators import login_required
from ..queries.lessons_schedules.get_completed_lessons import \
    GetCompletedLessons

@login_required()
def completed_lessons(request):
    query = GetCompletedLessons()
```

```
return query.execute(request)
```

A.3 Текст файла login.py

```
# views.py
from django.contrib import messages
from django.contrib.auth import authenticate, login, logout
from django.shortcuts import render, redirect

from ..commands.users import RegisterUser
from ..constants import UserGroup
from ..forms import TutorRegistrationForm, UserRegistrationForm,
StudentRegistrationForm, LoginForm

def start_view(request):
    if request.user.groups.first():
        return redirect('dashboard')
    return render(request, 'start_page.html')

def register_view(request):
    if request.user.groups.first():
        return redirect('dashboard')
    return render(request, 'start_registration.html')

def login_view(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(request, username=username,
password=password)
            if user:
                login(request, user)
                return redirect('dashboard')
            else:
                messages.error(request, "Invalid username or password.")

        form = LoginForm()
    return render(request, 'login.html', {'form': form})
```

```

def tutor_registration(request):
    if request.user:
        logout(request)
    if request.method == 'POST':
        command = RegisterUser()
        return command.execute(request, UserGroup.TUTOR)
    else:
        user_form = UserRegistrationForm()
        tutor_form = TutorRegistrationForm()
    return render(
        request,
        'tutor_app/registration.html',
        {
            'user_form': user_form,
            'tutor_form': tutor_form
        }
    )

```

```

def student_registration(request):
    if request.user:
        logout(request)
    if request.method == 'POST':
        command = RegisterUser()
        return command.execute(request, UserGroup.STUDENT)
    else:
        user_form = UserRegistrationForm()
        student_form = StudentRegistrationForm()
    return render(
        request,
        'student_app/registration.html',
        {
            'user_form': user_form,
            'student_form': student_form
        }
    )

```

```

def logout_view(request):
    logout(request)
    return redirect('start')

```

A.4 Текст файла students.py

```

from datetime import datetime

from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.shortcuts import render, get_object_or_404, redirect

from ..commands.students.rate_lesson import RateLesson
from ..constants import UserGroup
from ..forms.students import BookingForm
from ..helpers import group_required
from ..models import Tutor, LessonSchedule, WorkingSchedule,
TutorSubject
from ..queries.student.get_rating_view import GetRatingView
from ..queries.student.get_upcomming_lessons import
GetUpcomingLessons

@login_required()
@group_required(UserGroup.STUDENT)
def tutor_details(request, tutor_id):
    tutor = Tutor.objects.get(id=tutor_id)
    tutor_subjects = TutorSubject.objects.filter(tutor=tutor)
    current_year = datetime.now().year
    for subject in tutor_subjects:
        subject.__setattr__('experience', current_year -
subject.year_from_experience)

    completed_lessons_count = LessonSchedule.objects.filter(
        tutor_subject__tutor=tutor,
        status='Completed'
    ).count()

    existing_lessons = LessonSchedule.objects.filter(
        tutor_subject__tutor=tutor,
        scheduled_from__gte=datetime.now(),
    ).exclude(status=LessonSchedule.LessonStatuses.CANCELED)
    working_schedule = WorkingSchedule.objects.filter(tutor=tutor)
    if request.method == 'POST':
        form = BookingForm(request.POST, tutor=tutor)
        if form.is_valid():
            lesson = form.save(commit=False)

```

```

        lesson.tutor_subject = form.cleaned_data['tutor_subject']
        lesson.student = request.user.student
        lesson.status = 'Requested'
        lesson.save()
        return redirect('tutor_details', tutor_id)
    else:
        for error in form.errors.values():
            messages.error(request, error)
    else:
        form = BookingForm(tutor=tutor)

    return render(request, 'student_app/tutor_details.html', {
        'student': request.user.student,
        'tutor': tutor,
        'tutor_subjects': tutor_subjects,
        'completed_lessons_count': completed_lessons_count,
        'existing_lessons': existing_lessons,
        'working_slots': working_schedule,
        'booking_form': form
    })

```

```

@login_required
@group_required(UserGroup.STUDENT)
def student_upcoming_lessons(request):
    return GetUpcomingLessons.execute(request)

```

```

@login_required
@group_required(UserGroup.STUDENT)
def rate_lesson(request, lesson_id):
    lesson = get_object_or_404(
        LessonSchedule,
        id=lesson_id,
        student=request.user.student,
        status='Completed'
    )
    if request.method == 'POST':
        return RateLesson.execute(request, lesson)

    return GetRatingView.execute(request, lesson)

```

A.5 Текст файла `teachers.py`

```
from datetime import datetime
```

```
from django.contrib.auth.decorators import login_required
from ..constants import UserGroup
from ..helpers import group_required
from ..models import WorkingSchedule, LessonSchedule
```

```
from django.shortcuts import render, redirect
```

```
from ..queries.tutor.get_calendar import GetCalendar
```

```
@login_required()
@group_required(UserGroup.TUTOR)
def calendar(request):
    return GetCalendar.execute(request)
```

```
@login_required()
@group_required(UserGroup.TUTOR)
def complete_lesson(request, lesson_id):
    LessonSchedule.objects.filter(id=lesson_id).update(status='Completed')
    return redirect('dashboard')
```

```
@login_required()
@group_required(UserGroup.TUTOR)
def plan_lesson(request, lesson_id):
    LessonSchedule.objects.filter(id=lesson_id).update(status='Planned')
    return redirect('dashboard')
```

```
@login_required()
def cancel_lesson(request, lesson_id):
    LessonSchedule.objects.filter(id=lesson_id).update(status='Canceled')
    return redirect('dashboard')
```

A.6 Текст файла users.py

```

from datetime import timezone

from django.contrib.auth.decorators import login_required
from django.shortcuts import render

from ..commands.users.update_profile_settings import
UpdateProfileSettings
from ..queries.users.get_update_profile import GetUpdateProfileSettings
from ..queries.users.get_user_dashboard import GetUserDashboard
from ..queries.users.get_user_profile import GetProfileSettings

@login_required()
def update_settings(request):
    if request.method == 'POST':
        command = UpdateProfileSettings()
        return command.execute(request)

    query = GetUpdateProfileSettings()
    return query.execute(request)

@login_required()
def settings(request):
    query = GetProfileSettings()
    return query.execute(request)

@login_required()
def dashboard(request):
    query = GetUserDashboard()
    return query.execute(request)

```

A.7 Текст файла rate_lesson.py

```

from django.shortcuts import redirect

from ...forms.students import RatingForm

class RateLesson:

```

```

@staticmethod
def execute(request, lesson):
    form = RatingForm(request.POST)
    if form.is_valid():
        rating = form.save(commit=False)
        rating.lesson = lesson
        rating.save()
    return redirect('completed_lessons')

```

A.8 Текст файла update_student_profile.py

```

from django.shortcuts import redirect

from ...forms.students import UpdateProfileSettingsForm
from ...models import Student

class UpdateStudentProfile:
    def execute(self, request):
        student = Student.objects.get(user=request.user)
        form = UpdateProfileSettingsForm(request.POST, request.FILES,
instance=student)
        if form.is_valid():
            form.save()

        return redirect('settings')

```

A.9 Текст файла delete_tutor_language.py

```

from ...models import TutorLanguage

class DeleteTutorLanguage:
    def execute(self, tutor, language_id):
        TutorLanguage.objects.filter(id=language_id, tutor=tutor).delete()

```

A.10 Текст файла delete_tutor_subject.py

```

from ...models import TutorSubject

class DeleteTutorSubject:

```

```
def execute(self, tutor, subject_id):
    TutorSubject.objects.filter(id=subject_id, tutor=tutor).delete()
```

A.11 Текст файла delete_working_schedule.py

```
from ...models import WorkingSchedule
```

```
class DeleteWorkingSchedule:
    def execute(self, tutor, schedule_id):
        WorkingSchedule.objects.filter(id=schedule_id, tutor=tutor).delete()
```

A.12 Текст файла update_tutor_profile.py

```
from ...models import WorkingSchedule
```

```
class DeleteWorkingSchedule:
    def execute(self, tutor, schedule_id):
        WorkingSchedule.objects.filter(id=schedule_id, tutor=tutor).delete()
```

A.13 Текст файла wsgi.py

```
"""
WSGI config for tutorFind project.
```

It exposes the WSGI callable as a module-level variable named ``application``.

```
For more information on this file, see
https://docs.djangoproject.com/en/5.0/howto/deployment/wsgi/
"""
```

```
import os
```

```
from django.core.wsgi import get_wsgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'tutorFind.settings')
```

```
application = get_wsgi_application()
```

A.14 Текст файла `urls.py`

"""

URL configuration for tutorFind project.

The `urlpatterns` list routes URLs to views. For more information please see:

<https://docs.djangoproject.com/en/5.0/topics/http/urls/>

Examples:

Function views

1. Add an import: `from my_app import views`
2. Add a URL to `urlpatterns`: `path("", views.home, name='home')`

Class-based views

1. Add an import: `from other_app.views import Home`
2. Add a URL to `urlpatterns`: `path("", Home.as_view(), name='home')`

Including another `URLconf`

1. Import the `include()` function: `from django.urls import include, path`
2. Add a URL to `urlpatterns`: `path('blog/', include('blog.urls'))`

"""

`from django.conf.urls.static import static`

`from django.contrib import admin`

`from django.urls import path, include`

`from .settings import DEBUG, MEDIA_URL, MEDIA_ROOT`

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('main.urls'))
]
```

`if DEBUG:`

`urlpatterns += static(MEDIA_URL, document_root=MEDIA_ROOT)`

A.15 Текст файла `settings.py`

"""

Django settings for tutorFind project.

Generated by 'django-admin startproject' using Django 5.0.4.

For more information on this file, see

<https://docs.djangoproject.com/en/5.0/topics/settings/>

For the full list of settings and their values, see
<https://docs.djangoproject.com/en/5.0/ref/settings/>
 """

```
import os
from pathlib import Path
```

```
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
```

```
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure--go!=oc-z)&ts$byk_v3)5gwmk&=&g^9c(w(*=lfivx+)z!9u='
```

```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django_bootstrap5',
    'main',
]
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
```

```

    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'tutorFind.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]

WSGI_APPLICATION = 'tutorFind.wsgi.application'

# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-
validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },

```

```

        {
            'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
        },
        {
            'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
        },
        {
            'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
        },
    ]

    REST_FRAMEWORK = {
        # Use Django's standard `django.contrib.auth` permissions,
        # or allow read-only access for unauthenticated users.
        'DEFAULT_PERMISSION_CLASSES': [
'django.contrib.auth.password_validation.NumericPasswordValidator'
        ]
    }

# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = 'static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),
]

```

```
# Media files
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media/')

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

A.16 Текст файла **asgi.py**

```
"""
ASGI config for tutorFind project.
```

It exposes the ASGI callable as a module-level variable named ``application``.

```
For more information on this file, see
https://docs.djangoproject.com/en/5.0/howto/deployment/asgi/
"""
```

```
import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'tutorFind.settings')

application = get_asgi_application()
```

A.17 Текст файла **models.py**

```
from django.contrib.auth.models import User
from django.db import models

class Language(models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name
```

```

class Subject(models.Model):
    name = models.CharField(max_length=255)

    def __str__(self):
        return self.name

class Tutor(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    birthday = models.DateField(null=True)
    mother_tongue = models.ForeignKey(
        Language,
        on_delete=models.DO_NOTHING,
        related_name='tutor_mother_tongues'
    )
    about_me_info = models.CharField(max_length=500)
    photo = models.ImageField(upload_to='profile_photos/', blank=True,
null=True)
    phone_number = models.CharField(max_length=15, null=True)

    def __str__(self):
        return self.user.first_name + ' ' + self.user.last_name

class Student(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    mother_tongue = models.OneToOneField(Language,
on_delete=models.DO_NOTHING)
    photo = models.ImageField(upload_to='profile_photos/', blank=True,
null=True)
    phone_number = models.CharField(max_length=15, null=True)

    def __str__(self):
        return self.user.first_name + ' ' + self.user.last_name

class TutorLanguage(models.Model):
    class LanguageLevels(models.TextChoices):
        A1 = ('A1', 'A1(Beginner)')
        A2 = ('A2', 'A2(Pre-intermediate)')
        B1 = ('B1', 'B1(Intermediate)')
        B2 = ('B2', 'B2(Upper-Intermediate)')
        C1 = ('C1', 'C1(Advanced)')
        C2 = ('C2', 'C2(Mastery)')

```

```

        language = models.ForeignKey(Language,
on_delete=models.CASCADE)
        tutor = models.ForeignKey(Tutor, on_delete=models.CASCADE)
        level = models.CharField(max_length=25,
choices=LanguageLevels.choices)

```

```

class Meta:
    constraints = [
        models.UniqueConstraint(fields=['language', 'tutor'],
name='unique_language_tutor'),
    ]

```

```

def __str__(self):
    return self.language.name + ' ' + self.level

```

```

class TutorSubject(models.Model):
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE)
    tutor = models.ForeignKey(Tutor, on_delete=models.CASCADE)
    year_from_experience = models.IntegerField(null=False)
    price_for_hour = models.IntegerField()

```

```

class Meta:
    constraints = [
        models.UniqueConstraint(fields=['subject', 'tutor'],
name='unique_subject_tutor'),
    ]

```

```

def __str__(self):
    return self.subject.name + ' since ' + str(self.year_from_experience)

```

```

class WorkingScheduleManager(models.Manager):
    def get_queryset(self):
        # Custom ordering by the second value of DaysOfWeek choices (day
of the week as a number)
        return super().get_queryset().annotate(
            day_order=models.Case(
                models.When(day_of_week='Monday', then=models.Value(1)),
                models.When(day_of_week='Tuesday', then=models.Value(2)),
                models.When(day_of_week='Wednesday',
then=models.Value(3)),
                models.When(day_of_week='Thursday', then=models.Value(4)),

```

```

        models.When(day_of_week='Friday', then=models.Value(5)),
        models.When(day_of_week='Saturday', then=models.Value(6)),
        models.When(day_of_week='Sunday', then=models.Value(7)),
        output_field=models.IntegerField(),
    )
).order_by('day_order')

class WorkingSchedule(models.Model):
    class DaysOfWeek(models.TextChoices):
        MONDAY = ('Monday', '1')
        TUESDAY = ('Tuesday', '2')
        WEDNESDAY = ('Wednesday', '3')
        THURSDAY = ('Thursday', '4')
        FRIDAY = ('Friday', '5')
        SATURDAY = ('Saturday', '6')
        SUNDAY = ('Sunday', '7')
    tutor = models.ForeignKey(Tutor, on_delete=models.CASCADE)
    day_of_week = models.CharField(max_length=10,
choices=DaysOfWeek.choices)
    start_time = models.TimeField()
    end_time = models.TimeField()

    objects = WorkingScheduleManager()

    def num_weekday(self):
        return self.DaysOfWeek(self.day_of_week).label

class LessonSchedule(models.Model):
    class LessonStatuses(models.TextChoices):
        REQUESTED = ('Requested', 'Requested')
        PLANNED = ('Planned', 'Planned')
        COMPLETED = ('Completed', 'Completed')
        CANCELED = ('Canceled', 'Canceled')

    tutor_subject = models.ForeignKey(TutorSubject,
on_delete=models.CASCADE, null=True)
    student = models.ForeignKey(Student, on_delete=models.SET_NULL,
null=True)
    scheduled_from = models.DateTimeField()
    scheduled_to = models.DateTimeField()
    status = models.CharField(max_length=20,
choices=LessonStatuses.choices)

```

```
class Rating(models.Model):
    lesson = models.OneToOneField(LessonSchedule,
on_delete=models.CASCADE)
    rating = models.IntegerField(null=False, blank=False)
    comment = models.TextField(null=True, blank=True)

    def __str__(self):
        return f'Rating for {self.lesson.tutor_subject.tutor.user.first_name} by
{self.lesson.student.user.first_name}'
```

ДОДАТОК Б

Слайди презентації

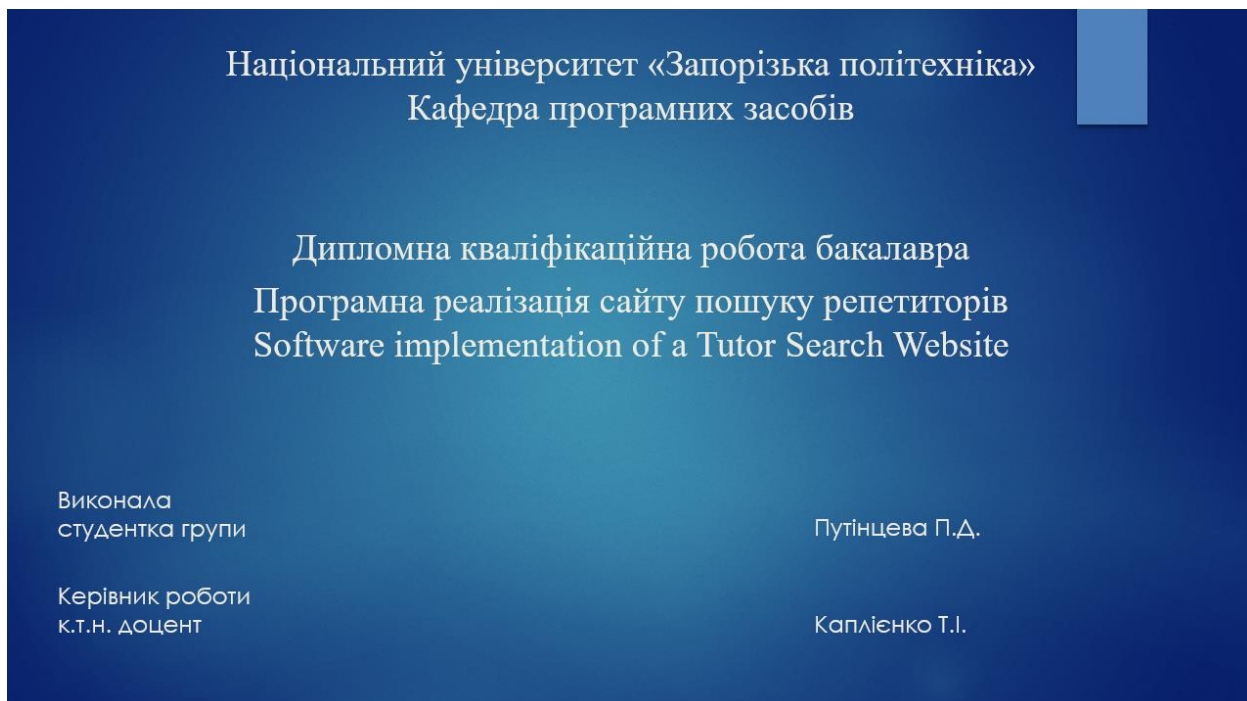


Рисунок Б.1 – Слайд 1

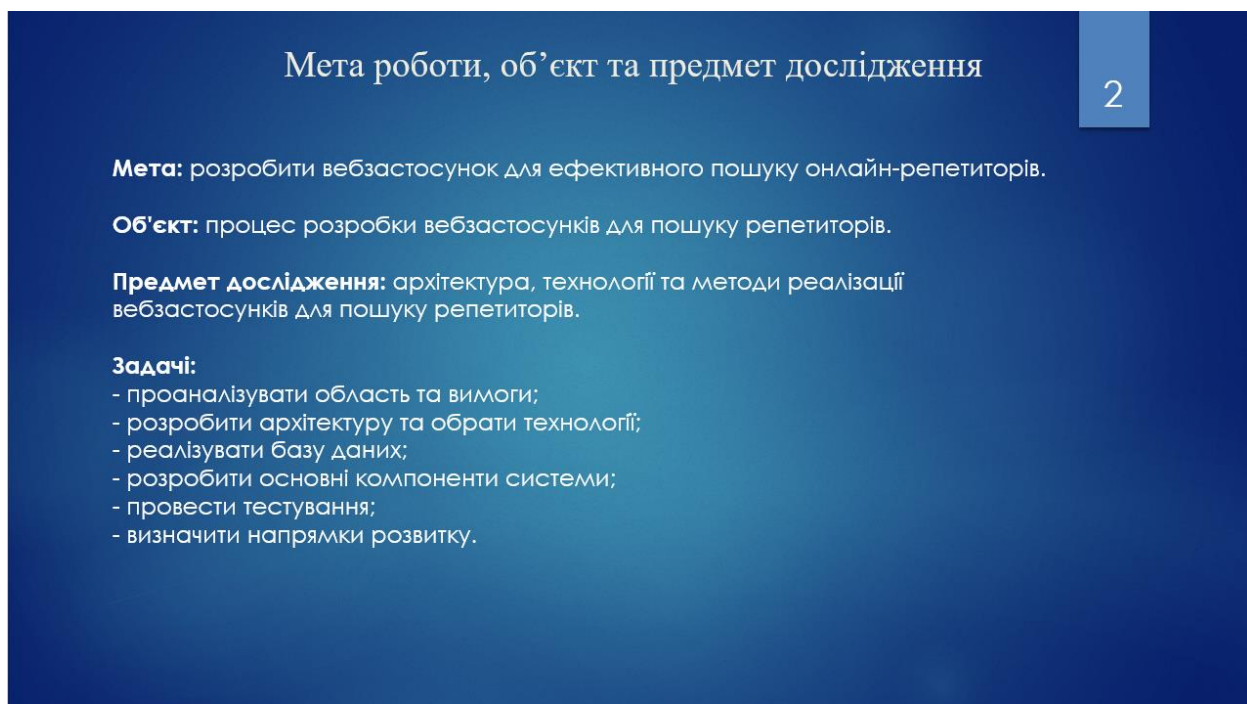


Рисунок Б.2 – Слайд 2

Порівняння існуючих аналогів

3

Характеристика	Preply	Buki	Superprof
Географічне охоплення	Міжнародне	Україна	Міжнародне
Мовна підтримка	50+ мов	Укр, рос, англ	10+ мов
Спеціалізація на предметах	Акцент на мовах	Рівномірний розподіл	Найбільша різноманітність
Форма занять	Лише онлайн	Онлайн та офлайн	Онлайн та офлайн
Верифікація репетиторів	Багаторівнева	Відгуки та рейтинги	Відгуки та рейтинги
Вбудована система занять	Так	Ні	Ні
Вбудована система оплати	Так	Ні	Ні
Внутрішній чат/відеозв'язок	Так	Ні	Ні
Комісія для репетиторів	До 33% (0% для українців)	Немає	Щомісячна абонплата
Можливість видалення профілю репетитором	Так	Лише приховати на певний термін	Ні

Рисунок Б.3 – Слайд 3

Вибір бекенд мови програмування

4

Мова програмування	Переваги	Недоліки
Python	- проста у вивченні та читанні коду; - багата екосистема бібліотек та фреймворків - динамічна типізація.	- нижча швидкодія порівняно з компільованими мовами; - глобальний інтерпретатор блокувань (GIL) в деяких версіях.
Java	- портативність (працює на різних платформах); - потужна екосистема та велика спільнота розробників; - статична типізація.	- багатослівність коду; - вища складність мови порівняно з Python.
JavaScript (Node.js)	- асинхронне програмування; - одна мова для фронтенду та бекенду; - багата екосистема пакетів (npm).	- callback hell (складність коду при вкладених викликах); - динамічна типізація (може призводити до помилок).
Ruby	- проста у читанні та написанні коду; - багата екосистема бібліотек та фреймворків (Rails); - конвенції коду (зменшує кількість рішень).	- нижча швидкодія порівняно з іншими мовами; - менша поширеність порівняно з Python, Java, JavaScript.
Go	- висока швидкодія та низька споживання пам'яті; - проста у вивченні та читанні коду; - вбудована підтримка конкурентності (горутини та канали).	- менша екосистема порівняно з іншими мовами; - відсутність деяких високорівневих функцій.

Рисунок Б.4 – Слайд 4

Вибір фронтенд мови програмування

5

Мова програмування	Переваги	Недоліки
Jinja2 (генерація HTML)	- генерація стандартної мови розмітки вебсторінок; - проста у вивченні та використанні.	- обмежена функціональність (лише структура та зміст); - відсутність динамічності та інтерактивності.
Jinja2 (генерація CSS)	- генерація стандартної мови стилізації вебсторінок; - відокремлення стилів від структури HTML.	- складність підтримки при великих проєктах; - відсутність логіки та динамічності.
JavaScript	- додає інтерактивність та динамічність вебсторінкам; - багата екосистема бібліотек та фреймворків.	- потенційні проблеми безпеки при неправильному використанні; - відмінності в реалізації між браузерами.
TypeScript	- статична типізація (зменшує кількість помилок); - покращена підтримка IDE та рефакторингу коду.	- додатковий крок компіляції перед виконанням; - потребує вивчення додаткового синтаксису та типів.
Dart	- розроблений для веб та мобільних застосунків; - можливість компіляції в JavaScript або в нативний код.	- менша екосистема порівняно з JavaScript; - потребує вивчення нової мови та інструментів.

Рисунок Б.5 – Слайд 5

Вибір вебсерверу

6

Вебсервер	Переваги	Недоліки
Amazon EC2	- широка підтримка та документація; - багато модулів та розширень.	- вища споживання ресурсів порівняно з деякими альтернативами; - складність налаштування для високонавантажених проєктів.
Nginx	- висока продуктивність та низьке споживання ресурсів; - ефективне обслуговування статичних файлів.	- менша кількість модулів та розширень порівняно з Apache; - складніша конфігурація для динамічних застосунків.
Microsoft IIS	- тісна інтеграція з платформою Windows та .NET; - графічний інтерфейс для налаштування.	- обмежена підтримка на не-Windows платформах; - менша продуктивність порівняно з деякими альтернативами.
Lighttpd	- легкий та швидкий вебсервер; - ефективне обслуговування статичних файлів.	- менша кількість функцій та розширень порівняно з іншими; - менша популярність та підтримка спільноти.
Caddy	- автоматичне налаштування HTTPS за замовчуванням; - проста конфігурація через файл Caddyfile.	- менша кількість функцій та розширень порівняно з іншими; - менша популярність та екосистема порівняно з іншими.

Рисунок Б.6 – Слайд 6

Вибір сервісу для побудови бази даних

7

База даних	Переваги	Недоліки
PostgreSQL	- потужна та надійна СУБД з відкритим кодом - підтримка розширених функцій (індекси, тригери, процедури); - хороша масштабованість та продуктивність.	- вища складність налаштування порівняно з деякими альтернативами; - вищі вимоги до ресурсів порівняно з легшими БД.
MySQL	- широка популярність та підтримка; - простота використання та налаштування; - велика кількість інструментів та ресурсів.	- обмежена підтримка деяких розширених функцій; - проблеми з продуктивністю при високих навантаженнях.
Oracle Database	- потужна та надійна комерційна СУБД; - багата функціональність та можливості; - підтримка великих корпоративних систем.	- висока вартість ліцензій та підтримки; - складність налаштування та адміністрування.
Microsoft SQL Server	- тісна інтеграція з платформою Windows та .NET; - багатий набір інструментів та функцій.	- обмежена підтримка на не-Windows платформах; - вища вартість ліцензій порівняно з відкритими альтернативами.
MongoDB	- високопродуктивна нереляційна (NoSQL) база даних; - гнучка схема даних (відсутність фіксованої структури); - масштабованість та розподіленість.	- відсутність підтримки транзакцій та зв'язків між колекціями; - обмежена підтримка складних запитів та агрегацій.
Redis	- висока продуктивність як сховище ;ключ-значення в пам'яті - підтримка різних структур даних (рядки, списки, хеші); - проста масштабованість через реплікацію та шардинг.	- обмежена підтримка складних структур даних; - відсутність підтримки складних запитів та транзакцій.

Рисунок Б.7 – Слайд 7

Структурна схема архітектури вебзастосунку

8



Рис. 1 – Схема архітектури вебзастосунку

Рисунок Б.8 – Слайд 8

ER-діаграма бази даних вебзастосунку

9

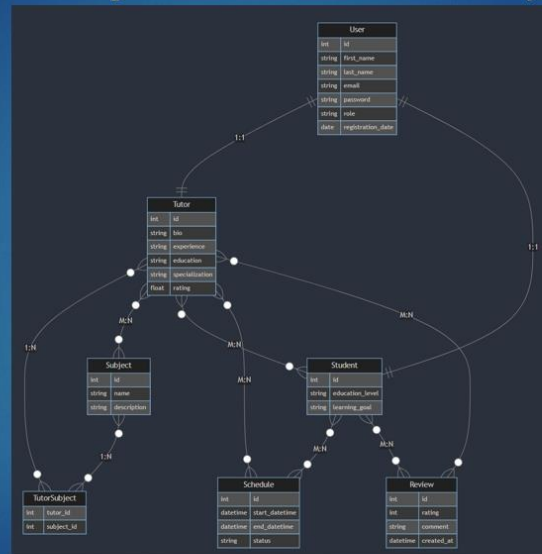


Рис. 2 - ER-діаграма сутностей та атрибутів БД застосунку

Рисунок Б.9 – Слайд 9

Реалізація основних компонентів системи

10

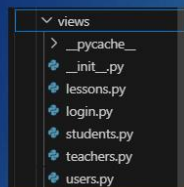


Рис. 3 – Модуль представлень та його класи

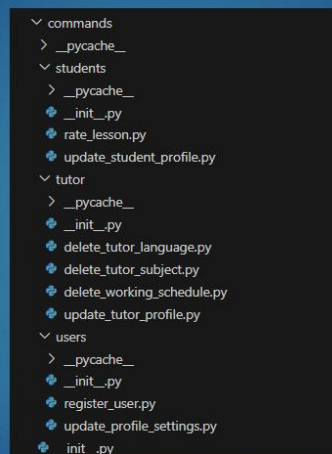


Рис. 4 – Модуль команд та його класи

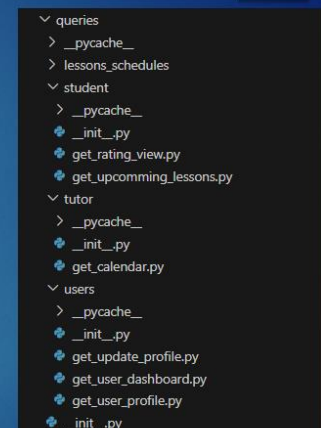


Рис. 5 – Модуль запитів та його класи

Рисунок Б.10 – Слайд 10

Інтерфейси застосунку

11

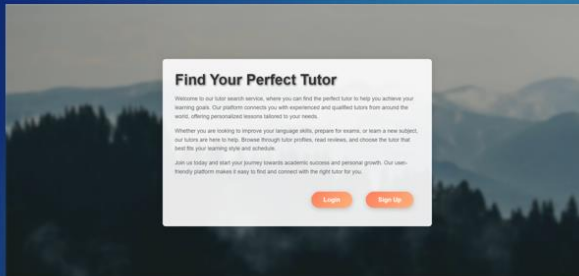
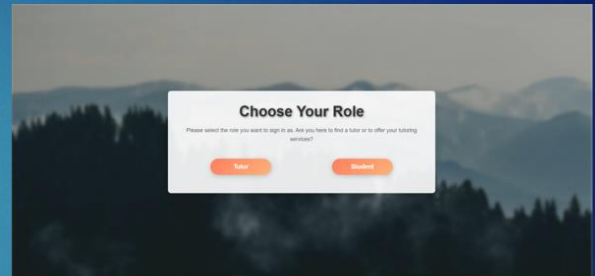
Рис. 6 – Сторінка
автентифікаціїРис. 7 – Сторінка вибору
ролі при реєстрації

Рисунок Б.11 – Слайд 11

Інтерфейси застосунку

12

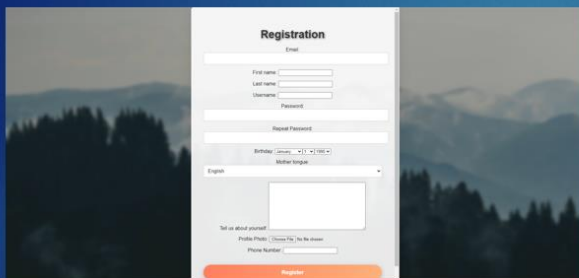
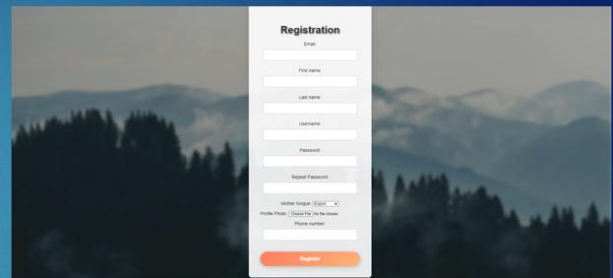
Рис. 8 – Сторінка форми
реєстрації викладачаРис. 9 – Сторінка форми
реєстрації учня

Рисунок Б.12 – Слайд 12

Інтерфейси застосунку

13

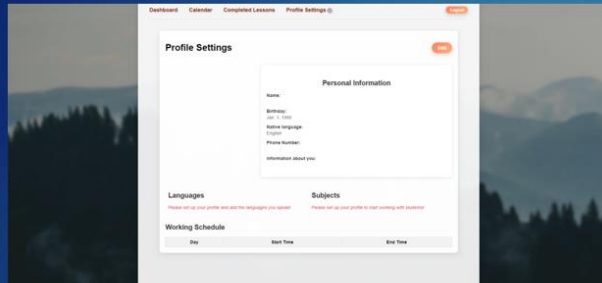


Рис. 10 – Сторінка профілю репетитора

Рис. 11 – Форма налаштувань профілю репетитора

Рисунок Б.13 – Слайд 13

Інтерфейси застосунку

14

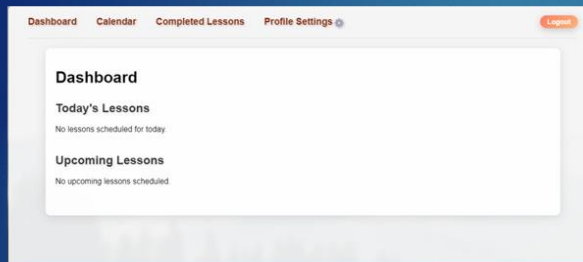


Рис. 12 – Головна сторінка особистого кабінету репетитора

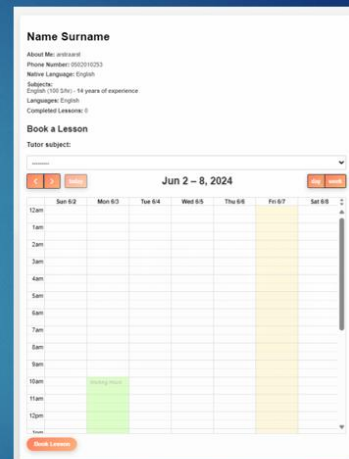


Рис. 13 – Сторінка відображення детальної інформації про репетитора

Рисунок Б.14 – Слайд 14

Інтерфейси застосунку

15

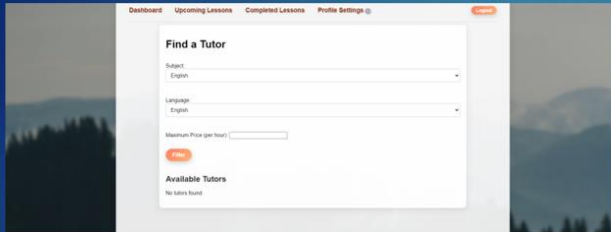


Рис. 14 – Головна сторінка
пошуку репетиторів

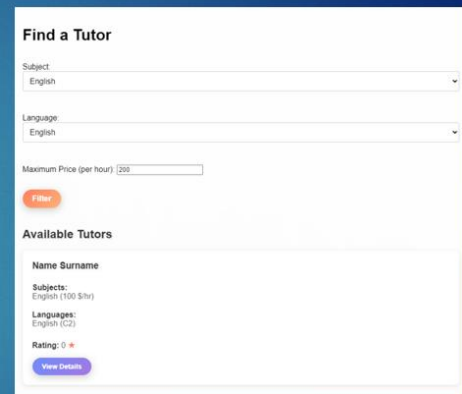


Рис. 15 – Відображення списку репетиторів
відфільтрованих за певними критеріями

Рисунок Б.15 – Слайд 15

Інтерфейси застосунку

16

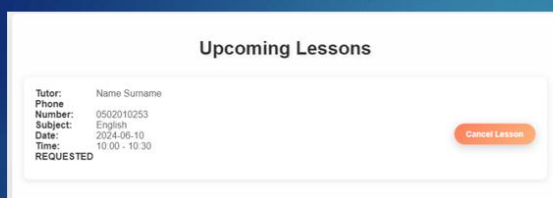


Рис. 16 – Сторінка
відображення списку
майбутніх занять

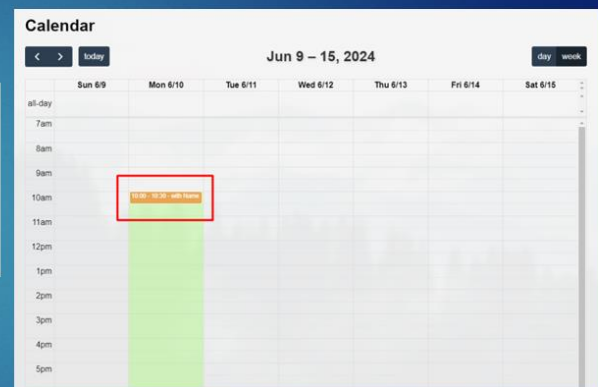


Рис. 17 – Сторінка відображення розкладу
репетитора, червоним виділено заплановане заняття

Рисунок Б.16 – Слайд 16

Інтерфейси застосунку

17

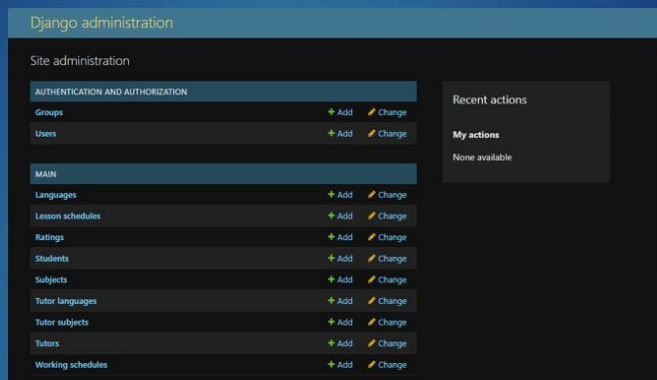


Рис. 18 – Сторінка адміністративної панелі та
модерації Django

Рисунок Б.17 – Слайд 17

Тестування застосунку

18

Функціональне тестування:

- Модульне - Так
- Інтеграційне - Так
- Системне - Так
- Приймальне - Так

Тестування безпеки:

- Автентифікації та авторизації - Так
- Валідації та фільтрації вхідних даних - Так
- Керування сесіями - Так
- Шифрування даних - Так

Тестування продуктивності та масштабованості:

- Визначення ключових показників продуктивності - Так
- Навантажувальне тестування - Так
- Профілювання коду - Так
- Оптимізація запитів до БД - Так
- Вертикальне масштабування - Так
- Масштабування БД - Так
- Кешування та оптимізація ресурсів - Так
- Відмовостійкість та збої - Так

Рисунок Б.18 – Слайд 18

Висновки

19

Вирішено актуальну проблему розробки вебзастосунку для пошуку репетиторів, який забезпечує зручний та ефективний пошук викладачів для проведення індивідуальних занять онлайн.

Проаналізовано предметну область, визначено вимоги, розроблено архітектуру на основі MVC з Django та Python, спроєктовано та реалізовано базу даних з PostgreSQL.

Реалізовано основні компоненти: реєстрація та автентифікація користувачів, особистий кабінет репетитора, пошук репетиторів, бронювання занять, адміністративна панель.

Проведено ретельне тестування функціональності, безпеки, продуктивності та масштабованості. Вебзастосунок відповідає вимогам та демонструє високу ефективність.

Визначено напрямки подальшого розвитку: розширення функціональності, покращення користувацького досвіду, маркетингові можливості, безпека, географічне охоплення, аналітика. Результати мають практичну цінність і можуть бути використані для подальшого розвитку онлайн-платформ у сфері додаткової освіти.

Рисунок Б.19 – Слайд 19