

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ МОВНОГО
ОБМІНУ МІЖ НОСІЯМИ МОВ

DEVELOPMENT OF WEB APPLICATION FOR LANGUAGE
EXCHANGE BETWEEN NATIVE SPEAKERS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-131

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПЕХОВА М.Д.

(ПРИЗВИЩЕ та ініціали)

Керівник ГЛАДКОВА О.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ДЬЯЧУК Т.С.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПЕХОВОЇ Марії Дмитрівни
 (ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебзастосунку для мовного обміну між носіями мов. Development of Web Application for Language Exchange between Native Speakers

керівник проєкту (роботи) к.т.н., ГЛАДКОВА ОЛЬГА МИКОЛАЇВНА,
 (науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 10 червня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Опис програми. 3. Проєктування вебзастосунку. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтроль	БСЛОВА А.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

(підпис) Марія ПЕХОВА
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Ольга ГЛАДКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 75 с., 5 табл., 32 рис., 2 дод., 17 джерел.

ВЕБЗАСТОСУНОК, PYTHON, JAVASCRIPT, БАЗА ДАНИХ, ПОШУК КОРИСТУВАЧІВ, ОБМІН ПОВІДОМЛЕННЯМИ.

Об'єкт дослідження – програмні засоби для розвитку мовних навичок та культурного обміну.

Предмет дослідження – комп'ютерні програми з пошуку партнерів для знайомств та ведення розмов.

Мета роботи – створення вебзастосунка для швидкого пошуку мовного партнера та спілкуванню через текстові повідомлення.

Матеріали, методи та технічні засоби: функціональне та об'єктно-орієнтоване програмування, мови програмування Python та JavaScript, мови розмітки сторінки HTML та стилів CSS, база даних PostgreSQL, технологія контейнеризації Docker.

Результати. Створено програмний застосунок з простим для використання інтерфейсом для швидкого пошуку користувачів-партнерів з мовного обміну та з можливістю обміну між ними повідомленнями.

Висновки. Розроблений вебзастосунок для мовного обміну забезпечить користувачам зручну платформу для пошуку, відбору за критеріями та спілкуванню з користувачами, що мають між собою спільні цілі.

Галузь використання – освіта, соціалізація та культурний обмін.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 75 pages, 5 tables, 32 figures, 2 appendixes, 17 sources.

WEB APPLICATION, PYTHON, JAVASCRIPT, DATABASE, USER SEARCH, MESSAGING.

Object of research - software tools for the development of language skills and cultural exchange.

The subject of the research is computer programs for finding partners for communication and conversation.

The purpose of the study is to create a web application for quick search of a language partner and communication via text messages.

Materials, methods and techniques: functional and object-oriented programming, Python and JavaScript programming languages, HTML page markup and CSS styles, PostgreSQL database, Docker containerization technology.

Results. A software application with an easy-to-use interface for quick search of users-partners in language exchange and with the ability to exchange messages between them has been created.

Conclusions. The developed web application for language exchange will provide users with a convenient platform for searching, selecting by criteria and communicating with users who share common goals.

Field of application - education, socialization and cultural exchange.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Огляд основних понять предметної області.....	10
1.2 Огляд існуючих аналогів застосунків з мовного обміну	11
1.2.1 Tandem.....	11
1.2.2 HelloTalk.....	12
1.2.3 Mixxer	13
1.2.4 MyLanguageExchange.....	15
1.2.5 Порівняльна таблиця існуючих аналогів.....	16
1.3 Постановка завдання.....	17
1.4 Висновки за розділом 1	17
2 Опис програми.....	19
2.1 Обґрунтування вибору мов програмування	19
2.1.1 Python як мова програмування серверної частини сервісу та порівняння з аналогами	20
2.1.2 JavaScript як мова програмування клієнтської частини сервісу та порівняння з аналогами	22
2.2 Обґрунтування вибору супровідних технологій.....	23
2.2.1 PostgreSQL як реляційна система керування базами даних (СКБД)	23
2.2.2 Docker як платформа контейнеризації програмних продуктів	25
2.3 Аналіз та обґрунтування вибору середовища розробки	26
2.4 Висновки за розділом 2	27
3 Проєктування вебзастосунку	28
3.1 Опис основних вимог до програми	28
3.2 Вимоги до технічних і програмних засобів.....	28

3.3 Архітектура програми.....	29
3.4 Користувацький інтерфейс	32
3.5 Файлова система.....	34
3.6 Висновки за розділом 3	34
4 Експлуатація, тестування та експериментальне дослідження програми.....	36
4.1 Опис застосування проєкту	36
4.2 Інструкція по експлуатації програми	36
4.2.1 Звернення до програми.....	36
4.2.2 Вхідні й вихідні дані	42
4.2.3 Тестування програмного забезпечення.....	42
4.3 Висновки за розділом 4	43
Висновки.....	44
Перелік джерел посилання	45
Додаток А Текст програми	47
Додаток Б Слайди презентації	69

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
CPU	– Central Processing Unit;
CSS	– Cascading Style Sheets;
gRPC	– Google Remote Procedure Call;
HTML	– HyperText Markup Language;
IDE	– Integrated Development Environment;
JS	– JavaScript;
NoSQL	– non Structured Query Language;
NPM	– Node Package Manager;
OC	– Objective-C;
ORM	– Object-relational mapping;
PyPI	– Python Package Index;
RAM	– Random Access Memory;
SQL	– Structured Query Language,
SPA	– Single Page Application;
T-SQL	– Transact Structured Query Language;
UI	– User Interface;
СКБД	– Систéма управління базами даних.

ВСТУП

В сучасному світі процес вивчення іноземних мов зазнав суттєвих трансформацій, зумовлених розвитком цифрових технологій та зростанням міжнародної взаємодії. Застосунки для мовного обміну забезпечують користувачам можливість безпосередньої комунікації з носіями мови в режимі реального часу, що значно підвищує ефективність мовної практики. Такі платформи є особливо цінними як для початківців, які прагнуть засвоїти базові навички, так і для просунутих користувачів, зацікавлених у самовдосконаленні, але вже у контексті автентичного мовного середовища. Крім лінгвістичних переваг, ці сервіси сприяють глибшому розумінню культурних особливостей та формуванню міжособистісних зв'язків.[1].

Виходячи з цього, розробка програмного продукту для пошуку мовних партнерів є дійсно актуальною, не зважаючи на велику кількість конкурентів та аналогів у сфері. Тому головною метою цієї роботи є створення програмного забезпечення у вигляді вебзастосунку, що дозволить користувачу швидко знаходити, відповідних його запиту, співрозмовників та забезпечить ненапружене спілкування.

Задача цього проєкту полягає у розробці продукту, що буде включати в себе:

- систему створення особистого профілю, з можливістю публікування інформації про себе;
- легку систему пошуку профілів інших користувачів за критеріями мов;
- реалізований функціонал для спілкування через особисті повідомлення, у вигляді чату.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд основних понять предметної області

Мовний обмін – це засіб для практичного використання мови. Він передбачає, що двоє людей, які розмовляють різними мовами, збираються разом на заздалегідь визначену кількість сесій, щоб кожен з них міг попрактикуватися у використанні мови іншого. Мовні обміни доповнюють формальне вивчення мови. Вони в основному служать для застосування усних мовних навичок на практиці. Крім того, мовні обміни є гарною можливістю отримати досвід і уявлення про аспекти інших культур [1].

Сесії мовного обміну можуть бути особистими або віртуальними, або поєднанням обох.

Онлайн-мовний обмін – це взаємодія між учасниками через інтернет, здебільшого за допомогою спеціалізованих платформ, месенджерів або відеозв'язку. Найголовнішими перевагами такого формату над іншими є:

- гнучкість, можна займатися у зручний час і в будь-якому місці;
- доступність, не довго потрібно шукати носіїв мови, а тим паче обмежуватись власним містом;
- вибір партнерів, є велика кількість користувачів із різними інтересами та рівнями мови;
- економія часу та коштів, немає потреби їздити на зустрічі з викладачем;
- практика живої мови, з усіма виразами, акцентами, інтонацією.

Зазвичай учасники самотійно, враховуючи потреби одне одного, домовляються про формат: скільки часу говорять однією мовою і скільки іншою; який стиль спілкування, вільний діалог, обговорення тем, розбір текстів тощо. Спілкування може бути як у реальному часі (відео- та аудіодзвінки), так і лише за допомогою обміну голосовими або текстовими повідомленнями у зручний час.

Об'єктом дослідження є програмні засоби для розвитку мовних навичок та культурного обміну. Предметом дослідження є функціональна та зручна платформа у вигляді вебзастосунку, яка надає користувачам як можливість ефективного пошуку партнерів самостійно, так і спосіб ведення розмов за допомогою текстового обміну.

1.2 Огляд існуючих аналогів застосунків з мовного обміну

1.2.1 Tandem

Tandem – це мобільний застосунок для мовного обміну, який дозволяє людям з усього світу практикувати іноземні мови у спілкуванні з носіями. Створений у 2015 році, сервіс швидко набув популярності серед різних категорій, що прагнуть удосконалити свої мовні навички завдяки реальному спілкуванню. Tandem доступний як у вигляді мобільного застосунку, так і у вебверсії, забезпечуючи користувачам гнучкий доступ до платформи з будь-якого пристрою [2].

Основною метою застосунку є забезпечення можливості міжкультурного мовного обміну шляхом безпосереднього спілкування між користувачами. Після реєстрації користувачі обирають мову, яку хочуть вивчати, та мову, якою володіють, після чого можуть знаходити співрозмовників. Кожен профіль у Tandem містить різноманітну корисну інформацію про користувача, що допомагає підібрати найбільш відповідного партнера для мовного обміну (рис. 1.1). Спілкування відбувається через текстові повідомлення, аудіо- та відеодзвінки, що дозволяє тренувати різні аспекти мови - як письмову, так і усну.

Серед найголовніших переваг Tandem можна виділити те, що ця платформа підтримує понад 300 мов, що є найбільшою кількістю серед аналогів і включає як популярні, так і рідкісні мови, та навіть мови жестів. Крім того, сервіс надає інструменти для інтерактивного навчання, а також забезпечує модерацію профілів та має систему безпеки, яка зменшує ризик

зловживань або неприйнятної поведінки з боку користувачів. Завдяки таким характеристикам Tandem став однією з провідних платформ у сфері неформального мовного навчання, поєднуючи сучасні технології з живим людським спілкуванням.

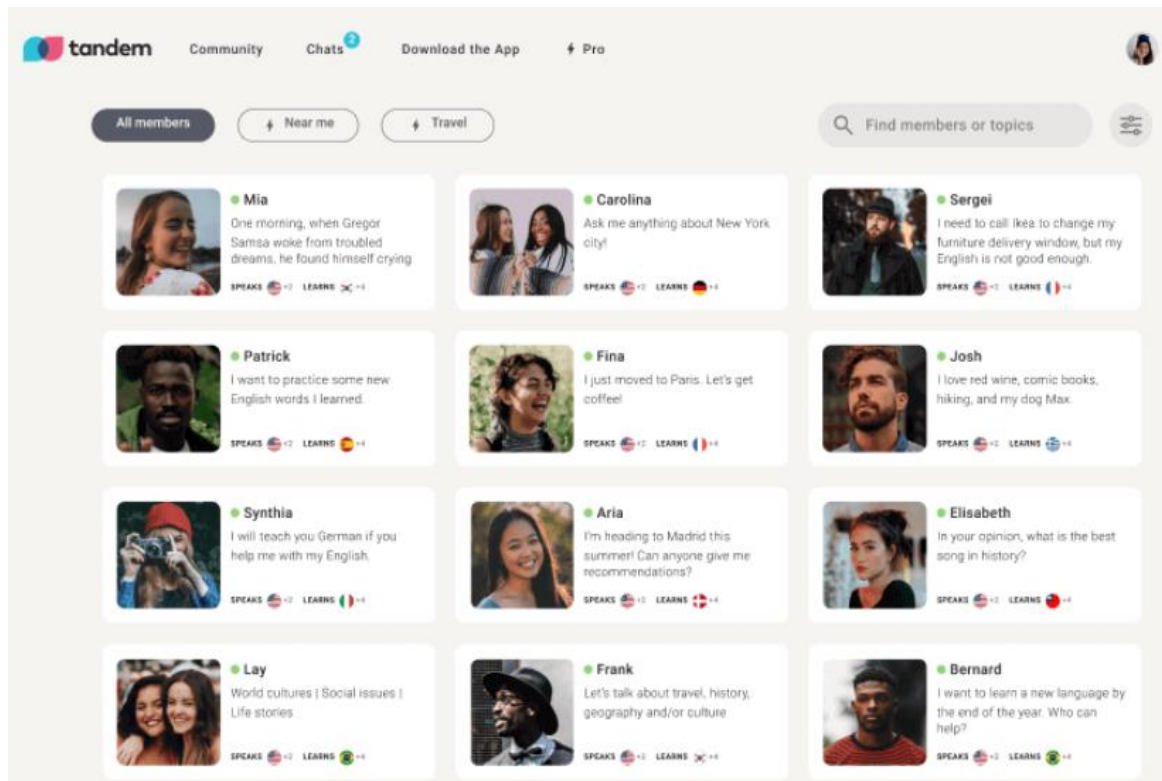


Рисунок 1.1 – Сторінка результатів пошуку у сервісі Tandem [5]

1.2.2 HelloTalk

HelloTalk – це платформа для вивчення іноземних мов, що функціонує у форматі соціальної мережі [3]. Заснована у 2012 році, вона орієнтована на організацію безпосереднього мовного обміну між носіями різних мов у всьому світі. HelloTalk доступна у вигляді мобільного застосунку для iOS та Android, що забезпечує широкий доступ до сервісу в режимі реального часу (рис. 1.2).

Алгоритм застосунку підбирає відповідні мовні пари, які можуть обмінюватися текстовими, голосовими повідомленнями та здійснювати дзвінки. Інтерфейс користувача включає детальний профіль з інформацією про мовні компетенції, інтереси, місцезнаходження (за бажанням) і коротку

біографічну довідку, що спрощує підбір релевантного партнера для комунікації.

До основних функціональних можливостей належать:

- підтримка понад 150 мов;
- вбудовані інструменти перекладу, транслітерації та голосового синтезу;
- тематичні спільноти (Moments) для обговорення культурних, освітніх та мовних питань.

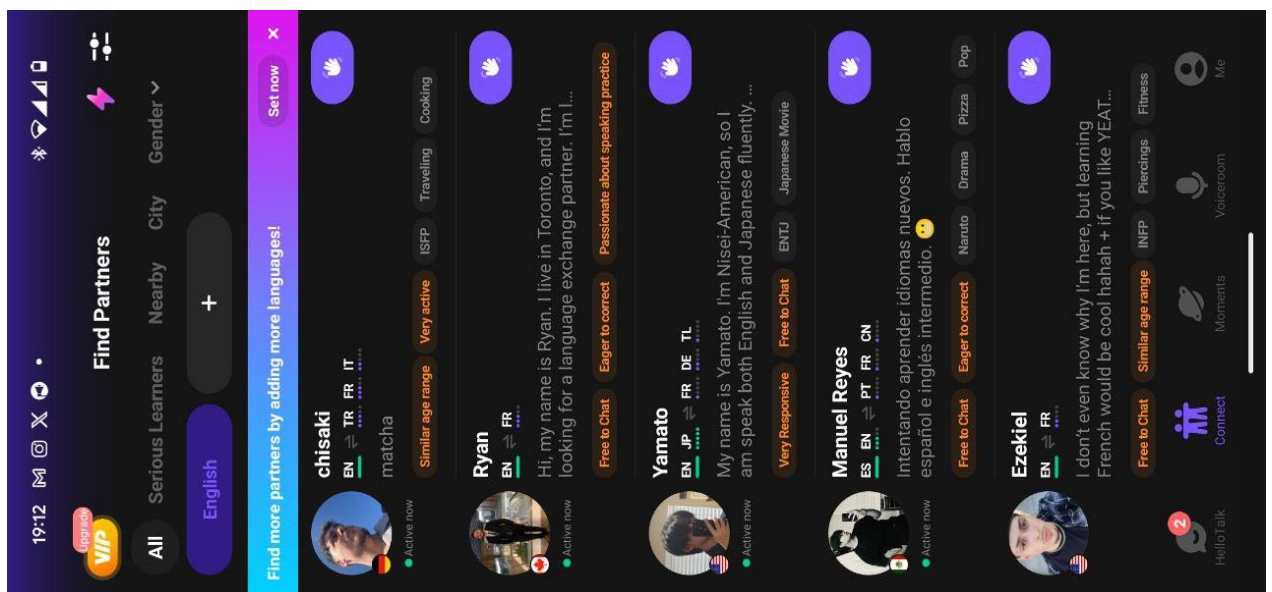


Рисунок 1.2 – Сторінка результатів пошуку партнерів у HelloTalk [6]

1.2.3 Mixxer

The Mixxer – це безкоштовна онлайн-платформа для мовного обміну, орієнтована на освітнє використання та академічне середовище. На відміну від комерційних мобільних застосунків, The Mixxer є веб-ресурсом із відкритим доступом, що підтримується академічною спільнотою. Основна мета платформи – створення умов для взаємонавчання шляхом мовного обміну між носіями мови та тими, хто її вивчає [4].

Платформа активно використовується викладачами як допоміжний інструмент для організації міжкультурної комунікації в межах навчальних

курсів. Користувачі реєструються на сайті, вказують мови, які вони знають, і мови, які прагнуть вивчати. Після цього вони можуть шукати мовних партнерів і налагоджувати контакт через текстові повідомлення або домовлятися про розмови в реальному часі за допомогою сторонніх сервісів, що зазначаються в профілі. Особливості The Mixxer:

- платформа некомерційна та відкрита для всіх охочих, але має чітку орієнтацію на академічне середовище;
- кожен профіль містить детальну інформацію про рівень володіння мовами, цілі вивчення та бажаний формат обміну;
- можливість участі у спеціальних мовних заходах, що проводяться у співпраці з навчальними закладами.

The Mixxer вирізняється акцентом на довгострокове освітнє партнерство, а не на швидкий обмін повідомленнями. Саме тому платформа здобула визнання серед викладачів іноземних мов і студентів, які шукають можливість регулярної мовної практики (рис. 1.3).

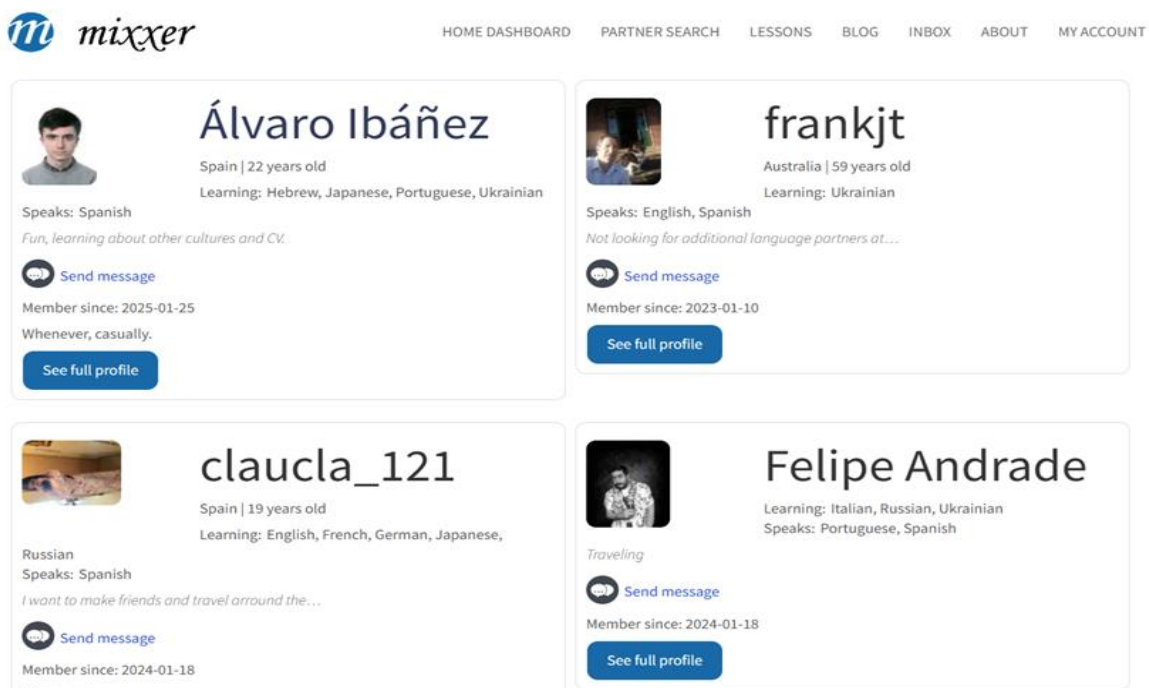


Рисунок 1.3 – Сторінка результатів пошуку Mixxer [11]

1.2.4 MyLanguageExchange

MyLanguageExchange – це одна з найперших веб-платформ для мовного обміну, створена у 2000 році з метою сприяти міжкультурному спілкуванню та взаємному навчанню іноземним мовам. Сервіс орієнтований на довготривалу комунікацію між користувачами, які прагнуть удосконалити мовні навички через живе спілкування з носіями мови [5].

Після реєстрації користувачі створюють профіль, зазначаючи мови, якими володіють, мови, які хочуть вивчати, рівень підготовки, інтереси та переваги щодо формату спілкування. Комунікація може здійснюватися за допомогою електронної пошти (pen-pal), голосових повідомлень або через живі розмови за допомогою сторонніх сервісів, таких як Skype, Zoom чи інші.

На відміну від мобільних застосунків, MyLanguageExchange не має офіційного мобільного інтерфейсу, але зберігає популярність завдяки своїй орієнтації на серйозне та методично виважене мовне навчання. Платформа активно використовується як індивідуальними користувачами, так і викладачами, які залучають її для позааудиторної практики студентів (рис. 1.4).

The screenshot displays the 'Members Search' interface on the MyLanguageExchange website. At the top, there is a navigation bar with links: 'How to', 'Penpals', 'Search', 'Teachers', 'Word Games', 'Chat', and 'Contact Us'. Below this, the main heading is 'Members Search - Find a Language Exchange Partner'. The search form is titled 'Members Search' and contains the following elements:

- A prompt: 'Find a language exchange partner who has:'
- 'Native language (required)': A dropdown menu with '- All -' selected.
- 'Practicing language': A dropdown menu with '- All -' selected.
- 'Country': A dropdown menu with '- All -' selected.
- 'City': A text input field with 'Optional' below it and a link 'Insert Accents'.
- A tip: 'Tip: use language versions of the city name. For example, search for Montreal and Montréal. Search for Rome and for Roma.'
- 'Age': A range selector with 'From: Optional' and 'To: Optional'.
- 'Gender': A dropdown menu with '-- All --' selected.
- 'Is a classroom': A dropdown menu with '-- All --' selected.
- 'Seeking a penpal exchange (via email)': A dropdown menu with '-- All --' selected.
- 'Seeking a live text chat exchange': A dropdown menu with '-- All --' selected.
- 'Seeking a voice chat exchange': A dropdown menu with '-- All --' selected.
- 'Seeking a live exchange in person': A dropdown menu with '-- All --' selected.
- 'First Name': A text input field with 'Optional' below it.
- 'Description contains:': A text input field with 'Optional' below it.
- 'Sort results by:': A dropdown menu with 'Login Date' selected.
- 'Save this search as:': A text input field with 'Optional' below it.
- A green 'Search' button at the bottom right.

Рисунок 1.4 – Сторінка пошуку сайту MyLanguageExchange [11]

1.2.5 Порівняльна таблиця існуючих аналогів

Вище були згадані одні з найвідоміших вебзастосунків для мовного обміну, що надають можливості пошуку та спілкування з іншими користувачами. Проте, крім ряду переваг, кожен з сервісів має свої недоліки.

Нижче представлена порівняльна таблиця цих програмних продуктів (табл. 1.1).

Таблиця 1.1 – Порівняння програмного забезпечення для пошуку партнерів для мовного обміну

Критерії для порівняння	Існуючі аналоги			
	Tandem	HelloTalk	The Mixxer	MyLanguageExchange
Наявність інтерфейсу веб-платформи	+	-	+	+
Наявність мобільного інтерфейсу	+	+	-	-
Наявність пошуку з фільтрами	+	+	+	+
Детальний опис профілів користувачів	+	-	+	+
Наявність платних розширених версій	+	+	-	-
Наявність чатів для обміну повідомленнями	+	+	+	+
Пряма можливість для переносу спілкування на інші ресурси	-	-	+	+
Кількість підтримуваних мов	300	180+	100+	163

1.3 Постановка завдання

Після вивчення предметної області та аналізу наявних успішних реалізацій для вирішення її проблем, сформулюємо технічне завдання.

Метою цього проєкту є розробка програмного забезпечення у формі вебзастосунку, до якого можемо виділити ряд функціональних вимог:

- можливість створення користувачем особистого профілю, зі стислою, але необхідною інформацією;
- реалізація пошуку, у якому буде легко орієнтуватись та знаходити підходящих користувачів;
- наявність чатів для швидкого обміну текстовими повідомленнями;
- надавати можливість користувачам швидкого перенесення спілкування на інші ресурси.

Тож система повинна бути спроектована з урахуванням можливостей майбутнього розширення та вдосконалення.

У процесі розробки інтерфейсу слід орієнтуватися на користувацькі потреби та забезпечити зручність взаємодії з системою. Це передбачає створення не лише інтуїтивно зрозумілого, а й продуктивного інтерфейсу. Він має надавати користувачам можливість без зусиль виконувати потрібні операції, не заважаючи концентрації на головних завданнях при роботі з додатком чи системою.

1.4 Висновки за розділом 1

У цьому розділі була проаналізована предметна область, визначені основні поняття, а також, для глибшого розуміння контексту, були досліджені найпопулярніші програмні альтернативи розроблюваного програмного проєкту. Аналіз їх функціоналу, визначення сильних та слабких сторін – базове розуміння подальшого розвитку проєкту.

Було сформульовано і визначено технічне завдання до роботи, що описує ключові функціональні можливості майбутнього проекту.

2 ОПИС ПРОГРАМИ

Вебзастосунок – являє собою специфічний різновид програмного забезпечення, створення якого стало можливим завдяки бурхливому зростанню мережі Інтернет, та функціонує за принципом архітектури "клієнт-сервер". Ключова відмінність полягає в тому, що сам вебдодаток розміщується та працює на серверній частині, тоді як клієнтська сторона одержує лише кінцеві результати обчислень. Функціонування вебсистеми базується на прийомі користувацьких запитів (від клієнта), їхній обробці та поверненні відповідних результатів. Транспортування запитів від клієнтської частини до серверної та отримання оброблених даних здійснюється через мережу Інтернет.

Кожен такий додаток відтворюється на дисплеї комп'ютера чи мобільного пристрою у вигляді веб-сторінки. Вона містить не лише інформаційний контент, а й інтерактивні елементи управління: кнопки, діалогові вікна та інші компоненти, що забезпечують виконання потрібних операцій для досягнення поставлених цілей.

2.1 Обґрунтування вибору мов програмування

При виборі мови програмування для створення проекту слід брати до уваги комплекс критеріїв, зокрема: швидкодію програмного продукту, ефективність роботи розробника, наявність необхідного інструментарію, рівень захищеності, можливості розширення системи та багато інших аспектів. Визначення мови програмування для розробки як серверної, так і клієнтської частини становить один із найважливіших кроків у процесі створення програмного забезпечення.

Тому, основними мовами програмування для розробки вебзастосунку були обрані: Python (із фреймворком Flask) для серверної частини застосунку

та JavaScript для клієнтської частини. Для побудови вебінтерфейсу використовується HTML/CSS із підтримкою шаблонізатора Jinja2.

Додатково, для збереження даних у реляційній базі даних застосовується SQL (через ORM SQLAlchemy) у зв'язці з PostgreSQL.

2.1.1 Python як мова програмування серверної частини сервісу та порівняння з аналогами

Python – це інтерпретована, високорівнева мова програмування загального призначення, яка вирізняється простим, читабельним синтаксисом, великою кількістю бібліотек та активною спільнотою [6]. У рамках розробки програмного продукту обрана як основна мова для розробки серверної частини застосунку на базі фреймворку Flask.

Flask є легким мікрофреймворком для створення вебзастосунків, який не нав'язує структури проєкту, дозволяє гнучко конфігурувати компоненти та швидко запускати готові прототипи [7]. Його використання дозволяє розробляти маршрути, обробляти запити, формувати динамічний вміст сторінок та організовувати роботу з базою даних за допомогою ORM.

Python також чудово підходить для реалізації асинхронної логіки (через Flask-SocketIO), що є критично важливою для функціонування частини майбутнього вебзастосунку з онлайн-чатом в реальному часі. Серед ключових переваг Python:

- простота у використанні та навчанні;
- активна підтримка та наявність тисяч готових рішень (наприклад, Flask, SQLAlchemy, WTForms);
- хороша читабельність коду та швидка розробка MVP;
- можливість масштабування у більші проєкти або переходу на складніші архітектури [8].

Завдяки цьому Python і Flask стали надійною основою для створення функціонального, масштабованого та підтримуваного бекенду для розробляемого програмного продукту [9].

Нижче наведено таблицю 2.1 з порівнянням Python (Flask) з його альтернативами JavaScript & Node.js, Ruby on Rails, PHP & Laravel.

Таблиця 2.1 – Порівняння характеристик Python з його основними аналогами

Характеристика	Python (Flask)	JavaScript & Node.js	PHP & Laravel	Ruby on Rails
Види продукту	Соц. мережі, стримінгові сервіси, сайти електронної комерції	Інтернет речей, програмне забезпечення для зв'язку в реальному часі	Системи управління контентом, сайти електронної комерції	Фінансове програмне забезпечення, стримінгові сервіси
Розмір проєкту	Мір, середнього розміру, великі	Середні, великі	Малі, середні	Малі, середні
Переваги	Простий синтаксис, масштабованість, швидке створення прототипів	Багатопотоковість, велика мережа зв'язків, вигідно для архітектури на основі мікросервісів	Хороша продуктивність, легке налагодження, дешево у виробництві	Надійність, зручні засоби розробки, високі стандарти якості
Недоліки	Незрілий доступ до бази даних, повільна продуктивність, залежить від сторонніх фреймворків та бібліотек	Може бути читатись по-різному різними тлумачами, може бути заблоковано будь-яким обчисленням	Небезпечний, важко керувати, не дозволяє змінюватися або модифікувати основну	Нестача гнучкості, висока ціна помилок, відсутність документації

		процесора	поведінку	
Простота використан ня	Відносно проста	Відносно проста	Відносно проста	Складніше, ніж Python/Node.js
Модуль- ність	Висока (PyPI)	Висока (NPM)	Висока (Composer)	Висока (Gems)

2.1.2 JavaScript як мова програмування клієнтської частини сервісу та порівняння з аналогами

JavaScript є стандартною мовою для розробки клієнтських (фронтенд) частин вебзастосунків. Вона дозволяє реалізовувати динамічну поведінку елементів, обробку подій, а також взаємодію з сервером без перезавантаження сторінки (AJAX, WebSocket). Станом на 2024 рік, 98.9% вебсайтів розроблено саме мовою JavaScript.

Як видно з табл. 2.1, для створення клієнтської частини програмного застосунку існує обмежена кількість варіантів, а саме лише JavaScript. За допомогою цієї мови можна здійснювати взаємодію з браузером, контролювати його функціонування, модифікувати контент і зовнішнє оформлення сторінок, а також забезпечувати асинхронний обмін інформацією з сервером [10].

JavaScript була обрана завдяки своїй універсальності, легкій інтеграції з HTML-шаблонами, стандартній підтримці всіма сучасними браузерами, а також бо JS відома великою кількістю допоміжних компонентів для розробки користувацьких інтерфейсів.

HTML, CSS і Jinja2 були обрані для побудови інтерфейсу. HTML використовується для структурування сторінок, CSS – для їхнього оформлення, а Jinja2 – як шаблонізатор для генерування динамічного HTML-коду зі сторони сервера.

Socket.IO – це бібліотека JavaScript, яка забезпечує двостороннє, подієво-орієнтоване з'єднання між клієнтом (браузером) і сервером у режимі реального часу [11]. Вона базується на WebSocket-технології, але з додатковими перевагами – наприклад, автоматичне перепідключення, підтримка fallback-протоколів, простий синтаксис подій тощо. Головними перевагами бібліотеки є: її легка інтеграція, підтримка масштабування (через Redis, кластеризацію) та високу надійність [12].

2.2 Обґрунтування вибору супровідних технологій

2.2.1 PostgreSQL як реляційна система керування базами даних (СКБД)

На сьогоднішній день розповсюджені дві основні категорії СКБД: реляційні та нереляційні, відомі як SQL та NoSQL відповідно. Реляційні системи баз даних є однією з найвживаніших моделей у світі, спрямованою на ефективне збереження та керування структурованою інформацією. Вони впорядковують дані у табличному форматі, що спрощує процеси пошуку, оновлення та маніпулювання інформацією через використання мови структурованих запитів (SQL). Реляційні бази даних є незамінними для бізнес-процесів, веб-ресурсів та програм, які вимагають складного керування даними та високої швидкодії. Такі системи демонструють відмінну продуктивність при активних операціях читання/запису з компактними та середніми обсягами даних. Вони також забезпечують прискорений пошук інформації завдяки створенню індексів полів даних для запитів та поєднання таблиць.

Нереляційна база даних зберігає інформацію без явних взаємозв'язків між елементами та без жорсткої структури. Замість організованих таблиць,

всередині системи розміщується множина різноманітних документів, включаючи зображення, відеоматеріали та навіть контент соціальних мереж. Нереляційні бази даних відзначаються високою продуктивністю, забезпечуючи паралельний доступ численних користувачів. Здатні містити необмежені обсяги наборів різних типів та форматів. Крім того, вони характеризуються значною гнучкістю щодо модифікації типів даних.

PostgreSQL являє собою потужну та розширювану об'єктно-реляційну СКБД, що пропонує обширний спектр можливостей для збереження та обробки інформації. PostgreSQL входить до числа найзатребуваніших відкритих СКБД та активно застосовується у веб-розробці, корпоративному програмуванні та наукових дослідженнях.

PostgreSQL створюється незалежно від будь-яких корпорацій; розвиток цього програмного рішення забезпечується завдяки колаборації численних фахівців, зацікавлених у використанні даної системи керування базами даних та інтеграції в неї новітніх технологічних рішень. Нижче наведено таблицю 2.2 з порівнянням найпопулярніших баз даних.

Таблиця 2.2 – Порівняння характеристик СКБД

Параметр	Тип бази даних	Легкість документації	Масштабованість	Легкість навчання
MySQL	SQL	середня	вертикальна, складна	середня
PostgreSQL	Object-relational, SQL	середня	вертикальна	складна
MongoDB	NoSQL, document-oriented	легка	горизонтальна	середня
Oracle	Multi-model, SQL	легка	вертикальна	складна
MSSQL Server	T-SQL	легка	вертикальна, складна	складна
Redis	NoSQL, key-value	легка	горизонтальна	середня

Серед найвагоміших переваг PostgreSQL виділяється її висока продуктивність та стабільність роботи, що перетворює її на оптимальне рішення для масштабних та ресурсомістких задач з обробки інформації. Додатково, система є повністю безплатною для застосування, що забезпечує її доступність для широкого спектра розробників та установ.

2.2.2 Docker як платформа контейнеризації програмних продуктів

Docker – це широко використовувана платформа для контейнеризації, яка дає змогу упакувати програму разом з усім її середовищем виконання в єдиний ізольований блок, який називається контейнером. На відміну від класичних віртуальних машин, Docker-контейнери не потребують повної віртуалізації апаратного забезпечення, оскільки працюють на основі операційної системи хоста, що забезпечує їхню легкість та високу продуктивність [14].

Ключова ідея використання Docker полягає в спрощенні та уніфікації процесу розгортання застосунків. Розробники можуть створювати контейнери, які включають всі потрібні бібліотеки, залежності та налаштування, і бути впевненими, що ці контейнери запускатимуться однаково у будь-якому середовищі: на локальному комп'ютері, у тестовому стенді чи у хмарі. Це дозволяє уникнути проблем, пов'язаних із "несумісністю середовищ".

Головна відмінність між контейнерами та віртуальними машинами полягає в принципі ізоляції.

У той час як віртуальні машини створюють повноцінні копії операційних систем, що споживають значні ресурси (CPU, RAM), контейнери використовують спільне ядро ОС хоста, але при цьому залишаються ізольованими один від одного.

Завдяки цьому, контейнери мають низку суттєвих переваг:

- вони легші та менш ресурсомні у порівнянні з віртуальними машинами;
- час запуску контейнера значно менший, ніж у віртуальної машини;
- сучасні системи оркестрації (наприклад, Kubernetes або Docker Swarm) створені саме для роботи з контейнерами, що робить їх більш зручними для масштабування застосунків;

Крім того, Docker має свій публічний репозиторій образів — DockerHub, де зберігаються тисячі готових рішень. Там можна знайти образи найпопулярніших баз даних, фреймворків, вебсерверів та інших інструментів. Це дозволяє швидко запускати потрібні сервіси, як для розробки, так і для розгортання у продакшн-середовищі [15].

2.3 Аналіз та обґрунтування вибору середовища розробки

Середовищем розробки програмного продукту було обрано PyCharm — інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains.

Цей потужний інструмент, орієнтований на професійну розробку програмного забезпечення на Python, який надає розробникам усі необхідні функції для зручної, швидкої та ефективної роботи. До таких функцій можна віднести: запуск та налагодження коду, інтегрований відлагоджувач, профілювання продуктивності застосунку, інтелектуальне автодоповнення та аналіз коду, зручну навігацію по проєкту, а також широкий набір інструментів для рефакторингу та генерації коду.

Серед альтернативних інструментів, які можуть бути використані для розробки на Python — Visual Studio Code. Хоча VS Code надає базовий функціонал, такий як підсвітка синтаксису, автодоповнення та інтеграція з Git, його функціональність залежить від сторонніх розширень. PyCharm, у свою чергу, забезпечує глибшу інтеграцію з екосистемою Python та надає вищий рівень автоматизації та зручності.

Однією з ключових переваг PyCharm є підтримка віртуальних середовищ та вбудованих інструментів для управління залежностями. Крім того, IDE підтримує інтеграцію з Docker і Docker Compose, дозволяючи розробникам запускати контейнери, налагоджувати застосунки в них і створювати нові образи безпосередньо зі середовища розробки.

PyCharm дозволяє ефективно працювати над проектами будь-якого рівня складності: від консольних скриптів до масштабованих вебзастосунків або мікросервісів. Вбудоване середовище тестування, налагодження, інструменти для роботи з базами даних, шаблонами, HTTP-запитами та REST API роблять його одним із найкращих виборів для розробки на Python.

2.4 Висновки за розділом 2

У другому розділі було розглянуті можливі варіанти мов програмування для написання серверної та клієнтської частин програмного продукту.

Для розробки серверної частини вебзастосунку було обрано Python на базі фреймворку Flask завдяки його простоті у використанні, активній підтримці, хорошій читабельності коду та швидкій розробці MVP, можливості масштабування у більші проекти або переходу на складніші архітектури

Для розробки клієнтської частини вебзастосунку була обрана JavaScript, у комбінації з HTML, CSS і Jinja2, завдяки компонентній архітектурі, логічній структурованості, повному контролю над стилями та можливості додання адаптивного дизайну.

Було також обрано супроводжуючі технології, як Docker для контейнеризації програмного продукту, а PostgreSQL як реляційну СКБД.

Загалом вибір даного набору технологій базується на таких їх перевагах як: висока гнучкість, можливість швидкої розробки, сучасний користувацький досвід і тд.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

3.1 Опис основних вимог до програми

Опис основних вимог становить один із найкритичніших етапів у процесі створення будь-якого програмного рішення. Він сприяє забезпеченню успішної та результативної реалізації застосунку, гарантуючи високу якість та відповідність потребам користувачів і бізнес-вимогам.

Перейдемо до переліку нефункціональних та функціональних вимог щодо майбутнього проєкту. До нефункціональних входять: назва вебзастосунку – українська, основна мова, простий шаблон користування.

Назва. Було обрано дати застосунку назву “TongoHub”, щоб виходила гра слів, де частина tongo- походить від англійського слова tongue (“язик”), але при цьому через співзвучність з відомим танцем “танго” – іконкою сайту було обрано зображення саме з натяком на танго.

До функціональних вимог входять:

- автентифікація користувача: (реєстрація користувача у системі, вхід користувача у систему (через електронну адресу), вихід користувача з системи);
- авторизація;
- пошук користувачів: (за віком, за username, за віком, за статтю, за зазначеними мовами);
- формування чатів з користувачами;
- обмін повідомленнями з користувачами.

3.2 Вимоги апаратного забезпечення і програмних засобів

Специфікації технічних та програмних компонентів встановлюють базові параметри, якими мають володіти клієнтські пристрої для належного функціонування вебзастосунку. Цей пункт деталізує необхідні характеристики

обладнання та програмного середовища, що можуть впливати на швидкодію, функціональні можливості та результативність продукту.

Вимоги до апаратного забезпечення:

- центральний процесор Intel Core i3-8100F з робочою частотою 3,6 ГГц або аналогічний [16];
- оперативна пам'ять: DDR3 з мінімальним обсягом 4 Гб [17];
- пристрої відображення: дисплей з роздільною здатністю не менш ніж 1280x800 пікселів.

Вимоги до програмного середовища:

- веббраузер (актуальна версія);
- мережеве підключення (стабільне для завантаження та експлуатації застосунку);
- підтримка технології JavaScript;
- операційна система: Windows 10 чи новіша, macOS 10.13 чи новіша, Linux (Ubuntu, Fedora тощо).

3.3 Архітектура програми

Архітектура програмного продукту визначає структурну організацію, складові елементи та механізми їх взаємодії, що впливає на робочі процеси, продуктивність та можливості розширення застосунку.

У даному пункті буде проаналізовано процес формування архітектури програми, визначено ключові елементи та їх взаємозалежності, а також розглянуто стратегії та методи створення ефективної та масштабованої архітектури для веб-системи.

У табл. 3.1 представлено перелік основних компонентів програмного рішення, що розробляється, разом з описом кожного з них. Усі елементи системи тим чи іншим способом пов'язані між собою, проте кожен компонент замість прямого виклику необхідних процедур та функцій з іншого елемента, використовуватиме чітко визначені шляхи взаємодії, які описують механізми

комунікації між компонентами. Такий підхід дозволяє мінімізувати ризик виникнення помилок при необхідності внесення змін всередині одного компонента без порушення функціонування інших залежних елементів.

Таблиця 3.1 – Перелік і опис основних компонентів системи

Назва модулю	Опис та призначення
User	Модуль реєстрації, аутентифікації та авторизації користувачів у системі. Також модуль дозволяє керувати своїм профілем.
Chat	Модуль, що дозволяє відправляти запити на особисте спілкування з іншими користувачами. Дозволяє також і сам обмін повідомленнями.
Searh	Модуль пошуку користувачів-партнерів для мовного обміну за ключовими категоріями (мова, вік і тд.). Дозволяє ще переглядати профілі інших користувачів.
Вебінтерфейс користувача	Модуль, який об'єднує можливості всіх перерахованих вище компонентів та забезпечує їх доступність через веб-інтерфейс.

Перші три елементи, що наведені, знаходяться на стороні вебсерверу. Вони взаємодіють з базами даних, обробляють HTTP запити, запускаються на окремій інфраструктурі, незалежній від клієнтської.

На сьогодні існують різноманітні архітектурні концепції для створення веб-додатків, зокрема: SOAP, gRPC, Rest API та GraphQL. Кожна з них володіє власними перевагами та обмеженнями, проте всі вони призначені для специфічних сценаріїв використання.

При створенні даного застосунку було застосовано багаторівневу модульну вебархітектуру. Клієнтську частину додатка було реалізовано як SPA. SPA являє собою вебзастосунок, який завантажується в браузер одноразово і під час взаємодії користувача з елементами інтерфейсу не вимагає перезавантаження сторінки щоразу, коли необхідно отримати або оновити інформацію на серверній стороні. При розробці серверна частина була реалізована за допомогою RESTful API (і частково WebSocket через Flask-SocketIO). Його варто використовувати через цілу низку переваг. RESTful API працюють на основі стандартних HTTP методів (GET, POST, PUT, PATCH, DELETE), які є звичними для більшості розробників. Цей підхід буде забезпечувати хорошу масштабованість: кожен клієнтський запит містить усі необхідні дані, що дає змогу легко масштабувати сервер у горизонтальному напрямку. А чітке розділення обов'язків між клієнтом і сервером дозволяє їм розвиватися незалежно, що позитивно впливає на підтримку та масштабування системи.

Загалом, як виглядав кінцевий архітектурний підхід програмного продукту зазначено в табл. 3.2.

Таблиця 3.2 – Комбінація архітектури застосунку

Шар	Технології	Роль
Презентаційний (UI)	HTML, CSS, JS, Jinja2, WebSocket	Інтерфейс користувача
Логіка застосунку	Flask (Blueprints), REST, Socket.IO	Обробка запитів, бізнес-логіка
Дані	SQLAlchemy, PostgreSQL або SQLite	Збереження та отримання даних
Комунікація	REST API + WebSocket	Клієнт ↔ сервер

Узагальнюючи, можна представити загальну архітектуру клієнтської та серверної частин застосунку разом із супутніми технологіями у формі спрощеної діаграми, зображеної на рис. 3.1.

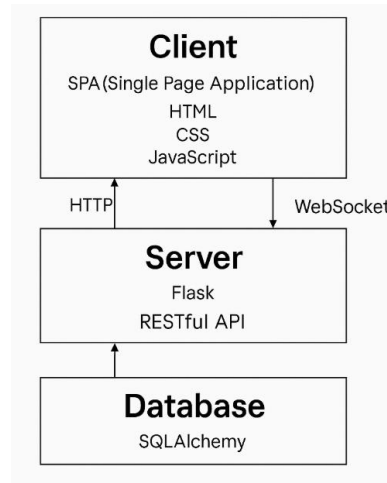


Рисунок 3.1 – Загальна архітектура клієнта і сервера

3.4 Користувацький інтерфейс

Створення користувацького інтерфейсу становить обов'язкову складову процесу розробки веб-додатка. Цей етап має критичну важливість, адже інтерфейс визначає характер взаємодії користувача з програмою та рівень задоволення від роботи із застосунком.

Користувацький інтерфейс (UI) у програмному продукті розроблений із урахуванням принципів зручності, інтуїтивності та функціональності, які забезпечують комфортну взаємодію користувача із системою. Він реалізований у вигляді вебінтерфейсу, що є частиною клієнтської частини застосунку.

Для розробки візуальної частини користувацького інтерфейсу проєкту було використано шаблонізатор Jinja2, який дозволяє динамічно формувати HTML-сторінки на стороні сервера. Це забезпечує ефективну інтеграцію даних, отриманих із бекенду (Flask), без необхідності використовувати повноцінний фронтенд-фреймворк.

Для стилізації та побудови інтерфейсних компонентів застосовувалася бібліотека Bootstrap, яка містить великий набір готових стилізованих елементів – кнопок, форм, навігаційних меню, списків, тощо. Завдяки цьому вдалось швидко створити адаптивний, чистий і сучасний дизайн, який коректно відображається на різних типах пристроїв, включно з мобільними (рис 3.2).

```
<div class="d-flex flex-wrap align-items-center justify-content-center justify-content-lg-start">
  <a href="/" class="d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none">
    
  </a>

  <ul class="nav col-12 col-lg-auto me-lg-auto mb-2 justify-content-center mb-md-0">
    <li><a href="{{ url_for('main.index') }}" class="nav-link px-2 text-white">Головна</a></li>
    <li><a href="{{ url_for('user.search') }}" class="nav-link px-2 text-white">Пошук</a></li>
    {% if current_user.is_authenticated %}
    <li><a href="{{ url_for('user.profile') }}" class="nav-link px-2 text-white">Мій Профіль</a></li>
    <li><a href="{{ url_for('chat.chats') }}" class="nav-link px-2 text-white">Мої Чати</a></li>
    {% endif %}
  </ul>

  <div class="text-end d-flex align-items-center gap-3">
    {% if current_user.is_authenticated %}
    <p class="text-white mb-0">Вітаємо, {{ current_user.username }}!</p>
    <a href="{{ url_for('user.logout') }}" type="button" class="btn btn-outline-light me-2">Вийти</a>
    {% else %}
    <a href="{{ url_for('user.login') }}" type="button" class="btn btn-outline-light me-2">Увійти</a>
    {% endif %}
  </div>
</div>
```

Рисунок 3.2 – Приклад використання Bootstrap

Окрім цього, для реалізації інтерактивних функцій на клієнтській частині використовувалися JavaScript та стандартні DOM API. Зокрема:

- для живого чату було інтегровано Socket.IO, що забезпечує двосторонній обмін повідомленнями в режимі реального часу;
- для попереднього перегляду медіа, вкладень і динамічного оновлення чатів використовуються прості JavaScript-механізми.

Також було враховано підтримку локалізації інтерфейсу, яку реалізовано через Flask-Babel. Це дозволяє перекладати текстові компоненти сторінки залежно від вибраної мови користувача, зберігаючи переклади у зовнішніх .po/.mo файлах.

Загалом, такий підхід до створення UI дозволив забезпечити простоту, зрозумілість та розширюваність інтерфейсу, який легко підтримувати і масштабувати в майбутньому.

3.5 Файлова система

Загальна файлова структура програмного продукту зображена на рисунку 3.3. Найважливішою директорією є `app`, де і збирається увесь застосунок.

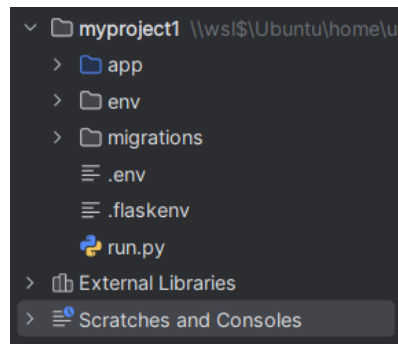


Рисунок 3.3 – Загальна файлова структура

Нижче, на рис. 3.4, зображена файлова структура директорії `app`. З рисунка видно, що `app` містить кілька інших директорій, кожна з яких відвідає або за головний модуль застосунку, або за допоміжний модуль, який використовується основними компонентами. За серверну частину застосунку відповідають директорії `models` та `routes`, в той час як за клієнтську – `templates` та `static`.

3.6 Висновки за розділом 3

Наведено детальну характеристику основних вимог як до програмного продукту, так і до апаратних, програмних компонентів. Надано докладний опис ключових елементів програмної системи, що розробляється, механізмів їх взаємодії, комунікації клієнтської частини застосунку із серверною

стороною та основних рішень щодо забезпечення безпеки користувацьких даних.

Окрему увагу приділено вибору рішень для дизайну вебінтерфейсу, використаним бібліотекам, а також структурі проекту — організації файлів та каталогів (рис. 3.4).

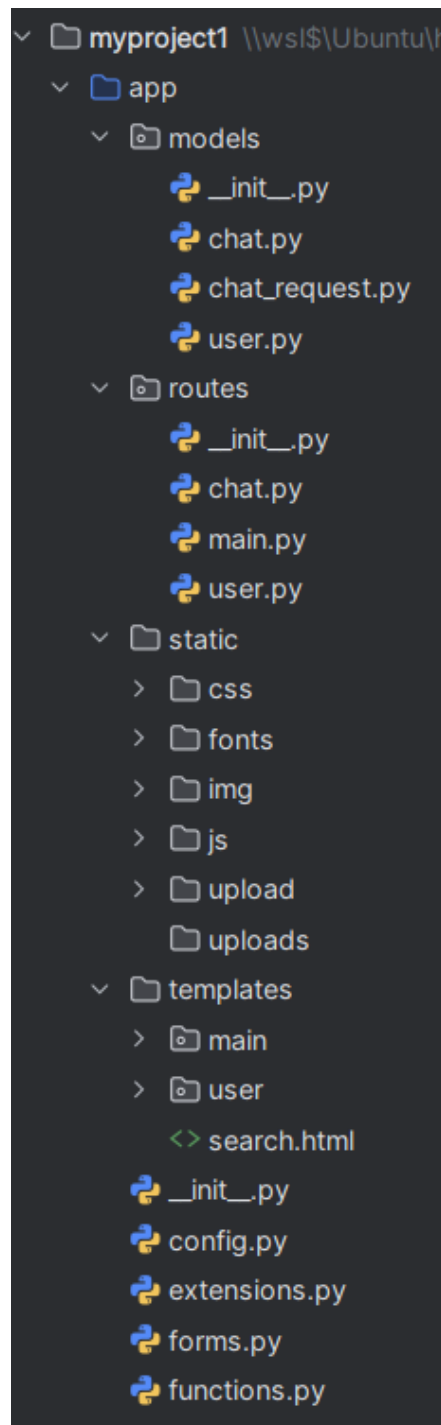


Рисунок 3.4 – Файлова структура директорії app

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування проєкту

Ціллю цього проєкту є створення вебзастосунку, призначенням якого є спрощення процесу пошуку партнерів для обміну мовами та культурним досвідом, допомога в налагодженні зв'язку між ними, а також становлення головним посередником у їх комунікації.

Розроблений вебзастосунок надає користувачу можливості реєструватися на сайті з інформацією, якою той готовий ділитись, а також такого самого інформаційного пошуку своїх співрозмовників. Окрім цього, застосунок дозволяє спілкуватися напряду, якщо користувачі не бажають користуватись сторонніми сервісами

4.2 Інструкція по експлуатації програми

4.2.1 Звернення до програми

Перш ніж користувачі зможуть працювати з інтерфейсом програмного продукту, потрібно ініціалізувати сервер для обслуговування статичних ресурсів, зокрема файлів HTML, CSS, JS, які застосовуються для відтворення інтерфейсу. Для розгортання всіх елементів програмної системи застосовується Docker. Після успішного запуску всіх складових системи, взаємодія користувачів з програмним рішенням повинна здійснюватися через браузер.

Головна сторінка застосунку відкривається після введення необхідної адреси. Вона складається лише з заголовку сервісу, де знаходяться не усі необхідні кнопки посилання, також з картинки, яка потрібна, щоб розбавити пустоту сторінки, ну і найголовніше – кнопки реєстрації (рис. 4.1)

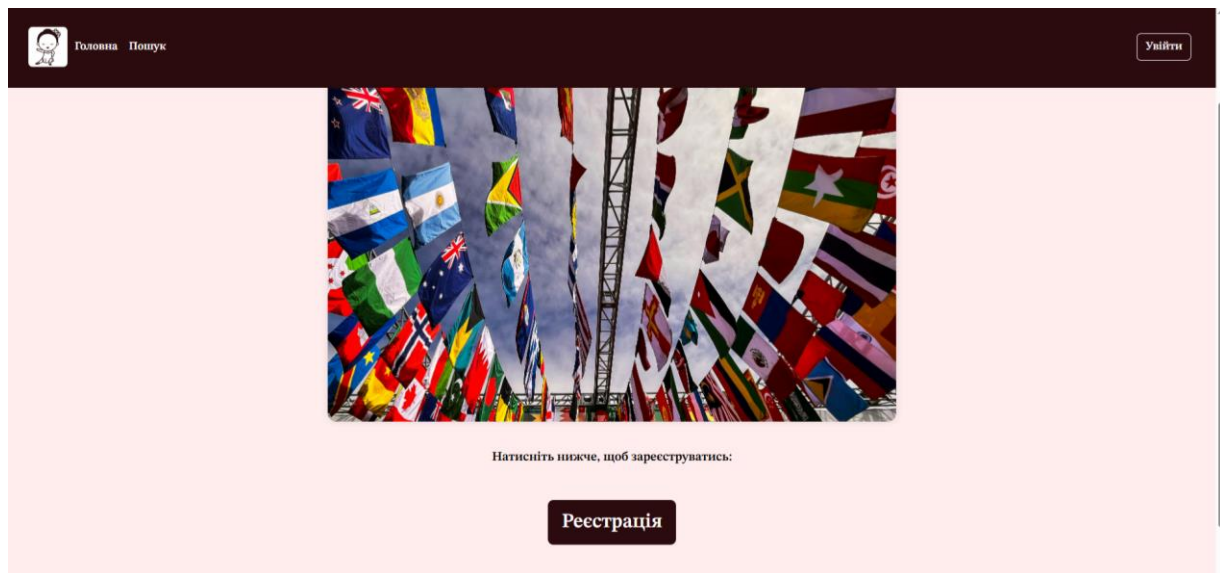


Рисунок 4.1 – Головна сторінка вебзастосунку

Заголовок містить кнопки “Пошук” та “Увійти”. При натисканні на обидві з них користувач переходитиме на сторінку з шаблоном авторизації (рис. 4.2), а при натисканні на іконку чи “Головну” буде повертатись на (рис. 4.1).

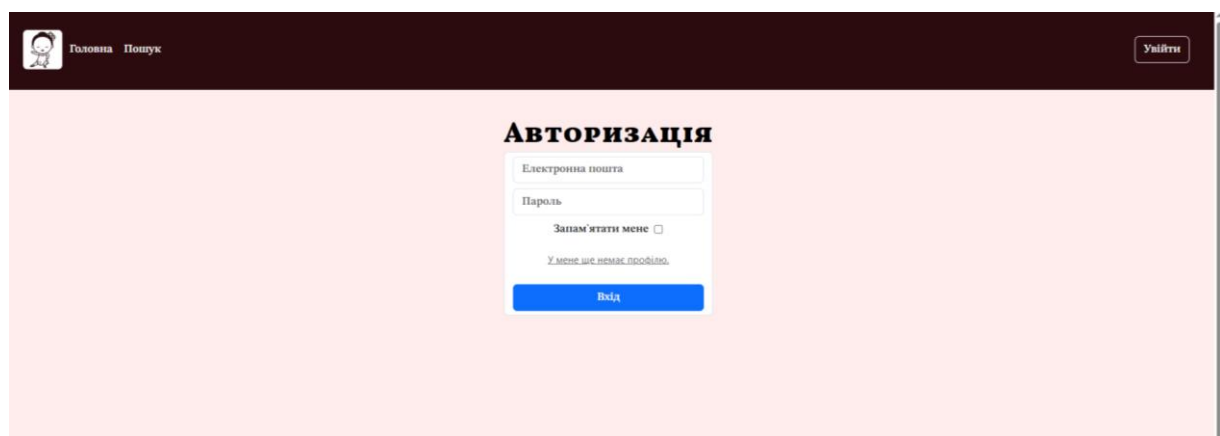


Рисунок 4.2 – Сторінка авторизації користувачів

Для подальшого не безсенсовного перебування на сайті необхідно пройти авторизацію, або реєстрацію, якщо користувач на платформі вперше. Перший шлях вже було зазначено, а щоб потрапити на сторінку з реєстрацією необхідно, або перейти через кнопку на головній, або при випадковому потраплянні на (рис. 4.2) натиснути у формі строку “У мене ще немає профілю”. Тоді користувача теж перекине на “Реєстрацію”(рис. 4.3-4.4).

 Головна Пошук

Увійти

РЕЄСТРАЦІЯ ПРОФІЛЮ

Електронна пошта

Ім'я користувача

Пароль

Підтвердити пароль

Вибрати файл Файл не вибрано

Опис

Особисте посилання

Стать

Вік

Країна

Мова, що вивчається

Рідна мова

Зареєструватись

© 2025 НУЗП, КНТ-131

Рисунок 4.3 – Сторінка реєстрації користувачів

Після реєстрації профілю користувач, або автоматично переходить на сторінку зі своїм профілем (рис. 4.4), або повертається на сторінку, куди обмежений доступ не авторизованим користувачам. У заголовку тепер з’явилися нові працюючі кнопки і привітання (рис. 4.5).

Мій профіль

Роба

Про себе: Вітаю!!

В процесі вивчення: Англійська

Рідна мова: Українська

Посилання: inst: prince

Стать: Чоловіча

Вік: 19

Країна: Україна

>

Рисунок 4.4 – Приклад щойно оформленого профілю користувача

 Головна Пошук Мій Профіль Мої Чати

Вітасмо, Рома! Вийти

Рисунок 4.5 – Заголовок після автентифікації користувача

Тепер після авторизації користувач може скористатись пошуком співрозмовника, або одразу заходити у “Мої чати”, хоча у тільки що зареєструвавшись користувачів їх звісно не буде (рис. 4.6).

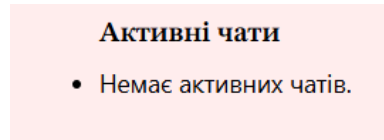


Рисунок 4.6 – Відсутність активних чатів

“Пошук” (рис. 4.7) надає можливість здійснювати пошук мовних партнерів серед таких самих вже зареєстрованих користувачів системи. Пошук може відбутись, як лише за одним вказаним параметром, так і детальнішим описом (рис. 4.8). Підходящі за запитом користувачі, після здійснення пошуку, будуть галереєю висіти на сторінці(рис. 4.9). Для більш детального перегляду зацікавившого профілю необхідно на нього перейти напряму.

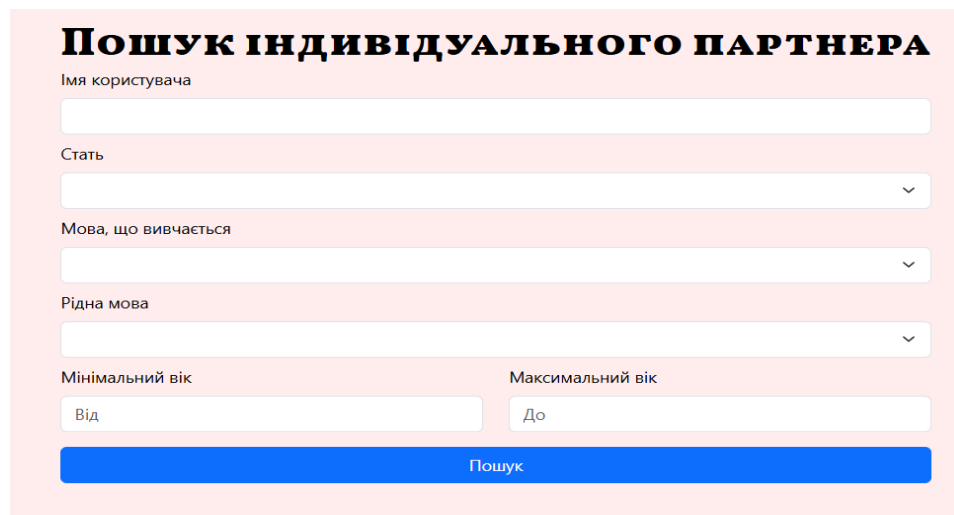
A screenshot of a web form titled 'Пошук індивідуального партнера' in bold black font. The form is set against a light pink background. It contains several input fields: a text field for 'Імя користувача', a dropdown menu for 'Стать', a dropdown menu for 'Мова, що вивчається', and a dropdown menu for 'Рідна мова'. Below these are two text fields for 'Мінімальний вік' (with 'Від' next to it) and 'Максимальний вік' (with 'До' next to it). At the bottom of the form is a wide blue button with the text 'Пошук' in white.

Рисунок 4.7 – Шаблон незаповненого пошуку

Пошук індивідуального ПАРТНЕРА

Ім'я користувача

Стать

Чоловіча

Мова, що вивчається

Українська

Рідна мова

Французька

Мінімальний вік

18

Максимальний вік

34

Пошук

Нічого не знайдено.

Рисунок 4.8 – Шаблон заповненого пошуку

Знайдені користувачі:



plotter

Про себе: Немає опису.

Стать: Чоловіча

Рідна мова: Польська

В процесі вивчення: Українська

[Переглянути профіль](#)



swan

Про себе: Немає опису.

Стать: Чоловіча

Рідна мова: Німецька

В процесі вивчення: Українська

[Переглянути профіль](#)

Рисунок 4.9 – Вигляд знайдених користувачів після виканового пошуку

Натиснувши “Переглянути профіль” користувач зможе ознайомитись з можливо більш детальною інформацією свого імовірного майбутнього співрозмовника. Також переглядаючи чужий профіль він зможе надіслати запит на формування чату (рис. 4.10)

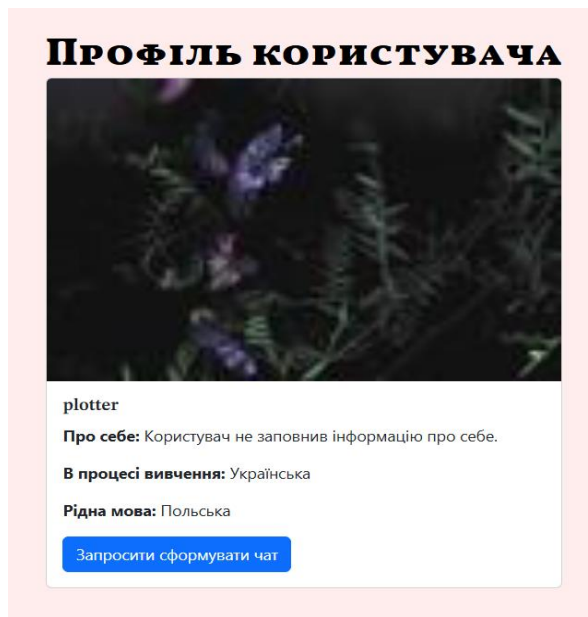


Рисунок 4.10 – Вигляд профілю та кнопка для запиту чату

Сторінка “Мої чати” відповідальна не тільки за активні чати, а також за запити на їх формування (рис. 4.11), можна відстежувати навіть ті, що ти відправив самостійно.

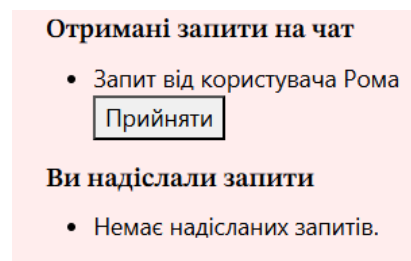


Рисунок 4.11 – Запити на формування чатів

Якщо користувач з того боку прийме запит, то автоматично між обома партнерами буде сформовано чат, окрему сторінку, де можна вільно обмінюватися повідомленнями (рис. 4.12). Чат згенерован таким чином. Що користувачам не доведеться оновлювати сторінку чи щось ще, усе відбуватиметься в режимі реального часу.



Рисунок 4.12 – Вигляд повідомлень у чаті між користувачами

Додатково, на (рис. 4.13) показано зовнішній вигляд повідомлень, що впливають при незвичних діях користувача.

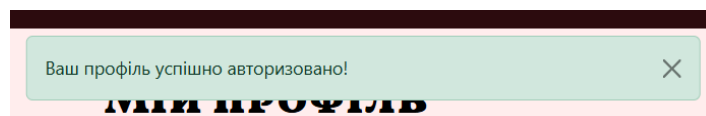


Рисунок 4.13 – Приклад повідомлення

4.2.2 Вхідні й вихідні дані

Вхідні дані: інформація про користувача, введена при реєстрації профілю, пошукові запити користувача, повідомлення відправлені користувачем

Вихідні дані: результати пошуку за запитом користувача, надсилання повідомлень про запит на чат.

4.2.3 Тестування програмного забезпечення

На стадії тестування програмного застосунка було здійснено кілька різноманітних типів перевірок.

Модульне тестування – перевірка окремих класів та функцій для підтвердження коректності роботи при різноманітних вхідних параметрах. Інтеграційне тестування – тестування кількох взаємопов'язаних елементів системи з використанням реальних даних. End-to-end тестування – контроль функціонування всього програмного веб-додатка з позиції кінцевого користувача.

Застосовуючи ці підходи до тестування, було проведено перевірку процедур реєстрації та авторизації в системі, процесів пошуку співрозмовників, механізмів відправлення запитів, отримання та створення чатів. Помилки, що повертаються із серверної частини, належним чином обробляються веб-інтерфейсом та відображаються для користувача у відповідних елементах інтерфейсу.

4.3 Висновки за розділом 4

У даному розділі розглянуто призначення та функціональні можливості створеного програмного забезпечення. Описано взаємодію користувача з проєктом та можливості, якими він володіє в процесі цієї взаємодії, що супроводжується візуальними ілюстраціями. Здійснено багаторівневе тестування, яке продемонструвало правильне функціонування проєкту.

ВИСНОВКИ

У процесі виконання дипломної роботи, об'єктом дослідження якої є програмні засоби для мовних навичок та культурного обміну, а предметом – комп'ютерні програми з пошуку партнерів для знайомств, було створено вебзастосунок, що забезпечує користувачів зручними інструментами не лише для пошуку, а й для комунікації з партнерами.

Було проведено огляд альтернатив застосунків для мовного обміну, завдяки чому були встановлені слабкі та сильні сторони таких засобів, що в свою чергу допомогло зі створенням функціоналу розробляемого застосунку для знайомств.

Було створено програмне рішення, яке надає користувачам можливість швидко та ефективно знаходити співрозмовників для культурного та мовного обміну, за допомогою ключових слів та деталей опису. Крім того, в продукті реалізована можливість прямого спілкування.

Під час тестування не було виявлено жодних системних збоїв або потенційних порушень безпеки стосовно доступу до інформації користувачів. Таким чином, можна зробити висновок – програмний засіб функціонує належним чином, відповідає встановленим вимогам і готовий до експлуатації..

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Student guide for exchange programs. Universitat de Barcelona. [Electronic resource]. – Access mode: https://www.ub.edu/sl/en/docs/guia_intercanvi_eng.pdf.
2. Tandem – Language Exchange App. [Electronic resource]. – Access mode: <https://tandem.net/>.
3. HelloTalk – Chat with native speakers. [Electronic resource]. – Access mode: <https://www.hellotalk.com/>.
4. The Mixxer – Language Exchange Community. [Electronic resource]. – Access mode: <https://www.language-exchanges.org/>.
5. MyLanguageExchange.com – Practice & Learn Languages. [Electronic resource]. – Access mode: <https://www.mylanguageexchange.com/>.
6. Python 3 documentation. [Electronic resource]. – Access mode: <https://docs.python.org/3/>.
7. Flask documentation. [Electronic resource]. – Access mode: <https://flask.palletsprojects.com/>.
8. Flask Tutorials – Real Python. [Electronic resource]. – Access mode: <https://realpython.com/tutorials/flask/>.
9. Why Python is so popular? Towards Data Science. [Electronic resource]. – Access mode: <https://towardsdatascience.com/why-python-is-so-popular-7633e3c86c4a>.
10. JavaScript documentation – MDN Web Docs. [Electronic resource]. – Access mode: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
11. Socket.IO documentation. [Electronic resource]. – Access mode: <https://socket.io/docs/v4/>.
12. Document Object Model (DOM) – MDN Web Docs. [Electronic resource]. – Access mode: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model.
13. Multifactor dimensionality reduction for detecting haplotype-haplotype

interaction / [Y. Jiang, R. Zhang, G. Liu et al.] // Fuzzy systems and knowledge discovery : Sixth international conference, Tianjin, 14-16 August 2009 : proceedings. – Los Alamitos: IEEE, 2009. – P. 241–245.

14. What is a Container? Docker. [Electronic resource]. – Access mode: <https://www.docker.com/resources/what-container/>.

15. Vinoji. Day 30: Exploring Docker Hub. Medium. [Electronic resource] / Vinoji. – Access mode: <https://medium.com/@vinoji2005/day-30-exploring-docker-hub-fdd4b5f3c6d4>.

16. Intel Core i3-8100F Processor. TechPowerUp. [Electronic resource]. – Access mode: <https://www.techpowerup.com/cpu-specs/core-i3-8100f.c2087>.

17. Double Data Rate (DDR). Cadence. [Electronic resource]. – Access mode: https://www.cadence.com/en_US/home/explore/double-data-rate-ddr.html.

ДОДАТОК А
Текст програми

A.1 Файл `__init__.py`

```

from flask import Flask
from .extensions import db, migrate, login_manager, socketio
from .config import Config

from .routes.user import user
from .routes.main import main
from .routes.chat import chat_bs

def create_app(config_class=Config):
    app = Flask(__name__)
    app.config.from_object(config_class)

    app.register_blueprint(user)
    app.register_blueprint(main)
    app.register_blueprint(chat_bs)

    db.init_app(app)
    migrate.init_app(app, db)
    login_manager.init_app(app)
    socketio.init_app(app)

    login_manager.login_view = 'user.login'
    login_manager.login_message = 'Ви не можете зараз отримати доступ до цієї сторінки. Необхідно спочатку авторизуватись.'
    login_manager.login_message_category = 'warning'

    with app.app_context():
        db.create_all()

    return app

```

A.2 Файл `config.py`

```

import os

class Config(object):
    APPNAME = 'app'
    ROOT = os.path.abspath(APPNAME)
    UPLOAD_PATH = '/static/upload'
    FULL_PATH = ROOT + UPLOAD_PATH

    UPLOADS_URL = '/static/uploads'
    UPLOADS_PATH = os.path.join(ROOT, 'static', 'uploads')

```

```

USER = os.environ.get('POSTGRES_USER', 'usermaru')
PASSWORD = os.environ.get('POSTGRES_PASSWORD', 'namjin')
HOST = os.environ.get('POSTGRES_HOST', '127.0.0.1')
PORT = os.environ.get('POSTGRES_PORT', 5532)
DB = os.environ.get('POSTGRES_DB', 'mydatb')

```

```

SQLALCHEMY_DATABASE_URI =
f'postgresql://{USER}:{PASSWORD}@{HOST}:{PORT}/{DB}'
SECRET_KEY = 'brsrcnn21347vn2lufm3c8lshffgcr4'
SQLALCHEMY_TRACK_MODIFICATIONS = True

```

A.3 Файл extentions.py

```

from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
from flask_bcrypt import Bcrypt
from flask_login import LoginManager
from flask_socketio import SocketIO

db = SQLAlchemy()
migrate = Migrate()
bcrypt = Bcrypt()
login_manager = LoginManager()
socketio = SocketIO(cors_allowed_origins="*")

```

A.4 Файл forms.py

```

from flask_wtf import FlaskForm
from flask_wtf.file import FileAllowed
from wtforms.fields.choices import SelectField
from wtforms.fields.numeric import IntegerField
from wtforms.fields.simple import StringField, PasswordField, SubmitField,
TextAreaField, FileField, BooleanField
from wtforms.validators import DataRequired, Length, EqualTo, ValidationError,
Optional
from .models.user import User

class RegistrationForm(FlaskForm):
    email = StringField('Електронна пошта', validators=[DataRequired(),
Length(min=2, max=100)])
    password = PasswordField('Пароль', validators=[DataRequired()])

```

```

confirm_password = PasswordField('Підтвердження паролю',
validators=[DataRequired(), EqualTo('password')])
username = StringField('Ім'я користувача', validators=[DataRequired(),
Length(min=2, max=20)])
avatar = FileField('Фото профілю', validators=[FileAllowed(['jpg', 'png'])])
bio = TextAreaField('Про себе', validators=[Optional()])
link = TextAreaField('Посилання', validators=[Optional()])

gender = SelectField('Стать', choices=[
    ('', ''),
    ('male', 'Чоловіча'),
    ('female', 'Жіноча'),
    ('other', 'Інша')
], validators=[Optional()])

age = SelectField('Вік', choices=[
    ('', ''),
    ] + [(str(i), str(i)) for i in range(13, 101)], validators=[Optional()])

country = SelectField('Країна', choices=[
    ('', ''),
    ('ua', 'Україна'),
    ('pl', 'Польща'),
    ('us', 'США'),
    ('de', 'Німеччина'),
    ('fr', 'Франція')
], validators=[Optional()])

learning_language = SelectField('Мова, що вивчається', choices=[
    ('', ''),
    ('en', 'Англійська'),
    ('ua', 'Українська'),
    ('pl', 'Польська'),
    ('de', 'Німецька'),
    ('fr', 'Французька')
], validators=[DataRequired()])

native_language = SelectField('Рідна мова', choices=[
    ('', ''),
    ('ua', 'Українська'),
    ('pl', 'Польська'),
    ('en', 'Англійська'),
    ('de', 'Німецька'),
    ('fr', 'Французька')
], validators=[DataRequired()])

```

```

submit = SubmitField('Зареєструватись')

def validate_email(self, email):
    user = User.query.filter_by(email=email.data).first()
    if user:
        raise ValidationError('Ця електронна адреса вже зареєстрована на сайті.')

def validate_username(self, username):
    user = User.query.filter_by(username=username.data).first()
    if user:
        raise ValidationError('Дане ім'я вже використовується. Будь ласка, оберіть інше.')

class LoginForm(FlaskForm):
    email = StringField('Електронна пошта', validators=[DataRequired(),
Length(min=2, max=50)])
    password = PasswordField('Пароль', validators=[DataRequired()])
    remember = BooleanField('Запам'ятати мене')
    submit = SubmitField('Вхід')

class SearchForm(FlaskForm):
    username = StringField('Ім'я користувача', validators=[Optional(),
Length(min=2, max=50)])
    learning_language = SelectField('Мова, що вивчається', choices=[
        ('', ''),
        ('en', 'Англійська'),
        ('ua', 'Українська'),
        ('pl', 'Польська'),
        ('de', 'Німецька'),
        ('fr', 'Французька')
    ], validators=[Optional()])
    native_language = SelectField('Рідна мова', choices=[
        ('', ''),
        ('ua', 'Українська'),
        ('pl', 'Польська'),
        ('en', 'Англійська'),
        ('de', 'Німецька'),
        ('fr', 'Французька')
    ], validators=[Optional()])
    gender = SelectField('Стать', choices=[
        ('', ''),
        ('male', 'Чоловіча'),
        ('female', 'Жіноча'),

```

```

('other', 'Інша')
], validators=[Optional()])
min_age = IntegerField('Мінімальний вік', validators=[Optional()])
max_age = IntegerField('Максимальний вік', validators=[Optional()])
submit = SubmitField('Пошук')

```

A.5 Файл `functions.py`

```

import os
import secrets

from PIL import Image
from flask import current_app
from werkzeug.datastructures import FileStorage
from werkzeug.utils import secure_filename

def save_picture(picture):
    if not picture or not isinstance(picture, FileStorage) or picture.filename == "":
        return None
    random_hex = secrets.token_hex(8)
    _, f_ext = os.path.splitext(picture.filename)
    picture_fn = random_hex + f_ext
    picture_path = os.path.join(current_app.config['FULL_PATH'], picture_fn)
    output_size = (125, 125)
    try:
        i = Image.open(picture)
        i.thumbnail(output_size)
        i.save(picture_path)
        return picture_fn
    except Exception as e:
        print(f"Помилка при збереженні зображення: {e}")
        return None

def save_attachment(file):
    if not file or file.filename == "":
        return None
    filename = secure_filename(file.filename)
    filepath = os.path.join(current_app.config['UPLOAD_PATH'], filename)
    file.save(filepath)
    return filename

LANGUAGE_LABELS = {
    'ua': 'Українська',
    'en': 'Англійська',

```

```

    'pl': 'Польська',
    'de': 'Німецька',
    'fr': 'Французька',
}

```

```

GENDER_LABELS = {
    'male': 'Чоловіча',
    'female': 'Жіноча',
    'other': 'Інша'
}

```

```

COUNTRY_LABELS = {
    'ua': 'Україна',
    'en': 'Велика Британія',
    'pl': 'Польща',
    'de': 'Німеччина',
    'fr': 'Франція',
}

```

A.6 Файл /routes/main.py

```

from flask import Blueprint, render_template

main = Blueprint('main', __name__)

@main.route('/index')
@main.route('/')

def index():
    return render_template('main/index.html')

```

A.7 Файл /routes/user.py

```

from flask import Blueprint, render_template, request, redirect, url_for, flash, abort
from flask_login import login_user, logout_user, login_required, current_user
from ..functions import save_picture, LANGUAGE_LABELS,
GENDER_LABELS, COUNTRY_LABELS
from ..forms import RegistrationForm, LoginForm, SearchForm
from ..extensions import db, bcrypt
from ..models.user import User
from ..models.chat import Chat
from ..models.chat_request import ChatRequest

user = Blueprint('user', __name__)

```

```

@user.route('/profile')
@login_required
def profile():
    return render_template('user/profile.html', user=current_user,
language_labels=LANGUAGE_LABELS,
gender_labels=GENDER_LABELS,
country_labels=COUNTRY_LABELS)

@user.route('/project')
def project():
    return render_template('project.html')

@user.route('/user/search', methods=['GET', 'POST'])
@login_required
def search():
    form = SearchForm()
    users = []

    if form.validate_on_submit():
        query = User.query.filter(User.id != current_user.id)

        if form.username.data:
            query = query.filter(User.username.ilike(f"%{form.username.data}%"))
        if form.learning_language.data:
            query = query.filter(User.learning_language ==
form.learning_language.data)
        if form.native_language.data:
            query = query.filter(User.native_language == form.native_language.data)
        if form.gender.data:
            query = query.filter(User.gender == form.gender.data)

        if form.min_age.data is not None:
            query = query.filter(User.age >= form.min_age.data)
        if form.max_age.data is not None:
            query = query.filter(User.age <= form.max_age.data)

        users = query.all()

    return render_template('search.html', form=form, users=users,
language_labels=LANGUAGE_LABELS, gender_labels=GENDER_LABELS)

@user.route('/profile/<int:user_id>')
@login_required
def view_profile(user_id):
    profile_user = User.query.get_or_404(user_id)

```

```

existing_chat = None
existing_request = None
if profile_user.id != current_user.id:
    existing_chat = Chat.query.filter(
        ((Chat.user1_id == current_user.id) & (Chat.user2_id == profile_user.id)) |
        ((Chat.user1_id == profile_user.id) & (Chat.user2_id == current_user.id))
    ).first()

    existing_request = ChatRequest.query.filter_by(
        from_user_id=current_user.id,
        to_user_id=profile_user.id,
        status='pending'
    ).first()

return render_template(
    'user/view_profile.html',
    user=profile_user,
    language_labels=LANGUAGE_LABELS, gender_labels=GENDER_LABELS,
    country_labels=COUNTRY_LABELS,
    existing_chat=existing_chat, existing_request=existing_request
)

@user.route('/user/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        avatar_filename = None
        if form.avatar.data and form.avatar.data.filename != "":
            avatar_filename = save_picture(form.avatar.data)
        user = User(username=form.username.data, email=form.email.data,
            password=hashed_password, avatar=avatar_filename, bio=form.bio.data or "",
            link=form.link.data or "", gender=form.gender.data or None,
            age=int(form.age.data) if form.age.data else None,
            country=form.country.data or None,
            learning_language=form.learning_language.data or None,
            native_language=form.native_language.data or None )
        try:
            db.session.add(user)
            db.session.commit()
            login_user(user)
            flash(f'{form.username.data}, ваш профіль успішно зареєстровано!',
                "success")
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('user.profile'))

```

```

except Exception as e:
    print(str(e))
    flash(f'Помилка при реєстрації.', "danger")
return render_template('user/register.html', form=form)

@user.route('/user/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            login_user(user, remember=form.remember.data)
            next_page = request.args.get('next')
            flash(f'Ваш профіль успішно авторизовано!', "success")
            return redirect(next_page) if next_page else redirect(url_for('user.profile'))
        else:
            flash(f'Помилка входу. Будь ласка, перевірте Email та пароль!',
                  "danger")
            return render_template('user/login.html', form=form)

@user.route('/user/logout', methods=['GET', 'POST'])
def logout():
    logout_user()
    return redirect(url_for('user.login'))

```

A.8 Файл /models/user.py

```

from datetime import datetime
from ..extensions import db, login_manager
from flask_login import UserMixin

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)

    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(150), nullable=False)
    username = db.Column(db.String(50), nullable=False)
    avatar = db.Column(db.String(200))
    bio = db.Column(db.String(500), nullable=True)

```

```

link = db.Column(db.String(50), nullable=True)
gender = db.Column(db.String(50), nullable=True)
age = db.Column(db.Integer, nullable=True)
country = db.Column(db.String(50), nullable=True)
learning_language = db.Column(db.String(50), nullable=False)
native_language = db.Column(db.String(50), nullable=False)
date = db.Column(db.DateTime, default=datetime.utcnow)

```

A.9 Файл /routes/chat.py

```

import os

from flask import Blueprint, render_template, request, redirect, url_for, flash, abort,
current_app
from flask_login import login_required, current_user
from flask_socketio import join_room, emit
from werkzeug.utils import secure_filename

from ..models.user import User
from ..models.chat import Chat, Message
from ..extensions import db, socketio
from ..models.chat_request import ChatRequest

chat_bs = Blueprint('chat', __name__)

@chat_bs.route('/chat/request/<int:to_user_id>', methods=['POST'])
@login_required
def send_chat_request(to_user_id):
    existing = ChatRequest.query.filter_by(from_user_id=current_user.id,
to_user_id=to_user_id).first()
    if not existing:
        db.session.add(ChatRequest(from_user_id=current_user.id,
to_user_id=to_user_id))
        db.session.commit()
        flash('Запит на чат надіслано', 'info')
        return redirect(url_for('user.view_profile', user_id=to_user_id))

@chat_bs.route('/chat/accept/<int:from_user_id>', methods=['POST'])
@login_required
def accept_chat_request(from_user_id):
    req = ChatRequest.query.filter_by(from_user_id=from_user_id,
to_user_id=current_user.id, status='pending').first()
    if req:
        req.status = 'accepted'
        db.session.add(Chat(user1_id=from_user_id, user2_id=current_user.id))

```

```

        db.session.commit()
        flash('Чат сформовано!', 'success')
        return redirect(url_for('chat.chats'))

@chat_bs.route('/chats')
@login_required
def chats():
    chats = Chat.query.filter(
        (Chat.user1_id == current_user.id) | (Chat.user2_id == current_user.id)
    ).all()

    chat_data = []
    for chat in chats:
        partner = chat.user2 if chat.user1 == current_user else chat.user1
        last_msg = Message.query.filter_by(chat_id=chat.id).order_by(Message.timestamp.desc()).first()
        chat_data.append({
            'chat': chat,
            'partner': partner,
            'last_msg': (last_msg.content[:50] + '...') if last_msg else 'Немає повідомлень'
        })

    incoming_requests = ChatRequest.query.filter_by(to_user_id=current_user.id,
        status='pending').all()

    outgoing_requests = ChatRequest.query.filter_by(from_user_id=current_user.id,
        status='pending').all()

    return render_template(
        'user/chats.html',
        chat_data=chat_data,
        chat_requests_in=incoming_requests,
        chat_requests_out=outgoing_requests
    )

@chat_bs.route('/chat/<int:chat_id>', methods=['GET', 'POST'])
@login_required
def chat(chat_id):
    chat = Chat.query.get_or_404(chat_id)
    if current_user.id not in [chat.user1_id, chat.user2_id]:
        abort(403)

    if request.method == 'POST':

```

```

        content = request.form.get('message')
        if content:
            msg = Message(chat_id=chat.id, sender_id=current_user.id,
content=content)
            db.session.add(msg)
            db.session.commit()
            return redirect(url_for('chat.chat', chat_id=chat_id))

    messages =
Message.query.filter_by(chat_id=chat_id).order_by(Message.timestamp).all()
    partner = chat.user2 if chat.user1 == current_user else chat.user1
    return render_template('user/chat_room.html', chat=chat, messages=messages,
partner=partner)

@socketio.on('join')
def handle_join(data):
    join_room(data['chat_id'])

@socketio.on('send_message')
def handle_send_message(data):
    msg = Message(chat_id=data['chat_id'], sender_id=data['user_id'],
content=data['message'])
    db.session.add(msg)
    db.session.commit()

    user = User.query.get(data['user_id'])
    emit('receive_message', {
        'message': data['message'],
        'user_id': data['user_id'],
        'username': user.username
    }, room=data['chat_id'])

@chat_bs.route('/chat/<int:chat_id>/history')
@login_required
def load_chat_history(chat_id):
    page = request.args.get('page', 1, type=int)
    messages =
Message.query.filter_by(chat_id=chat_id).order_by(Message.timestamp.desc()).pa
ginate(page, 20, False).items
    return render_template('partials/messages.html', messages=reversed(messages))

def save_attachment(file):
    if not file or file.filename == "":
        return None

```

```

filename = secure_filename(file.filename)
upload_path = current_app.config['UPLOADS_PATH']
os.makedirs(upload_path, exist_ok=True)
filepath = os.path.join(upload_path, filename)
file.save(filepath)
return filename

@chat_bs.route('/chat/<int:chat_id>/send', methods=['POST'])
@login_required
def send_message(chat_id):
    content = request.form.get('message')
    file = request.files.get('file')
    filename = None

    if file:
        filename = save_attachment(file)

    msg = Message(
        chat_id=chat_id,
        sender_id=current_user.id,
        content=content or "",
        attachment=filename
    )
    db.session.add(msg)
    db.session.commit()

    socketio.emit('receive_message', {
        'message': msg.content,
        'user_id': current_user.id,
        'username': current_user.username,
        'attachment': filename,
        'timestamp': msg.timestamp.strftime('%H:%M')
    }, room=str(chat_id))

    return "", 204

```

A.10 Файл /models/chat.py

```

from datetime import datetime
from ..extensions import db

class Chat(db.Model):
    id = db.Column(db.Integer, primary_key=True)

```

```

user1_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
user2_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

```

```

user1 = db.relationship('User', foreign_keys=[user1_id])
user2 = db.relationship('User', foreign_keys=[user2_id])

```

```

class Message(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    chat_id = db.Column(db.Integer, db.ForeignKey('chat.id'))
    sender_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    content = db.Column(db.Text)
    timestamp = db.Column(db.DateTime, default=datetime.utcnow)
    attachment = db.Column(db.String(255), nullable=True)

    chat = db.relationship('Chat', backref='messages')
    sender = db.relationship('User')

```

A.11 Файл /templates/search.html

```

{% extends 'main/base.html' %}

{% block title %}Пошук індивідуального партнера{% endblock %}

{% block content %}
    <form method="post" class="d-flex flex-column gap-2 mb-4">
        {{ form.hidden_tag() }}
        {{ form.username.label }} {{ form.username(class="form-control") }}
        {{ form.gender.label }} {{ form.gender(class="form-select") }}
        {{ form.learning_language.label }} {{ form.learning_language(class="form-select") }}
        {{ form.native_language.label }} {{ form.native_language(class="form-select") }}
    </form>

    <div class="row">
        <div class="col">
            {{ form.min_age.label(class="form-label") }}
            {{ form.min_age(class="form-control", placeholder="Від") }}
        </div>
        <div class="col">
            {{ form.max_age.label(class="form-label") }}
            {{ form.max_age(class="form-control", placeholder="До") }}
        </div>
    </div>

    {{ form.submit(class="btn btn-primary mt-2") }}
</form>

```

```
{% if users %}
<h3>Знайдені користувачі:</h3>
<div class="row row-cols-1 row-cols-md-2 g-4">
  {% for user in users %}
    <div class="col">
      <div class="card h-100">
        {% if user.avatar %}
          
        {% else %}
          
        {% endif %}
        <div class="card-body">
          <h2 class="card-title">{{ user.username }}</h2>
          <p class="card-text"><strong>Про себе:</strong> {{ user.bio or
'Немає опису.' }}</p>
          <p><strong>Стать:</strong> {{ gender_labels.get(user.gender,
'Невідомо') }}</p>
          <p><strong>Рідна мова:</strong> {{
language_labels.get(user.native_language, user.native_language) }}</p>
          <p><strong>В процесі вивчення:</strong> {{
language_labels.get(user.learning_language, user.learning_language) }}</p>
          <a href="{{ url_for('user.view_profile', user_id=user.id) }}"
class="btn btn-outline-primary">Переглянути профіль</a>
        </div>
      </div>
    </div>
  {% endfor %}
</div>
{% elif form.is_submitted() %}
<p>Нічого не знайдено.</p>
{% endif %}
{% endblock %}
```

A.12 Файл /templates/user/search.html

```
{% extends 'main/base.html' %}

{% block title %}Реєстрація профілю{% endblock %}

{% block content %}
  <form method="post" enctype="multipart/form-data" class="form-control h2-
style d-flex flex-column gap-2" >
    {{ form.csrf_token }}
```

```

{{ form.email(class="form-control", placeholder="Електронна пошта") }}
{% if form.email.errors %}
    <div class="invalid-feedback" style="display: block!important">
        {% for error in form.email.errors %}
            <span>{{ error }}</span>
        {% endfor %}
    </div>
{% endif %}

{{ form.username(class="form-control", placeholder="Ім`я користувача") }}
{% if form.username.errors %}
    <div class="invalid-feedback" style="display: block!important">
        {% for error in form.username.errors %}
            <span>{{ error }}</span>
        {% endfor %}
    </div>
{% endif %}

{{ form.password(class="form-control", placeholder="Пароль") }}
{{ form.confirm_password(class="form-control", placeholder="Підтвердити
пароль") }}

{{ form.avatar(class="form-control") }}
{{ form.bio(class="form-control", placeholder="Опис") }}
{{ form.link(class="form-control", placeholder="Особисте посилання") }}

{{ form.gender.label }}
{{ form.gender(class="form-select") }}
{{ form.age.label }}
{{ form.age(class="form-select") }}
{{ form.country.label }}
{{ form.country(class="form-select") }}
{{ form.learning_language.label }}
{{ form.learning_language(class="form-select") }}
{{ form.native_language.label }}
{{ form.native_language(class="form-select") }}
{{ form.submit(class="btn btn-primary mt-3") }}
</form>

{% endblock %}

```

A.13 Файл /static/css/main.css

```
@import url('fonts.css');
```

```

body {
  background: #ffeded;
  color: #000000;
  width: 100%;
  margin-left: auto;
  margin-right: auto;
}

.page {
  overflow-x: hidden;
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

.header {
  background-color: #2c0b0e;
  position: sticky;
  top: 0;
  z-index: 1000;
  padding: 27px;
  font-family: 'STIX', sans-serif;
  font-weight: 400;
}

.flash-absolute {
  position: absolute;
  top: 120px;
  left: 50%;
  transform: translateX(-50%);
  z-index: 10;
  width: 90%;
  max-width: 600px;
}

.register-button {
  background-color: #2c0b0e;
  color: white;
  padding: 10px 20px;
  font-size: 28px;
  font-family: 'STIX', sans-serif;
  border: none;
  border-radius: 8px;
  transition: background-color 0.3s ease;
}

```

```

}

.register-button:hover {
  background-color: #501419;
  cursor: pointer;
}

.center {
  display: flex;
  margin: auto;
}

.content-full {
  width: 100%;
  max-width: none;
  padding: 0;
  margin: 0;
}

.content {
  max-width: 1140px;
  padding: 40px 20px;
  flex: 1 1 auto;
  margin: 0 auto;
}

h1 {
  font-family: 'Vollkorn', sans-serif;
  font-weight: 900;
  margin: 0 auto;
}

h2 {
  font-family: 'STIX', sans-serif;
  font-size: 18px;
}

h3 {
  font-family: 'STIX', sans-serif;
  font-size: 64px;
}

.form-control.h2-style input,
.form-control.h2-style textarea,
.form-control.h2-style select,

```

```
.form-control.h2-style label {
  font-family: 'STIX', sans-serif;
  font-size: 16px;
}

.header__img {
  display: block;
  width: 60px;
  height: 60px;
}

.chat-avatar-wrapper {
  position: relative;
}

.chat-avatar {
  width: 48px;
  height: 48px;
  border-radius: 50%;
  object-fit: cover;
  margin-right: 14px;
  border: 1px solid #ccc;
}

.chat-unread-badge {
  position: absolute;
  top: -4px;
  right: -4px;
  background-color: #dc3545;
  color: white;
  font-size: 12px;
  font-weight: bold;
  padding: 2px 6px;
  border-radius: 999px;
  box-shadow: 0 0 2px rgba(0, 0, 0, 0.2);
}

.chat-container {
  width: 100%;
  max-width: none;
  height: 700px;
  margin: auto;
  display: flex;
  flex-direction: column;
  border: 1px solid #ccc;
}
```

```
border-radius: 8px;
overflow: hidden;
}

.messages {
  flex: 1;
  padding: 15px;
  overflow-y: scroll;
  background-color: #f8f9fa;
}

.message {
  display: inline-block;
  max-width: 60%;
  padding: 10px;
  border-radius: 20px;
  margin-bottom: 5px;
  position: relative;
  background-color: #d1e7dd;
}

.text-end .message {
  background-color: #cfe2ff;
  align-self: flex-end;
}

.timestamp {
  display: block;
  font-size: 0.75em;
  color: gray;
  margin-top: 5px;
  text-align: right;
}

.date-divider {
  text-align: center;
  font-weight: bold;
  color: #6c757d;
  margin: 15px 0;
}

.input-area {
  display: flex;
  padding: 10px;
  gap: 10px;
```

```
background-color: #fff;  
border-top: 1px solid #ccc;  
}
```

ДОДАТОК Б
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра на тему:

«РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ МОВНОГО
ОБМІНУ МІЖ НОСІЯМИ МОВ»

Виконала: студентка групи КНТ-131 Марія ПЕХОВА

Керівник: к.т.н., доцент каф. ПЗ Ольга ГЛАДКОВА

Рисунок Б.1 – Слайд 1

- ✓ Об'єкт дослідження – програмні засоби для розвитку мовних навичок та культурного обміну.
- ✓ Предмет дослідження – комп'ютерні програми з пошуку партнерів для знайомств та ведення розмов.
- ✓ Мета роботи – створення вебзастосунку для швидкого пошуку мовного партнера та спілкуванню через текстові повідомлення.

Рисунок Б.2 – Слайд 2

Порівняння програмного забезпечення для пошуку партнерів для мовного обміну

Критерії для порівняння	Існуючі аналоги			
	Tandem	HelloTalk	The Mixer	MyLanguageExchange
Наявність інтерфейсу веб-платформи	+	-	+	+
Наявність мобільного інтерфейсу	+	+	-	-
Наявність пошуку з фільтрами	+	+	+	+
Детальний опис профілів користувачів	+	-	+	+
Наявність платних розширених версій	+	+	-	-
Наявність чатів для обміну повідомленнями	+	+	+	+
Пряма можливість для переносу спілкування на інші ресурси	-	-	+	+
Кількість підтримуваних мов	300	180+	100+	163

Рисунок Б.3 – Слайд 3

Порівняння характеристик Python з його основними аналогами.

Характеристика	Python (Flask)	JavaScript & Node.js	PHP & Laravel	Ruby on Rails
Види продукту	Соц. мережі, стримінгові сервіси, сайти електронної комерції	Інтернет речей, програмне забезпечення для зв'язку в реальному часі	Системи управління контентом, сайти електронної комерції	Фінансове програмне забезпечення, стримінгові сервіси
Розмір проєкту	Мвр, середнього розміру, великі	Середні, великі	Малі, середні	Малі, середні
Переваги	Простий синтаксис, масштабованість, швидке створення прототипів	Багатопотоковий, велика мережа зв'язків, вигідно для архітектури на основі мікросервісів	Хороша продуктивність, легке налагодження, дешево у виробництві	Надійність, зручні засоби розробки, високі стандарти якості
Недоліки	Незрілий доступ до бази даних, повільна продуктивність, залежить від сторонніх фреймворків та бібліотек	Може бути читатись по-різному різними глумачами, може бути заблоковано будь-яким обчисленням процесора	Небезпеч-ний, важко керувати, не дозволяє змінюватися або модифікувати основну поведінку	Нестача гнучкості, висока ціна помилок, відсутність документації
Простота використання	Відносно проста	Відносно проста	Відносно проста	Складніше, ніж Python/Node.js
Модуль-ність	Висока (PyPI)	Висока (NPM)	Висока (Composer)	Висока (Gems)

Рисунок Б.4 – Слайд 4

Порівняння характеристик СКБД

Параметр	Тип бази даних	Легкість документації	Масштабованість	Легкість навчання
MySQL	SQL	середня	вертикальна, складна	середня
PostgreSQL	Object-relational, SQL	середня	вертикальна	складна
MongoDB	NoSQL, document-oriented	легка	горизонтальна	середня
Oracle	Multi-model, SQL	легка	вертикальна	складна
MSSQL Server	T-SQL	легка	вертикальна, складна	складна
Redis	NoSQL, key-value	легка	горизонтальна	середня

Рисунок Б.5 – Слайд 5

Перелік і опис основних компонентів системи

Назва модулю	Опис та призначення
User	Модуль реєстрації, аутентифікації та авторизації користувачів у системі. Також модуль дозволяє керувати своїм профілем.
Chat	Модуль, що дозволяє віправляти запити на особисте спілкування з іншими користувачами. Дозволяє також і сам обмін повідомленнями.
Search	Модуль пошуку користувачів-партнерів для мовного обміну за ключовими категоріями (мова, вік і т.д.). Дозволяє ще переглядати профілі інших користувачів.
Вебінтерфейс користувача	Модуль, що комбінує функціональність усіх вищезгаданих модулів та надає їй у вигляді вебінтерфейсу.

Рисунок Б.6 – Слайд 6

Комбінація архітектури застосунку

Шар	Технології	Роль
Презентаційний (UI)	HTML, CSS, JS, Jinja2, WebSocket	Інтерфейс користувача
Логіка застосунку	Flask (Blueprints), REST, Socket.IO	Обробка запитів, бізнес-логіка
Дані	SQLAlchemy, PostgreSQL або SQLite	Збереження та отримання даних
Комунікація	REST API + WebSocket	Клієнт↔сервер

Загальна архітектура клієнта і сервера

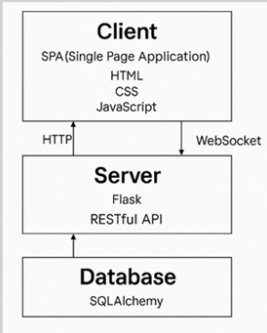
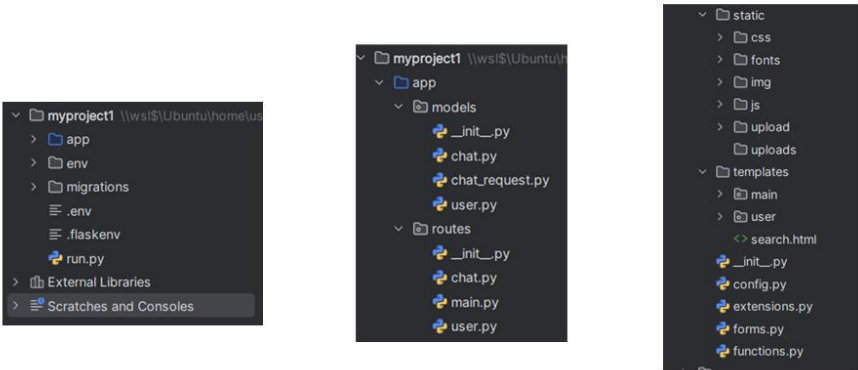


Рисунок Б.7 – Слайд 7



Файлова система

Рисунок Б.8 – Слайд 8

Приклад використання Bootstrap

```
<div class="d-flex flex-wrap align-items-center justify-content-center justify-content-lg-start">
  <a href="/" class="d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none">
    
  </a>

  <ul class="nav col-12 col-lg-auto me-lg-auto mb-2 justify-content-center mb-md-0">
    <li><a href="{{ url_for('main.index') }}" class="nav-link px-2 text-white">Головна</a></li>
    <li><a href="{{ url_for('user.search') }}" class="nav-link px-2 text-white">Пошук</a></li>
    {% if current_user.is_authenticated %}
    <li><a href="{{ url_for('user.profile') }}" class="nav-link px-2 text-white">Мій Профіль</a></li>
    <li><a href="{{ url_for('chat.chats') }}" class="nav-link px-2 text-white">Мої Чати</a></li>
    {% endif %}
  </ul>

  <div class="text-end d-flex align-items-center gap-3">
    {% if current_user.is_authenticated %}
    <p class="text-white mb-0">Вітаємо, {{ current_user.username }}!</p>
    <a href="{{ url_for('user.logout') }}" type="button" class="btn btn-outline-light me-2">Вийти</a>
    {% else %}
    <a href="{{ url_for('user.login') }}" type="button" class="btn btn-outline-light me-2">Увійти</a>
    {% endif %}
  </div>
</div>
```

Рисунок Б.9 – Слайд 9

Загальний вигляд інтерфейсу застосунку

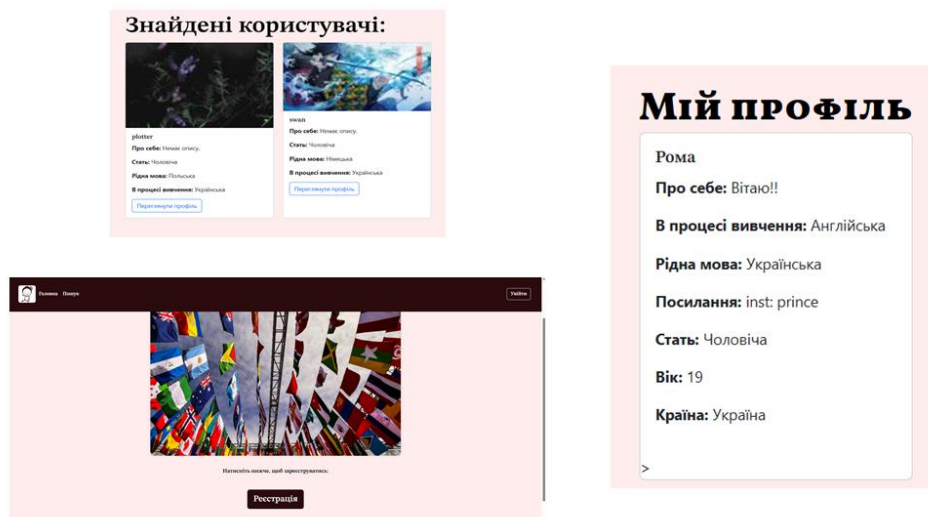


Рисунок Б.10 – Слайд 10

Висновки

- Було створено програмний продукт, що дозволяє користувачам швидко та ефективно знаходити співрозмовників для культурного та мовного обміну, за допомогою ключових слів та деталей опису. Крім того, в продукті реалізована можливість прямого спілкування.
- Під час перевірки не виявлено жодних системних помилок або можливих порушень безпеки щодо доступу до користувацьких даних. Отже, у висновку – програмний продукт працює коректно, відповідає вимогам і готовий до використання.

Рисунок Б.11 – Слайд 11