ВСТУП

У сучасних умовах розвитку цифрових технологій вивчення іноземних мов дедалі частіше відбувається в онлайн-форматі. Зростає кількість мовних шкіл, курсів та платформ, що потребують ефективних інструментів комунікації між викладачами та студентами. Особливо це актуально в умовах дистанційного та змішаного навчання, де якісне цифрове середовище є основою результативного освітнього процесу.

Більшість сучасних рішень охоплюють лише окремі аспекти взаємодії, що створює потребу у створенні комплексного веб-сервісу, адаптованого до конкретних освітніх умов. Відтак розробка веб-застосунку для підтримки комунікації в мовних курсах є актуальною інженерною задачею.

Об’єкт дослідження – система онлайн-комунікації в освітньому середовищі.

Предмет дослідження – методи та засоби створення веб-сервісу для комунікації між студентами та викладачами іноземних мов із використанням Python і фреймворку Django.

Мета роботи полягає у створенні функціонального веб-сервісу для ефективної взаємодії учасників навчального процесу з урахуванням вимог доступності, захисту даних, масштабованості та зручності використання.

Завданням роботи є аналіз інформативних ознак веб-сервісу для вивчення іноземної мови та потреб користувачів, проєктування структури системи, реалізація серверної та клієнтської частин, а також перевірка її працездатності.

Методи дослідження включають порівняльний аналіз аналогів, моделювання інформаційних систем та програмування з використанням Django.

Практичне значення роботи полягає у створенні веб-сервісу, який може бути впроваджений у мовні школи, заклади післядипломної освіти або на онлайн-курсах для організації навчального процесу та комунікації між його учасниками.

1 АНАЛІЗ НОРМАТИВНО-ДОКУМЕНТАЛЬНОГО ЗАБЕЗПЕЧЕННЯ ТА ІНФОРМАТИВНИХ ОЗНАК ВЕБ-СЕРВІСУ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ

1.1 Загальна характеристика інформаційної системи

Розроблювана система є веб-застосунком, призначеним для підтримки процесу організації та проведення курсів іноземних мов. Вона належить до категорії інформаційних систем освітнього призначення та реалізує комплексну взаємодію між студентами, викладачами та адміністраторами курсів.

Система забезпечує:

- реєстрацію та автентифікацію користувачів з розмежуванням ролей;
- створення, перегляд та управління навчальними курсами;
- організацію навчального процесу через призначення домашніх завдань і проведення онлайн-тестування;
- зберігання, обробку та перегляд результатів навчальної діяльності;
- інтегрований чат-форум для комунікації між учасниками курсу;
- бронювання занять та управління розкладом;
- систему сповіщень для оперативного інформування про зміни в курсах, новини та нагадування;
- зручний користувацький інтерфейс з адаптивною верстkою.

Застосунок реалізований з використанням фреймворку Django та мови програмування Python, що гарантує масштабованість, безпеку та підтримку сучасних веб-технологій. Його функціонал спрямований на покращення комунікації між учасниками освітнього процесу, забезпечення доступу до актуального навчального контенту, а також контроль за навчальним прогресом студентів.

Система відповідає актуальним вимогам до цифрових освітніх платформ і може бути легко інтегрована у структуру мовної школи чи освітнього центру з дистанційною формою навчання.

1.2 Нормативне, законодавче та технічне забезпечення

1.2.1 Законодавчі акти України

Закон України «Про Освіту» – визначає правові засади функціонування освітніх установ, підтримує інтеграцію цифрових інструментів у навчальний процес, регламентує використання електронних платформ для підвищення якості освіти. [15]

Закон України «Про захист персональних даних» – регламентує обробку, зберігання та захист персональних даних користувачів вебзастосунку, включаючи імена, контактні дані, результати тестів, домашні завдання тощо. [16]

Закон України «Про елктрону комерцію» – встановлює правові засади здійснення електронних правочинів, укладення договорів онлайн, а також регламентує оплату навчальних послуг через інтернет. [17]

1.2.2 Міжнародні стандарти та технічна документація

CEFR (Common European Framework of Reference for Languages) – визначає міжнародні стандарти рівнів володіння іноземною мовою. У застосунку рівень курсу може бути вказаний відповідно до цієї шкали. [19]

WCAG 2.1 (Web Content Accessibility Guidelines) – рекомендації щодо доступності вебконтенту для людей з порушеннями зору, слуху чи моторики. Вебсервіс частково дотримується цих принципів (контрастність, адаптивність, семантична структура). [18]

ISO/IEC 25010 – стандарт, що визначає вимоги до якості програмного забезпечення: функціональність, зручність використання, надійність, продуктивність, безпека тощо. Під час розробки вебсервісу враховано ці принципи як орієнтир. [20]

1.3 Інформативні ознаки веб-сервісу

1.3.1 Персональні дані користувача

У системі реєструються облікові записи користувачів з наступними персональними атрибутами:

- Ім’я та прізвище: використовуються для ідентифікації у профілі, в журналі відвідувань, чаті та рейтингах;
- Email: основний засіб автентифікації та комунікації;
- Роль у системі: передбачено дві основні ролі: студент та викладач. Вони визначають доступ до відповідних сторінок і функцій;
- Фото профілю: опціонально, завантажується через особистий кабінет;
- Дата реєстрації: фіксується автоматично при створенні акаунту;

1.3.2 Навчальні параметри

Кожен користувач має навчальний контекст, пов’язаний із курсами та результатами. До навчальних ознак належать:

- Назва та опис курсу: збережено у відповідній моделі, відображається у списку курсів і профілі користувача;
- Формат занять: система підтримує індивідуальні заняття з можливістю бронювання;
- Розклад занять: реалізовано через окремий модуль бронювання, де вказується викладач, дата, час;
- Домашні завдання: створюються викладачем, виконуються студентом і мають поле для коментаря;
- Тестові завдання: реалізовано функціонал проходження тестів, з фіксацією результату, дати та кількості спроб;

- Прогрес студента: автоматично фіксується у вигляді кількості зданих домашніх завдань і пройдених тестів;
- Оцінки та коментарі викладача: зберігаються у розділі домашніх завдань та тестів.

1.3.3 Форум

Веб-сервіс включає елементи, що забезпечують взаємодію між студентами та викладачами:

- Форум курсу: реалізовано функціонал обговорень у межах курсу.
- Сповіщення: повідомлення про нові завдання, тести, бронювання, які формуються автоматично і відображаються у відповідному шаблоні;
- Стан викладача: за активністю викладача на платформі можна оцінити його участь (оновлення курсів, оцінювання тощо);
- Можливість залишати відгуки/коментарі: поки реалізовано у вигляді простого поля, яке можна використовувати як відгук до завдань.

1.4 Зв’язки між об’єктами системи

У веб-застосунку "Language School" реалізовано кілька основних сутностей, між якими встановлено логічні зв’язки, що забезпечують узгоджену взаємодію між модулями навчання, користувацькими ролями та інтерфейсами. Основні зв’язки описано нижче:

- Один студент може бути зарахований на кілька курсів. Зв’язок реалізується через проміжну таблицю між моделями User і Course. У такий спосіб можна вести облік студентів, що навчаються на різних курсах одночасно.

- Кожен викладач може створювати й викладати кілька курсів. Один користувач з роллю teacher може бути пов’язаний як автор/викладач із багатьма записами моделі Course.
- Кожен курс може містити кілька домашніх завдань і тестів. Моделі Homework та Test мають зовнішній ключ до курсу, що дозволяє групувати навчальні матеріали за напрямами.
- Кожне заняття пов’язане з одним викладачем і студентом або декількома студентами. Модель BookingSlot реалізує зв’язок з користувачами-студентами та викладачами, а також містить дату й час заняття.
- Кожен тест містить кілька запитань, реалізовано через зв’язок Test → Question.
- Домашнє завдання має зв’язок із конкретним студентом, курсом та файлом відповіді. Модель HomeworkSubmission поєднує студента, відповідь і статус виконання.
- Форум курсу прив’язаний до конкретного курсу. Зв’язки: ForumMessage → Course.
- Система сповіщень зв’язана з користувачем. Кожне повідомлення містить поле отримувача та тип повідомлення (тест, завдання, бронювання).
- Користувач має один профіль, у якому зберігається додаткова інформація (ім’я, фото, роль), реалізовано через модель Profile, пов’язану з User.

1.5 Зразки документів

У веб-сервісі реалізовано низку електронних документів та форм, які забезпечують повний цикл взаємодії користувача із системою: від реєстрації до моніторингу навчального прогресу. Нижче наведено основні приклади таких документів.

1.5.1 Електронна форма реєстрації користувача

При створенні облікового запису користувач заповнює форму, яка включає такі поля:

- Ім’я: текстове поле;
- Прізвище: текстове поле;
- Email: унікальна адреса користувача;
- Пароль: конфіденційне поле з підтвердженням;
- Роль: вибір зі списку (студент / викладач);

Форма валідується на стороні сервера та забезпечує CSRF-захист.

1.5.2 Розклад занять

Доступний як для студентів, так і для викладачів у відповідному розділі. Реалізований у вигляді таблиці, яка містить:

- Дата заняття;
- Час;
- Викладач / Студент (відповідно до ролі перегляду);
- Посилання на зустріч;
- Контактну нформацю;
- Статус.

1.5.3 Форум курсу

У рамках форуму зберігаються записи:

- Автор повідомлення;
- Дата й час публікації;
- Вміст повідомлення;
- Відповіді на повідомлення.

Це дозволяє зберігати історію обговорень у межах навчального курсу.

1.5.4 Профіль користувача

Кожен користувач має персоналізовану сторінку профілю, яка відображає його основну інформацію та активність у системі. Профіль формується автоматично після реєстрації й може редагуватися самим користувачем.

Основні поля профілю:

- Ім’я та прізвище;
- Email;
- Роль користувача;
- Фото профілю;
- Курси;
- Рейтинг(якщо профіль викладача).

1.6 Опис предметної області

На ринку існує широкий спектр веб-застосунків і мобільних платформ, які дозволяють вивчати іноземні мови в інтерактивному форматі. Більшість із них орієнтовані на самостійне навчання або заняття з репетитором, і містять функції для побудови персонального навчального плану, інтеграцію ігор, тестів, живого спілкування, тощо. Серед популярних платформ виділяються такі сервіси, як Duolingo, Babbel, Rosetta Stone, Busuu та Preply.

Однак, незважаючи на широку функціональність, існуючі рішення мають певні обмеження – більшість з них не дозволяє організувати повноцінну комунікацію між студентами та викладачами, зберігати хід навчального процесу на рівні курсів, переглядати результати тестів у зручному форматі, або об’єднувати всі елементи у спільний веб-сервіс, що підходить для локальних мовних шкіл або навчальних центрів.

Щоб створити ефективний веб-застосунок для організації навчального процесу з іноземних мов, необхідно проаналізувати найпопулярніші онлайн-сервіси, що працюють у цій сфері, та оцінити їх функціональні можливості. Аналіз таких сервісів дозволяє виявити їх переваги, недоліки та потенціал для удосконалення, який може бути врахований під час розробки нового рішення.

Згідно з оглядом порталу TechRadar[12], до найпопулярніших платформ для вивчення мов у 2024 році належать:

1. Duolingo – безкоштовна гейміфікована платформа;
2. Babbel – фокусується на реальному спілкуванні;
3. Rosetta Stone – інтегрує візуальне навчання;
4. Busuu – пропонує спілкування з носіями мови;
5. Preply – платформа для індивідуальних занять з викладачем.

Ці сервіси будуть використані для подальшого аналізу.

Наукові джерела та практика користування такими системами дозволяють сформувати ключові критерії аналізу аналогів, які враховуються при створенні власного рішення:

- формат навчання: інтерактивне, відео, репетиторство, комбіноване;
- можливість комунікації: чати, форуми, відеозв’язок;
- адаптивність: наскільки система підлаштовується під рівень і прогрес користувача;
- наявність структурованих курсів: наявність тем, уроків, завдань, перевірки знань;
- зручність інтерфейсу: логіка навігації, швидкість завантаження, адаптивний дизайн;
- доступність мовних пар: які мови підтримуються для навчання;
- можливість зворотного зв’язку з викладачем;
- вбудовані тести та оцінювання результатів;
- персоналізація навчання (нагадування, статистика, розклад занять).

Проаналізувавши предметну область, можна зробити висновок про необхідність розробки веб-сервісу, який поєднує функції управління курсами, перевірки знань, обміну повідомленнями та забезпечення індивідуального доступу до навчальних матеріалів у межах повноцінної цифрової екосистеми для локальних мовних курсів.

1.7 Огляд аналогів

У процесі розробки веб-сервісу для організації навчання іноземним мовам важливо проаналізувати існуючі рішення на ринку. Це дозволяє виявити їхні сильні та слабкі сторони, а також визначити можливості для вдосконалення.

Першим розглянемо веб-сайт Duolingo[4] (рис. 1.1), який є однією з найпопулярніших безкоштовних платформ для вивчення мов, яка пропонує гейміфікований підхід до навчання. Користувачі отримують бали за правильні відповіді, проходять рівні, змагаються з друзями та отримують віртуальні нагороди за щоденну активність.

Платформа підтримує понад 40 мов, включаючи українську, англійську, іспанську, французьку, німецьку тощо. Duolingo має веб-версію та мобільні застосунки для Android і iOS, що робить її універсально доступною. Окрім звичайного режиму, Duolingo також пропонує "Duolingo for Schools" для вчителів.

Серед основних функціональних можливостей:

- інтерфейс у вигляді гри (гейміфікація),
- повторення слів і граматики за методом інтервального повторення,
- тематичні модулі (їжа, подорожі, робота тощо),
- базова аналітика прогресу.

Користувачі часто хвалять простоту, швидкість та заохочувальний характер платформи. Проте вказують і на недоліки: занадто прості вправи, штучні переклади, обмежений живий контекст, відсутність можливості спілкування з

викладачами. Також є коментарі про часту рекламу у безкоштовній версії, яка може заважати навчанню.

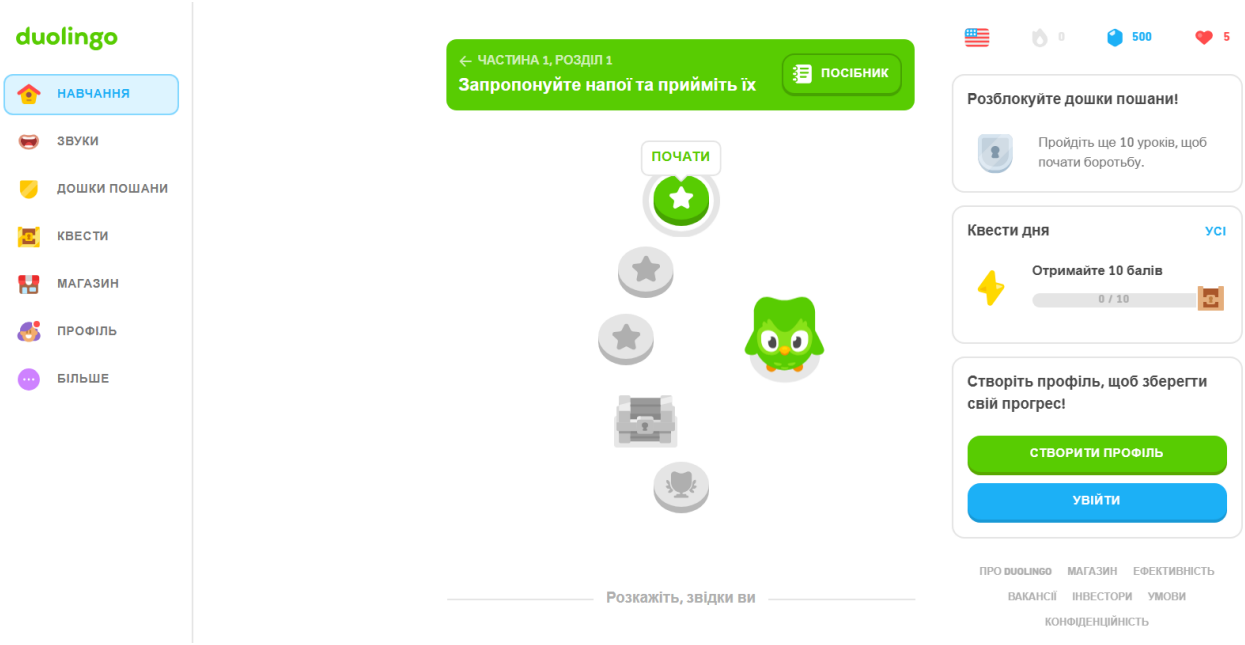


Рисунок 1.1 – Веб-сервіс вивчення мов Dualingo

Наступною розглянемо платформу Babel[2], яка зображена на рисунку 1.2. Це платний сервіс, який позиціонує себе як альтернатива Duolingo з більшим акцентом на реальне спілкування та граматику. Babbel пропонує курси, створені професійними лінгвістами, з фокусом на практичні діалоги, побудовані відповідно до рівня користувача.

Платформа пропонує:

– інтерактивні уроки тривалістю до 15 хв,
– вивчення через діалоги та вправи на вимову,
– розділи "граматика", "слова", "культура",
– синхронізацію прогресу між пристроями,
– офлайн-доступ до завантажених курсів.

Babbel доступний на комп’ютері та мобільних пристроях. Користувачі відзначають якість контенту, логічну структуру та відсутність зайвого – усе чітко, як у класичному підручнику. Недоліком є відсутність безкоштовної версії та обмеження на кількість доступних мов – лише близько 14.

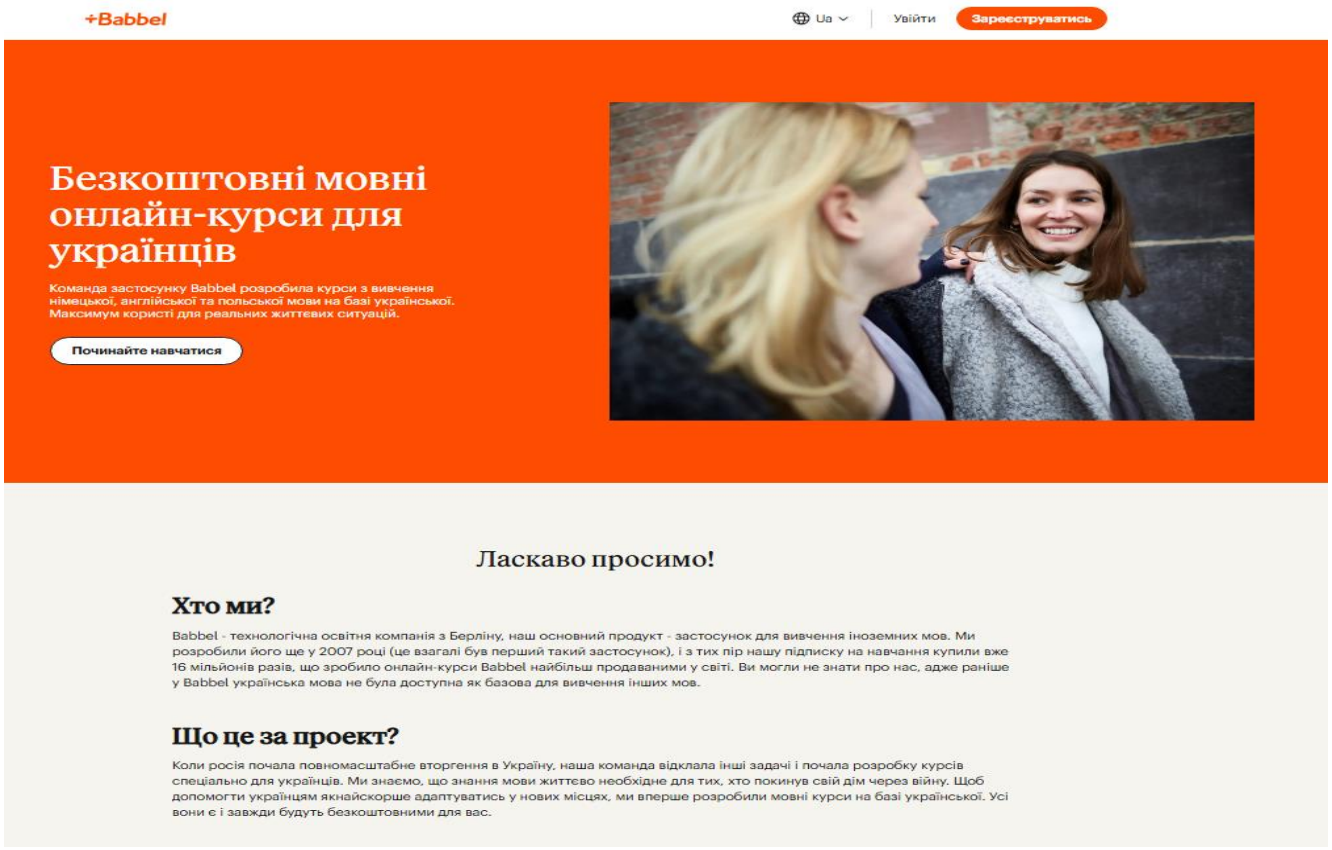


Рисунок 1.2 – Інтерфейс веб-сайту ua.babbel.com

Далі розглянемо Preply[8], зображений на рисунку 1.3. Це платформа, орієнтована на індивідуальні онлайн-заняття з викладачами. Вона функціонує як маркетплейс: студент обирає викладача за рейтингом, спеціалізацією, ціною та графіком. Після бронювання заняття проводиться через вбудований відеочат.

Preply підтримує понад 50 мов. Основний акцент – на персоналізації: користувач обирає рівень, тематику (для подорожей, бізнесу, TOEFL, IELTS тощо), кількість занять. Також реалізовано особистий кабінет, де відображаються майбутні уроки, історія занять, нотатки та домашні завдання.

Перевагами платформи є:

- реальні викладачі з досвідом,
- гнучкий графік,
- персоналізовані заняття,
- можливість підбору репетитора за мовою, країною, ціною.

Серед недоліків: відсутність безкоштовного навчання, варіативна якість викладання, необхідність самостійно організовувати навчальний процес.

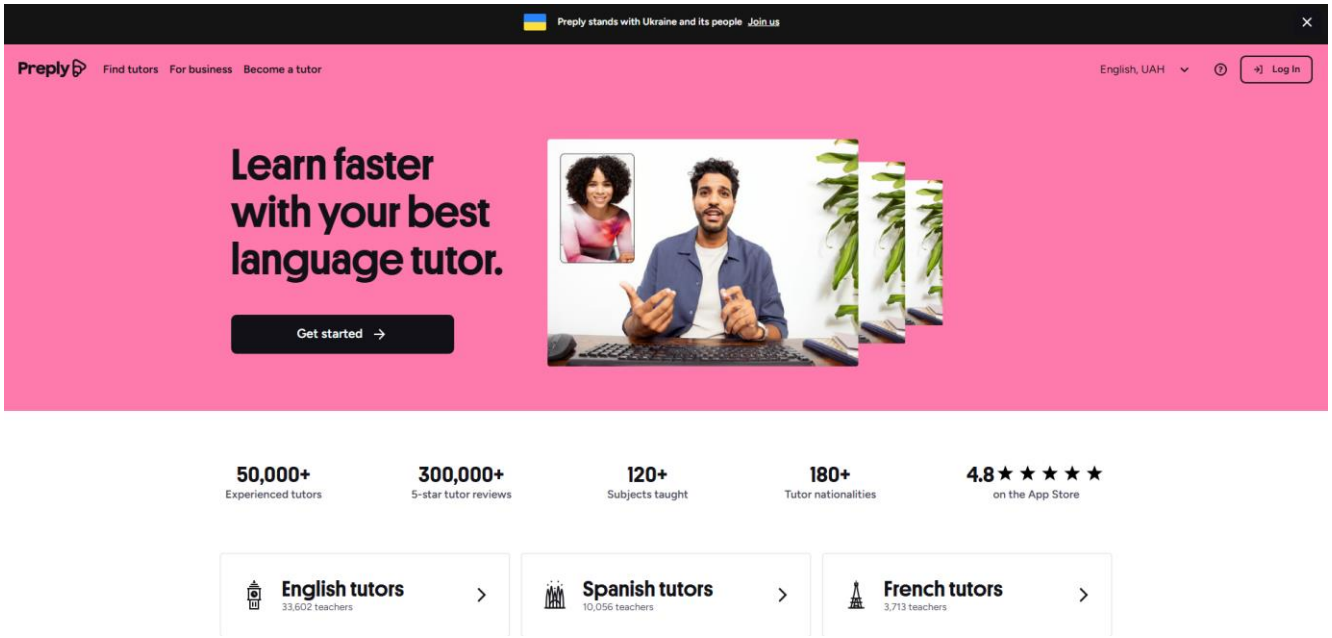


Рисунок 1.3 – Головна сторінка веб-сервісу preply.com

Наступний на огляді веб-сервіс Busuu[3], головну сторінку якого зображено на рисунку 1.4. Одним словом це мовна платформа, яка поєднує самостійне навчання з можливістю перевірки вправ реальними носіями мови. Платформа підтримує 12 мов, серед яких англійська, іспанська, французька, німецька, італійська, японська та інші. Busuu доступний як через веб-браузер, так і через мобільні додатки для iOS та Android.

Платформа пропонує такі функціональні можливості:

- структуровані курси з граматики, лексики, вимови;
- вправи на письмову та усну мову;
- персоналізовані уроки на основі цілей користувача;
- спільнота носіїв мови, які перевіряють ваші відповіді;
- інтеграція з календарем для планування уроків.

Busuu вирізняється тим, що вправи, які виконує користувач, можуть бути перевірені реальними людьми, що сприяє кращому запам’ятовуванню та практиці.

Також є функція створення індивідуального навчального плану з автоматичним нагадуванням про заняття.

Рисунок 1.4 – Вигляд веб-сервісу bussu.com

Серед переваг:

- живий контакт із носіями мови;
- адаптивність програми під цілі користувача;
- сучасний і зручний інтерфейс.

Недоліки, зазначені користувачами:

- обмежений доступ у безкоштовній версії (виправлення вправ доступне лише для преміум-акаунтів);
- деякі курси поверхневі й не охоплюють достатньо тем;
- без офлайн-режиму у базовому плані.

Тепер розглянемо сервіс Rosetta Stone[10], який є одним із найстаріших і найвідоміших брендів у галузі вивчення мов, що активно використовує метод "повного занурення", де навчання відбувається виключно мовою, яку ви вивчаєте.

Платформа підтримує понад 20 мов, серед яких основні європейські та азійські мови, включаючи китайську, арабську, японську.

Основні функції Rosetta Stone:

- методика іммерсії: усі слова, фрази, речення – без перекладу;
- вправи на вимову з розпізнаванням мови (технологія TruAccent);
- курси для мобільних пристроїв та ПК з синхронізацією прогресу;
- розмовні клуби та заняття з викладачем (у преміум-версії);
- спеціалізовані курси для бізнесу.

Користувачі позитивно оцінюють:

- природне засвоєння мови без перекладів;
- акцент на вимову та усне мовлення;
- стабільну роботу застосунку.

Недоліки:

- висока вартість підписки (особливо для кількох мов);
- повна відсутність пояснень рідною мовою;
- підходить не всім – потребує більше часу та самодисципліни.

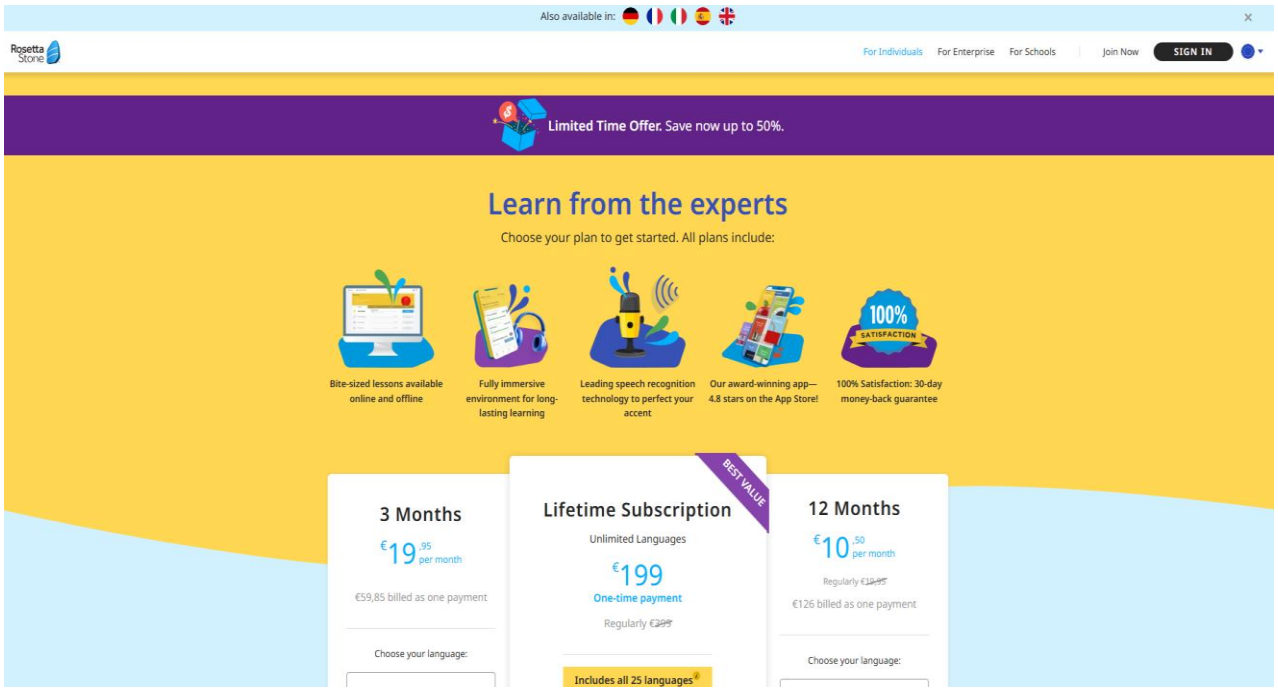


Рисунок 1.5 – Інтерфейс веб-застосунку rosettastone.com

З проведеного аналізу випливає, що хоча більшість платформ пропонують зручне й доступне навчання, жодна з них не забезпечує повноцінного середовища для організованого навчального процесу в групі, із централізованим керуванням курсами, інтегрованим тестуванням, обліком результатів та активною комунікацією між студентами та викладачами. Це створює потребу у створенні власного веб-сервісу, який об’єднає переваги вказаних платформ та усуне виявлені обмеження.

1.8 Постановка завдання

Метою роботи є розробка інформаційного веб-сервісу комунікації учасників курсів іноземних мов засобами Python та фреймворку Django. Система має забезпечити зручну та безпечну взаємодію між студентами, викладачами та адміністраторами, а також функціонально підтримувати навчальний процес у межах мовних курсів.

Завдання роботи:

1. Провести аналіз предметної області та огляд аналогічних мовних платформ, виявити їх сильні й слабкі сторони.
2. Визначити функціональні та нефункціональні вимоги до веб-сервісу, враховуючи специфіку освітнього середовища.
3. Розробити архітектурну модель системи, включно з базою даних, логікою взаємодії та інтерфейсними компонентами.
4. Реалізувати прототип веб-сервісу з ключовими модулями: реєстрація, автентифікація, управління курсами, обмін повідомленнями, тести та інше.
5. Виконати перевірку працездатності сервісу, оцінити його з точки зору продуктивності, доступності та зручності використання.
6. Підготувати технічну документацію, яка описує функціонування, структуру та можливості подальшого розвитку веб-сервісу.

Згідно інтерв'ю з Іваном Примаченком, співзасновником платформи Prometheus, опублікованим на сайті Укрінформ, кількість активних слухачів на платформі зросла на 40% порівняно з довоєнним періодом. Це свідчить про значне збільшення попиту на онлайн-освіту в Україні.[11]

Аналіз показує, що платформи Duolingo, Preply, Babbel, Busuu, Rosetta Stone – хоч і мають потужну функціональність, не поєднують усіх потрібних інструментів для повноцінної навчальної взаємодії в організованих курсах. Жодна з них не пропонує внутрішню систему управління курсами, інтегроване обговорення, облік прогресу й автоматизоване тестування у межах освітньої організації [12].

Отже, проєкт буде реалізовано як трикомпонентна система:

- Клієнтська частина: відповідає за користувацький інтерфейс, побудований з використанням шаблонів Django (HTML/CSS/JS). Забезпечує доступ до сторінок курсів, чатів, тестів, розкладу занять тощо.
- Серверна частина: реалізована на фреймворку Django, відповідає за обробку запитів, логіку взаємодії користувачів, авторизацію та передачу даних між модулями.
- База даних: забезпечує зберігання інформації про користувачів, ролі, курси, тести, повідомлення, результати навчання.

Функціональні вимоги до веб-сервісу:

- Реєстрація та авторизація користувачів з розподілом ролей: студент, викладач, адміністратор.
- Можливість створення та перегляду курсів, приєднання до них.
- Система чату для обміну повідомленнями між викладачем і студентами.
- Проведення тестів з автоматичною перевіркою результатів.
- Модуль завдань з прикріпленням файлів і переглядом оцінок.
- Календар занять та сповіщення.
- Особистий кабінет для перегляду прогресу, результатів, повідомлень.

Нефункціональні вимоги:

- Інтуїтивно зрозумілий і адаптивний інтерфейс.
- Підтримка мобільних пристроїв (через адаптивний дизайн).
- Безпека: автентифікація, захист персональних даних.
- Масштабованість: можливість легко розширити функціонал або інтегрувати сторонні сервіси.
- Продуктивність: система повинна забезпечувати швидку реакцію на запити користувача.

У цьому розділі було сформульовано цілі, задачі та архітектуру майбутнього веб-сервісу. Також визначено перелік функціональних і нефункціональних вимог. Створення веб-сервісу забезпечить гнучке керування освітнім процесом, об’єднає комунікацію між учасниками та дозволить ефективно організувати онлайн-навчання мовам у межах навчальних курсів.

2 РОЗРОБКА СТРУКТУРИ ВЕБ-СЕРВІСУ

2.1 Функціональні вимоги до веб-сервісу

Проаналізувавши предметну область та існуючі рішення, було визначено потребу у створенні веб-сервісу, який би забезпечував не лише процес вивчення іноземних мов, а й повноцінну комунікацію між усіма учасниками навчального процесу, тобто студентами, викладачами та адміністраторами.

З урахуванням цього, проект буде реалізовано у вигляді трикомпонентної архітектури:

- Клієнтська частина відповідає за відображення інтерфейсу у веб-браузері користувача. Саме тут студент або викладач взаємодіє з системою: переглядає курси, надсилає повідомлення, проходить тести або читає новини. Інтерфейс розробляється з використанням HTML, CSS і JavaScript, а також Django-шаблонів. Він забезпечує взаємодію з сервером через HTTP-запити, передачу форм, повідомлень та інших подій.
- Серверна частина, побудована на фреймворку Django (Python), відповідає за логіку обробки запитів, авторизацію, збереження інформації в базі даних, рендеринг шаблонів, а також реалізацію API-запитів до інших систем (наприклад, для надсилання електронних сповіщень).
- База даних (SQLite або PostgreSQL) містить усі структуровані дані: інформацію про користувачів, ролі, курси, завдання, результати тестів, повідомлення, сповіщення та інше. Вона підтримує реєстрацію, зв’язки між курсами та викладачами, зберігає історію тестів та чатів.

Основні функціональні вимоги до веб-сервісу:

- Реєстрація та автентифікація користувачів із розмежуванням ролей (студент або викладач);
- Перегляд і створення курсів, приєднання до курсу;

- Надсилання повідомлень між користувачами в чат-форумі курсу;
- Створення і проходження тестів з автоматичною перевіркою;
- Призначення домашніх завдань з можливістю завантаження файлів;
- Відображення оцінок, результатів тестування та виконаних завдань;
- Виведення сповіщень;
- Календар занять і сповіщення про зміни у розкладі;
- Кабінет користувача з персональною інформацією, курсами та повідомленнями.

Нефункціональні вимоги до веб-сервісу:

- Інтуїтивно зрозумілий інтерфейс українською мовою;
- Адаптивність: підтримка перегляду на мобільних телефонах, планшетах та ПК;
- Швидкість роботи: сторінки мають завантажуватись менш ніж за 2 секунди при стандартному з'єднанні;
- Безпека: автентифікація користувачів, захист персональних даних, уникнення XSS та CSRF-атак;
- Масштабованість: можливість легко додати нові модулі (відеоконференції, опитування, новини та інше);
- Резервне копіювання: регулярне збереження даних бази в окремі резервні копії.

У результаті реалізації поставлених функціональних вимог веб-застосунок надасть ефективний інструмент для керування курсами, спілкування між учасниками та підтримки повноцінного освітнього процесу.

2.2 UML-діаграми

UML (Unified Modeling Language) – уніфікована мова моделювання, що використовується розробниками програмного забезпечення для візуалізації процесів та роботи систем.

Це не мова програмування, скоріше набір правил та стандартів для створення діаграм. Вони дозволяють розробникам програмного забезпечення та інженерам «говорити однією мовою», не заглиблюючись у фактичний код свого продукту. Складання діаграм за допомогою UML – це чудовий спосіб допомогти іншим швидко зрозуміти складну ідею чи структуру[1].

Загалом, діаграма варіантів використання (use-case діаграма) слугує для визначення вимог до системи та розуміння взаємодії між користувачами (акторами) і функціональними можливостями веб-сервісу. Вона дозволяє описати основні сценарії використання системи з позиції зовнішніх учасників, таких як студенти, викладачі та адміністратори.

У розроблюваній системі беруть участь такі актори:

- Користувач – гість, який ще не авторизувався;
- Студент – авторизований користувач, який проходить курси;
- Викладач – користувач, який створює та веде курси;
- Адміністратор – відповідає за затвердження курсів.

Варіанти використання:

1. Реєстрація
2. Авторизація (
3. Запис на курс
4. Запис на заняття
5. Перегляд курсу
6. Проходження тестів
7. Завантаження домашньої роботи (
8. Чат-спілкування у форумі
9. Створення слоту заняття
10. Запит на створення курсу
11. Створення домашнього завдання та тестів
12. Оцінювання домашнього завдання
13. Редагування курсу

14.Підтвердження курсу

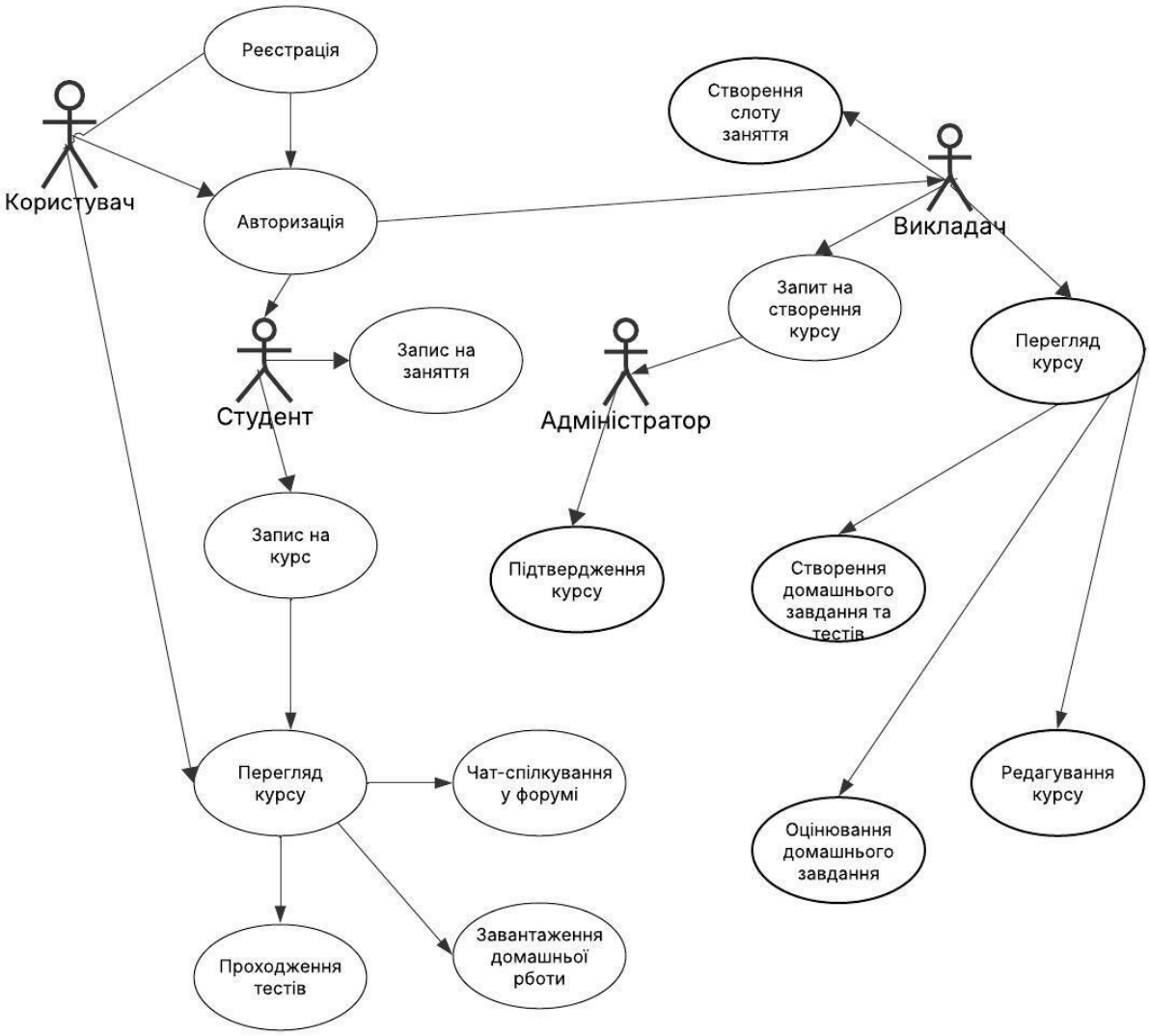


Рисунок 2.1 – Use-case діаграма

На рисунку 2.1 зображено use-case діаграму, де взаємодія між акторами і функціями системи відбувається так:

- Користувач може зареєструватися та авторизуватись у системі.
- Студент після авторизації може записуватись на курс і заняття, переглядати курс, проходити тести, надсилати домашні роботи та спілкуватися у форумі.

- Викладач має можливість подавати запит на створення курсу, створювати слоти занять, домашні завдання та тести, оцінювати роботи студентів і редагувати курс.
- Адміністратор виконує функцію затвердження курсів після їх створення викладачами.

Ця діаграма дозволяє узагальнити всі основні сценарії роботи користувачів з системою та є основою для подальшого моделювання структури, логіки й інтерфейсу веб-сервісу.

2.3 Архітектура веб-сервісу

В основі розробки веб-сервісу комунікації учасників курсів іноземних мов використано класичну клієнт-серверну архітектуру, що передбачає чіткий розподіл між клієнтською частиною (інтерфейсом користувача) та серверною частиною (логікою обробки даних).

Застосунок реалізовано з використанням фреймворку Django, який ґрунтується на шаблоні MTV (Model-Template-View). Вся логіка взаємодії між користувачем і системою відбувається через HTTP-запити, які обробляються на сервері, після чого користувач перенаправляється на нову HTML-сторінку або отримує відповідну відповідь.

У системі усі операції, включно з реєстрацією, входом, створенням курсів, проходженням тестів та надсиланням домашніх завдань, виконуються через традиційні форми та маршрути.

На рисунку 2.2 представлено загальну схему взаємодії між клієнтською та серверною частинами, яка відображає послідовність запитів і відповідей при роботі з веб-застосунком.

.

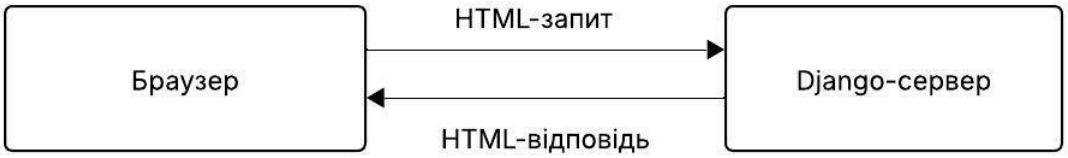


Рисунок 2.2 – Класична клієнт-серверна взаємодія у Django-застосунку

У результаті аналізу було побудовано структурну схему архітектури системи, зображену на рисунку 2.3. Система умовно поділяється на кілька основних компонентів:

- Клієнтська частина реалізована за допомогою HTML-шаблонів Django, які формуються сервером та передаються браузеру. CSS забезпечує оформлення інтерфейсу, а JavaScript використовується лише для незначних візуальних ефектів (без асинхронної взаємодії).
- Серверна частина відповідає за обробку логіки: реєстрацію, логін, створення та редагування курсів, перевірку відповідей, збереження результатів у базі даних, обробку форм і генерацію сторінок.
- База даних SQLite зберігає інформацію про користувачів, курси, заняття, тести, повідомлення, завдання та результати.

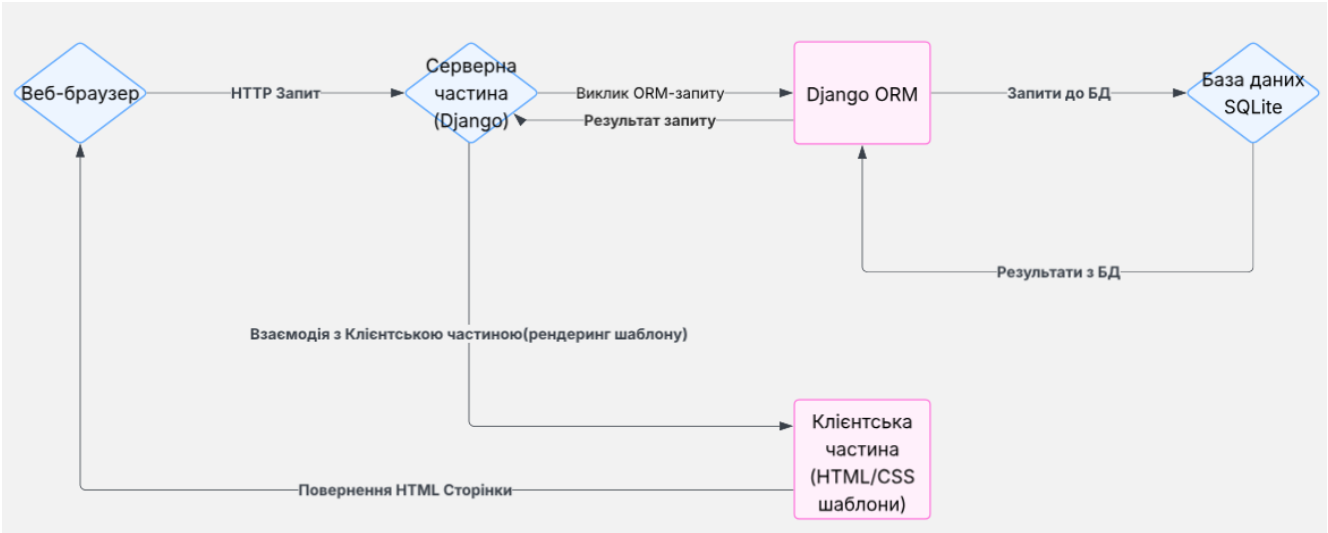


Рисунок 2.3 – Структурна схема архітектури веб-сервісу комунікації учасників курсів іноземних мов

Особливості реалізації:

1. Обробка форм реалізована за допомогою стандартного механізму Django Forms. Після надсилання форми користувач отримує відповідь у вигляді нової HTML-сторінки або повідомлення про помилку.
2. Маршрутизація забезпечується через URL-шляхи, які ведуть до відповідних представлень (views).
3. Безпека реалізована за допомогою вбудованих механізмів Django: CSRF-захист, автентифікація, обмеження доступу до функціоналу залежно від ролі користувача.
4. Підтримка ролей (гість, студент, викладач) реалізована через розширену модель користувача.

Таким чином, обрана архітектура забезпечує:

- простоту реалізації та підтримки;
- модульність коду;
- безпечну автентифікацію та зберігання даних;
- можливість масштабування функціоналу без порушення структури системи.

Дана структурна схема забезпечує стабільність, простоту реалізації та ефективну взаємодію між компонентами веб-сервісу. Усі запити користувача обробляються на сервері за допомогою фреймворку Django, після чого результати рендеряться у вигляді HTML-сторінок та повертаються клієнту. Такий підхід забезпечує передбачувану логіку, легку підтримку та розширюваність системи.

2.4 Обґрунтування вибору засобів розробки

При виборі серверної частини для розробки веб-сервісу комунікації учасників курсів іноземних мов необхідно враховувати низку важливих факторів: зручність у розробці, безпеку, продуктивність, масштабованість та підтримку з боку спільноти. Саме серверна частина відповідає за логіку обробки запитів,

взаємодію з базою даних, авторизацію користувачів та збереження результатів навчання, тому від правильного вибору технологій безпосередньо залежить ефективність функціонування всієї системи.

У сучасній веб-розробці найпоширенішими підходами до реалізації бекенду є використання таких фреймворків, як Django (Python), Node.js (JavaScript), та Spring Boot (Java). Усі вони мають свої переваги, недоліки та особливості реалізації.

З метою обґрунтованого вибору серверної технології для нашого проєкту було проаналізовано інформацію з сайтів Cloudways[9] та Geeksforgeeks[5] для декількох поширених рішень, на основі чого складено порівняльну таблицю (табл. 2.1), що охоплює ключові характеристики цих фреймворків.

Таблиця 2.1 – Порівняльна таблиця бекенд технологій

Критерій	Django	Node.js	Spring Boot
Мова	Python – проста та читаєма мова, з великою кількістю бібліотек	JavaScript – єдина мова для клієнта та сервера	Java – потужна, але складніша мова
Продуктивність	Висока для CRUD-додатків; обмежена для високонавантажених реальних часів	Висока завдяки неблокуючій архітектурі	Висока, особливо для великих корпоративних додатків
Масштабованість	Добра для середніх проєктів; масштабування можливе	Відмінна для мікросервісної архітектури	Відмінна для великих систем з мікросервісною архітектурою
Безпека	Високий рівень безпеки з вбудованими механізмами	Потребує додаткових налаштувань для забезпечення безпеки	Високий рівень безпеки з вбудованими механізмами
Швидкість розробки	Швидко за рахунок вбудованих функцій та багатограністю модулів	Середня; потребує вибору та налаштування додаткових модулів	Середня; потребує більше конфігурації

Кінець таблиці 2.1

Критерій	Django	Node.js	Spring Boot
Спільнота та підтримка	Велика спільнота, активна підтримка	Дуже велика спільнота, активна підтримка	Велика спільнота, особливо в корпоративному середовищі
Підходить для	CRUD-додатків, CMS, освітніх платформ	Реального часу, чатів, SPA	Великих корпоративних систем, фінансових додатків

Отже, на основі проаналізованої інформації та порівняння трьох бекенд технологій було вирішено обрати фреймворк Django, написаний мовою програмування Python. Саме цей фреймворк дозволить ефективно реалізувати функціонал, необхідний для нашого проєкту, тому що забезпечує швидку розробку, має потужну вбудовану ORM, систему автентифікації та адміністративну панель.

Хоча основним кандидатом для реалізації серверної частини веб-сервісу було обрано фреймворк Django, з метою перевірки доцільності цього вибору було також вирішено провести порівняльний аналіз ще одного популярного Python-фреймворку – Flask.

Flask, як мікрофреймворк, приваблює багатьох розробників своєю простотою, мінімалізмом та високою гнучкістю. Він дозволяє створювати легкі та швидкі застосунки без зайвих надбудов і є гарним вибором у випадках, коли розробник прагне мати повний контроль над структурою додатку.

Однак у нашому проєкті передбачалося реалізувати великий спектр функцій, серед яких:

- обробка аутентифікації користувачів із розмежуванням ролей;
- взаємодія з базою даних через ORM;
- створення адміністративної панелі для керування контентом;
- виведення структурованих HTML-шаблонів та обробка форм.

Такі вимоги роблять повнофункціональний фреймворк Django більш придатним, оскільки він уже містить більшість необхідних інструментів «із коробки», що значно пришвидшує розробку, зменшує кількість зовнішніх залежностей і сприяє кращій структурованості проєкту.

З огляду на це, на основі інформації з сайту Geeksforgeeks[4], було складено порівняльну таблицю (табл. 2.2), яка демонструє ключові відмінності між Django та Flask за низкою важливих характеристик.

Таблиця 2.2 – Порівняльна таблиця Django та Flask

Критерій	Django	Flask
Тип фреймворку	Повноцінний фреймворк з багатьма вбудованими функціями	Мікрофреймворк, легкий та гнучкий
Швидкість розробки	Висока завдяки вбудованим модулям	Висока для простих додатків; потребує додаткових бібліотек для складних функцій
Гнучкість	Менш гнучкий через вбудовану структуру	Висока гнучкість; розробник сам обирає компоненти
Підходить для	Великих додатків, які потребують швидкої розробки та стандартних функцій	Малих додатків, API, де потрібна максимальна гнучкість
Спільнота	Велика спільнота, багато ресурсів	Велика спільнота, активний розвиток

Аналізуючи наведені характеристики, можна ще раз зробити висновок, що хоча Flask є чудовим інструментом для реалізації невеликих або високоспеціалізованих веб-додатків, для нашого проєкту доцільніше обрати саме Django. Завдяки великій кількості вбудованих компонентів, високому рівню інтеграції, готовій системі авторизації та наявності адміністративної панелі, Django дозволяє значно прискорити розробку та уникнути потреби у підключенні сторонніх бібліотек для базових задач.

Крім того, структура Django краще підходить для підтримки масштабного коду, який включає взаємодію з базою даних, обробку форм, управління ролями користувачів і реалізацію системи сповіщень. Усе це робить Django оптимальним вибором для побудови стабільного та розширюваного веб-сервісу комунікації учасників курсів іноземних мов.

Також варто зазначити, що згідно даних з сайту Radixweb[7], Django є другим найкращим фреймворком, для написання бекенду у 2025 році. Згідно інформації з сайту його перевагами є:

- Він масштабований. Функція повторного використання коду Django спрощує для розробників налаштування зростаючого трафіку на вебсайті.
- Він SEO-орієнтований. Веб-сайти, створені на його основі, легко оптимізувати та зробити SEO-дружніми. Будучи популярним фреймворком, він має сильну спільноту, готову підтримати будь-кого, хто користується цим інструментом.
- Хоча Django – це фреймворк, що не використовує код, він орієнтований на роботу з ним без коду та має кілька можливостей, що працюють без коду. Існує кілька пакетів, які програмісти можуть використовувати, навіть не пишучи коду.
- Він пропонує інтегровані функції безпеки веб-застосунків , такі як захист від SQL-ін'єкцій, захист XSS, хешування паролів тощо. Тим самим захищаючи ваш веб-застосунок від зловмисних атак хакерів.

Django вважається найбільш підходящим для створення веб-сайту на основі баз даних, і це один з найбільш використовуваних бекенд-фреймворків.[7]

Отже, враховуючи наведені дані, актуальні галузеві рейтинги та відповідність функціональних можливостей Django потребам нашого проєкту, вибір саме цього фреймворку є цілком виправданим і стратегічно обґрунтованим рішенням.

2.5 Розробка структури бази даних веб-сервісу комунікації учасників курсів іноземних мов

Основою складовою веб-сервісу є база даних, що забезпечує зберігання, обробку та взаємодію користувачів із даними системи. Оскільки проєкт передбачає поділ на ролі і різнорівневий доступ до функцій, а також можливість бронювання занять, чатів та розділів із завданнями, вибір підходящої по структурі схеми бази даних є критичним.

Для реалізації в проєкті була обрана реляційна модель бази даних, реалізована за допомогою ORM Django. Схема узгоджується з класичною структурою ролевої системи: Гість, Студент, Викладач.

З метою формалізації структури інформаційної системи було побудовано логічну модель бази даних у вигляді ER-діаграми, яка наочно демонструє основні сутності, їх атрибути та взаємозв'язки. Це дозволяє виявити ключові елементи системи, встановити зв'язки між таблицями та визначити цілісність і послідовність даних, що зберігаються в БД.

ER-діаграма відображає основні сутності системи та зв'язки між ними. Усього є 24 таблиці, але основними є:

- auth_user: зберігає загальні дані про користувачів;
- accounts_profile: додаткова інформація про профіль (роль, аватар, біо);
- course_course: дані про курси;
- course_test, course_question, course_answer, course_testresult: структура тестів і відповідей;
- course_homework, course_homeworksubmission: завдання та їх подання;
- bookings_bookingslot, bookings_bookingslot_students: система бронювання занять;
- notifications_notification: зберігає сповіщення;
- course_forumessage: форум-записи;
- accounts_teachrating: оцінки студентів викладачам.

Інші службові таблиці (auth_permission, auth_group, django_session тощо) – відповідають за авторизацію, права доступу, логування та інше.

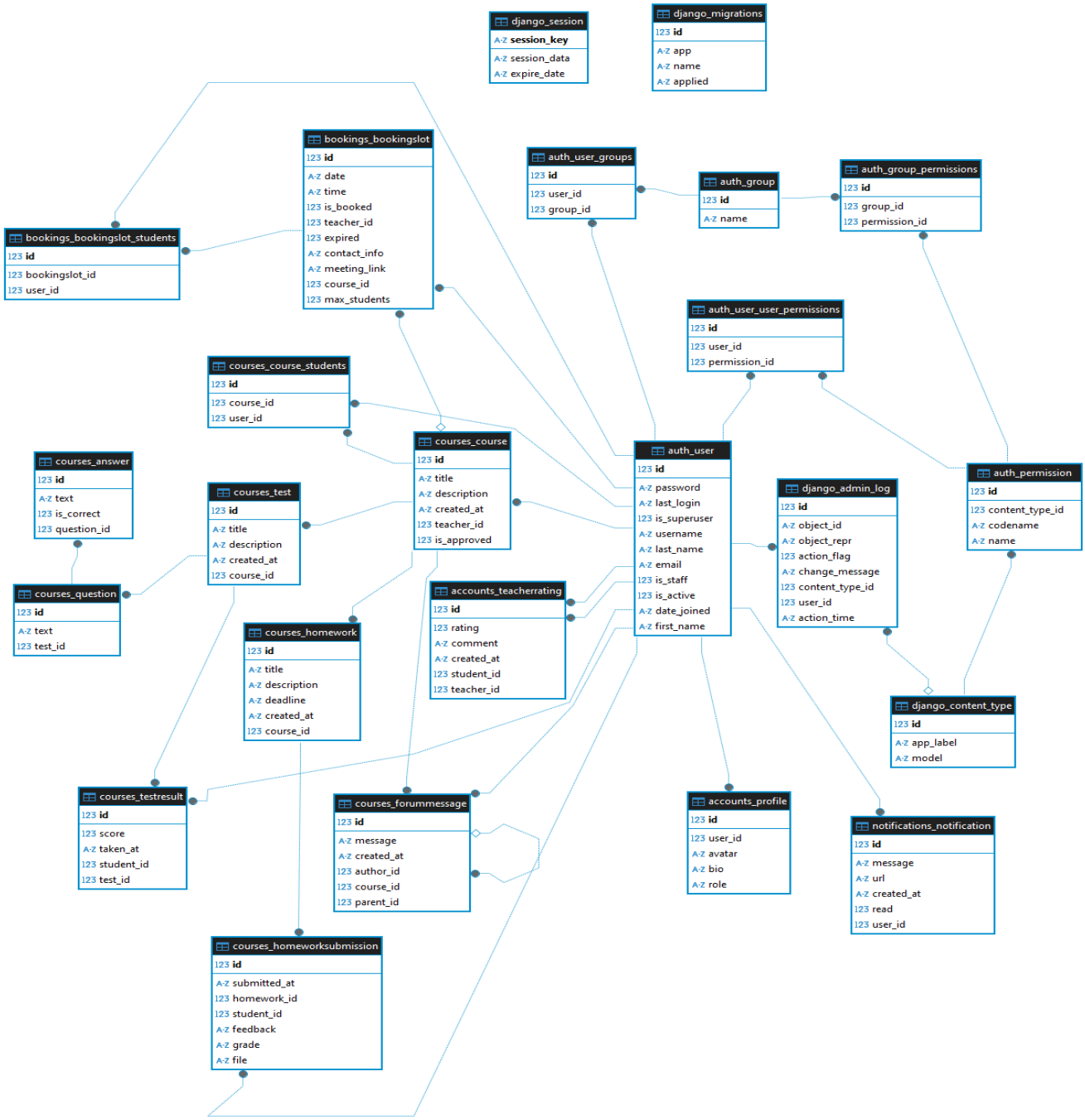


Рисунок 2.4 – ER-діаграма бази даних

Для повного розуміння логіки функціонування веб-сервісу необхідно розглянути ключові таблиці, що реалізують основні процеси взаємодії користувачів із системою. Нижче наведено опис основних таблиць, їх призначення та структури полів відповідно до логічної моделі.

Таблиця auth_user є однією з ключових у структурі бази даних і відповідає за збереження базової інформації про всіх користувачів веб-сервісу. Вона є стандартною у фреймворку Django і забезпечує автентифікацію та управління обліковими записами.

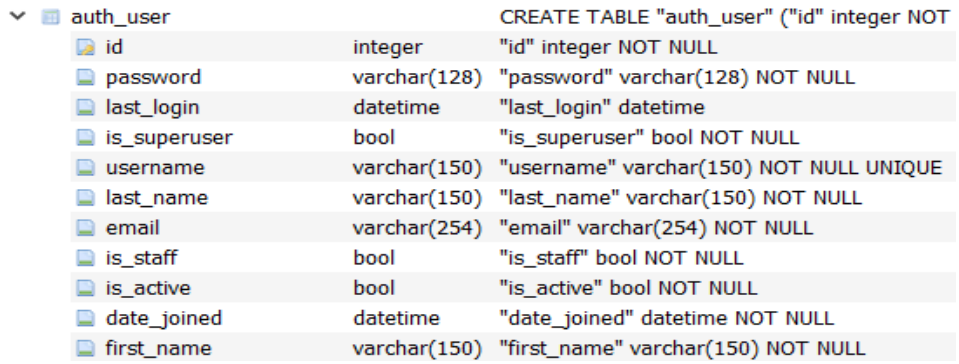


Рисунок 2.5 – Таблиця auth_user

Таблиця 2.3 – Опис таблиці auth_user

Поле	Тип даних	Призначення
id	Int	Первинний ключ користувача
password	Varchar	Хешований пароль користувача
last_login	Datetime	Дата і час останнього входу
is_super	Boolean	Вказує, чи має користувач права суперкористувача (адміністратора)
username	Varchar	Унікальне ім’я користувача
last_name	Varchar	Прізвище користувача
email	Varchar	Електронна пошта
is_staff	Boolean	Дозвіл доступу до адміністративної частини сайту
is_active	Boolean	Вказує, чи активовано обліковий запис
date_joined	Datetime	Дата реєстрації облікового запису

Ця таблиця взаємодіє з іншими через зовнішні ключі: наприклад, зв’язується з профілем користувача (accounts_profile) або записами бронювання, курсами, тестами, домашніми завданнями тощо. Таким чином, вона є ядром усіх ролей та операцій у системі.

Таблиця accounts_profile зберігає додаткову інформацію про користувача, яка не передбачена базовою таблицею auth_user. У ній міститься роль користувача (студент, викладач), біографічні дані та аватар.

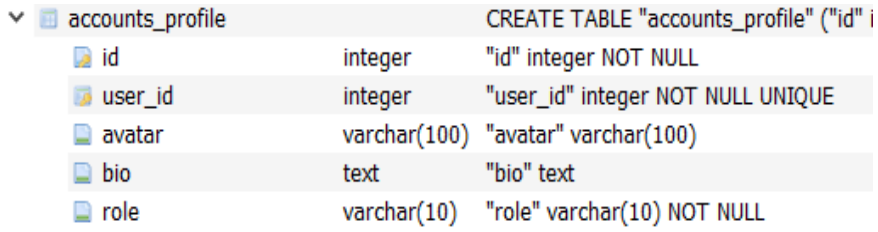


Рисунок 2.6 – Таблиця accounts_profile

Таблиця 2.4 – Опис таблиці accounts_profile

Поле	Тип даних	Призначення
id	Int	Первинний ключ профілю
user_id	Int	Зовнішній ключ на таблицю auth_user
role	Varchar	Роль користувача в системі
avatar	Varchar	Аватар користувача
bio	Text	Біографічна інформація користувача

Ця таблиця використовується для диференціації типів користувачів та відображення профілю в особистому кабінеті. На її основі реалізуються фільтрації, відображення карток профілю, а також логіка доступу до функцій відповідно до ролі користувача.

Таблиця courses_course містить основну інформацію про навчальні курси, доступні для студентів. Кожен запис представляє окремий курс із відповідними атрибутами, включаючи назву, опис, автора (викладача) та статус модерації.

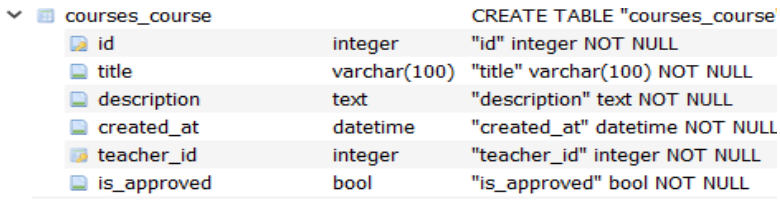


Рисунок 2.7 – Таблиця courses_course

Таблиця 2.5 – Опис таблиці courses_course

Поле	Тип даних	Призначення
id	Int	Первинний ключ курсу
teacher_id	Int	Зовнішній ключ на користувача-викладача
title	Varchar	Назва курсу
description	Text	Опис курсу
is_approved	Boolean	Статус схвалення курсу адміністратором
created_at	Datetime	Дата створення курсу

Курс є центральною одиницею навчання, з яким пов’язані тести, домашні завдання, повідомлення у форумі та слоти бронювання. Всі інші навчальні об’єкти через зовнішні ключі зв’язані саме з цією таблицею.

Таблиця courses_test зберігає інформацію про тести, які належать до певного курсу. Вона містить заголовок тесту, опис, дату створення та ознаку затвердження.

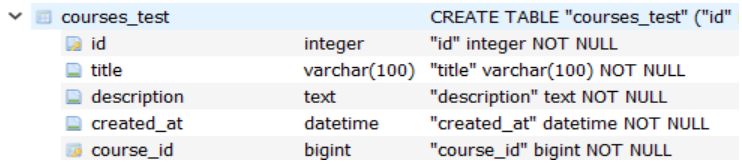


Рисунок 2.8 – Таблиця courses_test

Таблиця 2.6 – Опис таблиці courses_test

Поле	Тип даних	Призначення
id	Int	Первинний ключ тесту
course_id	BigInt	Зовнішній ключ на таблицю courses_course

Кінець таблиці 2.5

title	Varchar	Назва тесту
description	Text	Опис тесту
created_at	Datetime	Дата створення тесту

Тест пов’язаний з конкретним курсом і включає в себе перелік запитань, які зберігаються в таблиці courses_question. Студенти проходять тести, результати яких фіксуються у таблиці courses_testresult.

Таблиця course_question зберігає перелік запитань, що входять до тестів. Кожне запитання належить до конкретного тесту та містить текст формулювання, тип відповіді (текстова або вибір із варіантів), а також порядковий номер.

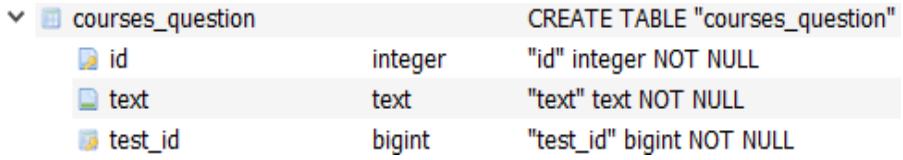


Рисунок 2.9 – Таблиця courses_question

Таблиця 2.7 – Опис таблиці courses_question

Поле	**Тип даних**	**Призначення**
id	Int	Первинний ключ запитання
test_id	BigInt	Зовнішній ключ на таблицю courses_test
text	Text	Текст запитання

Ця таблиця визначає структуру кожного тесту та є пов’язуючою ланкою між тестами (courses_test) та відповідями (courses_answer). Вона забезпечує гнучкість у налаштуванні тестових завдань відповідно до тематики курсу.

Таблиця courses_answer містить варіанти відповідей на запитання тесту. Вона застосовується у випадку, якщо тип запитання – вибір із кількох варіантів. Зв’язана із таблицею courses_question за зовнішнім ключем.

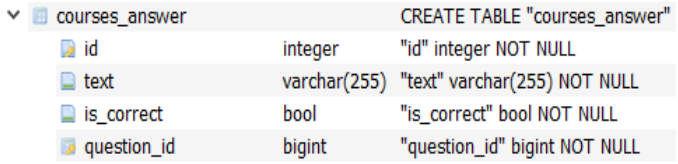


Рисунок 2.10 – Таблиця courses_answer

Таблиця 2.8 – Опис таблиці courses_answer

Поле	Тип даних	Призначення
id	Int	Первинний ключ відповіді
question_id	BigInt	Зовнішній ключ на courses_question
text	Varchar	Текст варіанту відповіді
is_correct	Boolean	Чи є варіант правильним

Ця таблиця дозволяє створювати структуровані запитання з декількома варіантами відповідей, які потім оцінюються автоматично при проходженні тесту студентом.

Таблиця courses_testresult зберігає результати проходження тестів студентами. Вона фіксує інформацію про те, який студент проходив який тест, його набраний бал, дату проходження та додаткові параметри.

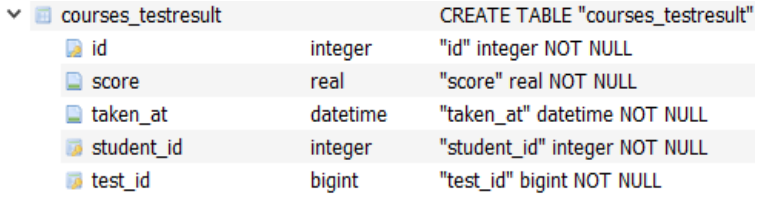


Рисунок 2.11 – Таблиця courses_testresult

Таблиця 2.9 – Опис таблиці courses_testresult

Поле	Тип даних	Призначення
id	Int	Первинний ключ результату
test_id	BigInt	Зовнішній ключ на таблицю courses_test

Кінець таблиці 2.9

student_id	Int	Зовнішній ключ на таблицю auth_user
score	Int	Набраний бал
taken_at	Datetime	Дата проходження тесту

Ця таблиця є ключовою для аналітики й контролю знань студентів. Вона дозволяє будувати рейтинги, обчислювати середні бали та фіксувати успішність по кожному курсу.

Таблиця courses_homework зберігає інформацію про домашні завдання, які викладач додає до курсу. Вона пов'язана з таблицею курсів та містить основні характеристики кожного завдання.

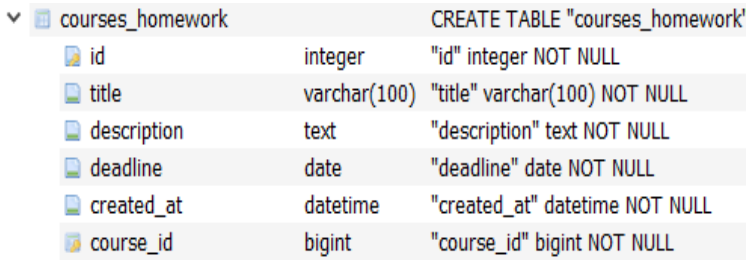


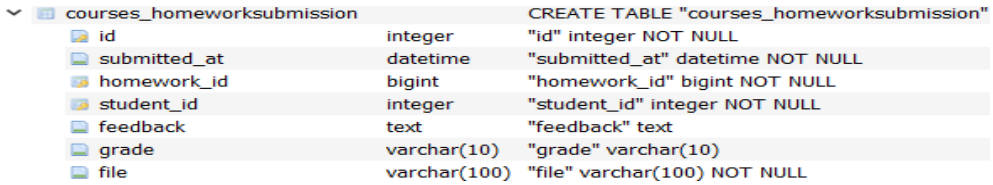
Рисунок 2.12 – Таблиця courses_homework

Таблиця 2.10 – Опис таблиці courses_homework

Поле	Тип даних	Призначення
id	Int	Первинний ключ завдання
course_id	BigInt	Зовнішній ключ на таблицю courses_course
title	Varchar	Назва завдання
description	Text	Опис завдання
deadline	Date	Кінцевий термін здачі
created_at	Datetime	Дата створення

Домашнє завдання відображається студентам у курсі після затвердження викладачем. Ця таблиця тісно пов'язана з таблицею courses_homeworksubmission, де фіксуються відповіді студентів. Вона дозволяє будувати рейтинги, обчислювати середні бали та фіксувати успішність по кожному курсу.

Таблиця courses_homeworksubmission містить інформацію про відповіді студентів на домашні завдання. Вона дозволяє фіксувати, хто саме і коли надіслав відповідь на конкретне завдання, а також – результат перевірки.



courses_homeworksubmission		CREATE TABLE "courses_homeworksubmission"
id	integer	"id" integer NOT NULL
submitted_at	datetime	"submitted_at" datetime NOT NULL
homework_id	bigint	"homework_id" bigint NOT NULL
student_id	integer	"student_id" integer NOT NULL
feedback	text	"feedback" text
grade	varchar(10)	"grade" varchar(10)
file	varchar(100)	"file" varchar(100) NOT NULL

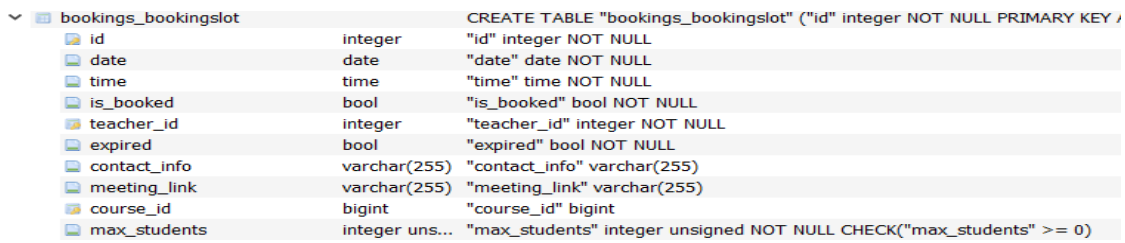
Рисунок 2.13 – Таблиця courses_ homeworksubmission

Таблиця 2.11 – Опис таблиці courses_homeworksubmission

Поле	Тип даних	Призначення
id	Int	Первинний ключ подання
homework_id	BigInt	Зовнішній ключ на таблицю courses_homework
student_id	Int	Зовнішній ключ на таблицю auth_user
feedback	Text	Фідбек
file	Varchar	Назва файлу
submitted_at	Datetime	Дата й час подання
grade	Varchar	Оцінка

Ця таблиця є важливою для реалізації двостороннього процесу перевірки домашніх завдань, оскільки дозволяє як фіксувати подачу, так і формувати зворотній зв’язок.

Таблиця bookings_bookingslot відповідає за зберігання інформації про доступні часові слоти для бронювання занять між студентами та викладачами. Кожен слот має конкретну дату, час початку та закінчення, а також прив’язується до викладача.



bookings_bookingslot		CREATE TABLE "bookings_bookingslot" ("id" integer NOT NULL PRIMARY KEY
id	integer	"id" integer NOT NULL
date	date	"date" date NOT NULL
time	time	"time" time NOT NULL
is_booked	bool	"is_booked" bool NOT NULL
teacher_id	integer	"teacher_id" integer NOT NULL
expired	bool	"expired" bool NOT NULL
contact_info	varchar(255)	"contact_info" varchar(255)
meeting_link	varchar(255)	"meeting_link" varchar(255)
course_id	bigint	"course_id" bigint
max_students	integer uns...	"max_students" integer unsigned NOT NULL CHECK("max_students" >= 0)

Рисунок 2.14 – Таблиця bookings_bookingslot

Таблиця 2.12 – Опис таблиці bookings_bookingslot

Поле	Тип даних	Призначення
id	Int	Первинний ключ бронювання
teacher_id	Int	Зовнішній ключ на таблицю auth_user
course_id	BigInt	Зовнішній ключ на таблицю courses_course
date	Date	Дата проведення заняття
time	Time	Час проведення заняття
is_booked	Boolean	Чи заброньовано
expired	Boolean	Чи пройшло вже заняття
contact_info	Varchar	Контакти
meeting_link	Varchar	Посилання на заняття
max_students	Int	Максимальна к-сть студентів

Ця таблиця є основою для реалізації системи планування занять і дозволяє викладачам створювати доступні вікна, які потім можуть бути заброньовані студентами через пов’язану таблицю bookings_bookingslot_students.

Таблиця bookings_bookingslot_students реалізує зв’язок між студентами та слотами бронювання. Вона використовується для фіксації, який студент забронював конкретний часовий проміжок із викладачем.

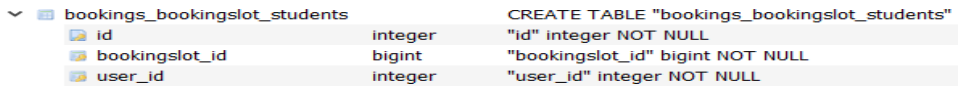


Рисунок 2.15 – Таблиця bookings_bookingslot_students

Таблиця 2.13 – Опис таблиці bookings_bookingslot_students

Поле	Тип даних	Призначення
id	Int	Первинний ключ запису
user_id	Int	Зовнішній ключ на таблицю auth_user
bookingslot_id	BigInt	Зовнішній ключ на таблицю bookings_bookingslot

Ця таблиця дозволяє забезпечити реєстрацію на заняття та унеможливлює багаторазове бронювання одного й того ж слоту кількома користувачами. Вона також може бути використана для перегляду історії занять конкретного студента або формування розкладу викладача.

Таблиця notifications_notification відповідає за збереження сповіщень, що надсилаються користувачам системи. Вона дозволяє реалізувати механізм повідомлення про важливі події, оновлення або нагадування.

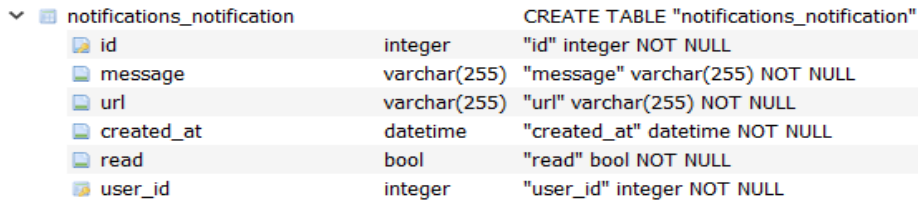


Рисунок 2.16 – Таблиця notifications_notification

Таблиця 2.14 – Опис таблиці notifications_notification

Поле	Тип даних	Призначення
id	Int	Первинний ключ сповіщення
user_id	Int	Зовнішній ключ на користувача, якому адресується сповіщення
user_id	Int	Зовнішній ключ на користувача, якому адресується сповіщення
message	Varchar	Повідомлення
url	Varchar	Посилання
read	Boolean	Чи прочитано
created_at	Datetime	Дата й час створення

Ця таблиця дає змогу реалізувати як просту систему сповіщень, так і складні механізми нагадувань про дедлайни, нові повідомлення у чатах або зміни в курсах.

Таблиця course_forumessage зберігає повідомлення, які користувачі публікують у форумі, пов’язаному з курсами. Це дозволяє студентам та викладачам вести обговорення безпосередньо в межах системи.

Рисунок 2.17 – Таблиця course_forumessage

Таблиця 2.15 – Опис таблиці course_forumessage

Поле	Тип даних	Призначення
id	Int	Первинний ключ повідомлення
course_id	BigInt	Зовнішній ключ на таблицю courses_course
parent_id	BigInt	Зовнішній ключ на цю ж таблицю, якщо це відповідь на інше повідомлення
author_id	Int	Зовнішній ключ на автора повідомлення
created_at	Datetime	Дата й час створення
message	Text	Повідомлення у форумі

Ця таблиця є важливою складовою для реалізації внутрішньої комунікації в межах курсу. Повідомлення можуть стосуватись обговорення теми заняття, організаційних питань чи обміну думками серед учасників.

Таблиця accounts_teachrating зберігає оцінки, які студенти виставляють викладачам після проходження курсів. Вона забезпечує механізм зворотного зв’язку між учасниками навчального процесу.

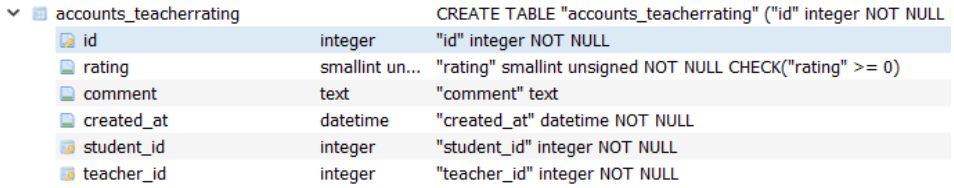


Рисунок 2.18 – Таблиця accounts_teachrating

Таблиця 2.16 – Опис таблиці accounts_teachrating

Поле	Тип даних	Призначення
id	Int	Первинний ключ оцінки
teacher_id	Int	Зовнішній ключ на вчителя у таблиці auth_user

Кінець таблиці 2.16

student_id	Int	Зовнішній ключ на студента у таблиці auth_user
rating	SmallInt	Рейтинг
comment	Text	Коментар
created_at	Datetime	Дата й час оцінки

Ця таблиця дозволяє формувати рейтинг викладачів, відображати оцінки на сторінках курсів і враховувати думку студентів для покращення якості викладання.

Таким чином, структура бази даних веб-сервісу комунікації учасників курсів іноземних мов була ретельно спроектована відповідно до потреб користувачів різних ролей. Завдяки логічному поділу таблиць на функціональні блоки, а саме: реєстрація, навчальний контент, тести, домашні завдання, бронювання, повідомлення та форуми – система забезпечує цілісність, масштабованість і зручність у подальшому супроводі. Вибір реляційної моделі та використання ORM Django дає змогу ефективно реалізувати зв’язки між сутностями, спростити роботу з даними та забезпечити високий рівень узгодженості в усіх компонентах веб-сервісу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ КОМУНІКАЦІЇ УЧАСНИКІВ КУРСІВ ІНОЗЕМНИХ МОВ

3.1 Розробка серверної частини та бази даних

Для реалізації серверної частини веб-сервісу було обрано фреймворк Django, який забезпечує швидку побудову структурованих та захищених веб-застосунків із вбудованим ORM (Object-Relational Mapping), що дає змогу взаємодіяти з базою даних без написання SQL-запитів.

На початковому етапі проєкту було створено новий Django-проєкт командою:

```
django-admin startproject LanguageSchool
```

Після цього було ініціалізовано окремі додатки (apps), кожен із яких відповідає за свою частину функціональності, зокрема:

- accounts: обробка профілів, реєстрації, ролей користувачів;
- course: курси, тести, завдання;
- bookings: бронювання занять;
- notifications: реалізація системи сповіщень;

Кожен додаток містить власні моделі, представлення (views) та шаблони, що дозволяє підтримувати модульну та легко масштабовану архітектуру.

У якості системи управління базами даних було обрано SQLite, яка входить до стандартного складу Django. Такий вибір є доцільним для невеликих або середніх проєктів, а також під час активної розробки, оскільки дозволяє уникнути додаткових налаштувань зовнішнього СУБД-сервера.

Підключення до бази даних реалізоване у файлі settings.py:

```
DATABASES = {
```

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

Для взаємодії з базою даних використовується Django ORM. Усі таблиці генеруються автоматично на основі Python-класів (моделей). Наприклад, модель профілю користувача:

```
    class Profile(models.Model):
        user = models.OneToOneField(User, on_delete=models.CASCADE)
        role = models.CharField(max_length=20,
choices=ROLE_CHOICES)
        bio = models.TextField(blank=True)
```

Це дозволяє працювати з базою на рівні об'єктів Python, а не сирих SQL-запитів. Міграції для створення структури БД виконуються командами:

```
    python manage.py makemigrations
    python manage.py migrate
```

ORM дозволяє легко витягувати, додавати, змінювати та видаляти дані. Наприклад, створення нового тесту:

```
    CourseTest.objects.create(
        course=course,
        title="Test 1",
        created_by=request.user
    )
```

Також варто зазначити, що усі ключові налаштування зосереджені у файлі settings.py, серед яких:

- INSTALLED_APPS: активні додатки;
- MIDDLEWARE: обробники запитів (включно з автентифікацією);
- AUTH_USER_MODEL: вказівка на власну модель користувача, якщо застосовується;
- STATIC_URL, MEDIA_URL, TEMPLATES: параметри обробки статичних і шаблонних файлів;
- LANGUAGE_CODE = 'en-us', TIME_ZONE = 'Europe/Kyiv': мовна локалізація; часова зона
- DEBUG = True, ALLOWED_HOSTS = []: налаштування для режиму розробки.

Таким чином, серверна частина нашого веб-сервісу базується на класичних рішеннях Django без сторонніх залежностей. Це дозволило забезпечити високу стабільність, гнучкість розробки та відповідність усім вимогам до безпеки та масштабування проєкту.

3.2 Розробка клієнтської частини

У даному проєкті клієнтська частина реалізована відповідно до класичної архітектури Django, що передбачає використання шаблонізатора та серверної генерації HTML-сторінок. Це забезпечує простоту розгортання, швидке завантаження сторінок без потреби у додатковому клієнтському фреймворку (наприклад, React чи Vue), а також тісну інтеграцію з серверною логікою.

Для розмежування логіки в шаблонах клієнтської частини використано структуру директорій, зображену на рисунку 3.1, яка відповідає окремим модулям сервісу. Зокрема:

- accounts/templates: містить шаблони, пов’язані з обліковими записами користувачів (реєстрація, вхід, профіль);

- courses/templates: шаблони, що відображають список курсів, інформацію про курс, домашні завдання, тести тощо;
- bookings/templates: шаблони для бронювання слотів занять між студентами та викладачами;
- notifications/templates: шаблони сповіщень;
- templates: коренева директорія з базовим шаблоном base.html, який підключається до всіх сторінок для забезпечення єдиного стилю інтерфейсу.

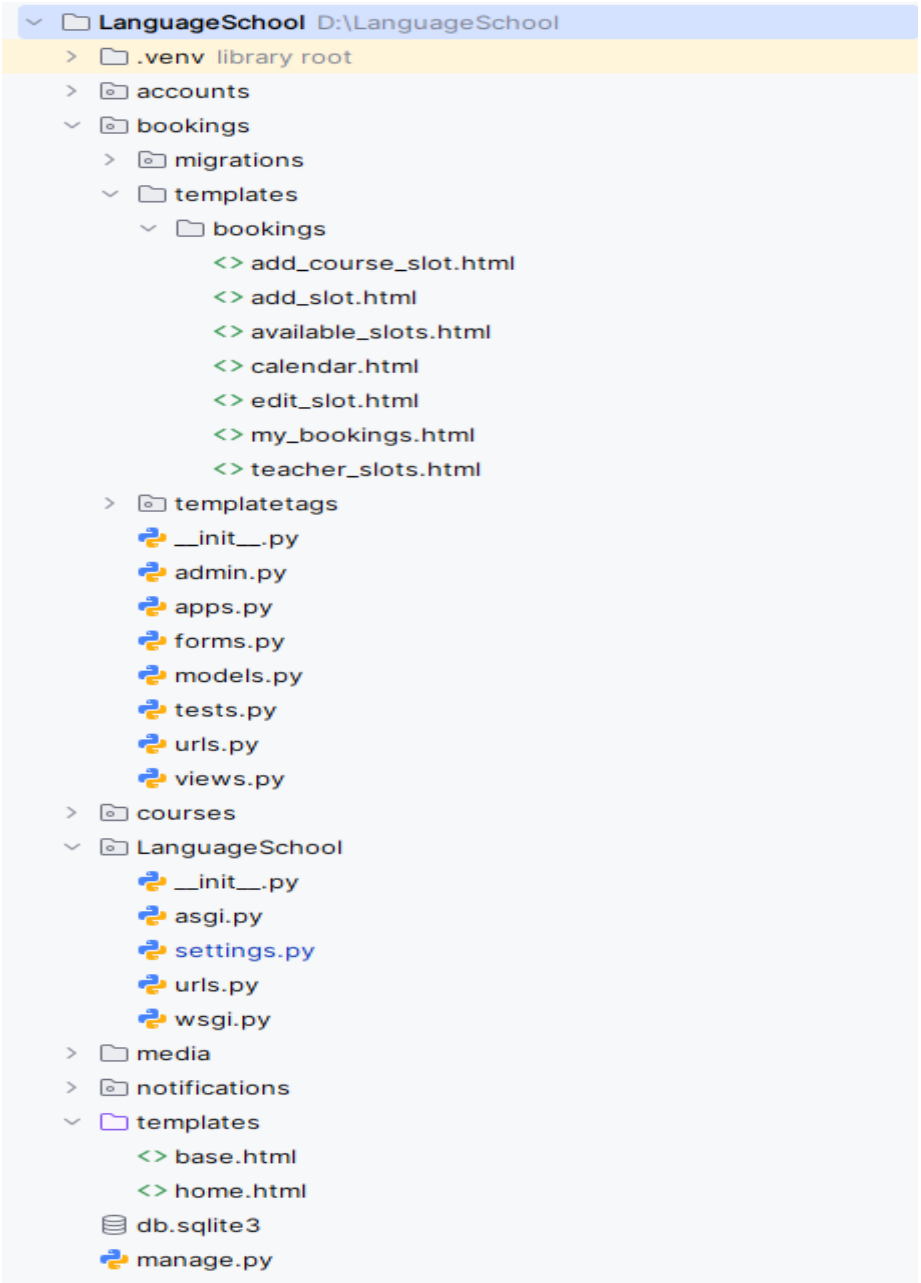


Рисунок 3.1 – Структура папок клієнтської частини веб-сервісуУ

Кожен шаблон наслідує структуру головного шаблону base.html, що забезпечує уніфікованість інтерфейсу: верхня панель навігації, футер, підключення CSS та JS-файлів.

Основні шаблони клієнтської частини:

- base.html: базовий шаблон, що містить основні теги HTML, підключення Bootstrap, заголовок сторінки, меню навігації. У цьому шаблоні розміщено блоки {% block content %}, які перевизначаються в дочірніх шаблонах.
- home.html (розташований у templates): головна сторінка, яка динамічно відображає вітальне повідомлення, кнопки входу/реєстрації або навігаційні елементи для авторизованих користувачів.
- login.html, register.html (у accounts/templates/accounts): форми входу та реєстрації. Вони використовують стандартні форми Django або кастомні форми з валідацією та CSRF-захистом.
- course_detail.html (у courses/templates/courses): сторінка з інформацією про курс, переліком викладачів, домашніх завдань, тестів тощо. Тут реалізовано перевірку ролей (викладач/студент) та відповідні функції (наприклад, редагування для викладача).
- booking_page.html (у bookings/templates/bookings): сторінка з можливістю обрати доступний слот бронювання, які витягуються з бази даних та групуються за викладачами та днями тижня.
- notifications.html (у notifications/templates/notifications): відображає список сповіщень, які зберігаються у відповідній моделі Notification.

А у шаблонах широко використовуються вбудовані теги Django, такі як:

- {% extends %}: для наслідування базових шаблонів;
- {% block %} та {% endblock %}: для визначення змінних областей вмісту;
- {% url %}: для генерації посилань за іменами маршрутів;

- {% if user.is_authenticated %}: умовне відображення елементів інтерфейсу в залежності від ролі користувача;
- {% csrf_token %}: для захисту від CSRF-атак у формах;
- {% for item in list %}: циклічний вивід даних.

Завдяки цим можливостям вдається досягти високого рівня динамічності сторінок без потреби у JavaScript-фреймворках.

У клієнтській частині проєкту для зручного створення форм також було використано стандартні можливості Django Forms. Завдяки ним реалізовано обробку вхідних даних, автоматичне створення HTML-форм, валідацію полів, обробку CSRF-токенів, а також виведення помилок. Всі основні форми, зокрема форми реєстрації, входу, створення курсів, тестів, домашніх завдань – побудовані на базі класів forms.Form або forms.ModelForm.

Форма входу (LoginForm) складається з двох полів: ім’я користувача або email, та пароль. При відправці форми проводиться валідація через метод clean, який перевіряє відповідність даних у базі. Якщо дані вірні, користувача аутентифікують за допомогою функції authenticate() і виконують вхід через login(request, user).

Форма реєстрації (RegisterForm) містить більше полів: ім’я, прізвище, логін, email, пароль та підтвердження пароля. Після перевірки унікальності логіна та email, створюється обліковий запис. Для забезпечення безпеки форма використовує PasswordInput, а також вбудовану функцію хешування пароля через set_password.

Окрім цього, реалізовані окремі форми для:

- створення домашніх завдань (HomeworkForm);
- створення/редагування тестів (TestForm);
- додавання питань до тесту (QuestionForm);
- бронювання часу (BookingSlotForm, StudentBookingForm).

Користувацький досвід диференціюється відповідно до ролі. Всі ролі (гість, студент, викладач) мають різні доступи, що реалізовано у шаблонах через перевірки if user.is_authenticated та if user.groups.

Гість:

– може переглядати головну сторінку, перелік курсів (без доступу до змісту);
– бачить кнопки "Увійти" та "Зареєструватися";
– не може бачити профілі, тести чи домашні завдання.

Студент:

– бачить лише ті курси, на які записаний;
– має доступ до перегляду та подачі домашніх завдань;
– проходить доступні для нього тести;
– бачить бронювання та новини по курсах.

Викладач:

– має можливість створювати нові курси;
– може створювати домашні завдання та тести;
– бачить всі відповіді студентів і може їх оцінювати;
– має доступ до календаря бронювань з можливістю додавати вільні слоти.

Усі шаблони адаптовані під ці ролі через умовні блоки, наприклад:

```
{% if user.groups.first.name == 'Teacher' %}
  <a href="{% url 'create_homework' course.id %}" class="btn btn-primary">Додати завдання</a>
{% endif %}
```

Таким чином клієнтська частина побудована з використанням перевірених практик Django, адаптована до ролей, та має зручний і адаптивний інтерфейс.

Завдяки динамічній поведінці форм та подій на сторінках курсу (course_detail.html) реалізовано можливість викладачу редагувати вміст за допомогою клієнтського JavaScript. Наприклад, при натисканні кнопки

«Редагувати» змінюється видимість блоків з формами, що дозволяє зручно змінювати опис або додавати завдання. Для цього використовується клас Bootstrap collapse та JS-події типу onclick.

На сторінці тестів (take_test.html) студенти бачать форму з питаннями, в яких відповіді можуть бути як текстові, так і з варіантами вибору. Для зручності введення відповіді підтримується автоматична перевірка заповненості обов'язкових полів перед надсиланням, а також індикація помилок без повного релоаду сторінки.

У формі створення домашнього завдання (create_homework.html) перед відправкою здійснюється валідація дати дедлайну. JavaScript-код перевіряє, чи встановлена дата не є меншою за поточну, що запобігає випадковому створенню недійсних дедлайнів.

Сторінка бронювання (available_slots.html) дозволяє студенту вибрати доступний слот. Вільні слоти динамічно завантажуються через JavaScript методом fetch, що запитує дані з серверу у вигляді JSON і оновлює DOM без перезавантаження. Це реалізовано як через модальні вікна (modal), так і таблиці зі слотами (table-striped).

Усі шаблони застосунку об'єднані через головний шаблон base.html, який містить навігаційне меню. У залежності від ролі користувача та факту автентифікації, навігаційна панель динамічно відображає відповідні пункти меню. Це реалізовано за допомогою конструкцій if user.is_authenticated та user.groups.first.nam. Приклад:

```
    {% if user.is_authenticated %}
  <li class="nav-item">
    <a href="{% url 'profile' %}" class="nav-link">👤 Профіль</a>
  </li>
  <li class="nav-item">
    <a href="{% url 'notifications' %}" class="nav-link position-
relative">
```

```
      🔔
      {%       if       unread_notifications       >       0       %}
        <span  class="position-absolute  top-0  start-100  translate-
middle            badge            rounded-pill            bg-danger">
          {{                  unread_notifications                  }}
        </span>
      {%                            endif                            %}
    </a>
  </li>
  <li                                                  class="nav-item">
    <a class="nav-link" href="{% url 'dashboard' %}">📋 Кабінет</a>
  </li>
  <li                                                  class="nav-item">
    <a  class="nav-link"  href="{%  url  'logout'  %}">🚪  Вийти</a>
  </li>
{%                                 else                                %}
  <li                                                  class="nav-item">
    <a  class="nav-link"  href="{%  url  'login'  %}">🔐  Увійти</a>
  </li>
  <li                                                  class="nav-item">
    <a    class="nav-link"    href="{%    url    'register'    %}">📝
Реєстрація</a>
  </li>
{% endif %}
```

Таке динамічне меню забезпечує адаптацію інтерфейсу до користувача та покращує користувацький досвід без потреби в окремій маршрутизації на клієнті.

Функціонал приєднання до курсу студентом реалізовано через шаблон course_detail.html, який у залежності від ролі користувача відображає відповідну кнопку:

```
    {% if user.is_authenticated and user.groups.first.name ==
'Student' and not enrolled %}
      <form method="post" action="{% url 'enroll_course' course.id
%}">
        {% csrf_token %}
        <button type="submit" class="btn btn-success">Приєднатися
до курсу</button>
      </form>
    {% endif %}
```

View-функція enroll_in_course перевіряє, чи студент ще не приєднаний до курсу, після чого створює відповідний зв'язок у базі даних через ManyToMany-зв’язок Course.students.add(request.user). Приклад функції enroll_in_course:

```
    @login_required
def enroll_in_course(request, course_id):
    course = get_object_or_404(Course, id=course_id)

    if request.user.profile.role != 'student':
        messages.error(request, 'Лише студенти можуть записуватися
на курси.')
        return redirect('dashboard')

    course.students.add(request.user)
    messages.success(request, f'✅ Ви записалися на курс
"{course.title}".')
    return redirect('course_detail', course_id=course.id)
```

Після приєднання оновлюється інтерфейс: замість кнопки "Приєднатися" студент бачить розклад, завдання та тести.

Календар бронювань реалізований на основі сторінки available_slots.html. Студент бачить розклад доступних слотів викладачів, згрупованих по днях тижня.

Дані для відображення надходять із серверної частини за допомогою Django context і включають:

- список викладачів;
- доступні тайм-слоти;
- уже заброньовані часи.

Для зручності взаємодії використовуються компоненти Bootstrap (таблиці, модальні вікна). JavaScript дозволяє динамічно відкривати вікно для бронювання, заповнювати форму, а також валідувати вибраний слот на боці клієнта.

При натисканні кнопки "Забронювати", спрацьовує fetch-запит, який надсилає POST-запит до відповідного URL з ID слота і даними користувача, після чого DOM оновлюється – слот позначається як зайнятий.

Інтерфейс курсу викладача містить додаткові елементи управління у шаблоні course_detail.html:

- Кнопка “Додати завдання”, що веде до форми створення;
- Кнопка “Редагувати тест”, яка відкриває форму редагування;
- Кнопка “Перегляд результатів” для доступу до аналітики.

Усі ці елементи обгорнуті в приблизну конструкцію:

```
{% if user.groups.first.name == 'Teacher' %}
  <a class="btn btn-primary" href="{% url 'edit_test' test.id %}">Редагувати</a>
{% endif %}
```

На сторінці редагування тесту (edit_test.html) реалізовано форму з динамічним додаванням запитань. Поля форми мають JavaScript-перевірку: не можна додати порожнє питання або питання без варіантів. Також реалізовано інтерактивне видалення блоку запитання без перезавантаження сторінки.

У рамках клієнтської частини веб-сервісу для курсів іноземних мов реалізовано чітко структуровану логіку роботи з ключовими функціональними

блоками: домашніми завданнями, тестами, сповіщеннями та системою повідомлень. Кожна з цих підсистем адаптована до ролей користувачів і реалізована за допомогою шаблонів Django, Bootstrap-компонентів та JavaScript-функцій для покращення динамічності взаємодії.

Кожне домашнє завдання створюється викладачем із відповідної сторінки курсу (course_detail.html), де відображається список вже доданих завдань. Перехід до створення здійснюється через кнопку "Додати завдання", яка веде на сторінку create_homework.html.

Форма створення побудована на базі HomeworkForm і містить:

- назву;
- опис;
- дату дедлайну;
- вибір курсу.

Перед відправкою форма перевіряється на коректність введення дати (дата не повинна бути меншою за поточну), а також на заповненість обов’язкових полів. Якщо всі умови виконано, запит передається серверу, і після збереження об'єкта Homework користувач отримує відповідне повідомлення.

На стороні студента завдання відображаються на сторінці курсу, де кожне з них має кнопку "Надіслати розв'язок", що веде до форми submit_homework.html. Студент може прикріпити відповідь у вигляді тексту або файлу. Після відправки завдання викладач отримає сповіщення і можливість оцінити виконання.

Система тестування реалізована за допомогою двох основних шаблонів:

- test_detail.html: перегляд і редагування тесту (для викладача);
- take_test.html: проходження тесту (для студента).

Для викладача:

У test_detail.html реалізовано форму редагування тесту з можливістю додавання питань і варіантів відповіді. Кожне питання подається як блок, що містить:

- текст питання;

– тип (відкрите, вибір одного/декількох);

– варіанти (якщо застосовно).

JavaScript дозволяє додавати нові блоки питань динамічно, а також видаляти існуючі без перезавантаження. Перевірки наявності заповненого тексту питання та варіантів виконуються одразу у браузері.

Для студента:

У шаблоні take_test.html студент бачить адаптовану форму з запитаннями, яка залежить від типу тесту. При натисканні на кнопку "Надіслати відповіді", JavaScript:

– перевіряє наявність заповнених полів;

– формує запит до серверної частини через стандартну POST-відправку;

– у разі помилки, повертає повідомлення про невідповідність.

Після завершення тесту, система записує результати у базу (модель TestResult) та надсилає сповіщення викладачу.

А для перегляду результатів тестів створено шаблон test_results_teacher.html, який надає викладачам повний огляд результатів тесту. У табличному форматі виводиться:

– ім’я студента з посиланням на його публічний профіль;

– отриманий бал у відсотках;

– дата проходження.

Приклад частини шаблону:

```
<tr>
  <td><a href="{% url 'public_profile' result.student.username %}">{{ result.student.get_full_name|default:result.student.username }}</a></td>
  <td><strong>{{ result.score }}%</strong></td>
  <td>{{ result.taken_at|date:"d.m.Y H:i" }}</td>
</tr>
```

Таким чином викладач має змогу відслідковувати успішність учнів та при потребі перейти до оцінювання чи аналізу відповідей окремого студента.

Для студента ж створено окрему сторінку test_result_student.html для перегляду власного результату, на якій він бачить бал у відсотках та дату проходження тесту. Оформлення забезпечує емоційне підкріплення досягнутого результату завдяки компонентам Bootstrap та зручній навігації.

У шаблоні course_detail створено можливість перегляду домашніх завдань, де студент бачить список домашніх завдань доступних у курсі:

```
    {% if homeworks %}
  <ul class="list-group">
    {% for hw in homeworks %}
      <li   class="list-group-item   d-flex   justify-content-between
align-items-center">
        <a  href="{%  url  'homework_detail'  hw.id  %}">{{  hw.title
}}</a>
        <span class="badge bg-secondary">{{ hw.deadline|date:"d.m.Y"
}}</span>
      </li>
    {% endfor %}
  </ul>
{% else %}
  <p class="text-muted">Домашніх завдань ще немає.</p>
{% endif %}
```

На цій сторінці студент бачить умову завдання, дедлайн та форму завантаження відповіді. Після надсилання відповідь зберігається в базі, а студент отримує повідомлення про успішну подачу.

Викладач бачить додаткову кнопку “Додати домашнє завдання”, яка веде на create_homework.html, де заповнюється назва, опис, дедлайн. У шаблоні homework_submissions.html викладач бачить список усіх поданих відповідей студентів з можливістю:

– прочитати текст відповіді;

– виставити оцінку;

– залишити коментар або фідбек.

Оцінки, коментарі, файли та дата здачі завдання зберігаються у відповідній моделі HomeworkSubmission:

```
class HomeworkSubmission(models.Model):
    homework = models.ForeignKey(Homework, on_delete=models.CASCADE,
related_name='submissions')
    student = models.ForeignKey(User, on_delete=models.CASCADE)
    file = models.FileField(upload_to='homework_submissions/')
    submitted_at = models.DateTimeField(auto_now_add=True)
    grade = models.CharField(max_length=10, blank=True, null=True)
    feedback = models.TextField(blank=True, null=True)

    def __str__(self):
        return f"{self.student.username} → {self.homework.title}"
```

Таким чином студент може редагувати коментар до завдання або вкладення, до тих пір поки не буде виставлена оцінка викладачем. А вже після виставлення оцінки він зможе переглянути її у відповідному інтерфейсі.

Також було розроблено систему сповіщень, де усі користувачі мають доступ до сторінки notifications.html, яка відображає список сповіщень, що стосуються:

– нових завдань;

– відповідей на тести;

– підтвердження бронювань.

Модель Notification містить поля:

– заголовок сповіщення;

– зміст;

- отримувача;
- дату;
- статус прочитано/непрочитано.

На клієнтській стороні за допомогою конструкції unread_notifications у навігаційному меню відображається індикатор кількості непрочитаних повідомлень:

```
{% if unread_notifications > 0 %}
  <span class="badge bg-danger">{{ unread_notifications }}</span>
{% endif %}
```

При відкритті сторінки відбувається зміна статусу сповіщення на "прочитано". Це реалізовано через виклик відповідного методу у view-контролері.

Також, на основі шаблону course_forum.html у вебсервісі реалізовано повноцінний чат-форум для обговорення між учасниками курсу. Ця функціональність дозволяє підтримувати асинхронну текстову комунікацію між студентами та викладачами у межах конкретного курсу, що суттєво покращує залученість та зручність навчального процесу.

Заголовок сторінки виводить назву курсу, а нижче – форму створення повідомлення, список усіх основних повідомлень і відповідей до них. Логіка побудована навколо наступних елементів:

- Форма надсилання повідомлення включає поле для тексту повідомлення (form.message) та приховане поле form.parent, що використовується для відповіді на конкретне повідомлення.
- При натисканні на кнопку «Відповісти», JavaScript-обробник заповнює значення прихованого поля parent ID відповідного повідомлення і фокусує textarea, показуючи відповідний placeholder типу “Відповідь на повідомлення #12”.

– Повідомлення відображаються у вигляді списку, кожне з яких містить ім’я автора, дату та час створення, саме повідомлення, кнопку відповіді та вкладені блоки-відповіді.

Відповіді виводяться з невеликим відступом і фоном (Bootstrap bg-light та border-start), що візуально групує їх із відповідним кореневим повідомленням.

Скрипт, включений у шаблон, реалізує мінімальну, але зручну інтерактивність:

```
document.addEventListener('DOMContentLoaded', function () {
  document.querySelectorAll('.reply-link').forEach(link => {
    link.addEventListener('click', function (e) {
      e.preventDefault();
      const parentId = this.getAttribute('data-parent-id');
      const messageField = document.querySelector('textarea[name=message]');
      const parentField = document.querySelector('input[name=parent]');
      parentField.value = parentId;
      messageField.focus();
      messageField.placeholder = "Відповідь на повідомлення #" + parentId;
    });
  });
});
```

Це забезпечує зручну UX-функцію – можливість відповісти на конкретне повідомлення без перезавантаження сторінки чи складної взаємодії.

Також реалізовано повноцінну функціональність профілів користувачів з урахуванням ролей та можливістю редагування особистої інформації. Відображення профілю реалізується через шаблон profile.html, який динамічно показує дані поточного користувача: ім’я, email, роль, опис та аватар.

Користувачі з роллю викладача додатково бачать середній рейтинг, який формується на основі оцінок студентів, а також список курсів, які вони викладають. Студенти, у свою чергу, бачать список курсів, на які вони підписані. Це реалізовано через логіку в шаблоні:

```
{% if profile.role == 'student' %}
  <h4>🎓 Курси студента</h4>
  ...
{% elif profile.role == 'teacher' %}
  <h4>🧑🏫 Курси викладача</h4>
  ...
{% endif %}
```

Форма редагування edit_profile.html використовує шаблони Django Forms, які дозволяють редагувати не тільки стандартні поля користувача, а й розширені поля профілю такі як біографія, аватар тощо. Кожне поле стилізоване через Bootstrap-класи:

```
{{ field|add_class:"form-control" }}
```

Кнопка переходу до редагування відображається в самому профілі, а оновлення даних виконується через POST-запит після валідації.

Публічний перегляд профілю іншого користувача реалізований у шаблоні public_profile.html, який дозволяє студентам оцінити викладача або переглянути, які курси він веде. Якщо поточний користувач є студентом і відвідує профіль викладача, він бачить кнопку “Оцінити викладача”.

Інтерфейс форм стилізовано за допомогою бібліотеки Bootstrap 5. Усі поля автоматично отримують класи form-control для уніфікованого вигляду. Наприклад:

```
    <div class="mb-3">
        <label       for="id_username"       class="form-label">Ім’я
користувача</label>
        {{ form.username }}
        {{ form.username.errors }}
    </div>
```

Також, щоб покращити UX, всі форми супроводжуються повідомленнями про помилки, а у разі успішної дії – флеш-повідомленнями (messages.success, messages.error), які відображаються у шаблоні base.html:

```
    {% if messages %}
      {% for message in messages %}
        <div  class="alert  alert-{{  message.tags  }}">{{  message
}}</div>
      {% endfor %}
    {% endif %}
```

Саме завдяки використанню Bootstrap 5 та механізму флеш-повідомлень, форми не лише мають уніфікований та естетичний вигляд, але й забезпечують користувачам зручний досвід взаємодії. Автоматичне застосування стилів спрощує підтримку та розширення інтерфейсу, а система повідомлень гарантує оперативний зворотний зв’язок щодо введених даних. Такий підхід сприяє покращенню UX та підвищенню загальної доступності форми.

3.3 Перевірка працездатності веб-сервісу

Спершу, щоб перевірити працездатність веб-сервісу потрібно запустити веб-сервер. Він запускається в терміналі Django-проекту в середовищі розробки PyCharm або якщо встановлено середовище Python бібліотеками Django в

директорії проекту в місці розташування файлу manage.py. Команда для запуску серверу через термінал:

```
python manage.py runserver
```

Це дозволяє розгорнути веб-сервіс локально за адресою http://127.0.0.1:8000, де можна протестувати всі основні функції системи.

При переході за посиланням відображається домашня сторінка (рис. 3.2), на якій пропонується виконати вхід або реєстрацію. Для гостей також доступна базова інформація про відкриті курси (з коротким описом) та список викладачів з можливістю перегляду їхніх публічних профілів.

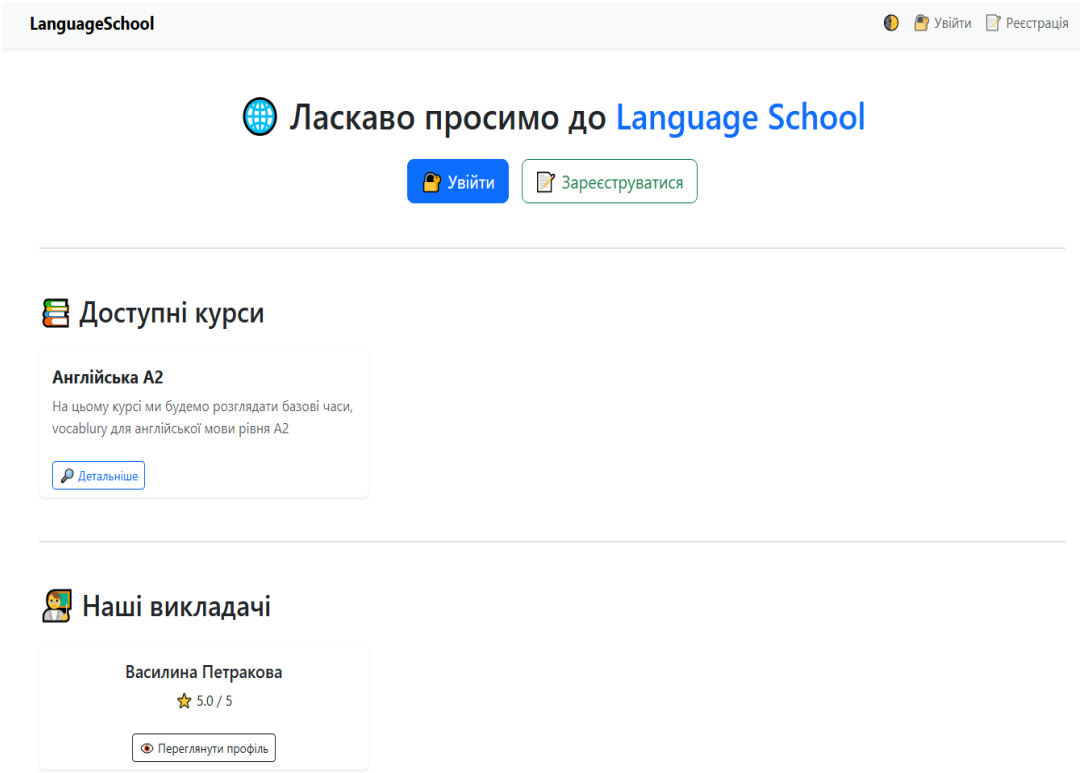


Рисунок 3.2 – Домашня сторінка

У варіанті входу в кабінет, необхідно натиснути на кнопку «Увійти», після чого перекидає на сторінку з формою входу(рис. 3.3). В цій формі доступні для заповнення поля ім’я користувача або email та пароль.

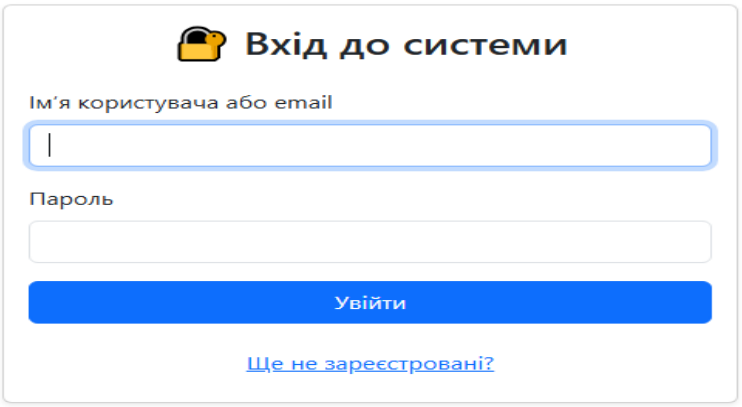


Рисунок 3.3 – Форма логіну користувача

Після натискання на кнопку «Зареєструватися» користувач переходить на форму реєстрації (рис. 3.4). Тут заповнюються поля: ім'я користувача, email, ім'я, прізвище, роль (викладач або студент), пароль та підтвердження пароля. Якщо дані введено некоректно, система виводить відповідне повідомлення про помилку.

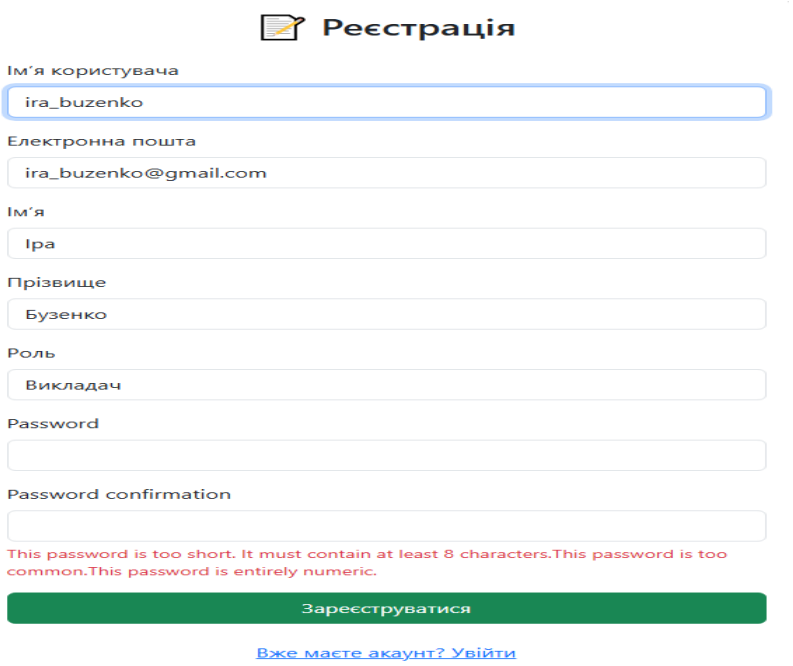


Рисунок 3.4 – Форма реєстрації користувача

Після успішної реєстрації та входу до системи, користувач перенаправляється до особистого кабінету (рис. 3.5). У випадку викладача інтерфейс надає можливість:

– переглянути всі курси,
– створити новий курс,
– додати вільний час для занять,
– переглядати свої бронювання.

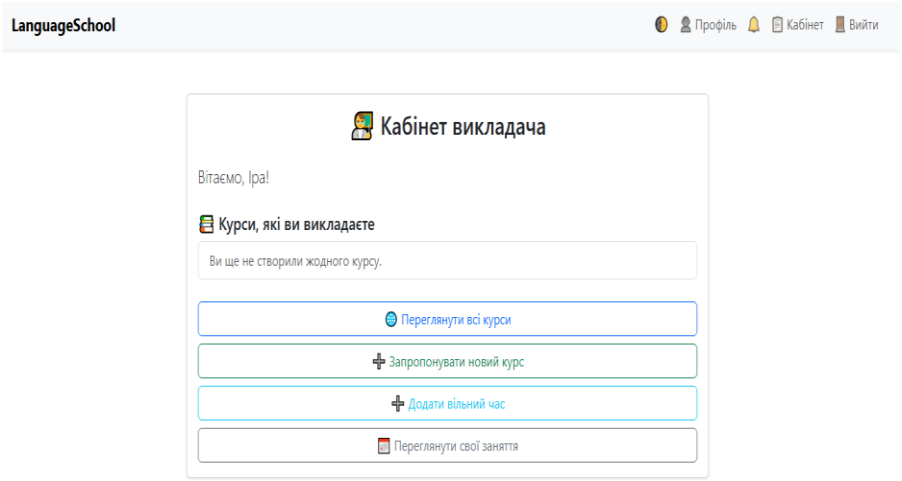


Рисунок 3.5 – Кабінет викладача після входу

При бажанні користувач може переглянути свій профіль (рис. 3.6), де відображається його ім’я, email, роль, коротка біографія та середній рейтинг. Кнопка "Редагувати профіль" відкриває форму редагування (рис. 3.7), у якій можна оновити ім’я, опис, email або завантажити аватар. Після збереження даних профіль оновлюється і зміни відображаються на сторінці, що зображено на рисунку 3.8.

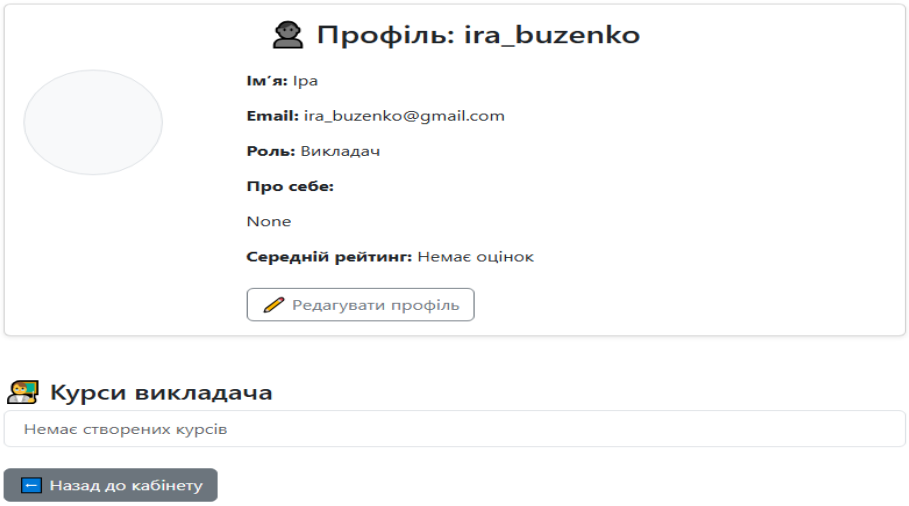


Рисунок 3.6 – Кабінет викладача після входу

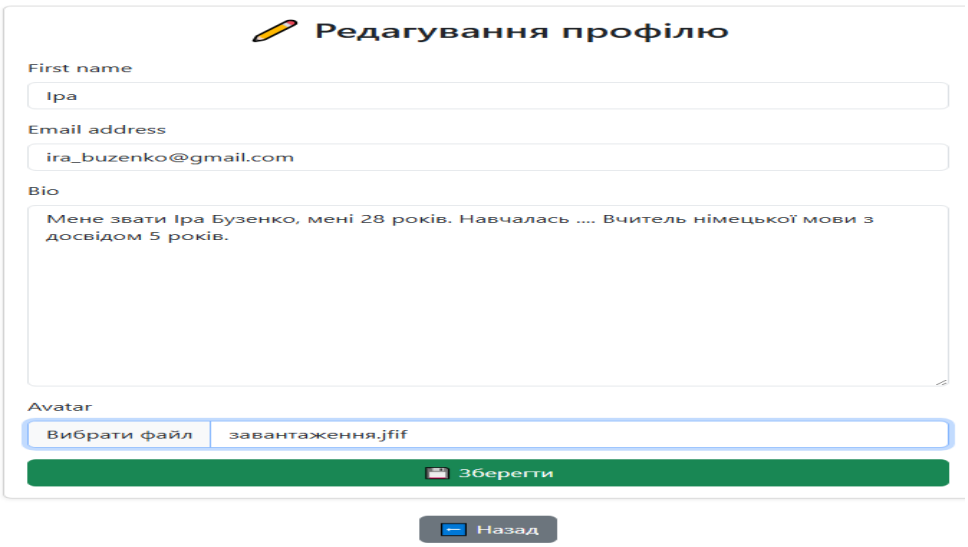


Рисунок 3.7 – Форма редагування профілю

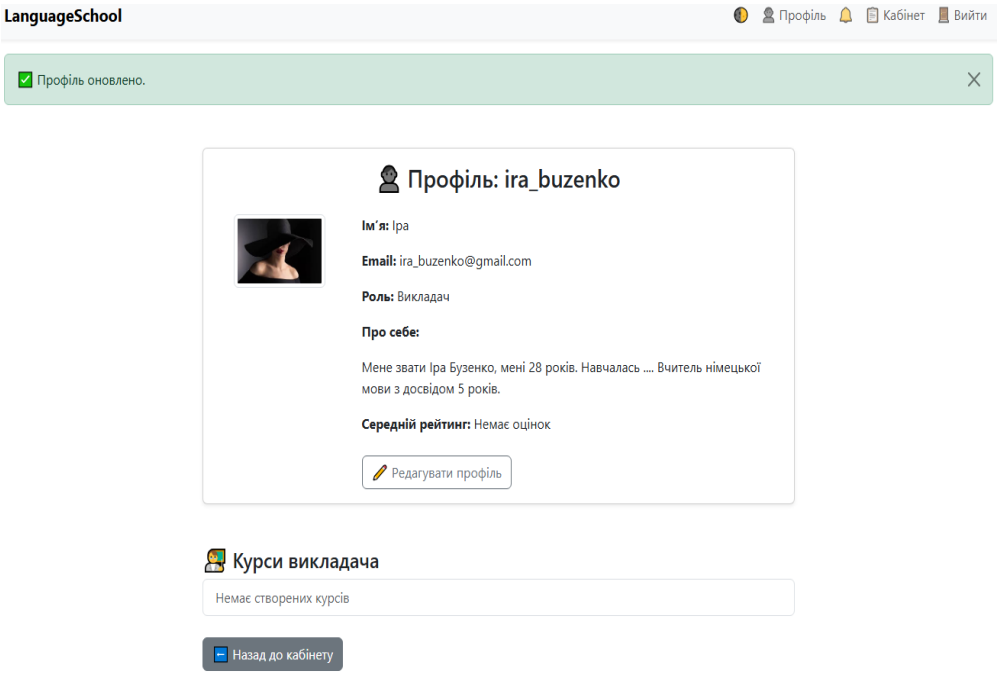


Рисунок 3.8 – Оновлений профіль

Наступним функціоналом розглянемо додавання вільного часу для занять, де викладач має змогу додати вільний час для проведення онлайн-зустрічі (рис. 3.9). У формі заповнюються дата, час, максимальна кількість студентів, посилання на зустріч та контактна інформація. Після збереження ці дані з'являються у

списку занять (рис. 3.10), де відображаються дата, час, статус, контактні дані та можливість редагування або видалення.

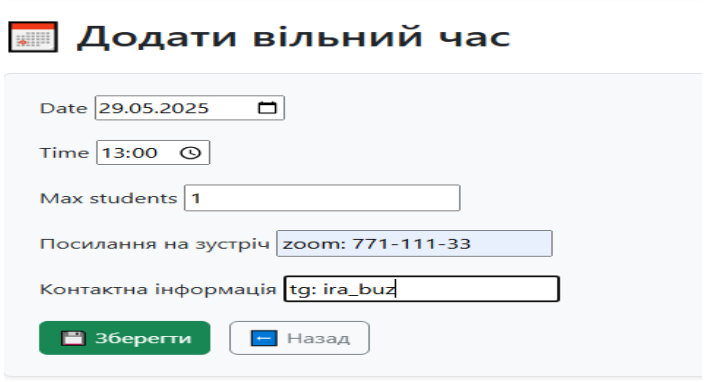


Рисунок 3.9 – Форма додавання вільного слота для заняття

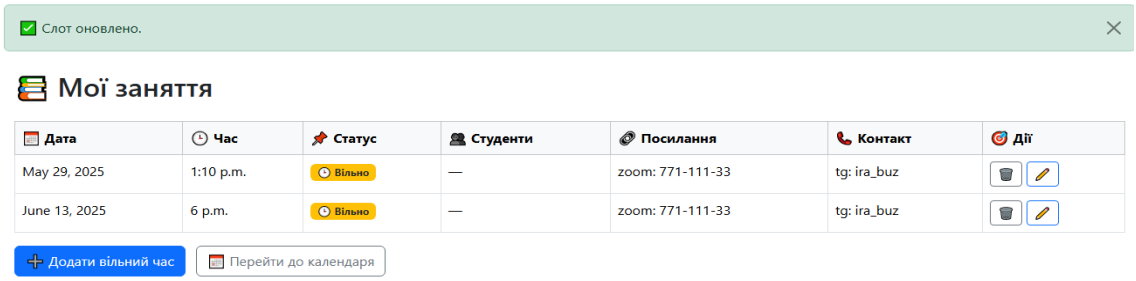


Рисунок 3.10 – Таблиця запланованих занять викладача

Також у особистому кабінеті викладач може запропонувати новий курс (рис. 3.11), нажавши на відповідну кнопку та в вілповідній формі заповнити назву та опис. Після натискання кнопки "Надіслати на підтвердження", відображається відповідне повідомлення (рис. 3.12), курс зберігається в системі та стає доступним для перегляду тільки після схвалення адміністратором.

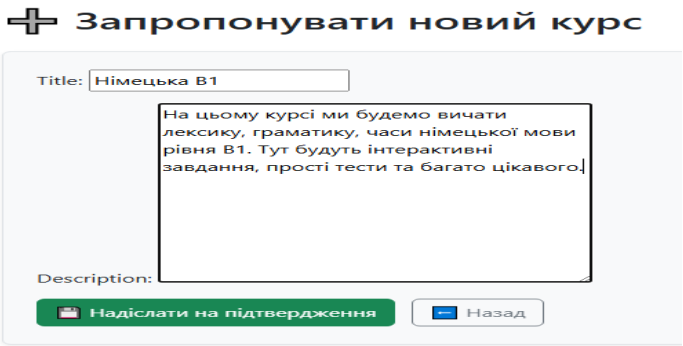


Рисунок 3.11 – Форма додавання курсу

Рисунок 3.12 – Повідомлення про додавання курсу

Далі, адміністратор схвалює курс через адмін панель в налаштуваннях курсу ставить галочку навпроти «is approved» (рис. 3.13), після чого схвалений курс відображається як і для гостів так і для авторизованих користувачів з можливістю до нього доєднатися (рис. 3.14).

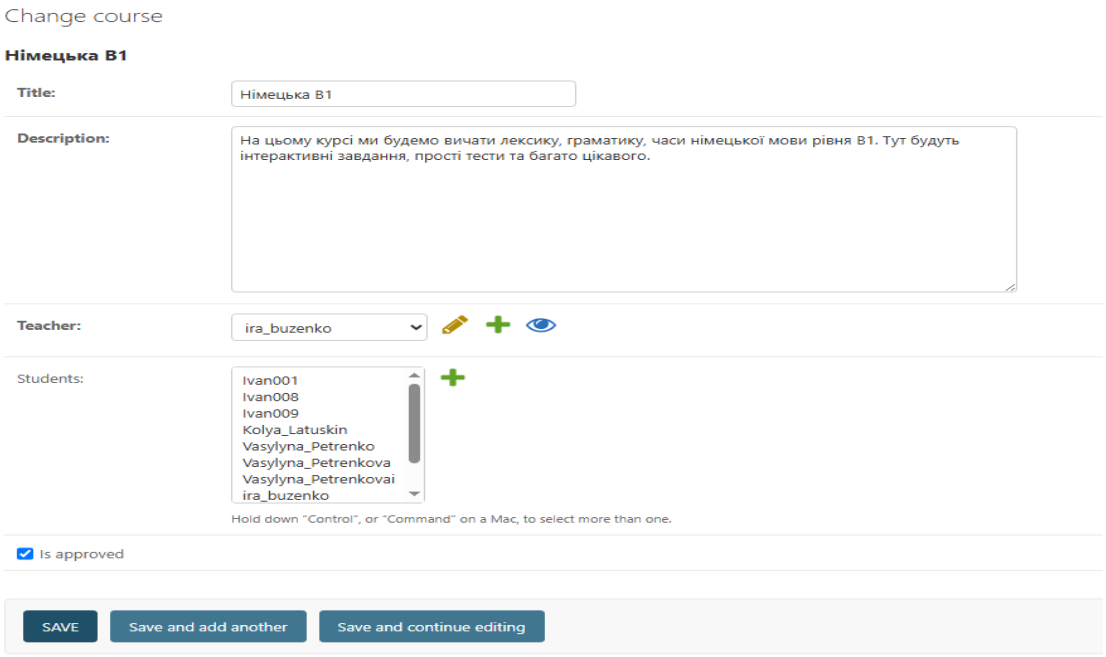


Рисунок 3.13 – Схвалення курсу в адмін панелі

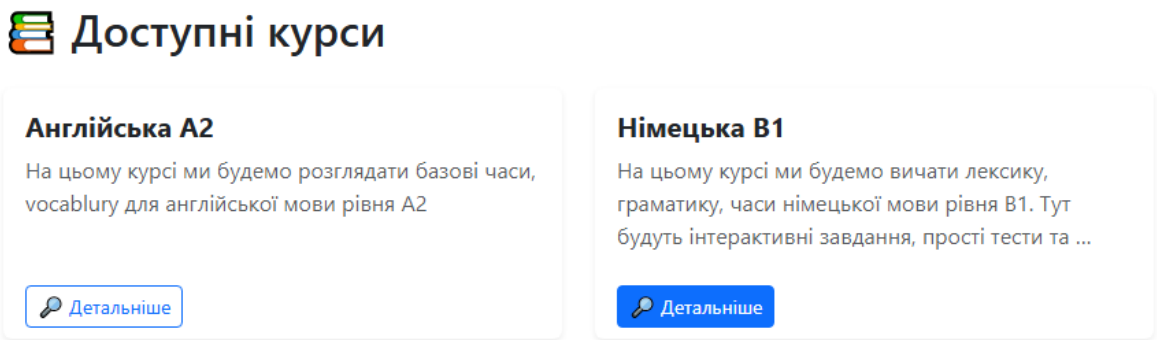


Рисунок 3.14 – Схвалені курси

При переході до детальної сторінки курсу (рис. 3.15), викладач бачить розділи:

– Домашні завдання,
– Тести,
– Заняття курсу.

Звідси можна створити домашнє завдання (рис. 3.16), де вказується назва, опис та дедлайн. Завдання з'являється на сторінці курсу з можливістю редагування або видалення (рис. 3.17).

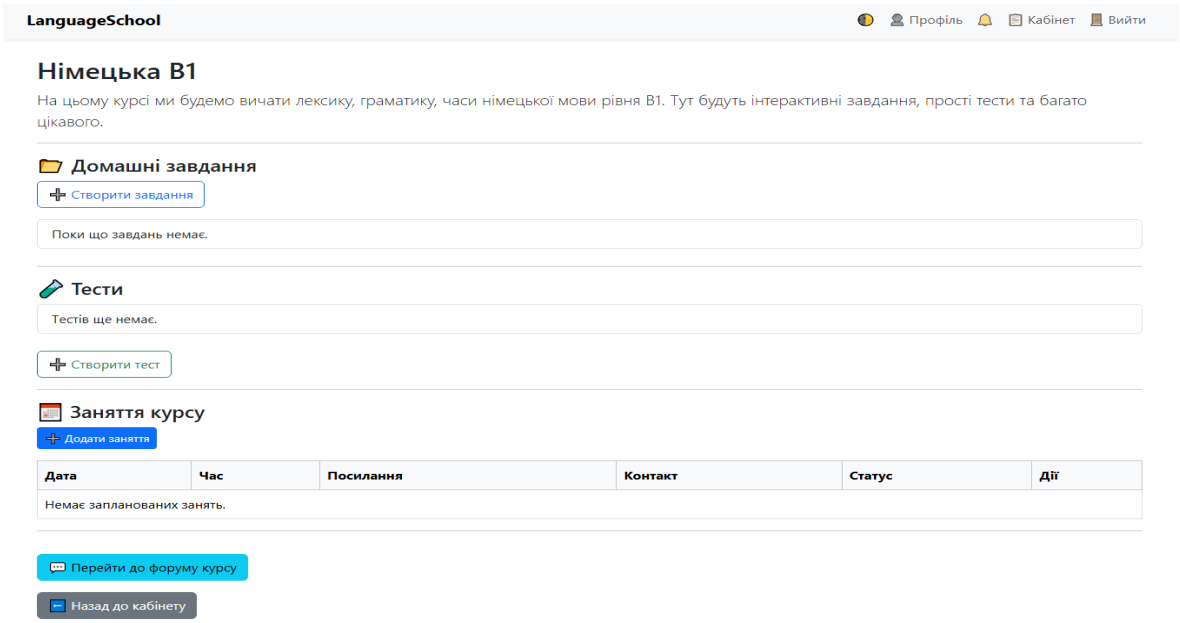


Рисунок 3.15 – Вигляд курсу для викладача

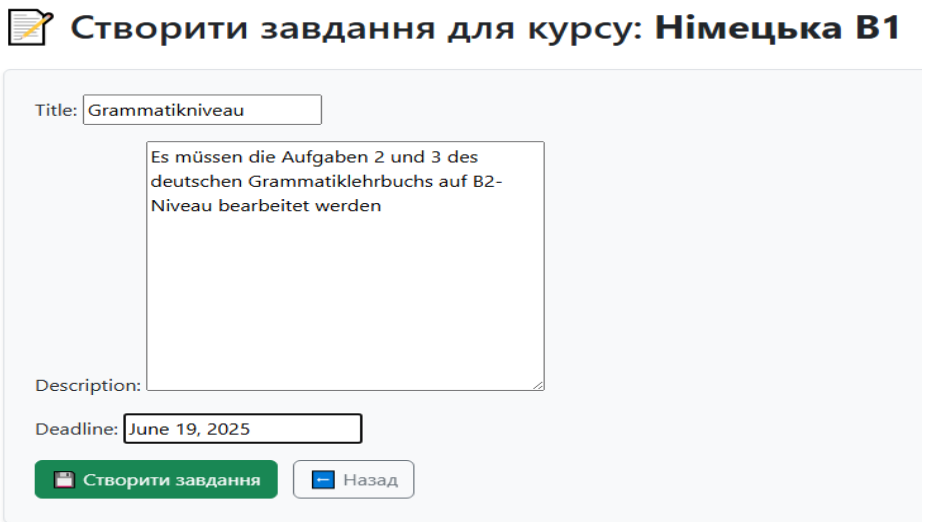


Рисунок 3.16 – Форма створення домашнього завдання

Рисунок 3.17 – Створене завдання на сторінці курсу

Також, викладач може переглянути надіслані відповіді студентів на домашні завдання. Для кожної відповіді можна відкрити файл, виставити оцінку у форматі x/10 та залишити текстовий фідбек. Після натискання кнопки “Зберегти оцінку” дані фіксуються в системі (рис. 3.18).

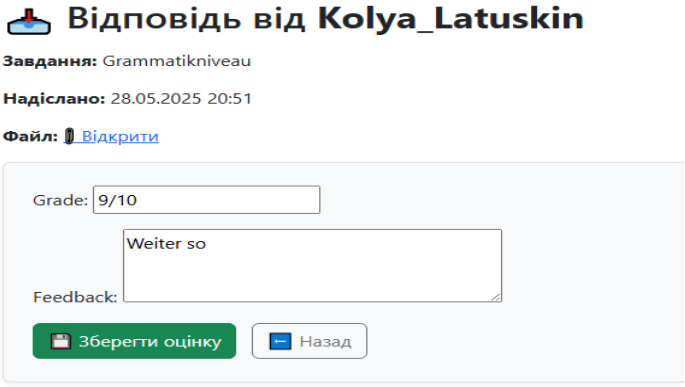


Рисунок 3.18 – Форма оцінювання відповіді

У розділі курсу викладач бачить список усіх студентських відповідей до кожного домашнього завдання. Відображається ім’я студента, файл-відповідь, дата надсилання, оцінка та можливість перейти до детального перегляду (рис. 3.19).

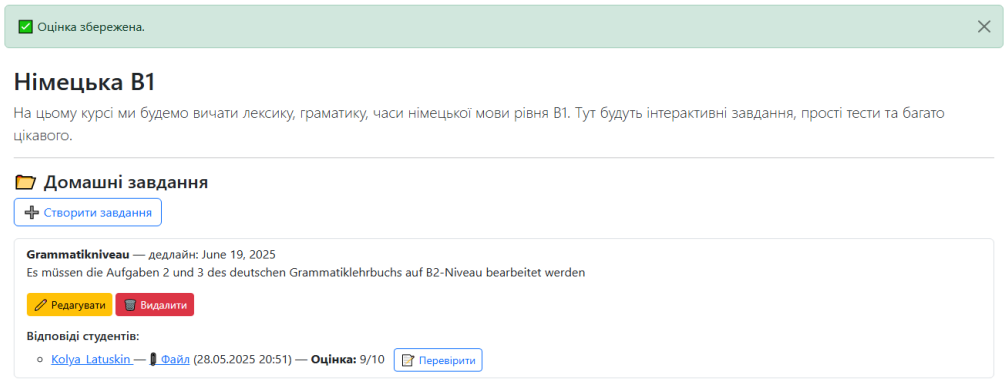


Рисунок 3.19 – Відображення оцінених домашніх завдань студентів

Тест створюється аналогічним чином. Викладач вводить назву та опис (рис. 3.20), додає запитання з варіантами відповіді та вказує правильний варіант (рис. 3.21). У результаті тест відображається у списку з кнопками перегляду результатів, редагування та видалення (рис. 3.22).

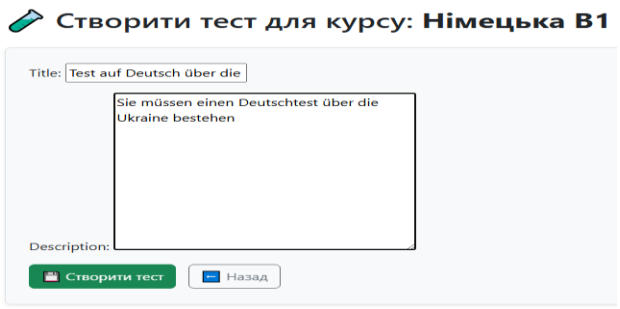


Рисунок 3.20 – Форма створення тесту

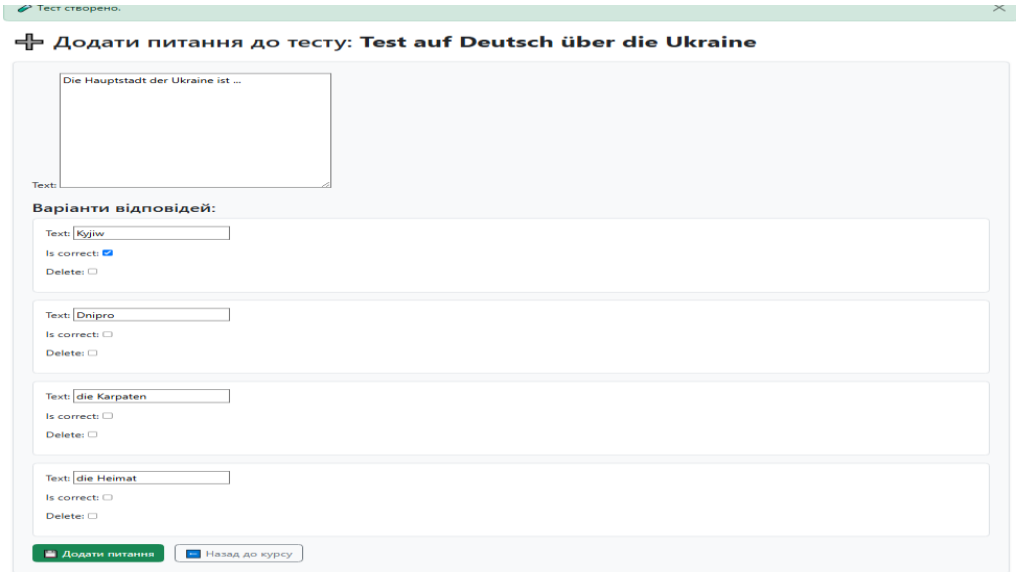


Рисунок 3.21 – Форма додавання запитань до тесту

Рисунок 3.22 – Відображення тесту

Викладач може переглянути результати тестів кожного студента. На сторінці відображаються ім’я студента, набрані бали (у відсотках) та дата проходження тесту (рис. 3.23).

Рисунок 3.23 – Результати проходження тесту студентом

Також у курсі можна добавити слот заняття, так само як і звичайне заняття, де після додавання у деталях курсу будуть відображатися відповідні слоти (рис. 3.24).

Рисунок 3.24 – Відображення занять у курсі

Наступним розглянемо вже інтерфейс студента, де після успішної реєстрації та входу з роллю «Студент», користувач потрапляє до «Кабінету студента». На головній панелі (рис. 3.25) виводиться привітальне повідомлення з іменем користувача, а також перелік його курсів. Нижче розташовані кнопки для перегляду всіх доступних курсів, запису на заняття та перегляду своїх занять.

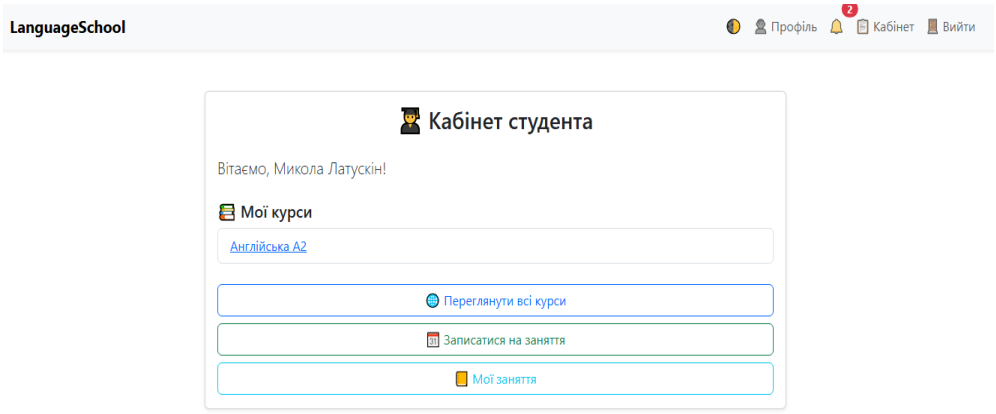


Рисунок 3.25 – Кабінет студента з доступними діями

При натисканні на кнопку «Переглянути всі курси» відкривається список курсів, до яких можна приєднатися. Кожен курс містить опис, ім’я викладача та кнопку «Детальніше» (рис. 3.26).

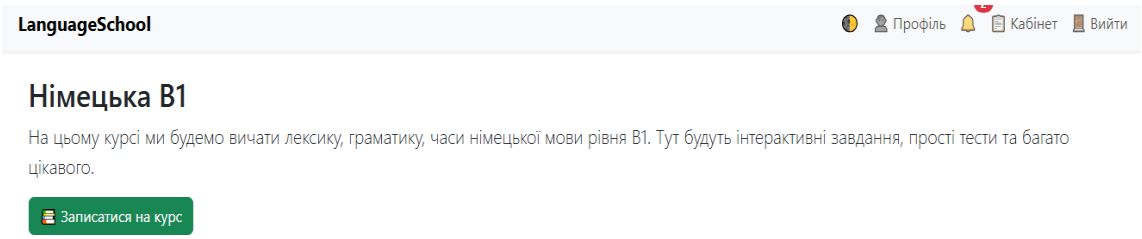


Рисунок 3.26 – Кабінет студента з доступними діями

При натисканні на кнопку «Переглянути всі курси» відкривається список курсів, до яких можна приєднатися. Кожен курс містить опис, ім’я викладача та кнопку «Детальніше» (рис. 3.27).

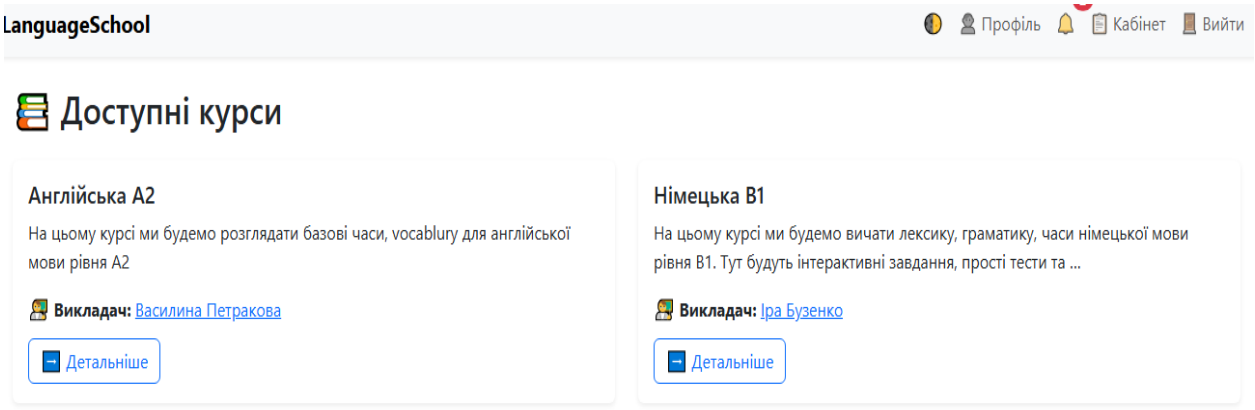


Рисунок 3.27 – Доступні курси

Далі, після перегляду детальної інформації про курс, студент має можливість записатися на нього за допомогою кнопки «Записатися на курс» (рис. 3.28). У разі успішної реєстрації на курс виводиться відповідне повідомлення, і відкривається повна структура курсу (рис. 3.29).

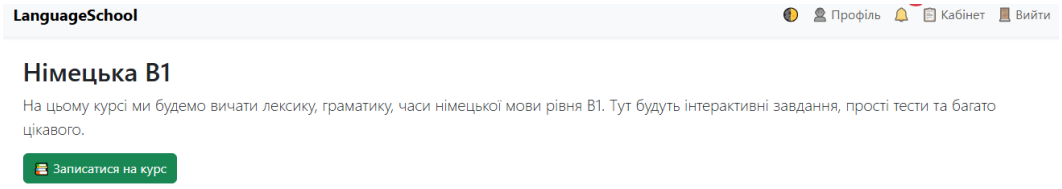


Рисунок 3.28 – Запис на курс

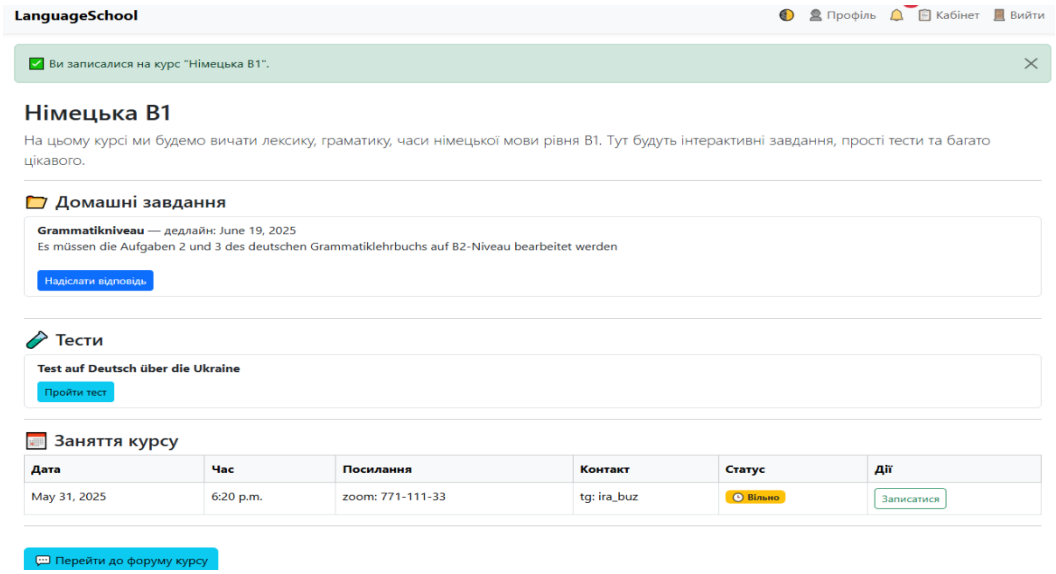


Рисунок 3.29 – Запис на курс

У розділі «Домашні завдання» студент може надіслати свою відповідь, прикріпивши файл. Для цього реалізована форма (рис. 3.30), після чого виводиться підтвердження про успішну відправку (рис. 3.31).

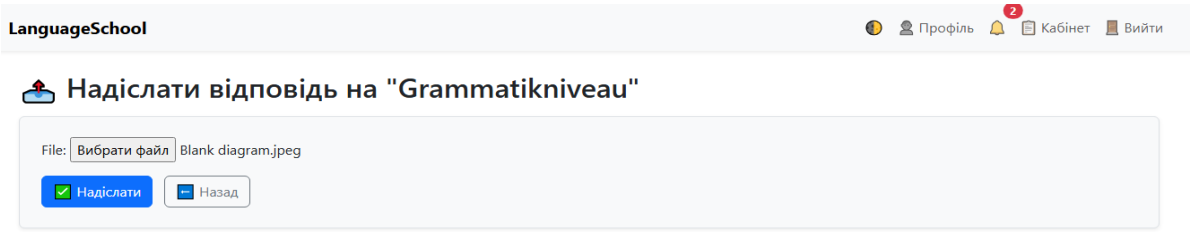


Рисунок 3.30 – Форма надсилання домашнього завдання

Рисунок 3.31 – Завдання із зазначенням, що відповідь була надіслана

У розділі «Тести» студент має можливість пройти тест, натиснувши кнопку «Пройти тест». Інтерфейс тесту реалізований у вигляді запитань із вибором відповіді (рис. 3.33). Після завершення тесту студент бачить відсоток правильних відповідей (рис. 3.32).

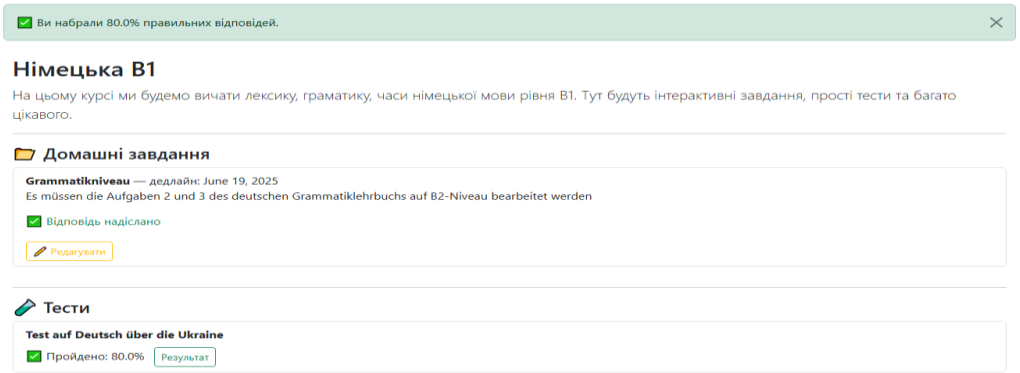


Рисунок 3.32 – Результат проходження тесту: відображення відсотка правильних відповідей

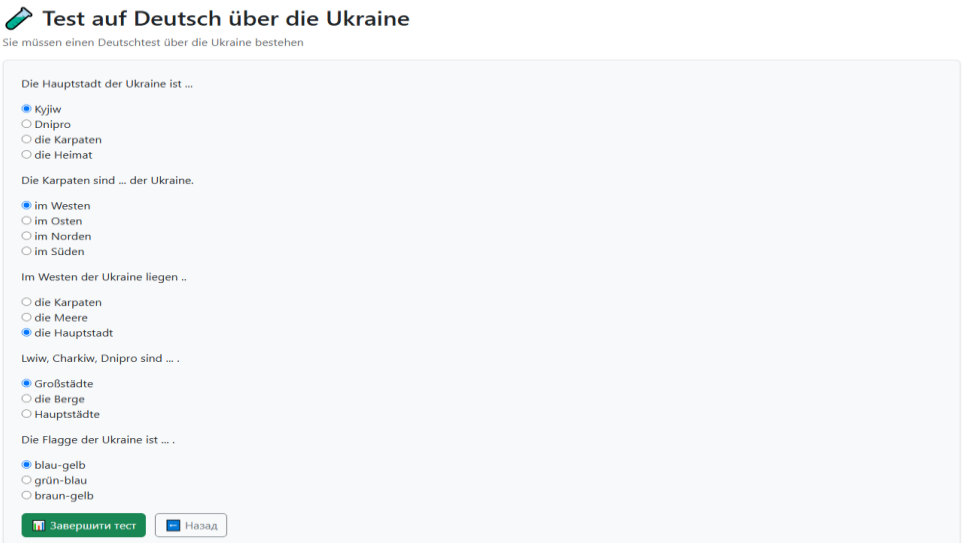


Рисунок 3.33 – Результат проходження тесту: відображення відсотка правильних відповідей

Також, при натисканні кнопки «Записатися на заняття» студент бачить таблицю всіх доступних слотів із зазначенням викладача, дати, часу та кнопкою запису (рис. 3.34). Після натискання з’являється підтвердження про успішний запис.

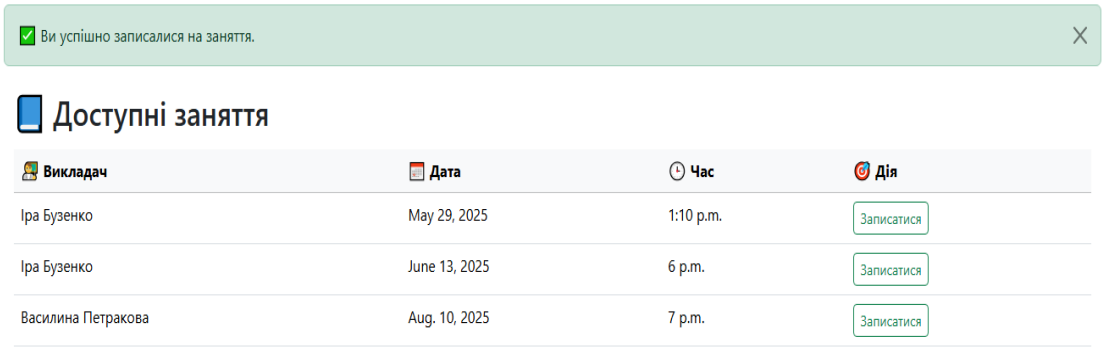


Рисунок 3.34 – Інтерфейс перегляду доступних занять

У розділі «Мої заняття» студент має змогу переглядати перелік записаних занять. Для кожного заняття вказані дата, час, ім’я викладача, статус, а також кнопка для скасування участі (рис. 3.35).

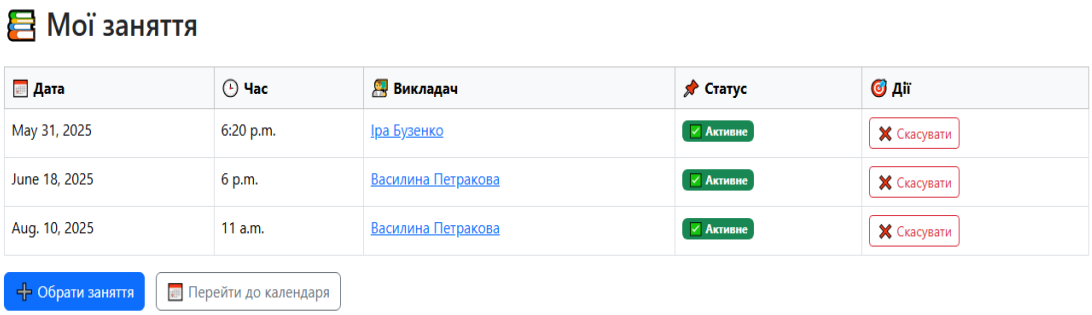


Рисунок 3.35 – Список занять, на які студент записаний

Також реалізовано календар занять, де студент або викладач бачить інтерактивне місячне представлення запланованих подій. При натисканні на заняття відкривається модальне вікно з детальною інформацією: назва заняття, дата та час початку, контакт викладача, посилання на зустріч. Також доступна кнопка скасування участі у занятті (рис. 3.36).

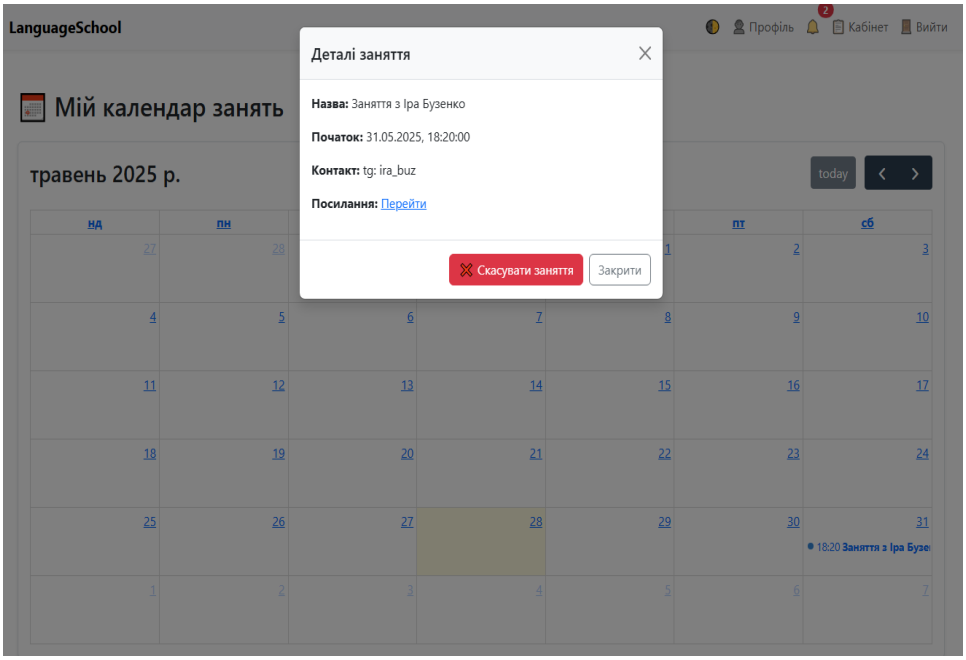


Рисунок 3.36 – Відображення календаря занять із деталями події

Наступним ще розглянемо форум, де відображаються теми та повідомлення учасників. Повідомлення можуть містити запитання до викладача чи обговорення домашніх завдань. Є можливість відповіді на кожне повідомлення (рис. 3.37).

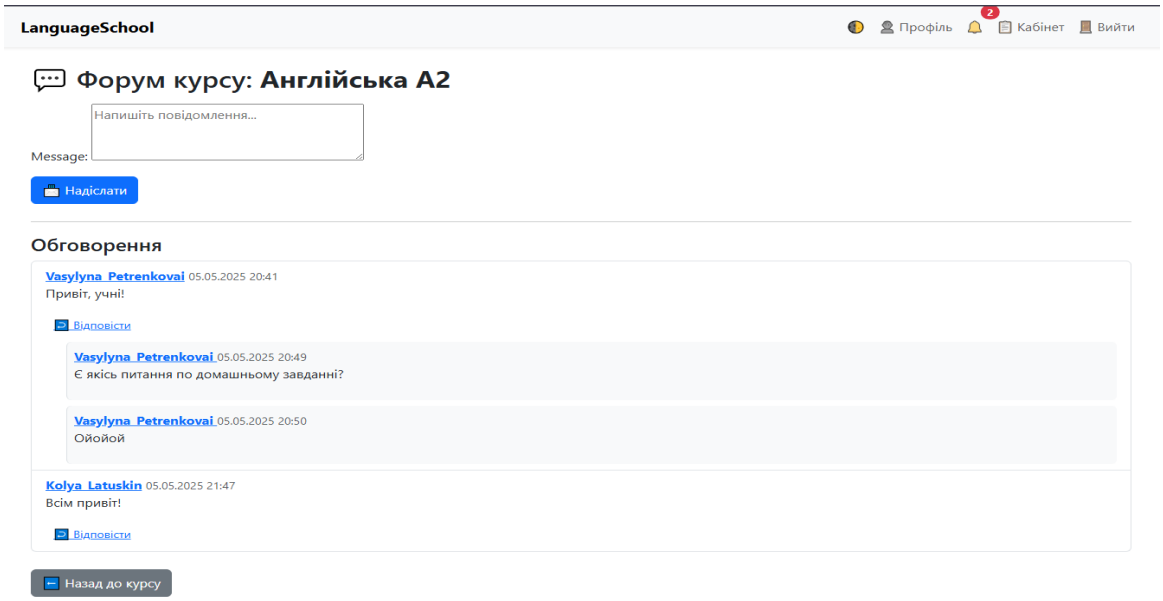


Рисунок 3.37 – Форум курсу

Також не менш важливим є система сповіщень, де у верхній частині під виглядом іконки дзвіночка є підсвітка сповіщень і є можливість перейти на

сторінку зі сповіщеннями. Тут відображаються нові події: створення тестів і завдань у відповідних курсах. Непрочитані сповіщення позначаються міткою “Нове” (рис. 3.38).

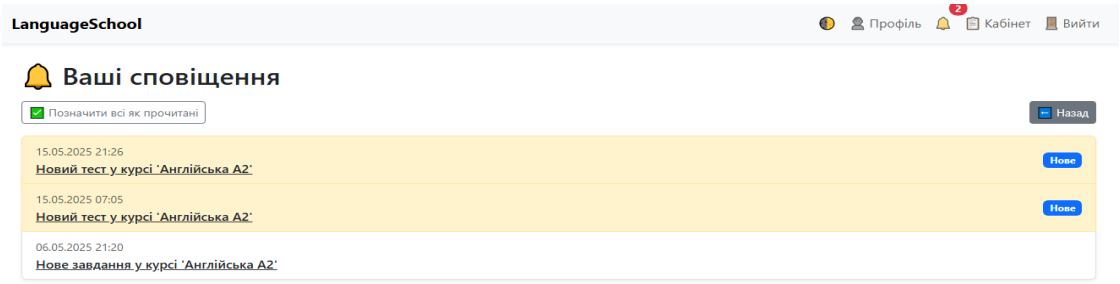


Рисунок 3.38 – Сторінка перегляду сповіщень користувача

Ну і на кінець розглянемо інтерфейс оцінювання викладача. Для цього на сторінці профілю викладача доступна форма оцінювання: вибір балу від 1 до 5 та текстовий коментар (рис. 3.39). Після збереження відгук впливає на відображення середнього рейтингу викладача (рис. 3.40).

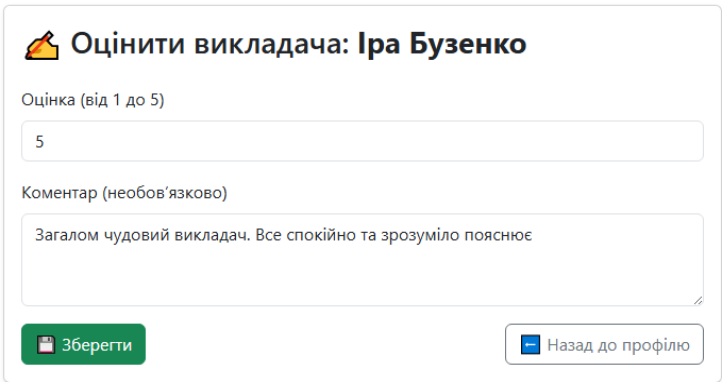


Рисунок 3.39 – Форма оцінювання викладача

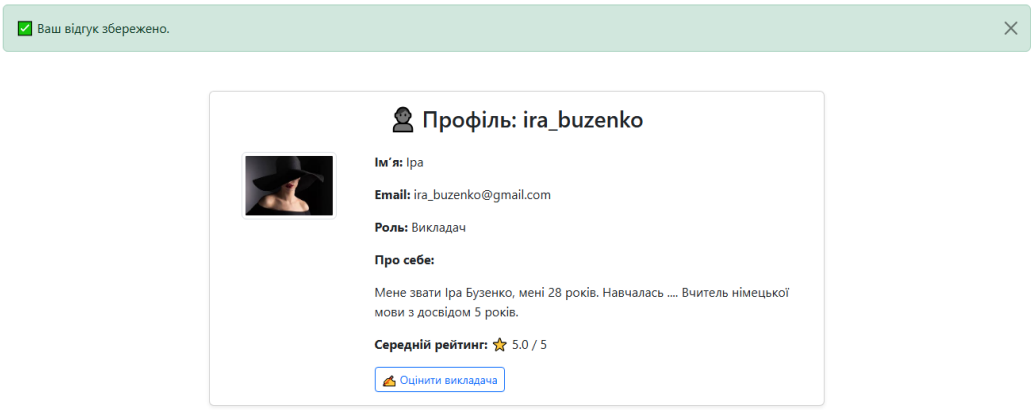


Рисунок 3.40 – Відображення збереженого рейтингу у профілі викладача

Покрокова перевірка інтерфейсу на основі реальних сценаріїв показала, що всі основні модулі функціонують відповідно до технічного завдання. Взаємодія між користувачами забезпечена інтуїтивно зрозумілим інтерфейсом, а інструменти керування навчальним процесом повністю доступні і коректно обробляються.

Таким чином, можна зробити висновок, що веб-сервіс відповідає поставленим функціональним вимогам, а реалізована система є зручною, стабільною та придатною до використання у навчальному середовищі для організації та супроводу мовних курсів.

ВИСНОВКИ

В роботі проведено аналіз предметної області, виокремлена інформаційна система та її інформативні ознаки. Проаналізовані аналоги, вставновлені їх переваги та недоліки. Згідно проведеного аналізу визначено основні функціональні вимоги до веб-сервісу комунікації учасників мовних курсів. Вивчено типові рішення, реалізовані в аналогічних платформах, окреслено роль кожного типу користувача. Також сформульовано головну мету проєкту – створення інтуїтивно зрозумілого, зручного та функціонально повного інструменту для навчального середовища.

Обґрунтовано вибір технологій для реалізації веб-додатку: фреймворку Django на мові Python, бази даних SQLite, HTML/CSS/Bootstrap для побудови інтерфейсу. Розроблено структуру бази даних, реалізовано моделі для зберігання інформації про курси, користувачів, заняття, завдання, тести, повідомлення тощо. Здійснено поділ функціональності між ролями користувачів, описано структуру шаблонів інтерфейсу та навігацію сайтом.

Розроблено і описано клієнтську частину веб-сервісу, а також здійснено перевірку його працездатності. З наведенням скріншотів було продемонстровано покрокову взаємодію з системою: від реєстрації користувача до приєднання до курсу, створення та виконання завдань і тестів, спілкування на форумі, управління розкладом занять, редагування профілю та оцінювання викладачів. Проведена перевірка на працездатність підтвердила коректну роботу всіх компонентів.

Таким чином, у результаті виконаної роботи було створено повнофункціональний веб-сервіс, що забезпечує ефективну взаємодію між викладачами та студентами мовних курсів. Сервіс дозволяє автоматизувати процеси навчання, комунікації, планування занять і контролю знань. Розроблене рішення має потенціал до масштабування та подальшого розвитку відповідно до потреб освітніх закладів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Каграманова Ю. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. *Dou.ua*. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 16.05.2025).
2. Babbel для україни. *Babbel*. URL: https://ua.babbel.com/ (дата звернення: 29.04.2025).
3. Busuu - learn languages online: start for free. *Busuu*. URL: https://www.busuu.com/ (дата звернення: 29.04.2025).
4. Differences between django vs flask - geeksforgeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/differences-between-django-vs-flask/ (дата звернення: 17.05.2025).
5. Django vs spring boot: which is better for your website - geeksforgeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/django-vs-spring-boot/ (дата звернення: 17.05.2025).
6. Dualingo - найкращий у світі спосіб вивчити англійську мову. *Dualingo*. URL: https://www.duolingo.com/learn (дата звернення: 29.04.2025).
7. Gadhavi M. 10 best backend frameworks in 2025. *Radixweb*. URL: https://radixweb.com/blog/best-backend-frameworks (дата звернення: 17.05.2025).
8. Learn with the best online language tutors. *Preply*. URL: https://preply.com/ (дата звернення: 29.04.2025).
9. Rehman A. Django vs. node.js: which is better for web development in 2025?. *Cloudways*. URL: https://www.cloudways.com/blog/django-vs-nodejs/ (дата звернення: 17.05.2025).
10. Rosetta stone® - sprachkurse. *Rosetta Stone® - Sprachkurse*. URL: https://www.rosettastone.com/ (дата звернення: 29.04.2025).
11. Ukrinform. Повномасштабна війна збільшила попит на онлайн-курси -

експерт. *Укрінформ - актуальні новини України та світу*. URL: https://www.ukrinform.ua/rubric-society/3781918-povnomasstabna-vijna-zbilsila-popit-na-onlajnkursi-ekspert.html?utm_source=chatgpt.com (дата звернення: 30.04.2025).

12. Wolfe B. M. Best language learning app of 2025. *TechRadar*. URL: https://www.techradar.com/best/best-language-learning-apps (дата звернення: 29.04.2025).

13. Django ORM (Querysets) · HonKit. *Choose a language · HonKit*. URL: https://tutorial.djangogirls.org/en/django_orm/ (дата звернення: 17.05.2025).

14. Django. *Django Project*. URL: https://www.djangoproject.com/ (дата звернення: 17.05.2025).

15. Про освіту. *Офіційний вебпортал парламенту України*. URL: https://zakon.rada.gov.ua/laws/show/2145-19#Text (дата звернення: 28.04.2025).

16. Про захист персональних даних. *Офіційний вебпортал парламенту України*. URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 28.04.2025).

17. Про електронну комерцію. *Офіційний вебпортал парламенту України*. URL: https://zakon.rada.gov.ua/laws/show/675-19#Text (дата звернення: 28.04.2025).

18. Web Content Accessibility Guidelines (WCAG) 2.1. *W3C*. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 28.04.2025).

19. Common European Framework of Reference for Languages (CEFR). *COE.INT*. URL: https://www.coe.int/en/web/common-european-framework-reference-languages (дата звернення: 28.04.2025).

20. ISO 25010. *PORTAL ISO 25000*. URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 (дата звернення: 28.04.2025).