

Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Технології створення програмних продуктів»

за темою

«Polylingua»

Пояснювальна записка (комплексної роботи, для проєктної команди)

Частина № 2

Виконав(ла):

студент(ка) 3-го курсу

групи АІ-195

Урбанович Маргарита

Костянтинівна

Перевірив:

Блажко Олександр

Анатолійович

Одеса-2021

Анотація

В курсовій роботі розглядається процес створення програмного продукту «Polylingua» на основних етапах: визначення вимог до програмного продукту, планування процесів розробки, проектування, конструювання, верифікація, розгортання та валідація програмного продукту.

Робота виконувалась в команді з декількох учасників: Савченко Б. І., Урбанович М. К.

Тому вміст пояснювальної записки розділів «1 Вимоги до програмного продукту» та «2 Планування процесу розробки програмного продукту» співпадає зі вмістом пояснювальної записки інших учасників проєктної команди. у розділах «Проектування», «Конструювання» та «Верифікація» детальніше описано лише одну частину з урахуванням планів проведених робіт з розділу з описом особливостей конструювання:

- структур даних моделі «нереляційна» в системі керування базами даних «MongoDB»;
- програмних модулів в інструментальному середовищі «Visual Studio Code» з використанням фреймворку «NodeJs» та мови програмування «JavaScript».

Поточну версію пояснювальної записки до результатів роботи розміщено на *GitHub*-репозиторії за адресою: <https://github.com/Ayashma/Polylingua>.

Перелік скорочень

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП – програмний продукт

UML – уніфікована мова моделювання

Зміст

	стор.
1 Вимоги до програмного продукту	8
1.1 Визначення потреб споживача	8
1.1.1 Ієрархія потреб споживача	8
1.1.2 Деталізація матеріальної потреби	9
1.2 Бізнес-вимоги до програмного продукту	9
1.2.1 Опис проблеми споживача	9
1.2.1.1 Концептуальний опис проблеми споживача	9
1.2.1.2 Опис цільової групи споживача	10
1.2.1.3 Метричний опис проблеми споживача	10
1.2.2 Мета створення програмного продукту	13
1.2.2.1 Проблемний аналіз існуючих програмних продуктів	13
1.2.2.2 Мета створення програмного продукту	13
1.2.3 Назва програмного продукту	14
1.2.3.1 Гасло програмного продукту	14
1.2.3.2 Логотип програмного продукту	14
1.3 Вимоги користувача до програмного продукту	15
1.3.1 Пригодницька історія користувача програмного продукту	15
1.3.2 Історія користувача програмного продукту	16
1.3.3 Діаграма прецедентів програмного продукту	17
1.3.4 Сценарії використання прецедентів програмного продукту	17

1.4 Функціональні вимоги до програмного продукту	22
1.4.1. Багаторівнева класифікація функціональних вимог	22
1.4.2 Функціональний аналіз існуючих програмних продуктів	25
1.5 Нефункціональні вимоги до програмного продукту	27
1.5.1 Опис зовнішніх інтерфейсів	27
1.5.1.1 Опис інтерфейсів користувача	27
1.5.1.1.1 Опис INPUT-інтерфейсів користувача	27
1.5.1.1.2 Опис OUTPUT-інтерфейсів користувача	28
1.5.1.2 Опис інтерфейсу із зовнішніми пристроями	33
1.5.1.3 Опис програмних інтерфейсів	36
1.5.1.4 Опис інтерфейсів передачі інформації	36
1.5.1.5 Опис атрибутів продуктивності	36
2 Планування процесу розробки програмного продукту	38
2.1 Планування ітерацій розробки програмного продукту	38
2.2 Концептуальний опис архітектури програмного продукту	40
2.3 План розробки програмного продукту	40
2.3.1 Оцінка трудомісткості розробки програмного продукту	40
2.3.2 Визначення дерева робіт з розробки програмного продукту	44
2.3.3 Графік робіт з розробки програмного продукту	45
2.3.3.1 Таблиця з графіком робіт	45
2.3.3.2 Діаграма Ганта	47
3 Проектування програмного продукту	49

3.1 Концептуальне та логічне проектування структур даних програмного продукту	49
3.1.1 Концептуальне проектування на основі <i>UML</i> -діаграми концептуальних класів	49
3.1.2 Логічне проектування структур даних	50
3.2 Проектування програмних класів	53
3.3 Проектування алгоритмів роботи методів програмних класів	57
3.4 Проектування тестових сценаріїв верифікації роботи програмних модулів	58
3.4.1 Проектування тестових сценаріїв верифікації функціональних вимог	58
3.4.2 Проектування тестових сценаріїв верифікації нефункціональних вимог	59
3.4.3 Створення матриці відповідності вимог до програмного продукту	61
4 Конструювання програмного продукту	63
4.1 Особливості конструювання структур даних	63
4.2 Особливості конструювання програмних модулів	66
4.2.1 Конструювання програмної структури з урахуванням спеціалізованого <i>Framework</i> для <i>FrontEnd</i> -компонент архітектури (за наявністю)	67
4.2.2 Особливості використання спеціалізованих програмних бібліотек та API (за наявністю)	68
4.3 Модульне тестування програмних модулів	70
5 Верифікація програмного продукту	74

5.1 Тестування апаратно-програмних інтерфейсів програмного продукту	74
5.2 Тестування інтерфейсу користувача програмного продукту	75
5.3 Тестування часу реакції програмного продукту на дії користувача	76
6 Розгортання та валідація програмного продукту	79
6.1 Інструкція з встановлення системного програмного забезпечення	79
6.2 Інструкція з використання програмного продукту	79
6.3 Результати валідації програмного продукту	91
Висновки	93
Додаток А	94

1 Вимоги до програмного продукту

1.1 Визначення потреб споживача

1.1.1 Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

- фізіологія (вода, їжа, житло, сон);
- безпека (особиста, здоров'я, стабільність),
- приналежність (спілкування, дружба, любов),
- визнання (повага оточуючих, самооцінка),
- самовираження (вдосконалення, персональний розвиток).

На рисунку 1.1 представлено одну ієрархію потреби споживача, яку хотілося б задовольнити, використовуючи майбутній програмний продукт.



Рис. 1.1 – Приклад ієрархії потреби споживача

1.1.2 Деталізація матеріальної потреби

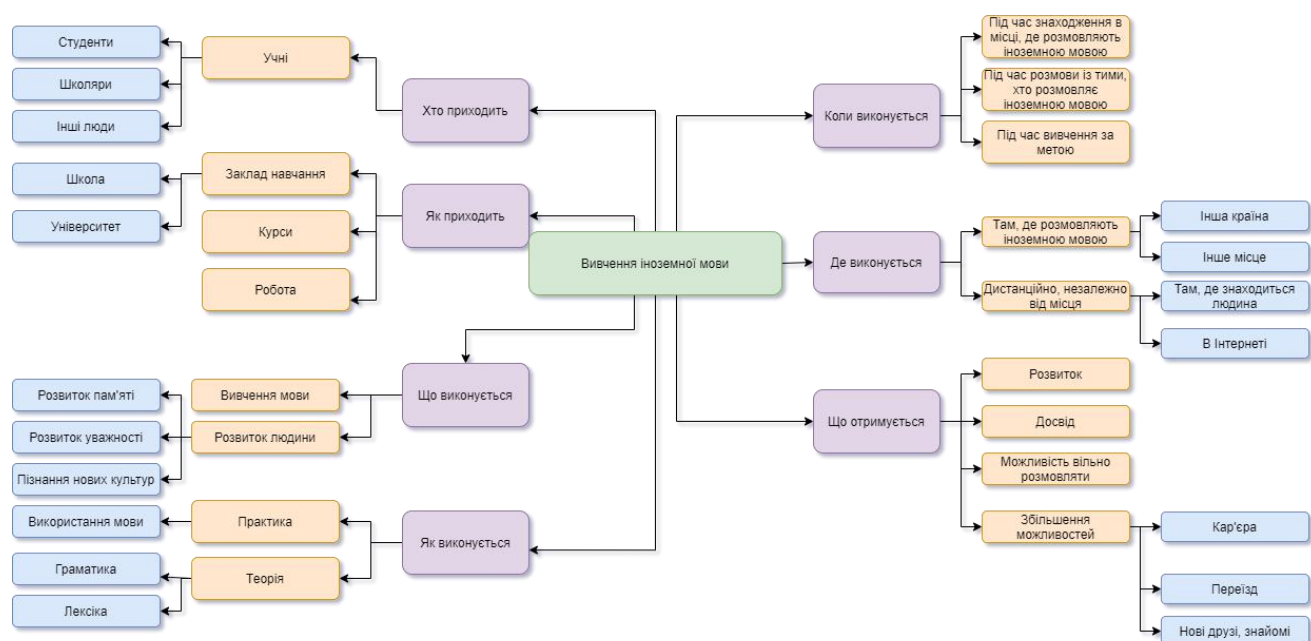


Рис. 1.1.2 – Mindmap

1.2 Бізнес-вимоги до програмного продукту

1.2.1 Опис проблеми споживача

1.2.1.1 Концептуальний опис проблеми споживача

Табл. 1.2.1.1 - Метричний опис проблеми споживача

№	Загальний опис проблеми	Метричні показники незадоволеності споживача
1	<u>Неможливо</u> організувати повноцінне вивчення мови без практики і мотивації.	<u>Низький рівень доступності</u> практики при вивченні іноземної мови.

1.2.1.2 Опис цільової групи споживачів

Програмний продукт призначений як для школярів, так і для дорослої аудиторії. Через неформальність способу вивчення матеріалу можна припустити, що споживачами буде молодь, тобто школяри та дорослі віком 14-25 років. Через те, що програмний продукт розрахований на вивчення іноземних мов, можна вважати, що аудиторія буде інтернаціональною. Стать також не можна точно визначити, тому що програмний продукт призначений для всіх. Типовий споживач даного програмного продукту може виглядати як підліток будь-якої статі, який користується платформою для школи, або через свої власні інтереси щодо вивчення іноземних мов.

1.2.1.3 Метричний опис проблеми споживача

Метрика інформаційної потреби споживача програмного продукту “Polylingua” - вік споживачів, їх рід діяльності, рівень знання англійської мови, рівень мотивації, успіхи у навчанні на мовних курсах та задоволення від навчання, спосіб вивчення мов та доступність вивчення мов.

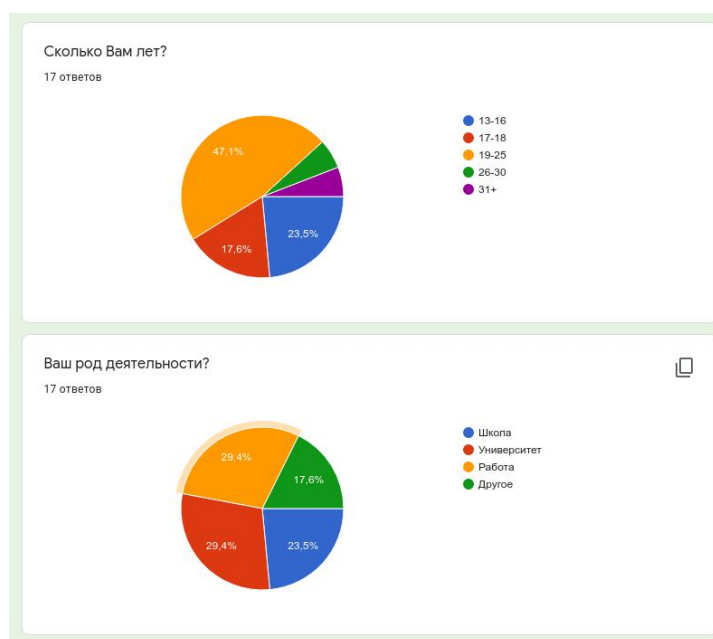


Рис. 1.2.1.3.1 – Графіки (метрика) результатів опитування

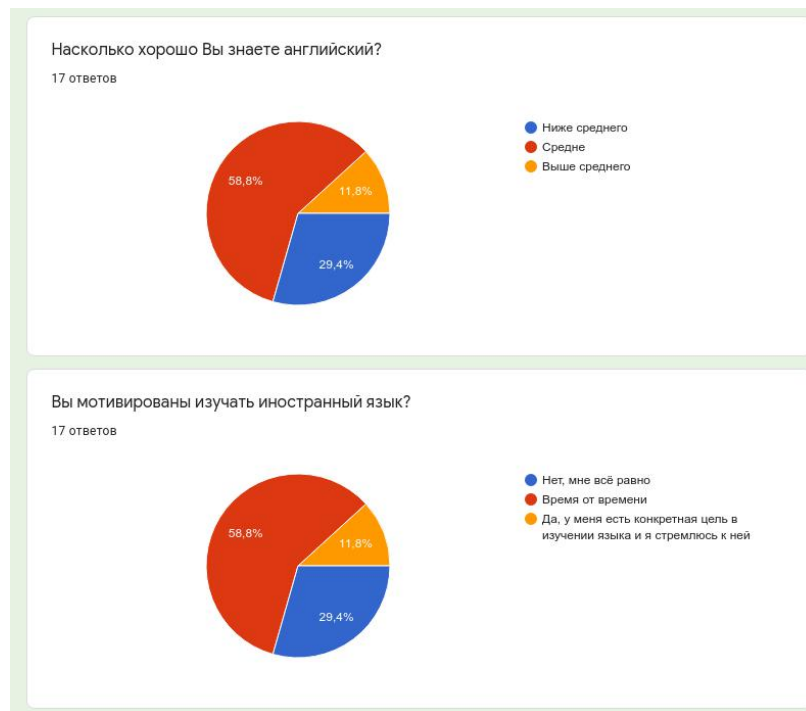


Рис. 1.2.1.3.2 – Графіки (метрика) результатів опитування

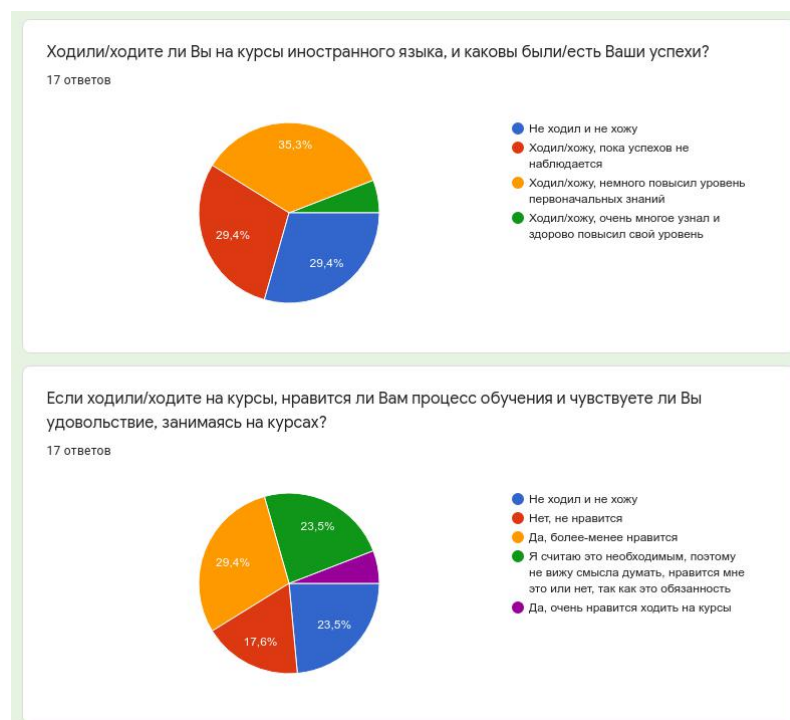


Рис. 1.2.1.3.3 – Графіки (метрика) результатів опитування

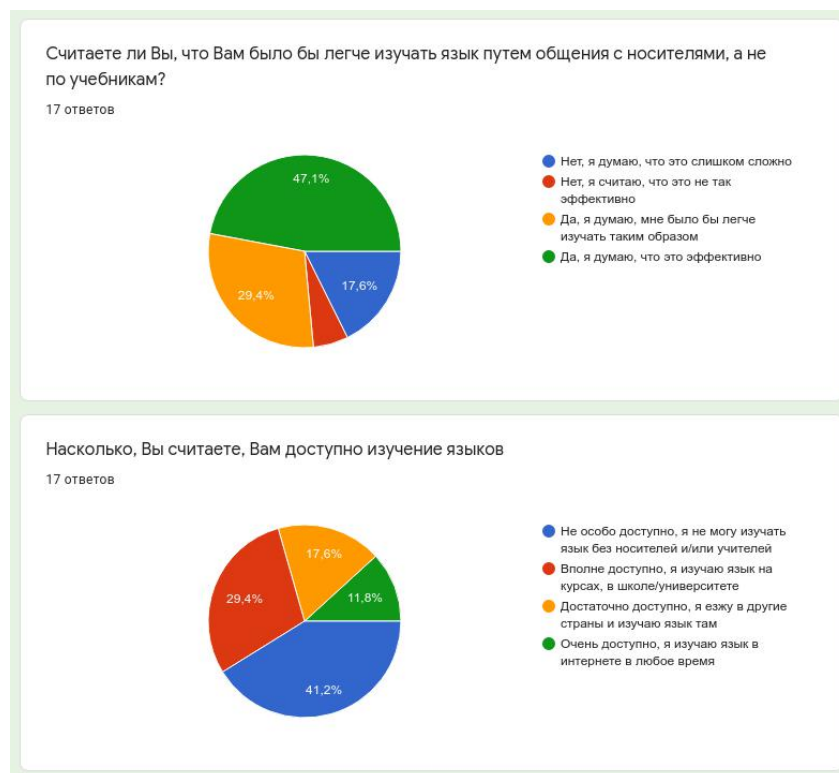


Рис. 1.2.1.3.4 – Графіки (метрика) результатів опитування

Через дані, отримані від споживачів, можна зробити висновок, що більшість з них відповідає цільовій категорії програмного продукту, тобто споживачами є молодь - школяри та дорослі віком 13-25 років. Споживачі навчаються в університеті, або працюють.

Також можна побачити, що споживачі вважають, що їм було б легше та ефективніше навчатися шляхом розмов із носіями мови. Більшість споживачів не дуже задовольняють результати навчання на курсах, в них немає постійної мотивації вивчати мови.

Головна проблема, яку намагається вирішити “Polylingua” - збільшення доступності практики та навчання іноземним мовам. Потрібність вирішення цієї проблеми позначається тим, що багато споживачів не вважають вивчення мов доступним.

1.2.2 Мета створення програмного продукту

1.2.2.1 Проблемний аналіз існуючих програмних продуктів

Табл. 1.2.2.1 - Проблемний аналіз існуючих ПП

№	Назва продукту	Вартість	Ступінь готовності	Примітка
1	Tandem	Безкоштовно	1	Месенджер, не можна ділитися контентом із іншими користувачами
2	HelloTalk	Безкоштовно	1	Багато контенту із обмеженнями для безкоштовної версії
3	Speaky	Безкоштовно	1	Мало користувачів, немає потоку нових співрозмовників
4	InterPals	Безкоштовно	1	Користувачі можуть загрузати пости, але лише фотографії
5	ITalki	Безкоштовно	1	Немає можливості розмовляти із іншими користувачами, діалог можна вести лише із викладачами

1.2.2.2 Мета створення програмного продукту

Метою створення програмного продукту Polylingua є збільшення доступності ефективного та легкого отримання знань в області іноземних мов шляхом обміну знань із іншими користувачами продукту.

За допомогою даного програмного продукту користувачі будуть мати змогу розмовляти один із одним, тим самим отримують корисні навички щодо вивчення іноземних мов, ділитися своїми знаннями із іншими людьми.

1.2.3 Назва програмного продукту

Назва ПП — "POLYLINGUA"

1.2.3.1 Гасло програмного продукту

Polylingua — доступний спосіб пізнання світу

1.2.3.2 Логотип програмного продукту



Рис. 1.2.3.2 – Логотип ПП

1.3 Вимоги користувача до програмного продукту

1.3.1 Пригодницька історія користувача програмного продукту (за бажанням членів проектної команди)

Антуан - студент, живе у Франції. Кілька років тому він поїхав з батьками до Іспанії, де провів відмінну літо. Саме тоді він дізнався, що йому дуже подобається іспанська мова і культура. Антуан вирішив вивчити іспанську мову, щоб йому було легше орієнтуватися під час поїздки і щоб спілкуватися з місцевими мешканцями, так він почав вивчати мову шляхом живого спілкування з іспанцями, але зрозумів, що з нульовими знаннями це дуже і дуже важко.

З навчання в поїздки нічого не вийшло, тому що на відпочинку хочеться відпочивати, а не наполегливо вчитися, і тому, приїхавши додому після канікул, Антуан записався на курси іспанської, купив підручники і став наполегливо займатися. Через деякий час яскраві враження від поїздки забулися, і Антуан зрозумів, що йому не вистачає мотивації для подальшого вивчення мови.

Втомившись сидіти за підручниками, Антуан закинув вивчення мови. Він був засмучений і розповів про це своєму другу Ремі.

Вислухавши історію Антуана про відсутність мотивації, друг знайшов в цій історії себе - коли він готувався до поїздки в Німеччину по навчанню, він ніяк не міг почати вивчати мову, і в підсумку, приїхавши в країну, він був абсолютно фрустрований довгими невимовними словами і ніяк не міг зрозуміти нічого, що відбувалося навколо. Після невдалої поїздки Ремі знайшов в інтернеті програмний продукт "Polylingua", зареєстрований в якому, він знайшов собі багато німецьких друзів, які з радістю вирішили йому допомогти у вивченні своєї мови. Натомість Ремі допомагав своїм друзям з французьким. Так, проводячи вільний час в дружній обстановці, Ремі і не помітив, як значно підвищив свій практично нульовий рівень німецької, і тепер міг висловлюватися на багато тем на незнайомому раніше йому мовою.

Ремі розповів про це Антуану і порадив йому зареєструватися в "Polylingua", де він обов'язково знайде іспаноязичних співрозмовників, які йому допоможуть у вивченні.

Антуан зареєструвався і знайшов собі кілька друзів, які вчили французьку, з яким Антуан міг їм допомогти. Через деякий час користування програмним продуктом, Антуан відчув, що тепер він щоранку прокидається з мотивацією вивчати мову, так як вивчення відбувається в процесі спілкування з друзями. Крім мотивації, він помітив значні поліпшення в своїх знаннях, які він отримав не з підручників, а завдяки програмному продукту "Polylingua", і, безумовно, своїм старанням.

Поїхавши до Іспанії через рік після знайомства з "Polylingua", він отримав ще більше задоволення від поїздки, ніж в перший раз, тому що тепер він добре орієнтувався і міг спілкуватися з місцевими мешканцями і вивчати культуру країни через спілкування з людьми.

1.3.2 Історія користувача програмного продукту

1. Як гість, я можу зареєструватися в ПП.
2. Як користувач, я можу переглядати свій профіль та профілі інших користувачів.
3. Як користувач, я можу редагувати свій профіль.
4. Як користувач, я можу публікувати свої статті у блог.
5. Як користувач, я можу вести особисті співбесіди із іншими користувачами.
6. Як користувач, я можу вийти зі свого аккаунта.
7. Як користувач, я можу видалити свій аккаунт.
8. Як адміністратор, я можу керувати базою даних.

1.3.3 Діаграма прецедентів програмного продукту

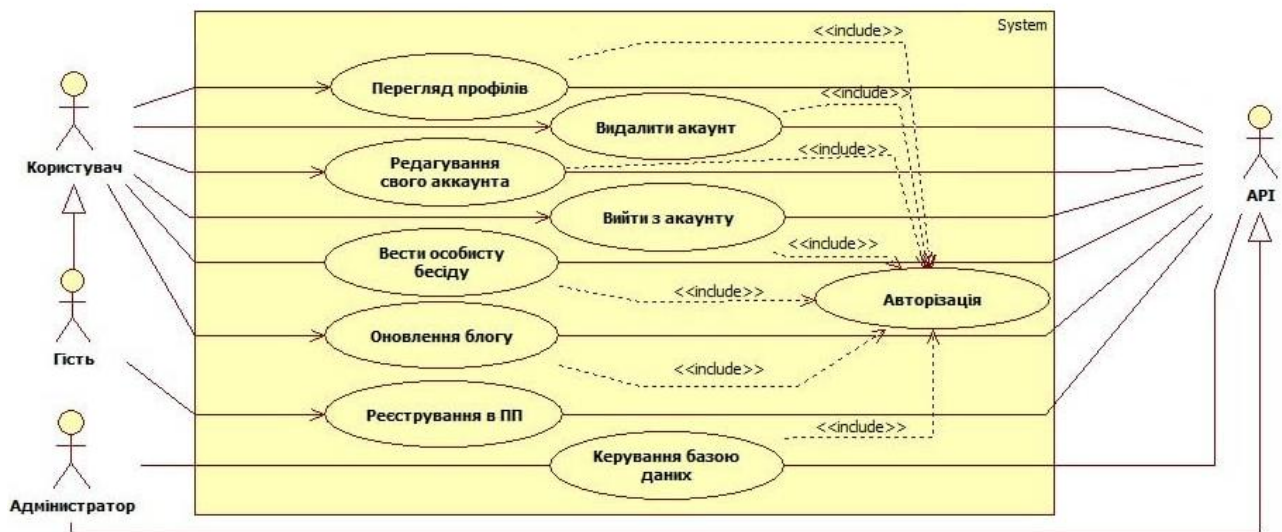


Рис. 1.3.3 – Діаграма прецедентів програмного продукту

1.3.4 Сценаріїв використання прецедентів програмного продукту

- Назва прецеденту: Реєстрація користувача
- Передумови початку виконання сценарію: перейти в призначений для реєстрації користувача інтерфейс ПП
- Актори: Гість, користувач, адміністратор
- Зацікавлені актори: Гість
- Результат: Успішна реєстрація користувача
- Основний успішний сценарій:
 1. ПП пропонує користувачеві ввести дані: Username, ім'я, пароль, підтвердження паролю, рідну мову, мову, яку користувач хоче вивчати, вступ про себе.
 2. Користувач передає дані ПП.
 3. ПП успішно зареєструвало користувача.
- Альтернативні сценарії:

2.1. Якщо користувач вводить некоректні дані, система попереджає його та підказує, що потрібно змінити для успішної реєстрації аккаунта.

- Назва прецеденту: Редагування профілю
- Передумови початку виконання сценарію: Авторизація
- Актори: Гість, користувач, адміністратор
- Зацікавлені актори: Користувач
- Результат: Успішне редагування профілю.
- Основний успішний сценарій:
 1. ПП пропонує користувачеві ввести дані, які він хоче відкоригувати: Username, пароль.
 2. Користувач передає дані ПП.
 3. ПП успішно редагує профіль користувача.
- Альтернативні сценарії:
 - 2.1. Якщо користувач вводить некоректні дані, система попереджає його та підказує, що потрібно змінити для успішного редагування профілю.
 - 3.1. Якщо користувач не хоче редагувати дані, він скасовує зміни.
- Назва прецеденту: Публікація статей до блогу
- Передумови початку виконання сценарію: Авторизація
- Актори: Гість, користувач, адміністратор
- Зацікавлені актори: Користувач
- Результат: Успішна публікація статті
- Основний успішний сценарій:
 1. ПП пропонує користувачеві ввести інформацію, яку він бажає опублікувати: Текст/посилання.
 2. Користувач передає дані ПП.
 3. Користувач успішно публікує статтю.
- Альтернативні сценарії:

3.1. Якщо користувач не бажає опублікувати статтю, він може не публікувати її.

- Назва прецеденту: Перегляд профілю.
- Передумови початку виконання сценарію: Авторизація
- Актори: Гість, користувач, адміністратор
- Зацікавлені актори: Користувач
- Результат: Успішний перегляд профілю.
- Основний успішний сценарій:
 1. ПП пропонує користувачеві перегляд/пошук профілів, які можуть його зацікавити відповідно до вивчаємих мов користувачів, імені, міста, країни або рідної мови.
 2. Користувач обирає профіль іншого користувача.
 3. Користувач успішно переглядає профіль.
- Альтернативні сценарії:
 - 2.1. Якщо користувач не бажає переглядати профіль, він може не обирати його.
- Назва прецеденту: Відправлення повідомлення.
- Передумови початку виконання сценарію: Авторизація
- Актори: Гість, користувач, адміністратор
- Зацікавлені актори: Користувач
- Результат: Успішне відправлення повідомлення іншому користувачеві.
- Основний успішний сценарій:
 1. ПП пропонує користувачеві перегляд/пошук профілів, які можуть його зацікавити відповідно до вивчаємих мов користувачів.
 2. Користувач обирає профіль іншого користувача.
 3. Користувач вводить дані, які він бажає відправити іншому користувачеві: Текст/посилання.
 4. Користувач передає дані ПП.

- Альтернативні сценарії:
 - 2.1. Користувач обирає профіль із тих, з якими він вже контактував, тобто обирає із іншої вкладки.
 - 3.1. Користувач може не вводити дані.
 - 4.1. Користувач може не відправляти своє повідомлення, а вийти з діалогу.

- Назва прецеденту: Вихід з аккаунту.
- Передумови початку виконання сценарію: Авторизація
- Актори: Гість, користувач, адміністратор
- Зацікавлені актори: Користувач
- Результат: Успішний вихід з акаунту.
- Основний успішний сценарій:
 1. Користувач відправляє запит ПП на вихід з профілю.
 2. Користувач успішно виходить з аккаунту.
- Альтернативні сценарії:
 - 2.1. Користувач може не відправляти запит та не виходити з аккаунту.

- Назва прецеденту: Видалення аккаунту.
- Передумови початку виконання сценарію: Авторизація
- Актори: Гість, користувач, адміністратор
- Зацікавлені актори: Користувач
- Результат: Успішне видалення аккаунту.
- Основний успішний сценарій:
 1. Користувач відправляє запит ПП на видалення свого акаунту.
 2. ПП попереджує користувача, що він не зможе повернути свій аккаунт, якщо видалить його, та всі дані будуть стерті.
 3. Користувач підтверджує свій запит видалення акаунту.
 4. Користувач успішно видаляє аккаунт.

- Альтернативні сценарії:
 - 2.1. Користувач може вийти із вкладки та не видаляти аккаунт.
 - 3.1. Користувач обирає не видаляти аккаунт, та виходить із цієї вкладки.
- Назва прецеденту: Керування базою даних.
- Передумови початку виконання сценарію: Авторизація
- Актори: Гість, користувач, адміністратор
- Зацікавлені актори: Адміністратор
- Результат: Успішне проведення операцій керування базою даних.
- Основний успішний сценарій:
 - 1. Адміністратор обирає операцію, яку він хоче зробити над даними.
 - 2. ПП пропонує адміністратору ввести дані відповідно до обраної операції.
 - 3. Адміністратор передає дані ПП відповідно до обраної операції.
 - 4. Адміністратор успішно проводить операцію керування базою даних.
- Альтернативні сценарії:
 - 1.1. Адміністратор не обирає операцію.
 - 3.1. Адміністратор вводить некоректні дані, система попереджає його та підказує, що потрібно змінити для успішного проведення операції керування базою даних.

1.4 Функціональні вимоги до програмного продукту

1.4.1. Багаторівнева класифікація функціональних вимог

Табл. 1.4.1 - Багаторівнева класифікація функціональних вимог

Ідентифікатор функції	Назва функції
FR 1	Авторизація користувача в програмному продукті
FR 1.1	Створення запиту у користувача на отримання його параметрів ідентифікації та аутентифікації
FR 1.2	Передача від користувача його параметрів ідентифікації та аутентифікації
FR 1.3	Успішна авторизація користувача в програмному продукті
FR 1.3.1	Повідомлення користувача про некоректно введені дані
FR 2	Перегляд профілів
FR 2.1	Створення запиту у користувача на вибір профілю, який він може захотіти переглянути
FR 2.2	Передача від користувача профілю, який він захотів переглянути
FR 2.3	Успішне передача даних обраного користувачем профіля користувачеві
FR 3	Редагування користувачем свого профілю
FR 3.1	Передача від користувача запиту на редагування свого профілю

FR 3.2	Створення запиту у користувача на отримання параметрів, які він хоче відредагувати
FR 3.3	Передача від користувача параметрів, які було змінено
FR 3.4	Успішне редагування свого профілю
FR 3.4.1	Повідомлення користувача про некоректно введені дані
FR 4	Публікація статей до блогу
FR 4.1	Створення запиту у користувача на отримання даних, необхідних для публікації статті
FR 4.2	Передача від користувача даних, які він передав з метою публікації статті
FR 4.3	Успішне публікація статті
FR 5	Ведення користувачем особистої співбесіди із іншими користувачами ПП
FR 5.1	Створення запиту у користувача на вибір профілю, якому він може надіслати повідомлення
FR 5.2	Передача від користувача профілю, якому він хоче надіслати повідомлення
FR 5.3	Створення запиту у користувача на отримання повідомлення, яке користувач хоче надіслати співрозмовнику
FR 5.4	Передача від користувача повідомлення

FR 5.5	Успішне відправлення користувачем повідомлення
FR 6	Реєстрація гостя в ПП
FR 6.1	Створення запиту у гостя на отримання параметрів, необхідних для реєстрації
FR 6.2	Передача від гостя параметрів
FR 6.3	Успішна реєстрація користувача в ПП
FR 6.3.1	Повідомлення гостя про некоректно введені дані
FR 7	Видалення акаунту в ПП
FR 7.1	Передача від користувача запросу на видалення акаунту
FR 7.2	Створення запиту у користувача на підтвердження дії
FR 7.3	Передача від користувача підтвердження
FR 7.4	Успішне видалення акаунту
FR 7.3.1	Передача від користувача відмовлення від видалення акаунту
FR 7.4.1	Аккаунт не видаляється
FR 8	Вихід з акаунту
FR 8.1	Передача від користувача запросу на вихід з акаунту

FR 8.2	Успішний вихід з акаунту
FR 9	Керування БД
FR 9.1	Створення запиту у адміністратора на операцію керування БД
FR 9.2	Передача від користувача параметрів
FR 9.3	Успішно виконана операція керування БД
FR 9.3.1	Повідомлення адміністратора про некоректно введені дані

1.4.2 Функціональний аналіз існуючих програмних продуктів

Табл. 1.4.2 - Функціональний аналіз існуючих програмних продуктів

Ідентифікатор функції	Tandem	HelloTalk	Speaky	InterPals	ITalki
FR 1.1	+	+	+	+	+
FR 1.2	+	+	+	+	+
FR 1.3	+	+	+	+	+
FR 1.3.1	+	+	+	+	+
FR 2.1	+	+	+	+	+
FR 2.2	+	+	+	+	+
FR 2.3	+	+	+	+	+
FR 3.1	+	+	+	+	+

Продовження таблиці 1.4.2

FR 3.2	+	+	+	+	+
FR 3.3	+	+	+	+	+
FR 3.4	+	+	+	+	+
FR 3.4.1	+	+	+	+	+
FR 4.1	-	+	-	+	+
FR 4.2	-	+	-	+	+
FR 4.3	-	+	-	+	+
FR 5.1	+	+	+	+	+
FR 5.2	+	+	+	+	+
FR 5.3	+	+	+	+	+
FR 5.4	+	+	+	+	+
FR 5.5	+	+	+	+	+
FR 6.1	+	+	+	+	+
FR 6.2	+	+	+	+	+
FR 6.3	+	+	+	+	+
FR 6.3.1	+	+	+	+	+
FR 7.1	+	+	+	+	+
FR 7.2	+	+	+	+	+
FR 7.3	+	+	+	+	+
FR 7.4	+	+	+	+	+
FR 7.3.1	+	+	+	+	+

Продовження таблиці 1.4.2

FR 7.4.1	+	+	+	+	+
FR 8.1	+	+	+	+	+
FR 8.2	+	+	+	+	+

1.5 Нефункціональні вимоги до програмного продукту

1.5.1 Опис зовнішніх інтерфейсів

1.5.1.1 Опис інтерфейсів користувача

1.5.1.1.1 Опис INPUT-інтерфейсів користувача

Табл. 1.5.1.1.1 - Опис INPUT-інтерфейсів користувача

Ідентифікатор функції	Засіб INPUT-потoku
FR 1.2	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 2.2	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 3.1	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 3.3	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 4.2	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран

Продовження таблиці 1.5.1.1.1

FR 5.2	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 5.4	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 6.2	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 7.1	2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 7.2	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 7.3	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 7.3.1	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран
FR 8.1	стандартна комп'ютерна клавіатура, 2/3-кнопочний маніпулятор типу "миша", сенсорний екран

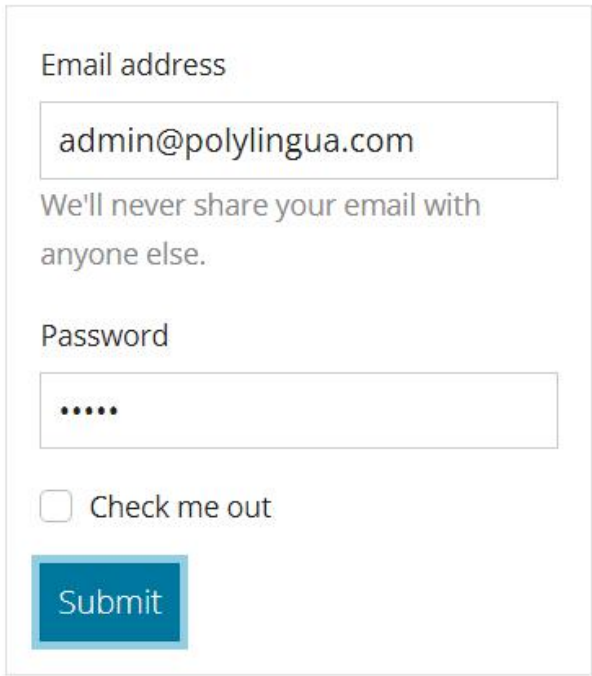
1.5.1.1.2 Опис OUTPUT-інтерфейсів користувача

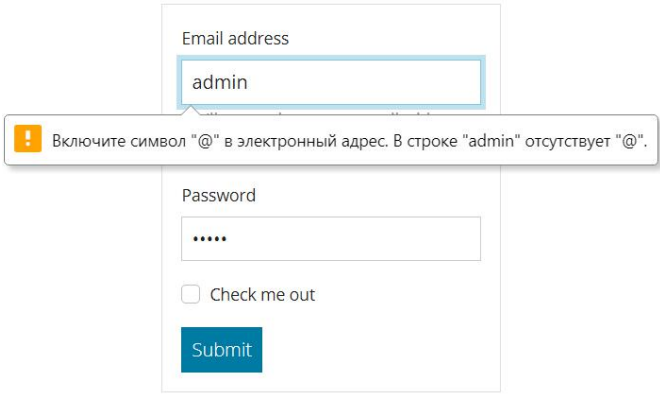
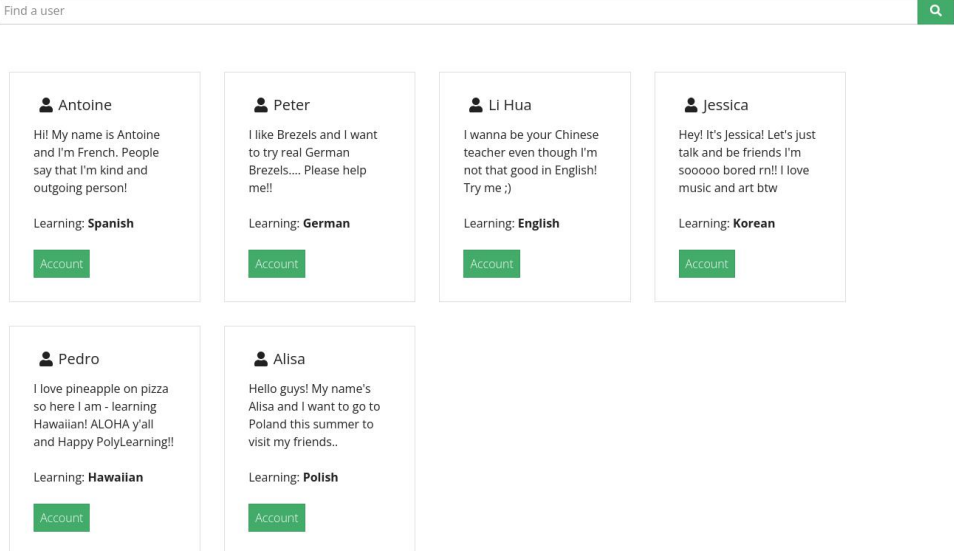
Табл. 1.5.1.1.2 - Опис OUTPUT-інтерфейсів користувача






Ідентифікатор функції	Засіб OUTPUT-потoku
FR 1.1	графічний інтерфейс
FR 1.3	графічний інтерфейс

FR 1.3.1	графічний інтерфейс
FR 2.2	графічний інтерфейс
FR 3	графічний інтерфейс
FR 4.2	графічний інтерфейс
FR 5.3	графічний інтерфейс
FR 6	графічний інтерфейс
FR 7	графічний інтерфейс
FR 8	графічний інтерфейс

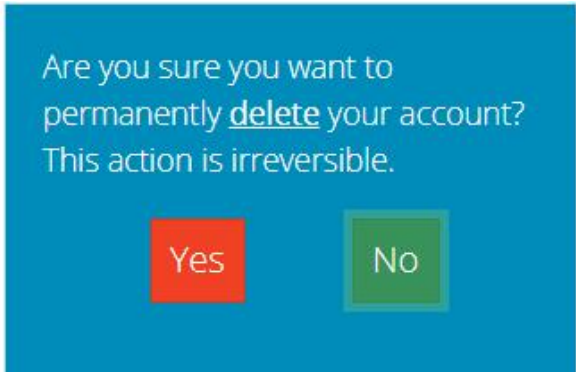
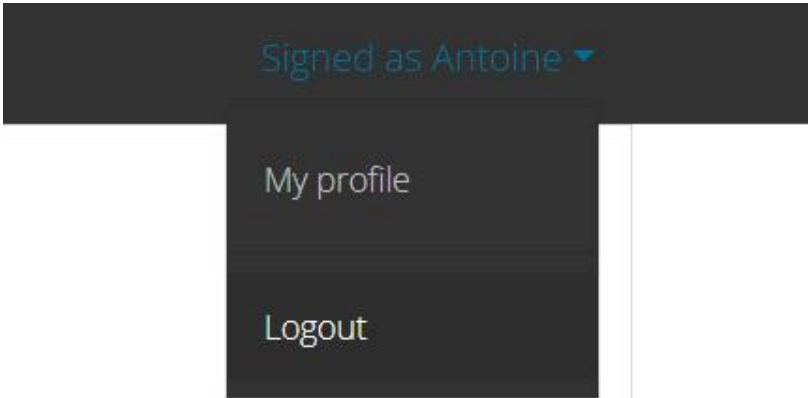
Табл. 1.5.1.1.3- Опис зовнішніх інтерфейсів

Ідентифікатор функції	Особливості використання
FR 1.1	<p>Вікно авторизації користувача в програмному продукті</p> 

FR 1.3.1	<p>Вікно авторизації користувача в програмному продукті при введенні некоректних даних</p> 
FR 2.2	<p>Перегляд користувачів ПП</p> 

FR 3	<p style="text-align: center;">Редагування користувачем свого профілю</p> <div> <p>Editing a profile</p> <p>New password</p> <input type="password" value="....."/> <p>Confirm new password</p> <input type="password" value="....."/> <p>Change the language you want to learn/practice</p> <div>English ▾</div> <p>Choose your level in this language</p> <p> <input type="radio"/> Beginner <input type="radio"/> Elementary <input checked="" type="radio"/> Pre-intermediate <input type="radio"/> Intermediate <input type="radio"/> Upper Intermediate <input type="radio"/> Advanced </p> <p>Submit</p> </div>
FR 4.2	<p style="text-align: center;">Публікація поста в блог</p> <div> <p>Write a new post</p> <p>Publish</p> <div> <p> Antoine</p> <p>Hi! My name is Antoine and I'm French. People say that I'm kind and outgoing person!</p> <p>Last summer I was in Barcelona with my parents and I fell in love with Spanish language. So I'm really looking forward to all the people speaking Spanish.</p> <p>As a native French speaker I can help you with my language, and you are going to help me in my language learning path. Please text me if you are interested!!!!</p>  </div> </div>
FR 5.3	<p style="text-align: center;">Особиста співбесіда користувачів</p> <div> <div> <div>  Antoine Bonjour! Oui! Comment tu t'appelles? </div> <div>  Jessica Hi. Saw you in this app today </div> <div>  Li Hua Hey, wanna learn some Chinese? </div> </div> <div> <p>Hi! I want to start learning French, would you mind helping me?</p> <p>Bonjour! Oui! Comment tu t'appelles?</p> <p>Write a message here</p> </div> </div>

FR 6	<p>Реєстрація гостя в ПП (перше вікно)</p> <div data-bbox="729 297 1185 1140"> <p>Registration</p> <p>Email address</p> <input type="text" value="Enter email"/> <p>We'll never share your email with anyone else.</p> <p>Username</p> <input type="text" value="Enter username"/> <p>Your Username must be 4-16 characters.</p> <p>Password</p> <input type="password" value="Enter your password"/> <p>Repeat your password</p> <input type="password" value="Repeat your password"/> <p>Next</p> </div>
FR 6	<p>Реєстрація гостя в ПП (наступне друге вікно)</p> <div data-bbox="651 1254 1272 2051"> <p>Native language</p> <div data-bbox="686 1388 1238 1462"> Ukrainian </div> <p>The language you want to learn</p> <div data-bbox="686 1563 1238 1637"> English </div> <p>Choose your level in this language</p> <p> <input type="radio"/> Beginner <input type="radio"/> Elementary <input type="radio"/> Pre-intermediate <input checked="" type="radio"/> Intermediate <input type="radio"/> Upper Intermediate <input type="radio"/> Advanced </p> <p>Confirm registration</p> </div>

FR 7	<p>Видалення акаунту</p> 
FR 8	<p>Вихід з акаунту</p> 

1.5.1.2 Опис інтерфейсу із зовнішніми пристроями

Табл. 1.5.1.2 - Опис інтерфейсу із зовнішніми пристроями

Ідентифікатор функції	Зовнішній пристрій
FR 1.1	Desktop, Laptop, Смартфон
FR 1.2	Desktop, Laptop, Смартфон
FR 1.3	Desktop, Laptop, Смартфон
FR 1.3.1	Desktop, Laptop, Смартфон
FR 2.1	Desktop, Laptop, Смартфон
FR 2.2	Desktop, Laptop, Смартфон

FR 2.3	Desktop, Laptop, Смартфон
FR 3	Desktop, Laptop, Смартфон
FR 3.1	Desktop, Laptop, Смартфон
FR 3.2	Desktop, Laptop, Смартфон
FR 3.3	Desktop, Laptop, Смартфон
FR 3.4	Desktop, Laptop, Смартфон
FR 3.4.1	Desktop, Laptop, Смартфон
FR 4	Desktop, Laptop, Смартфон
FR 4.1	Desktop, Laptop, Смартфон
FR 4.2	Desktop, Laptop, Смартфон
FR 4.3	Desktop, Laptop, Смартфон
FR 5	Desktop, Laptop, Смартфон
FR 5.1	Desktop, Laptop, Смартфон
FR 5.2	Desktop, Laptop, Смартфон
FR 5.3	Desktop, Laptop, Смартфон
FR 5.4	Desktop, Laptop, Смартфон
FR 5.5	Desktop, Laptop, Смартфон
FR 6	Desktop, Laptop, Смартфон
FR 6.1	Desktop, Laptop, Смартфон
FR 6.2	Desktop, Laptop, Смартфон
FR 6.3	Desktop, Laptop, Смартфон

FR 6.3.1	Desktop, Laptop, Смартфон
FR 7	Desktop, Laptop, Смартфон
FR 7.1	Desktop, Laptop, Смартфон
FR 7.2	Desktop, Laptop, Смартфон
FR 7.3	Desktop, Laptop, Смартфон
FR 7.4	Desktop, Laptop, Смартфон
FR 7.3.1	Desktop, Laptop, Смартфон
FR 7.4.1	Desktop, Laptop, Смартфон
FR 8	Desktop, Laptop, Смартфон
FR 8.1	Desktop, Laptop, Смартфон
FR 8.2	Desktop, Laptop, Смартфон

1.5.1.3 Опис програмних інтерфейсів

Для реалізації більшості функцій ПП знадобитися:

а) Для WEB-версії: версія браузера, що підтримує технічні вимоги WEB-реалізації ПП.

б) Для смартфона: версія ОС, що підтримує додатки з технічними вимогами ПП, Версія браузера, що підтримує технічні вимоги WEB-реалізації ПП.

1.5.1.4 Опис інтерфейсів передачі інформації

Для реалізації більшості функцій ПП знадобитися такі технічні зовнішні засоби як:

а) Провідні інтерфейси - Ethernet (при необхідності).

б) Безпроводні інтерфейси - WI-FI (при необхідності).

1.5.1.5 Опис атрибутів продуктивності

Табл. 1.5.1.5 - Опис атрибутів продуктивності

Ідентифікатор функції	Максимальний час реакції ПП на дії користувачів, секунди
FR 1.1	3
FR 1.2	3
FR 2.2	5
FR 3.2	3
FR 4.1	3
FR 4.3	3

Продовження таблиці 1.5.1.5

FR 5.2	3
FR 6.2	3
FR 6.4	3
FR 7.2	5
FR 8.1	2
FR 8.2	3

Максимальною кількістю користувачів, яких одночасно може обслуговувати програмний продукт залежить від реалізації продукту та обраного серверу.

2 Планування процесу розробки програмного продукту

2.1 Планування ітерацій розробки програмного продукту

З метою забезпечення вимог таких рекомендацій IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненність функціональних вимог до ПП, визначено функціональні пріоритети, які будуть використані при плануванні ітерацій розробки ПП. Результати представлено в таблиці 2.1

Табл. 2.1 - Опис атрибутів продуктивності

Ідентифікатор функції	Функціональні залежності	Вплив на досягнення мети, %	Пріоритет функції
FR 1	FR 7	5	M
FR 1.1	-	3	M
FR 1.2	-	2	M
FR 2	FR 1	20	S
FR 2.1	-	10	S
FR 2.2	-	10	S
FR 3	FR 1	5	S
FR 3.1	-	2	S
FR 3.2	-	3	S
FR 4	FR 1	30	M
FR 4.1	-	15	M
FR 4.2	-	15	M
FR 5	FR 1	30	M

Продовження таблиці 2.1

FR 5.1	-	5	M
FR 5.2	-	5	M
FR 5.3	-	10	M
FR 5.4	-	10	M
FR 6	-	10	M
FR 6.1	-	4	M
FR 6.2	-	6	M
FR 7	FR 1	0	C
FR 7.1	-	0	C
FR 7.2	-	0	C
FR 7.3	-	0	C
FR 8	FR 1	0	S
FR 8.1	-	0	S
FR 9	FR 1	0	S
FR 9.1	-	0	S
FR 9.2	-	0	S

2.2 Концептуальний опис архітектури програмного продукту

Архітектурний тип програмного продукту - Rich WEB Application (RWA).

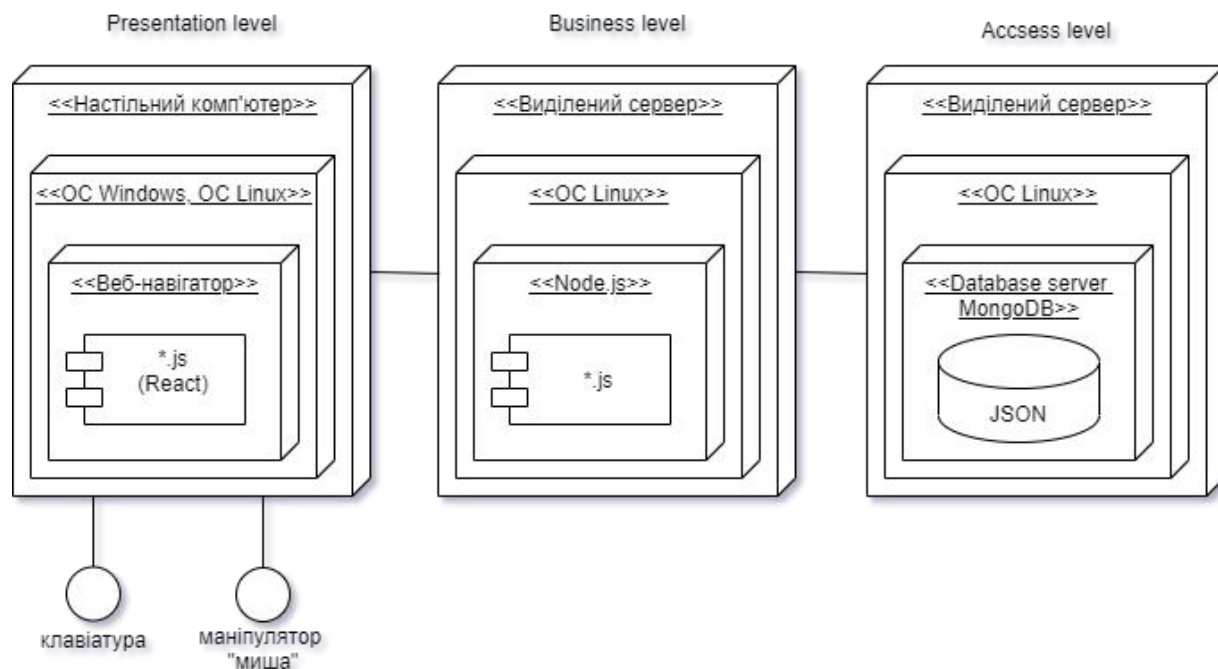


Рис. 2.2 – Діаграма прецедентів програмного продукту

2.3 План розробки програмного продукту

2.3.1 Оцінка трудомісткості розробки програмного продукту

Табл. 2.3.1.1 - Визначення вагових показників акторів А

Актор	Ваговий коефіцієнт
Гість	3
Користувач	3
Адміністратор	3

$$A = 3 * 3 = 9$$

Табл. 2.3.1.2 - Визначення вагових показників прецедентів UC

Прецедент	Кількість кроків сценарію	Ваговий коефіцієнт
Авторизація користувача в програмному продукті	4	10
Перегляд профілів	3	5
Редагування користувачем свого профілю	5	10
Публікація статей до блогу	3	5
Ведення користувачем особистої співбесіди із іншими користувачами ПП	5	10
Реєстрація гостя в ПП	4	10
Видалення акаунту в ПП	6	10
Вихід з акаунту	2	5
Керування базою даних	4	10

$$UC = 6*10 + 4*5 = 80$$

$$UUCP = 9 + 80 = 89$$

Табл. 2.3.1.3 - Визначення технічної складності проекту

Показник	Опис показника	Вага
T1	Розподілена система	1
T2	Висока продуктивність (пропускна здатність)	3
T3	Робота кінцевих користувачів в режимі он-лайн	5
T4	Складна обробка даних	1
T5	Повторне використання коду	1
T6	Простота установки	1
T7	Простота використання	5
T8	Переносимість	4
T9	Простота внесення змін	4
T10	Паралелізм	2
T11	Спеціальні вимоги до безпеки	5
T12	Безпосередній доступ до системи з боку зовнішніх користувачів	1
T13	Спеціальні вимоги до навчання користувачів	0

$$TCF = 0,6 + (0,01 * (ST_i * Вага_i))$$

$$TCF = 0.6 + (0.01 * (1*2 + 3*1 + 5*1 + 1*(-1) + 1*1 + 1*0.5 + 5*0.5 + 4*2 + 4*1 + 2*1 + 5*1 + 1*1 + 0*1)) = 0.6 + (0.01*33) = 0.6+0.33 = 0.93$$

Табл. 2.3.1.4 - Визначення рівня кваліфікації розробників

Показник	Опис показника	Вага
F1	Знайомство з технологією	2

F2	Досвід розробки додатків	3
F3	Досвід використання об'єктно-орієнтованого підходу	2
F4	Наявність провідного аналітика	0
F5	Мотивація	5
F6	Стабільність вимог	3
F7	Часткова зайнятість	2
T8	Складні мови програмування	5

$$EF = 1,4 + (-0,03 * (SF_i * Bar_{ai}))$$

$$EF = 1.4 + (-0.3 * (2 * 1.5 + 3 * 0.5 + 2 * 1 + 0 * 0.5 + 5 * 1 + 3 * 2 + 2 * (-1) + 5 * (-1))) = 1.4 + (-0.03 * 10.5) = 1.4 - 0.315 = 1.085$$

$$UCP = 89 * 0.93 * 1.085 = 89.81$$

2.3.2 Визначення дерева робіт з розробки програмного продукту

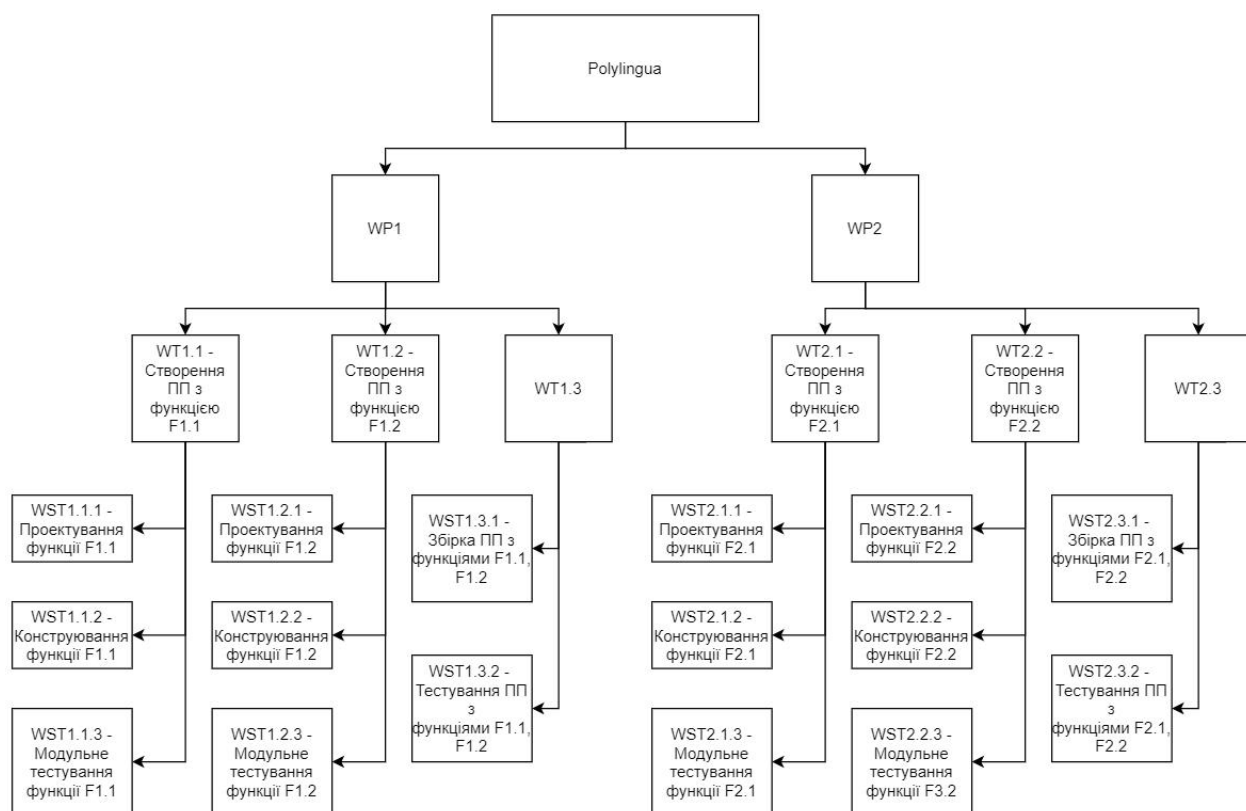


Рис. 2.3.1 – WBS-дерево робіт (WP1, WP2)

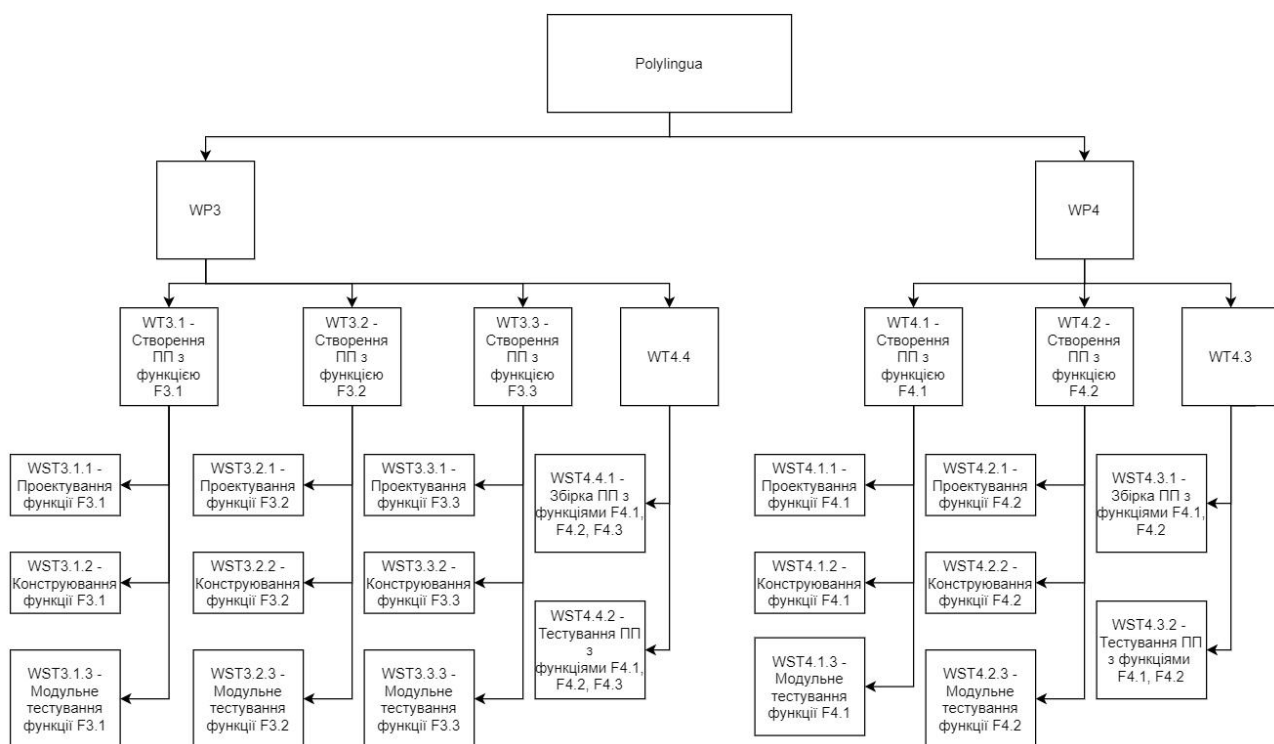


Рис. 2.3.2 – WBS-дерево робіт (WP3, WP4)

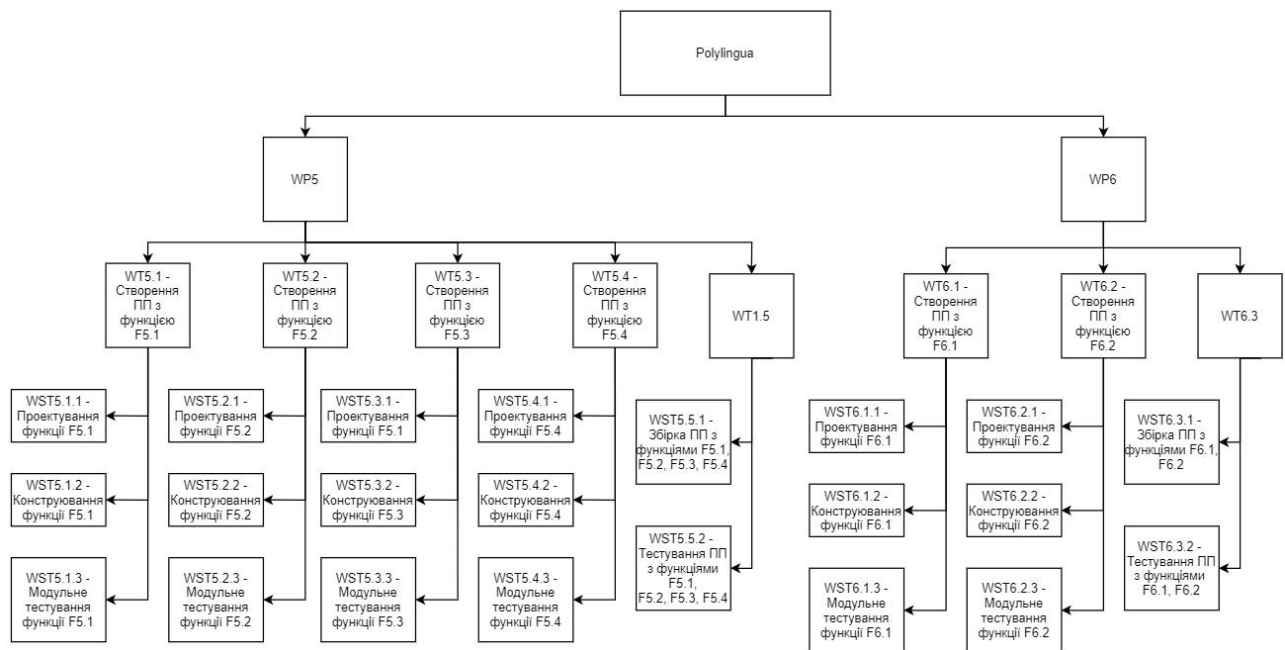


Рис. 2.3.3 – WBS-дерево робіт (WP5, WP6)

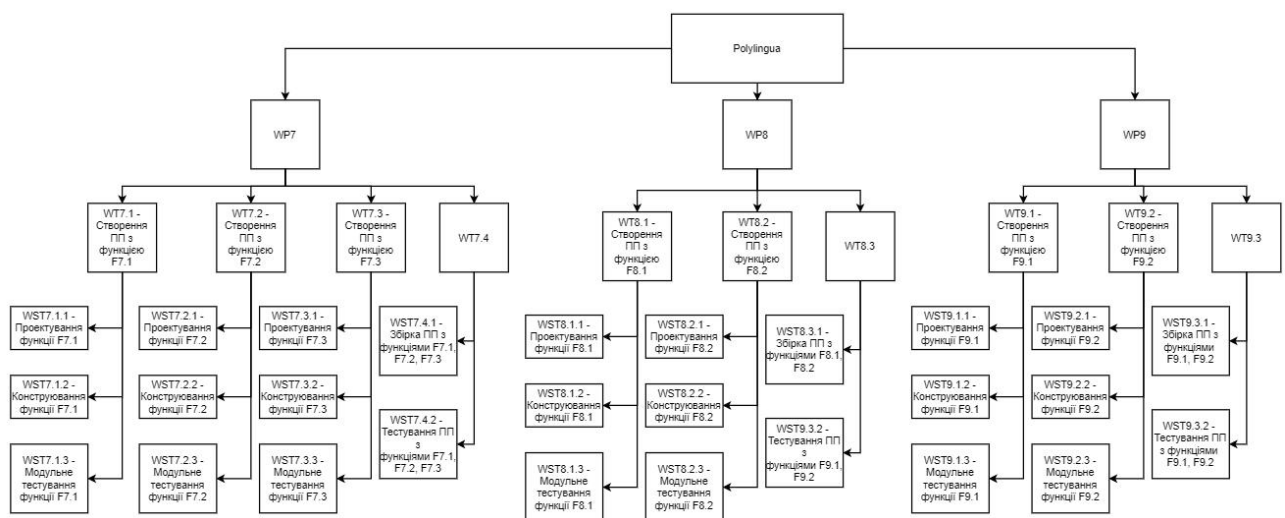


Рис. 2.3.4 – WBS-дерево робіт (WP7, WP8, WP9)

2.3.3 Графік робіт з розробки програмного продукту

2.3.3.1 Таблиця з графіком робіт

Табл. 2.3.3.1 - Визначення дерева робіт

Підзадача	Виконавець	Підзадача	Виконавець
WST1.1.1	Савченко Б.	WST3.1.1	Урбанович М.

Продовження таблиці 2.3.3.1

WST1.1.1	Савченко Б.	WST2.1.1	Урбанович М.
WST1.1.2	Савченко Б.	WST2.1.2	Урбанович М.
WST1.1.3	Савченко Б.	WST2.1.3	Урбанович М.
WST1.2.1	Савченко Б.	WST2.2.1	Урбанович М.
WST1.2.2	Савченко Б.	WST2.2.2	Урбанович М.
WST1.2.3	Савченко Б.	WST2.2.3	Урбанович М.
WST1.3.1	Савченко Б.	WST2.3.1	Урбанович М.
WST1.3.2	Савченко Б.	WST2.3.2	Урбанович М.
WST3.3.3	Урбанович М.	WST5.2.3	Урбанович М.
WST3.4.1	Урбанович М.	WST5.3.1	Савченко Б.
WST3.4.2	Урбанович М.	WST5.3.2	Савченко Б.
WST4.1.1	Савченко Б.	WST5.3.3	Савченко Б.
WST4.1.2	Савченко Б.	WST5.4.1	Урбанович М.
WST4.1.3	Савченко Б.	WST5.4.2	Урбанович М.
WST4.2.1	Савченко Б.	WST5.4.3	Урбанович М.
WST4.2.2	Савченко Б.	WST5.5.1	Савченко Б.
WST4.2.3	Савченко Б.	WST5.5.2	Савченко Б.
WST4.3.1	Савченко Б.	WST6.1.1	Савченко Б.
WST4.3.2	Савченко Б.	WST6.1.2	Савченко Б.
WST5.1.1	Савченко Б.	WST6.1.3	Савченко Б.
WST5.1.2	Савченко Б.	WST6.2.1	Савченко Б.

Продовження таблиці 2.3.3.1

WST5.1.3	Савченко Б.	WST6.2.2	Савченко Б.
WST5.2.1	Урбанович М.	WST6.2.3	Савченко Б.
WST5.2.2	Урбанович М.	WST6.3.1	Савченко Б.
WST5.3.2	Савченко Б.	WST8.2.2	Урбанович М.
WST7.1.1	Савченко Б.	WST8.2.3	Урбанович М.
WST7.1.2	Савченко Б.	WST8.3.1	Урбанович М.
WST7.1.3	Савченко Б.	WST8.3.2	Урбанович М.
WST7.2.1	Савченко Б.	WST9.1.1	Савченко Б.
WST7.2.2	Савченко Б.	WST9.1.2	Савченко Б.
WST7.2.3	Савченко Б.	WST9.1.3	Савченко Б.
WST7.3.1	Савченко Б.	WST9.2.1	Урбанович М.
WST7.3.2	Савченко Б.	WST9.2.2	Урбанович М.
WST7.3.3	Савченко Б.	WST9.2.3	Урбанович М.
WST7.4.1	Савченко Б.	WST9.3.1	Савченко Б.
WST7.4.2	Савченко Б.	WST9.3.2	Савченко Б.
WST8.1.1	Урбанович М.	-	-
WST8.1.2	Урбанович М.	-	-
WST8.1.3	Урбанович М.	-	-
WST8.2.1	Урбанович М.	-	-

2.3.3.2 Діаграма Ганта

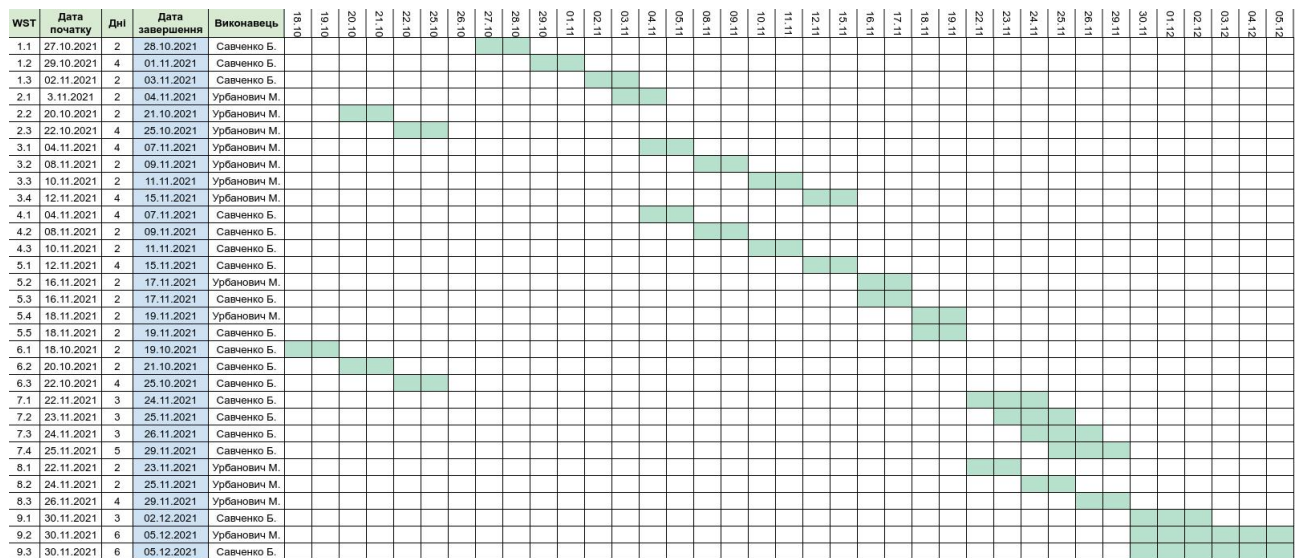


Рис. 2.3.3.2 – Діаграма Ганта

3 Проектування програмного продукту

3.1 Концептуальне та логічне проектування структур даних програмного продукту

3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів

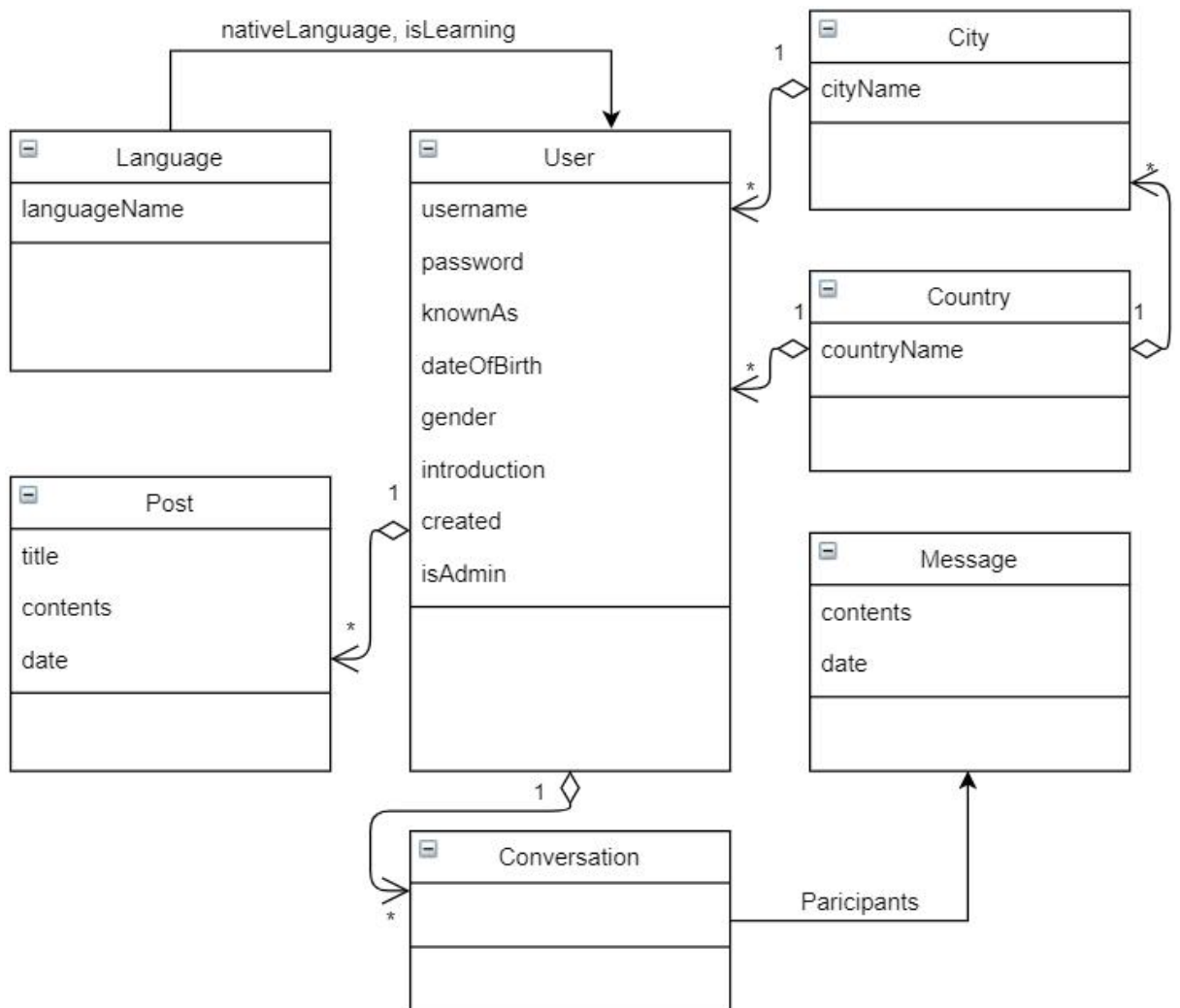


Рис. 3.1.1 – UML-діаграма концептуальних класів

3.1.2 Логічне проектування структур даних

```
4  const userSchema = new mongoose.Schema({
5      username: {
6          type: String,
7          required: true,
8          unique: true,
9          lowercase: true,
10     },
11     password: {
12         type: String,
13         required: true,
14         minlength: 6,
15     },
16     knownAs: {
17         type: String,
18         required: true,
19         lowercase: true,
20     },
21     nativeLanguage: {
22         type: String,
23         required: true,
24     },
25     isLearning: {
26         type: String,
27         required: true,
28     },
29     dateOfBirth: {
30         type: Date,
31         required: true,
32     },
33     gender: {
34         type: String,
35         required: true,
36         enum: ['Male', 'Female', 'Unknown']
37     },
38     country: {
39         type: String,
40         required: true,
41     },
42     city: {
43         type: String,
44         required: true,
45     },
46     introduction: {
47         type: String,
48         required: true,
49     },
50     created: {
51         type: Date,
52         required: true,
53         default: Date.now
54     },
55     isAdmin: {
56         type: Boolean,
57         required: true,
58         default: false
59     },
60 }, {
61     timestamps: true
62 })
```

Рис. 3.1.2.1 – Логічне проектування структур даних (UserSchema)

```
3  const languageSchema = new mongoose.Schema({
4      languageName: {
5          type: String,
6          required: true,
7          unique: true,
8      },
9  }, {
10     timestamps: true
11 })
```

Рис. 3.1.2.2 – Логічне проектування структур даних (LanguageSchema)

```

3  const countrySchema = new mongoose.Schema({
4    countryName: {
5      type: String,
6      required: true,
7      unique: true,
8    },
9  }, {
10    timestamps: true,
11  })

```

Рис. 3.1.2.3 – Логічне проектування структур даних (CountrySchema)

```

3  const citySchema = new mongoose.Schema({
4    cityName: {
5      type: String,
6      required: true,
7      unique: true,
8    },
9    countryId: {
10      type: String,
11      required: true,
12    },
13  }, {
14    timestamps: true
15  })

```

Рис. 3.1.2.4 – Логічне проектування структур даних (CitySchema)

```

3  const postSchema = new mongoose.Schema({
4    title: {
5      type: String,
6      trim: true,
7      required: true
8    },
9    contents: {
10      type: String,
11      trim: true,
12      required: true
13    },
14    userId: {
15      type: String,
16      required: true,
17    },
18    sectionId: {
19      type: String,
20      required: true,
21    },
22    date: {
23      type: Date,
24      required: true,
25      default: Date.now
26    },
27  }, {
28    timestamps: true
29  })

```

Рис. 3.1.2.5 – Логічне проектування структур даних (postSchema)

```

3  const messageSchema = new mongoose.Schema({
4    conversationId: {
5      type: String,
6      required: true
7    },
8    sender: {
9      type: String,
10     required: true,
11   },
12   contents: {
13     type: String,
14     trim: true,
15     required: true
16   },
17   date: {
18     type: Date,
19     required: true,
20     default: Date.now
21   },
22 }, {
23   timestamps: true,
24 })

```

Рис. 3.1.2.6 – Логічне проектування структур даних (MessageSchema)

```

3  const conversationSchema = new mongoose.Schema({
4    participants: {
5      type: Array,
6      required: true,
7    }
8  }, {
9    timestamps: true,
10 })

```

Рис. 3.1.2.7 – Логічне проектування структур даних (conversationSchema)

3.2 Проектування програмних функцій

Табл. 3.2 – Проектування програмних функцій

Ідентифікатор функції	Назва функції	Функція
FR 1	Авторизація користувача в програмному продукті	userLogin()
FR 1.1	Створення запиту у користувача на отримання його параметрів ідентифікації та аутентифікації	
FR 1.2	Передача від користувача його параметрів ідентифікації та аутентифікації	
FR 1.3	Успішна авторизація користувача в програмному продукті	
FR 1.3.1	Повідомлення користувача про некоректно введені дані	
FR 2	Перегляд профілів	
FR 2.1	Створення запиту у користувача на вибір профілю, який він може захотіти переглянути	getUsers()
FR 2.2	Передача від користувача профілю, який він захотів переглянути	getUsers(id)
FR 2.3	Успішне передача даних обраного користувачем профіля користувачеві	
FR 3	Редагування користувачем свого профілю	

FR 3.1	Передача від користувача запиту на редагування свого профілю	editProfile(id)
FR 3.2	Створення запиту у користувача на отримання параметрів, які він хоче відредагувати	
FR 3.3	Передача від користувача параметрів, які було змінено	
FR 3.4	Успішне редагування свого профілю	
FR 3.4.1	Повідомлення користувача про некоректно введені дані	
FR 4	Публікація статей до блогу	
FR 4.1	Створення запиту у користувача на отримання даних, необхідних для публікації статті	createPost(postTitle, postBody)
FR 4.2	Передача від користувача даних, які він передав з метою публікації статті	
FR 4.3	Успішна публікація статті	
FR 5	Ведення користувачем особистої співбесіди із іншими користувачами ПП	createMessage(senderId, recieverId, messageBody)
FR 5.1	Створення запиту у користувача на вибір профілю, якому він може надіслати повідомлення	

FR 5.2	Передача від користувача профілю, якому він хоче надіслати повідомлення	
FR 5.3	Створення запиту у користувача на отримання повідомлення, яке користувач хоче надіслати співрозмовнику	
FR 5.4	Передача від користувача повідомлення	
FR 5.5	Успішне відправлення користувачем повідомлення	
FR 6	Реєстрація гостя в ПП	register(username, password)
FR 6.1	Створення запиту у гостя на отримання параметрів, необхідних для реєстрації	
FR 6.2	Передача від гостя параметрів	
FR 6.3	Успішна реєстрація користувача в ПП	
FR 6.3.1	Повідомлення гостя про некоректно введені дані	
FR 7	Видалення акаунту в ПП	deleteAccount (accountId)
FR 7.1	Передача від користувача запиту на видалення акаунту	

Продовження таблиці 3.2

FR 7.2	Створення запиту у користувача на підтвердження дії	
FR 7.3	Передача від користувача підтвердження	
FR 7.4	Успішне видалення акаунту	
FR 7.3.1	Передача від користувача відмовлення від видалення акаунту	
FR 7.4.1	Акаунт не видаляється	
FR 8	Вихід з акаунту	logout()
FR 8.1	Передача від користувача запиту на вихід з акаунту	
FR 8.2	Успішний вихід з акаунту	
FR 9	Керування базою даних	deletePost(postId)
FR 9.1	Створення запиту у адміністратора на операцію керування БД	
FR 9.2	Передача від користувача параметрів	

FR 9.3	Успішно виконана операція керування БД	
FR 9.3.1	Повідомлення адміністратора про некоректно введені дані	

3.3 Проектування алгоритмів роботи методів програмних функцій

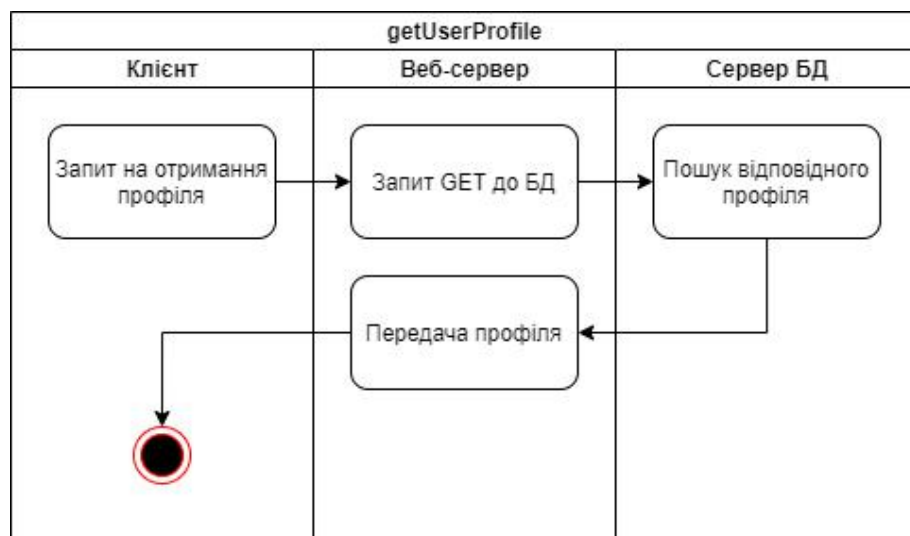


Рис. 3.3.2 – Алгоритм роботи методу getUserProfile

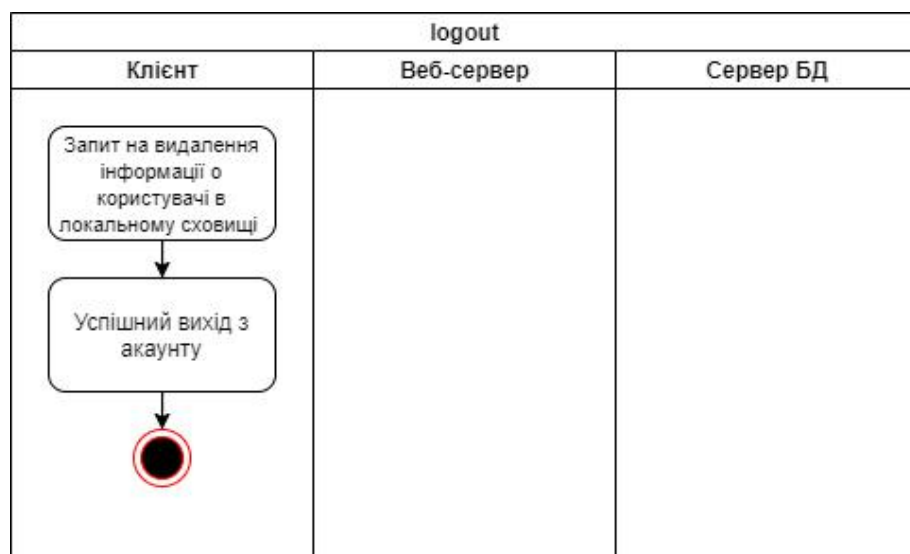


Рис. 3.3.8 – Алгоритм роботи методу logout

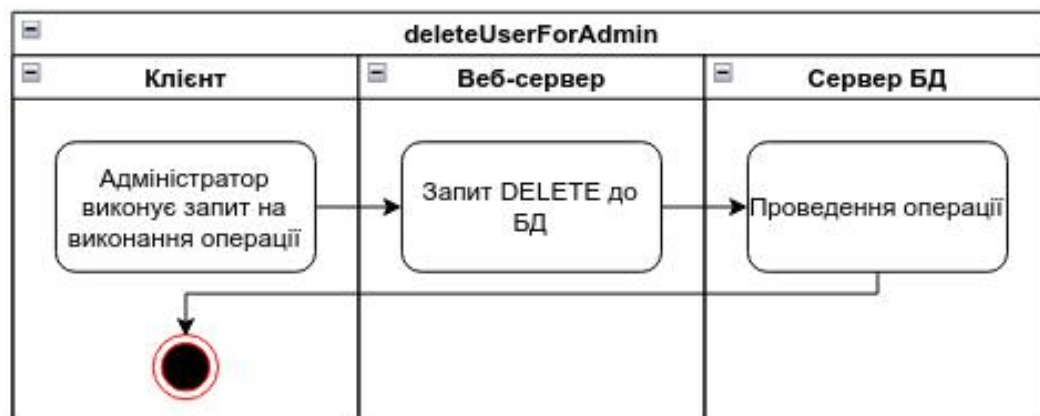


Рис. 3.3.9 – Алгоритм роботи методу deleteUser

3.4 Проектування тестових сценаріїв верифікації роботи програмних модулів

3.4.1 Проектування тестових сценаріїв верифікації функціональних вимог

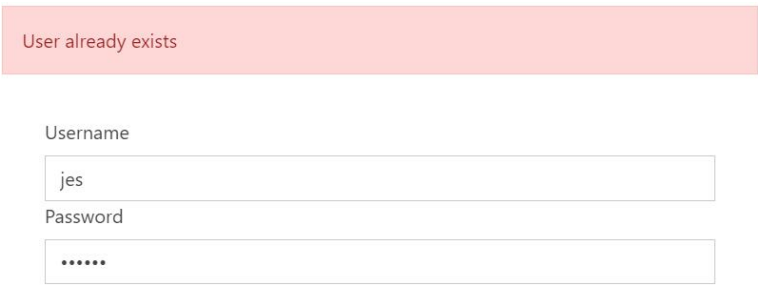
Табл. 3.4.1 – Проектування тестових сценаріїв верифікації функціональних вимог (BlackBox-методи)


FR ID	Test Case ID	Опис значень вхідних даних до методу (процедури, функції)	Опис очікуваних значень результату виконання методу
FR 4	TC1	username = username = jes (не повинна збігатися з вже існуючими користувачами) password = 123456 password confirmation = 1234567890	Рядок з повідомленням про помилку: username вже існує.
FR 4	TC2	username = andrew123 password = 123456 password confirmation = 1234567890	Рядок з повідомленням про помилку: введені паролі не збігаються,

FR 4	TC3	username = andrew123 password = 123456 password confirmation = 123456	Успішно відредагований профіль
------	-----	---	--------------------------------------

3.4.2 Проектування тестових сценаріїв верифікації нефункціональних вимог

Табл. 3.4.2 – Проектування тестових сценаріїв нефункціональних вимог

NFR ID	Test Case ID	Опис дій з ПП на рівні інтерфейсу користувача	Опис очікуваних значень результату виконання методу
NFR4	TC1	<ol style="list-style-type: none"> Для елементу “username” ввести значення jes Для елементу “password” ввести значення 123456 Натиснути на кнопку відправлення форми 	Рядок з повідомленням про помилку: username вже існує.
Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача NFR4, TC12			

NFR4	TC2	<ol style="list-style-type: none"> 1. Для елемента “username” ввести значення andrew123 2. Для елемента “password” ввести значення 123456 3. Для елемента «confirm password» ввести значення 1234567890 4. Натиснути на кнопку відправлення форми 	Рядок з повідомленням про помилку: введені паролі не збігаються
<p>Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача NFR4, TC3</p>		 <p>The screenshot shows a web form with three input fields: 'Username' (containing 'andrew123'), 'Password' (containing '*****'), and 'Confirm Password' (containing '*****'). Above the fields is a red error message box that says 'Passwords don't match'.</p>	
NFR4	TC14	<ol style="list-style-type: none"> 1. Для елемента “username” ввести значення andrew123 2. Для елемента “password” ввести значення 123456 3. Для елемента «confirm password» ввести значення 123456 4. Натиснути на кнопку відправлення форми 	Успішно відредагований профіль

<p>Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача NFR4, TC14</p>	<div style="background-color: #d4edda; padding: 5px; margin-bottom: 10px;">Profile successfully updated!</div> <p>Username</p> <input type="text" value="andreww"/> <p>Password</p> <input type="password" value="*****"/> <p>Confirm Password</p> <input type="password" value="*****"/>
--	---

3.4.3 Створення матриці відповідності вимог до програмного продукту

Табл. 3.4.3 – Матриця відповідності вимог до програмного продукту

FR ID	NFRID	TCID
FR7	NFR7	TC1
FR7	NFR7	TC2
FR7	NFR7	TC3
FR7	NFR7	TC4
FR7	NFR7	TC5
FR7	NFR7	TC6
FR7	NFR7	TC7
FR7	NFR7	TC8

Продовження таблиці 3.4.3

FR7	NFR7	TC9
FR7	NFR7	TC10
FR7	NFR7	TC11
FR ID	NFRID	TCID
FR4	NFR 4	TC12
FR4	NFR 4	TC13
FR4	NFR 4	TC14
FR4	NFR 4	TC15
FR4	NFR 4	TC16

4 Конструювання програмного продукту

4.1 Особливості конструювання структур даних

```
4  const userSchema = new mongoose.Schema({
5      username: {
6          type: String,
7          required: true,
8          unique: true,
9          lowercase: true,
10     },
11     password: {
12         type: String,
13         required: true,
14         minlength: 6,
15     },
16     knownAs: {
17         type: String,
18         required: true,
19         lowercase: true,
20     },
21     nativeLanguage: {
22         type: String,
23         required: true,
24     },
25     isLearning: {
26         type: String,
27         required: true,
28     },
29     dateOfBirth: {
30         type: Date,
31         required: true,
32     },
33     gender: {
34         type: String,
35         required: true,
36         enum: ['Male', 'Female', 'Unknown']
37     },
38     country: {
39         type: String,
40         required: true,
41     },
42     city: {
43         type: String,
44         required: true,
45     },
46     introduction: {
47         type: String,
48         required: true,
49     },
50     created: {
51         type: Date,
52         required: true,
53         default: Date.now
54     },
55     isAdmin: {
56         type: Boolean,
57         required: true,
58         default: false
59     },
60 }, {
61     timestamps: true
62 })
```

Рис. 4.1.1 – Конструювання структур даних (UserSchema)

```
3  const languageSchema = new mongoose.Schema({
4      languageName: {
5          type: String,
6          required: true,
7          unique: true,
8      },
9  }, {
10     timestamps: true
11 })
```

Рис. 4.1.2 – Конструювання структур даних (LanguageSchema)

```

3  const countrySchema = new mongoose.Schema({
4    countryName: {
5      type: String,
6      required: true,
7      unique: true,
8    },
9  }, {
10    timestamps: true,
11  })

```

Рис. 4.1.3 – Конструювання структур даних (CountrySchema)

```

3  const citySchema = new mongoose.Schema({
4    cityName: {
5      type: String,
6      required: true,
7      unique: true,
8    },
9    countryId: {
10      type: String,
11      required: true,
12    },
13  }, {
14    timestamps: true
15  })

```

Рис. 4.1.4 – Конструювання структур даних (CitySchema)

```

3  const postSchema = new mongoose.Schema({
4    title: {
5      type: String,
6      trim: true,
7      required: true
8    },
9    contents: {
10      type: String,
11      trim: true,
12      required: true
13    },
14    userId: {
15      type: String,
16      required: true,
17    },
18    sectionId: {
19      type: String,
20      required: true,
21    },
22    date: {
23      type: Date,
24      required: true,
25      default: Date.now
26    },
27  }, {
28    timestamps: true
29  })

```

Рис. 4.1.5 – Конструювання структур даних (postSchema)


```

3  const messageSchema = new mongoose.Schema({
4    conversationId: {
5      type: String,
6      required: true
7    },
8    sender: {
9      type: String,
10     required: true,
11   },
12   contents: {
13     type: String,
14     trim: true,
15     required: true
16   },
17   date: {
18     type: Date,
19     required: true,
20     default: Date.now
21   },
22 }, {
23   timestamps: true,
24 })

```

Рис. 4.1.6 – Конструювання структур даних (MessageSchema)

```

3  const conversationSchema = new mongoose.Schema({
4    participants: {
5      type: Array,
6      required: true,
7    }
8  }, {
9    timestamps: true,
10 })

```

Рис. 4.1.7 – Конструювання структур даних (conversationSchema)

4.2 Особливості конструювання програмних модулів

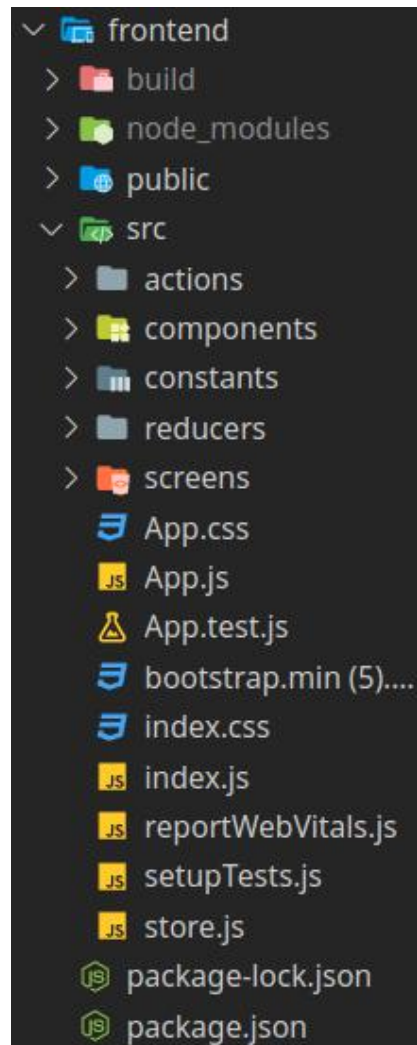


Рис. 4.2 – Конструювання програмних модулів

4.2.1 Конструювання програмної структури з урахуванням спеціалізованого Framework для FrontEnd-компонент архітектури (за наявності)

```
const getUserProfile = asyncHandler(async (req, res) => {
  const user = await User.findById(req.params.id);

  if (user) {
    res.json({
      _id: user._id,
      username: user.username,
      password: user.password,
      knownAs: user.knownAs,
      gender: user.gender,
      nativeLanguage: user.nativeLanguage,
      isLearning: user.isLearning,
      country: user.country,
      dateOfBirth: user.dateOfBirth,
      city: user.city,
      introduction: user.introduction,
      isAdmin: user.isAdmin,
      token: generateToken(user._id),
    })
  } else {
    res.status(404).json({ message: 'User not found' });
  }
})
```

Рис. 4.2.1.1 – Конструювання програмної структури (getUserProfile)

```
const getCurrentUserProfile = asyncHandler(async (req, res) => {
  const user = await User.findById(req.user._id);

  if (user) {
    res.json({
      _id: user._id,
      username: user.username,
      password: user.password,
      knownAs: user.knownAs,
      gender: user.gender,
      nativeLanguage: user.nativeLanguage,
      isLearning: user.isLearning,
      country: user.country,
      dateOfBirth: user.dateOfBirth,
      city: user.city,
      introduction: user.introduction,
      isAdmin: user.isAdmin,
      token: generateToken(user._id),
    });
  } else {
    res.status(404).json({ message: 'User not found' });
  }
})
```

Рис. 4.2.1.2 – Конструювання програмної структури (grtCurrentUserProfile)

```
const getConversationByUser = asyncHandler(async (req, res) => {
  const conversation = await Conversation.find({
    participants: { $in: [req.params.userId] }
  })

  res.json(conversation);
})
```

Рис. 4.2.1.3 – Конструювання програмної структури
(getConversationByUser)

```
const createConversation = asyncHandler(async (req, res) => {
  const newConversation = await Conversation.create({
    participants: [req.body.sender, req.body.reciever]
  })

  if (newConversation) {
    res.status(201).json({
      _id: newConversation._id,
      participants: newConversation.participants,
    })
  } else {
    res.status(400).json({ message: "Invalid conversation data" });
  }
})
```

Рис. 4.2.1.4 – Конструювання програмної структури (createConversation)

4.2.2 Особливості використання спеціалізованих програмних бібліотек та API (за наявності)

Axios – це бібліотека з відкритим вихідним кодом, що дозволяє робити запити HTTP. Вона надає методи `.get()`, `.post()` та `.delete()`.

React — JavaScript-бібліотека з відкритим вихідним кодом для розробки інтерфейсів користувача. (Використовується в реалізації сторони клієнта (frontend) ПП).

React-Bootstrap – це бібліотека React-компонентів, що підходять для повторного використання, яка реалізує можливості популярного шаблону Bootstrap від Twitter.

Redux — бібліотека керування станом для програм, написаних на JavaScript. Вона допомагає писати програми, які поведуться стабільно/передбачувано, працюють на різних оточеннях (клієнт/сервер/нативний код) та легко тестуються.

Socket.IO - це бібліотека, яка забезпечує двосторонній обмін даними між браузером та сервером у реальному часі та на основі подій.

timeago.js - це проста бібліотека, яка використовується для форматування datetime за допомогою оператора `*** time ago`. наприклад: «3 години тому» (використовується у реалізації діалогів, постів та коментарів у ПП).

Font Awesome – це набір інструментів для веб-сайтів, який містить іконки та логотипи соціальних мереж. Реалізація SVG з JavaScript для Font Awesome включає API, який можна використовувати з JavaScript для розширеного рендерингу значків та більшого контролю над деякими функціями.

bcryptjs - оптимізований bcrypt JavaScript з нульовими залежностями. bcrypt — адаптивна криптографічна хеш-функція формування ключа, що використовується для захищеного зберігання паролів (використовується для хешування паролів та ін.).

Dotenv - це модуль, у якого немає залежностей. Він призначений для завантаження змінних оточення з файлу `.env` в `process.env`. Використання цього механізму дозволяє зберігати установки програм окремо від їх коду.

Express — це швидкий та мінімалістичний веб-фреймворк для Node.js, який дає розробнику досить великий рівень свободи у створенні веб-серверів.

JSON Web Token (JWT) - це компактний, безпечний для URL-адрес засіб представлення вимог, що передаються між двома сторонами.

Mongoose представляє спеціальну ODM-бібліотеку (Object Data Modelling) для роботи з MongoDB, яка дозволяє зіставляти об'єкти класів та документи колекцій із бази даних.

Multer — це програмне забезпечення проміжного шару для Express, призначене для обробки даних типу `multipart/form-data`. Такі дані в основному надходять на сервер при розвантаженні файлів.

nodemon – це інструмент, який допомагає розробляти програми на основі node.js шляхом автоматичного перезапуску програми node при виявленні змін файлів у каталозі.

Модуль path надає утиліти для роботи з шляхами до файлів та каталогів.

4.3 Модульне тестування програмних модулів

Для реалізації модульного тестування був використаний фреймворк Mocha та бібліотека Chai.

Mocha — це багатофункціональна платформа для тестування JavaScript, що працює на Node.js і в браузері, що робить асинхронне тестування простим і цікавим. Mocha-тести виконуються послідовно, забезпечуючи гнучкі та точні звіти, а також відображаючи неперехоплені винятки у правильні тестові випадки.

Chai — це бібліотека підтверджень для node та браузера, яку можна чудово поєднати з будь-якою платформою тестування JavaScript.

Тестування API додавання міст.

Тест пройдено, якщо після відправлення коректних даних був отриманий об'єкт, який має такі властивості, як: cityName.

```
25     describe('/POST city', () => {
26         it('it should post a new city', (done) => {
27             chai.request('http://127.0.0.1:5000')
28                 .post('/api/countries')
29                 .send(country)
30             let city = new City({
31                 cityName: "Seoul",
32                 countryId: country.id
33             });
34             chai.request('http://127.0.0.1:5000')
35                 .post('/api/cities/' + country.id)
36                 .send(city)
37                 .end((err, res) => {
38                     res.status.should.be.equal(201);
39                     expect(res.body).to.be.an('object')
40                     expect(res.body).to.have.a.property('cityName');
41                     done();
42                 });
43         });
44     });
```

Рис. 4.3.1 – Тестування програмного модулю (city)



```
City API
/POST city
✓ it should post a new city (139ms)
```

Рис. 4.3.2 – Результат тестування програмного модулю (city)

Тест успішно пройдено.

Тестування API отримання міст.

Тест пройдено, якщо отримано не пустий масив.

```
describe('/GET cities', () => {
  it('it should GET all the cities for a country', (done) => {
    chai.request('http://127.0.0.1:5000')
      .get('/api/cities/' + country.id)
      .end((err, res) => {
        res.status.should.be.equal(200);
        expect(res.body).to.be.an('array').that.is.empty;
        done();
      });
  });
});
```

Рис. 4.3.3 – Тестування програмного модулю (cities)



```
/GET cities
✓ it should GET all the cities for a country (89ms)
```

Рис. 4.3.4 – Результат тестування програмного модулю (cities)

Тест успішно пройдено.

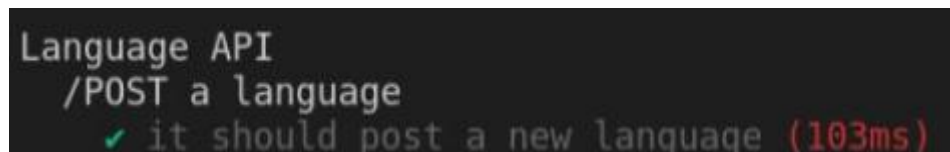
Тестування API додавання мов.

Тест пройдено, якщо після відправлення коректних даних був отриманий об'єкт, який має такі властивості, як: `languageName`.

```
20     describe('/POST a language', () => {
21         it('it should post a new language', (done) => {
22             let language = {
23                 languageName: "Korean",
24                 flagUrl: "27718288"
25             }
26             chai.request('http://127.0.0.1:5000')
27                 .post('/api/languages/')
28                 .send(language)
29                 .end((err, res) => {
30                     res.status.should.be.equal(201);
31                     expect(res.body).to.be.an('object')
32                     expect(res.body).to.have.a.property('languageName');
33                     expect(res.body).to.have.a.property('flagUrl');
34                     done();
35                 });
36         });
37     });
```

Рис.

Рис. 4.3.5 – Тестування програмного модулю (language)



```
Language API
/POST a language
✓ it should post a new language (103ms)
```

Рис. 4.3.6 – Результат тестування програмного модулю (language)

Тест успішно пройдено.

Тестування API отримання мов.

Тест пройдено, якщо отримано не пустий масив.

```
describe('/GET languages', () => {
  it('it should GET all the languages', (done) => {
    chai.request('http://127.0.0.1:5000')
      .get('/api/languages')
      .end((err, res) => {
        res.status.should.be.equal(200);
        expect(res.body).to.be.an('array').that.is.empty;
        done();
      });
  });
});
```

Рис. 4.3.7 – Тестування програмного модулю (languages)

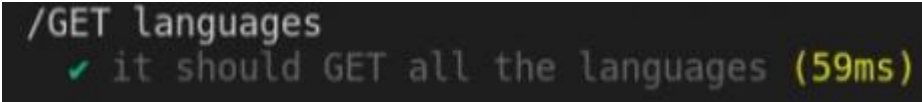


Рис. 4.3.8 – Результат тестування програмного модулю (languages)

Тест успішно пройдено.

5 Верифікація програмного продукту



5.1 Тестування апаратно-програмних інтерфейсів програмного продукту

Табл. 5.1 – Тестування апаратно-програмних інтерфейсів ПП

Тип умови (Hard/Soft)	Опис вимог	Опис реальних умов
Hard	CPU: frequency ≥ 2 Ghz RAM: size ≥ 2 Gb	CPU: AMD A4-4000 APU with Radeon HD Graphics, 3000Mhz (3.00 GHz), 1Core RAM: 2.00 GB
Soft	OS: Windows 7, 8, 10, Linux	Windows 7 64bit
Soft	Web browser: Google Chrome, Microsoft Edge, Mozilla Firefox, Opera	Web browser: Opera 60

5.2 Тестування інтерфейсу користувача програмного продукту

Табл. 5.2 – Тестування інтерфейсу користувача програмного продукту

NFR ID	Test Case ID	Опис дій з ПП на рівні інтерфейсу користувача	Результат тестування (Passed / Failed / Blocked)
NFR4	TC12	4. Для елемента “username” ввести значення jes 5. Для елемента “password” ввести значення 123456 6. Натиснути на кнопку відправлення форми	Passed
Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача NFR4, TC12			
NFR4	TC13	5. Для елемента “username” ввести значення andrew123 6. Для елемента “password” ввести значення 123456 7. Для елемента «confirm password» ввести значення 1234567890 8. Натиснути на кнопку відправлення форми	Passed
Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача NFR4, TC13			

Продовження таблиці 5.2

NFR4	TC3	1. Для елементу “username” ввести значення andrew123 2. Для елементу “password” ввести значення 123456 3. Для елементу «confirm password» ввести значення 123456 4. Для елементу «native language» ввести значення “spanish” 5. Для елементу «date of birth» ввести значення 07.16.2014	Passed
------	-----	---	--------

5.3 Тестування часу реакції програмного продукту на дії користувача

Табл. 5.3 – Тестування часу реакції програмного продукту на дії користувача

FR (назва)	Максимальний час реакції ПП на дії користувачів, секунди	Кількість проведених тестів	Результат тестування, секунди	Відносна похибка вимірювань, %
Створення запиту у користувача на отримання його параметрів ідентифікації та аутентифікації	3	2	103 мс	1
Передача від користувача його параметрів ідентифікації та аутентифікації	3	2	336 мс	9.2

Продовження табл. 5.3

Передача від користувача профілю, який він захотів переглянути	3	2	68 мс	1.5
Передача від користувача запиту на редагування свого профілю	3	2	95 мс	3.2
Передача від користувача параметрів, які було змінено	3	2	200 мс	1
Передача від користувача даних, які він передав з метою публікації статті	3	2	9 мс	0
Передача від користувача профілю, якому він хоче надіслати повідомлення	3	2	60 мс	1.4
Передача від користувача повідомлення	3	2	99 мс	4

Продовження табл. 5.3

Передача від гостя параметрів	5	2	95 мс	5.3
Передача від користувача запросу на видалення акаунту	2	2	132 мс	0
Передача від користувача параметрів	5	2	92 мс	1.1

6 Розгортання та валідація програмного продукту

6.1 Інструкція з встановлення системного програмного забезпечення

- 1) Встановити Node.js.
- 2) Встановити bash (якщо Windows).
- 3) Встановити IDE, в якому можна працювати із JS.
- 4) Клонувати код з github, вказаний в додатку А.
- 5) Перейти в папку Web в проєкті, встановити пакети node для серверу

командою:

```
npm install
```

- 6) Перейти в папку Web/frontend, встановити пакети node для клієнту

командою:

```
npm install
```

- 7) В папці з попереднього пункту створити оптимізовану збірку клієнтської частини.

```
npm run build
```

- 8) На даному етапі проєкт встановлено на локальну машину розробника та створено оптимізовану збірку клієнту, далі, для розгортання серверу, наприклад на Heroku:

- 9) Встановити на локальну машину Heroku CLI.

- 10) Зареєструватися в Heroku.

- 11) Увійти в свій акаунт Heroku командою

```
heroku login
```

- 12) За допомогою статті в додатку А розгорнути ПП на Heroku.

6.2 Інструкція з використання програмного продукту

1. Авторизація користувача

Для того, щоб авторизуватися в ПЗ, необхідно:

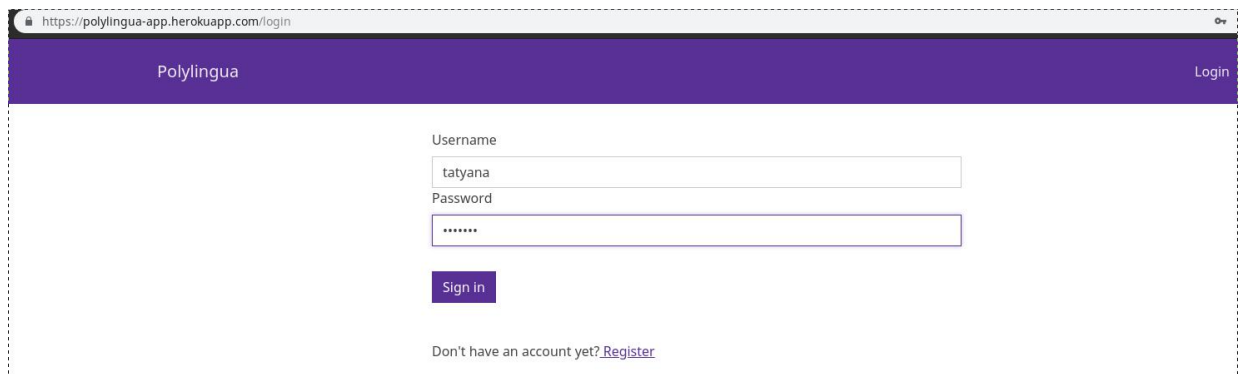
- 1) Зайти на головну сторінку ПЗ: <https://polylingua-app.herokuapp.com>.

2) Необхідно мати аккаунт в ПЗ, якщо аккаунту немає – перейти по посиланню до форми реєстрації, за необхідністю перейти до інструкції до прецеденту 6 (Реєстрація користувача).

Don't have an account yet? [Register](#)

Рис. 6.2.1 Посилання до форми реєстрації

3) Якщо аккаунт вже існує, треба ввести значення коректні (існуючі в БД ПЗ) Username, Password.



https://polylingua-app.herokuapp.com/login

Polylingua Login

Username
tatyana

Password

Sign in

Don't have an account yet? [Register](#)

Рис. 6.2.2 Форма авторизації користувача

4) Після введення даних, натиснути кнопку Sign in, введені дані буде перевірено на коректність.

4.1) Якщо дані коректні, користувач побачить головну сторінку ПЗ із листом користувачів.

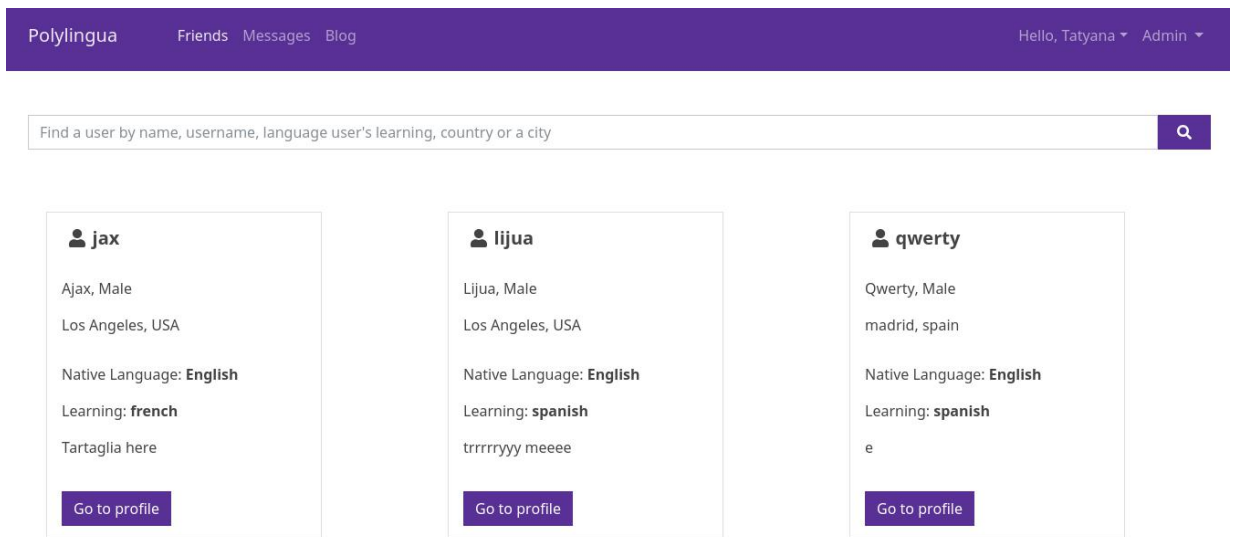


Рис. 6.2.3 Головна сторінка ПЗ із листом користувачів

4.2) Якщо дані некоретні, програма повідомить про це, тоді користувач повинен перейти до пункту 3 інструкції до прецеденту.



Рис. 6.2.4 Помилка перевірки на коректність даних при авторизації
(неправильний Username)

Invalid password

Username

tatyana

Password

.....

Sign in

Рис. 6.2.5 Помилка перевірки на коректність даних при авторизації
(неправильний Password)

2. Перегляд профілів

1.1) Для перегляду профілю треба обрати один з аккаунтів на головній сторінці:

1.1.2) Натиснути на кнопку “Go to profile”

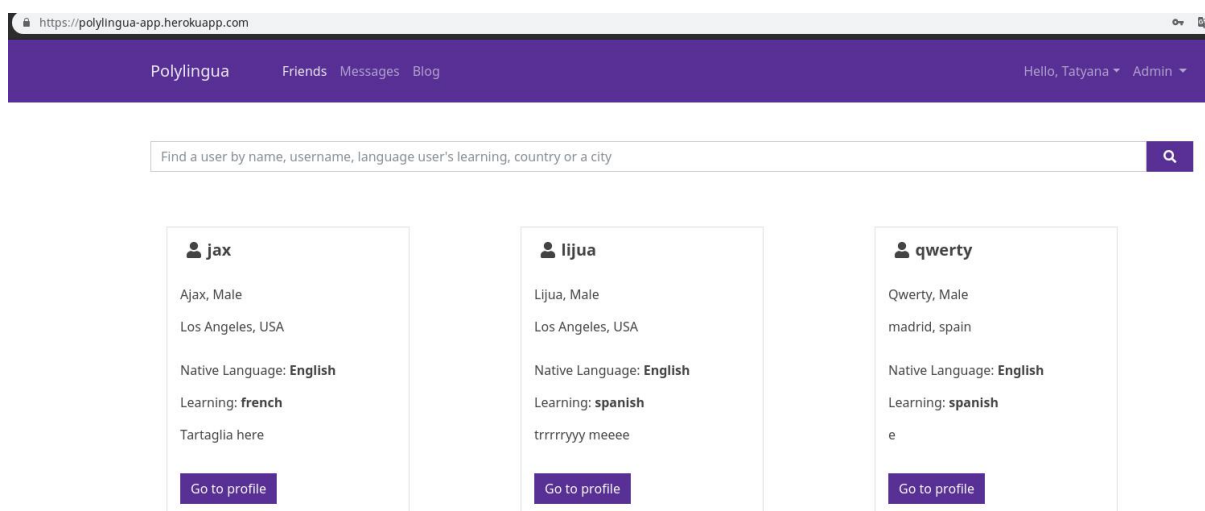


Рис. 6.2.6 Головна сторінка ПЗ із листом користувачів

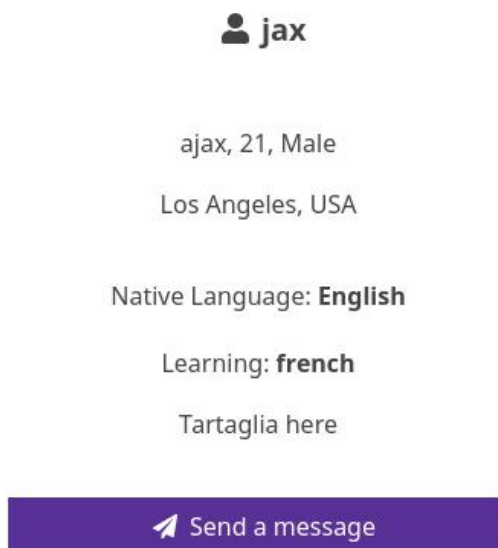


Рис. 6.2.7 Профіль іншого користувача

1.2) Або обрати свій профіль

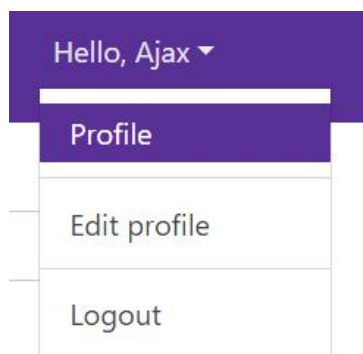


Рис. 6.2.8 Контекстне меню для вибору особистого профілю користувача



Рис. 6.2.9 Особистий профіль користувача

3. Редагування профіля

1) Для того, щоб відредагувати свій профіль, потрібно обрати в шапці сайту пункт “Edit profile”

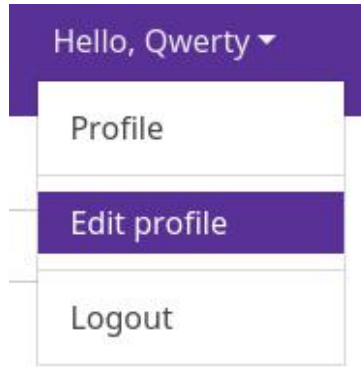


Рис. 6.2.10 Контекстне меню для вибору редагування особистого профілю користувача

2) На сторінці редагування профілю заповнити ті поля форми, які потрібно змінити, після чого натиснути “Update profile”

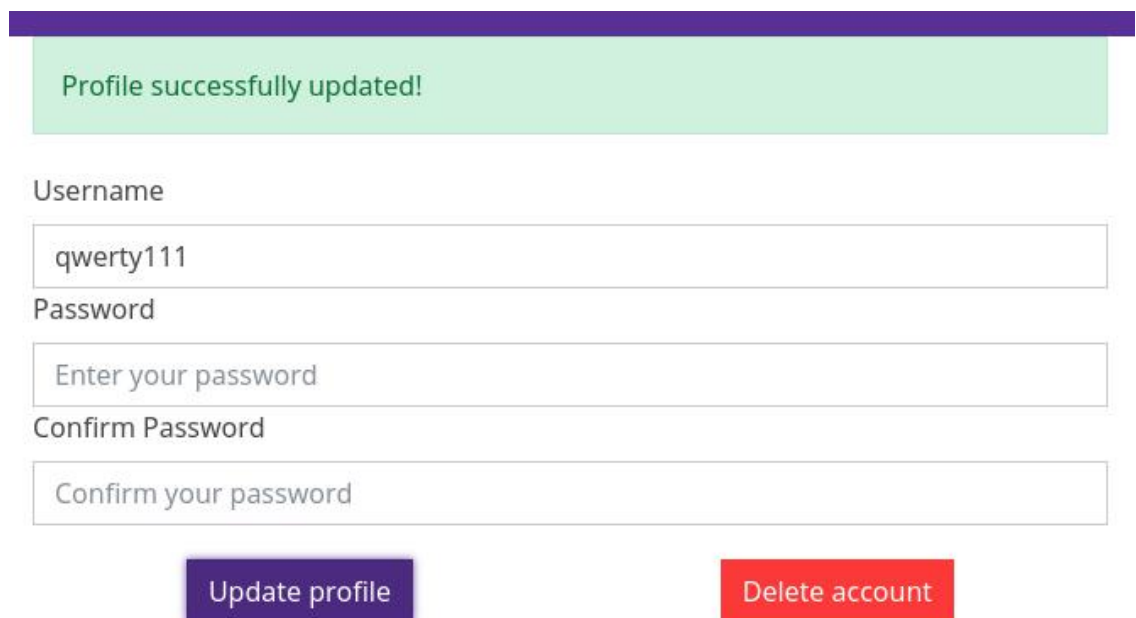
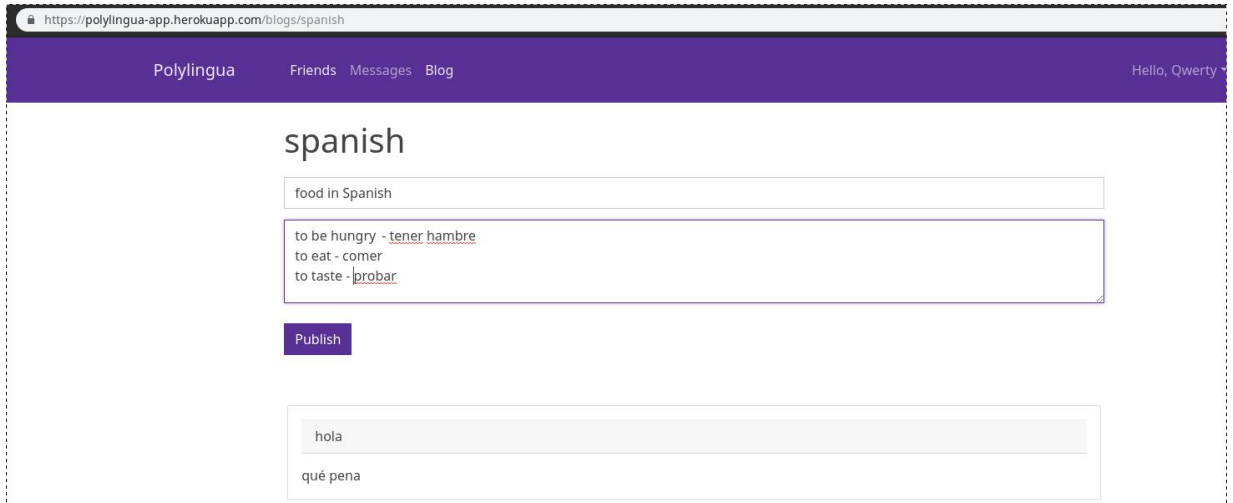
A screenshot of a web form for updating a user profile. At the top, a green banner displays the message 'Profile successfully updated!'. Below this, the form contains four input fields: 'Username' (containing 'qwerty111'), 'Password' (with placeholder text 'Enter your password'), 'Confirm Password' (with placeholder text 'Confirm your password'), and two buttons at the bottom: a dark blue 'Update profile' button and a red 'Delete account' button.

Рис. 6.2.11 Форма та успішне редагування особистого профілю користувача

4. Ведення блогу

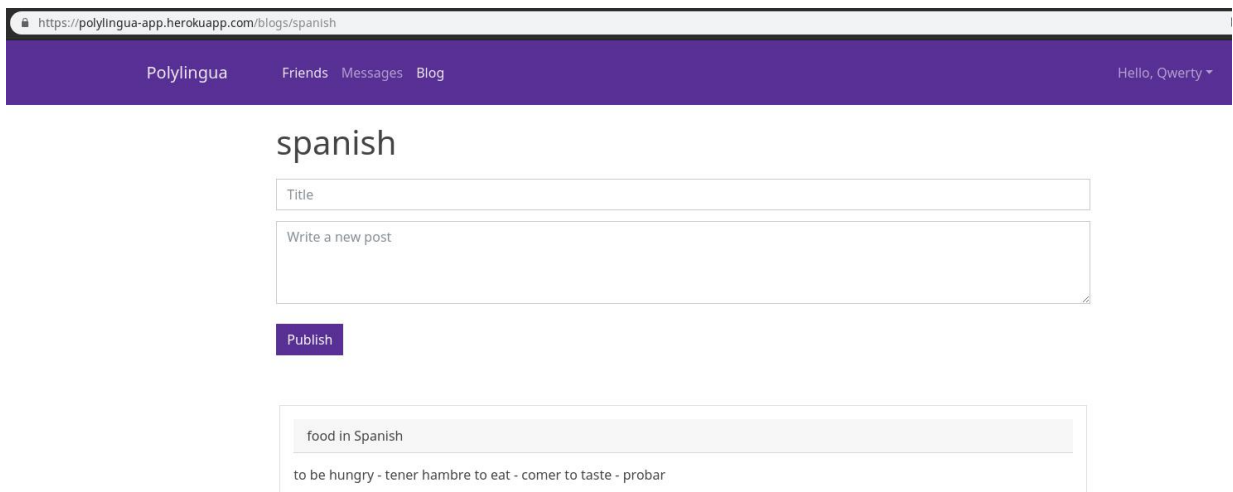
1) На сторінці блогу треба заповнити форму, після чого натиснути кнопку “Publish”



The screenshot shows a web browser window with the URL `https://polylingua-app.herokuapp.com/blogs/spanish`. The page has a purple header with the text "Polylingua" and navigation links "Friends", "Messages", and "Blog". On the right of the header, it says "Hello, Qwerty". The main content area is titled "spanish". It contains a text input field with the placeholder "food in Spanish". Below it is a larger text area containing the text: "to be hungry - tener hambre", "to eat - comer", and "to taste - probar". There is a purple "Publish" button below the text area. At the bottom, there is a preview section showing a card with the text "hola" and "qué pena".

Рис. 6.2.12 Сторінка з формою для оформлення нової публікації (поста) у блозі

2) Після натискання на кнопку сторінка оновиться та новий пост з'явиться знизу.



The screenshot shows the same web browser window as before, but the form has been updated. The "Title" field now contains "food in Spanish". The "Write a new post" text area now contains the text: "to be hungry - tener hambre", "to eat - comer", and "to taste - probar". The purple "Publish" button is still present. The preview section at the bottom now shows a card with the text "hola" and "qué pena".

Рис. 6.2.13 Сторінка блогу з успішним опублікуванням нової публікації (поста)

5. Повідомлення

1) Для того, щоб написати іншому користувачеві повідомлення, необхідно обрати користувача на сторінці користувачів, зайти на його сторінку та натиснути кнопку “Send a message”.

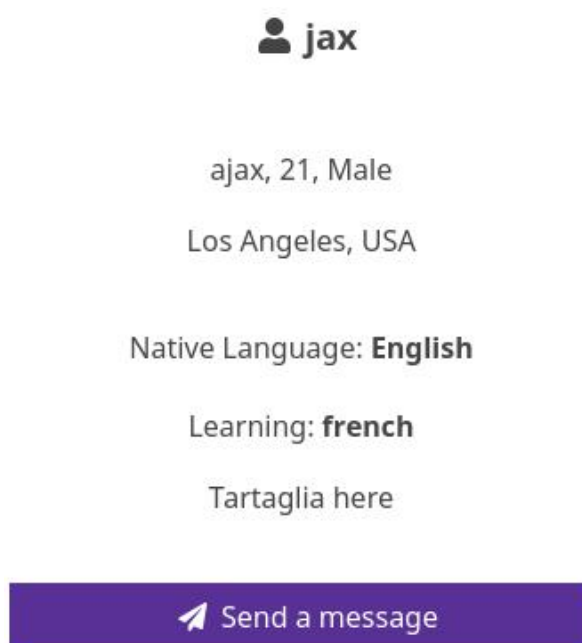


Рис. 6.2.14 Профіль іншого користувача

2) Після натискання на кнопку Вас буде перенаправлено на сторінку із повідомленнями

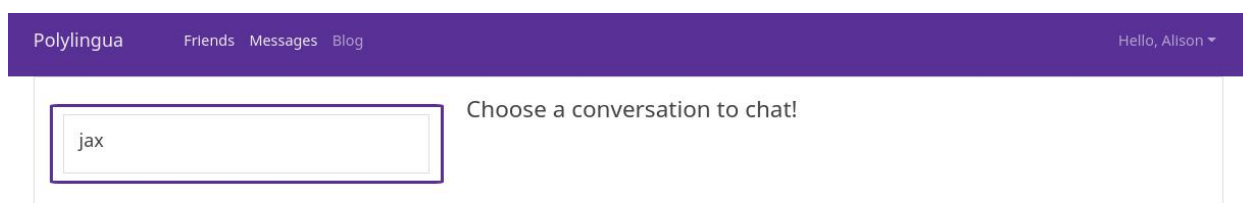


Рис. 6.2.15 Сторінка з повідомлення з іншими користувачами ПЗ

3) Натиснути на чат та написати повідомлення

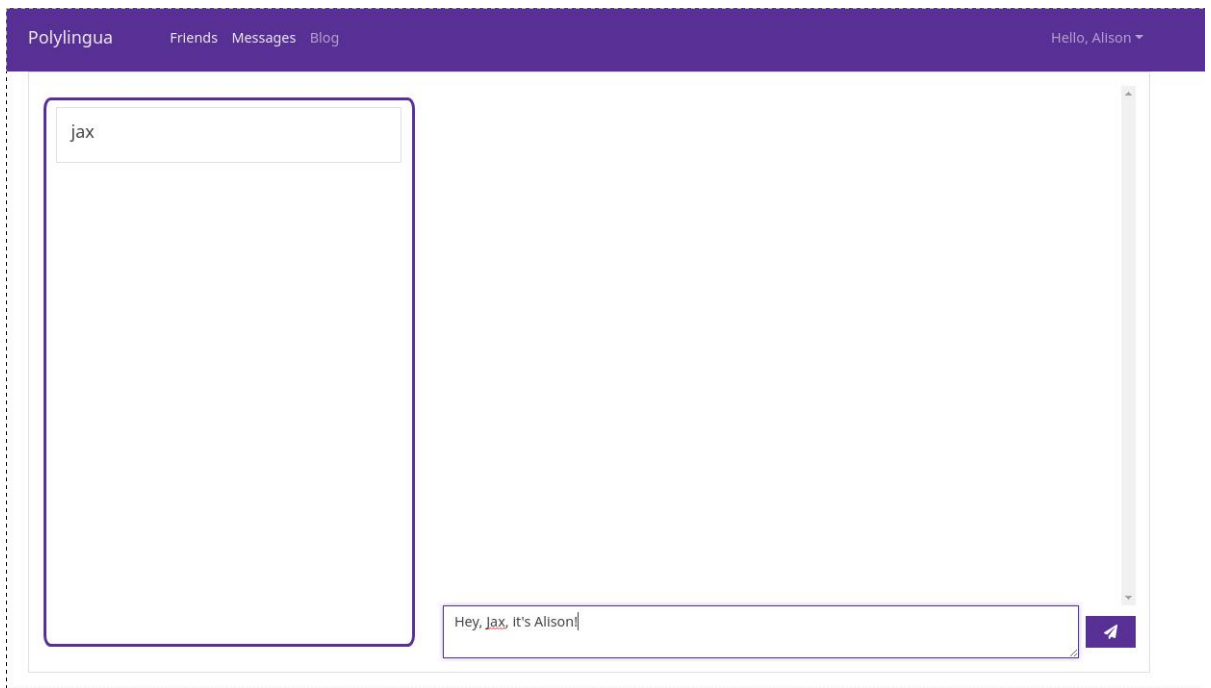


Рис. 6.2.16 Сторінка з заповненою формою для відправки повідомлення

4) Натиснути на кнопку із паперовим літаком

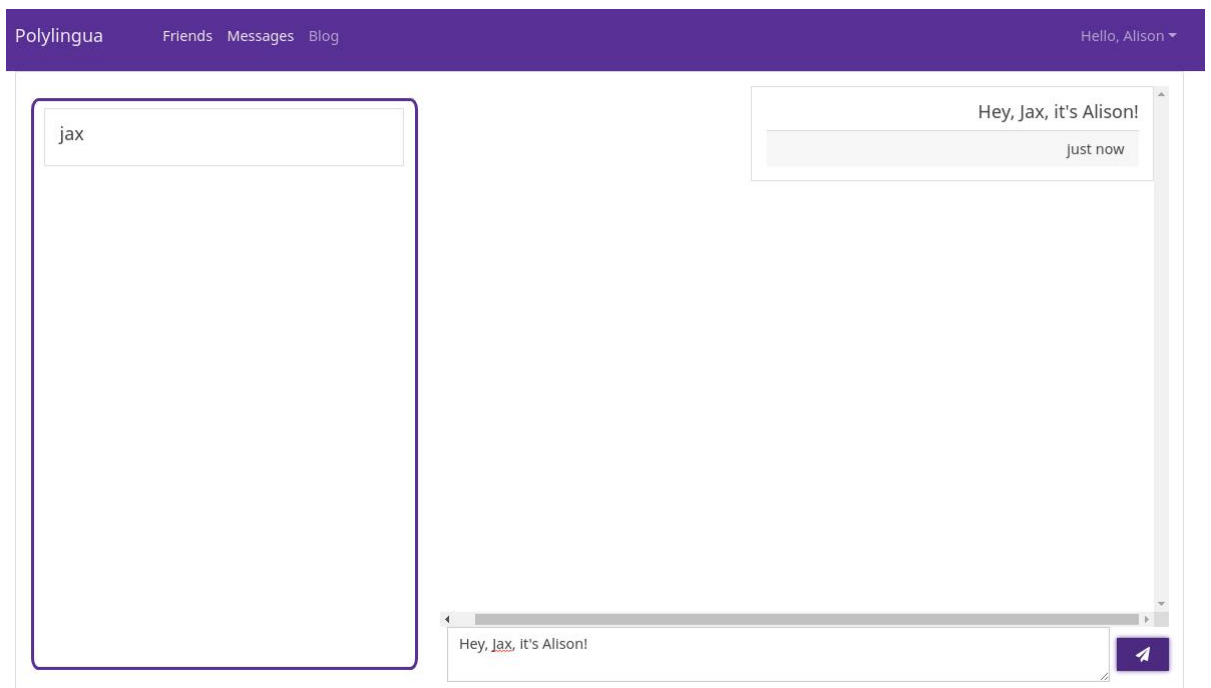
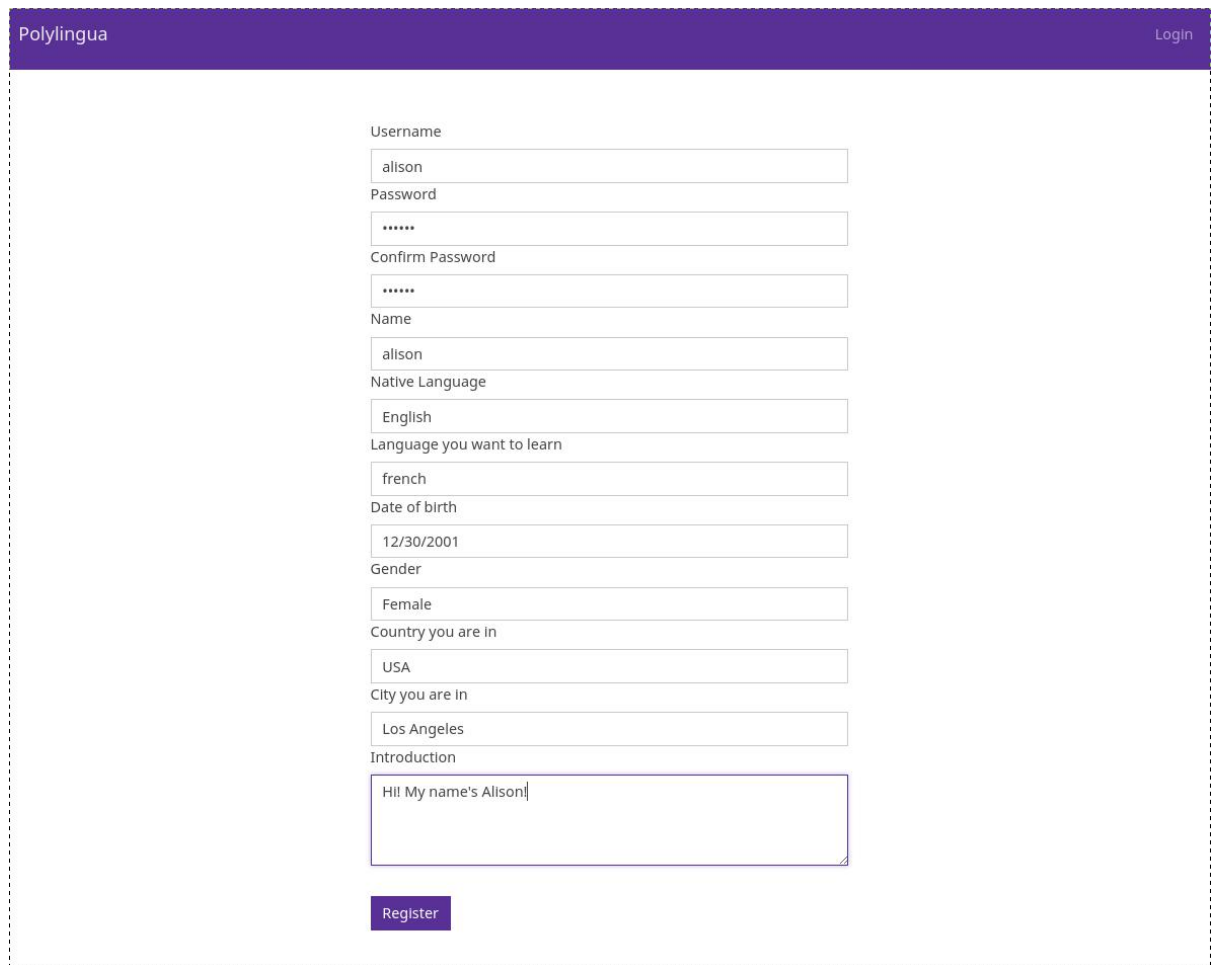


Рис. 6.2.17 Сторінка з успішним відправленням повідомлення іншому користувачу ПЗ

6. Реєстрація

1) Для реєстрації необхідно ввести коректні значення в поля форми (Username, Password, Confirm Password (повинний бути однаковим із Password), Name, Native Language, Language you want to learn (не може бути однаковим із Native Language), Date of birth (користувач повинен бути старше 13 років), Gender, Country you are in, City you are in, Introduction).



The screenshot shows a web form titled "Polylingua" with a "Login" link in the top right corner. The form contains the following fields:

- Username:
- Password:
- Confirm Password:
- Name:
- Native Language:
- Language you want to learn:
- Date of birth:
- Gender:
- Country you are in:
- City you are in:
- Introduction:

At the bottom of the form is a purple "Register" button.

Рис. 6.2.18 Сторінка із формою реєстрації нового користувача в ПЗ

2) Після натискання на кнопку “Register” користувач буде успішно зареєстрований та буде здійснен вхід в аккаунт, або зверху форми будуть написані повідомлення щодо некоректно введених даних, тоді повторити дії з пункту 1 інструкції до прецеденту.

7. Видалення аккаунту

1) Для видалення свого аккаунту користувач повинен перейти на сторінку редагування профілю та натиснути красну кнопку “Delete account”

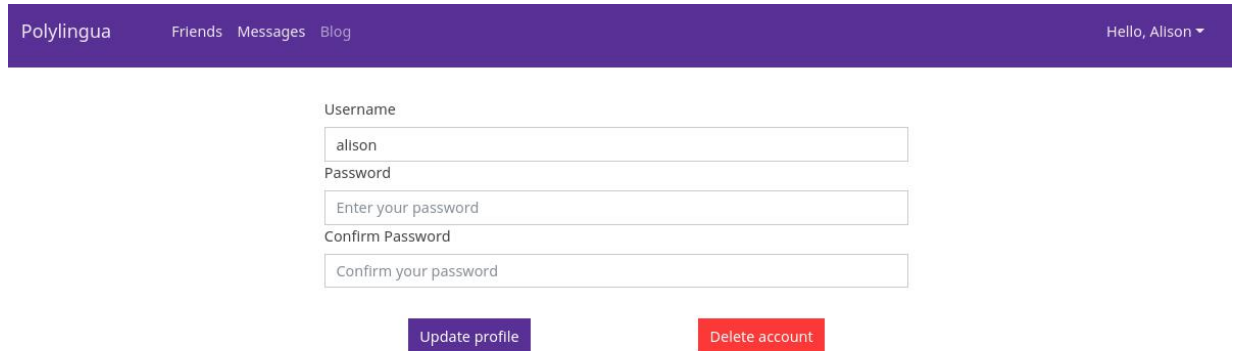


Рис. 6.2.19 Форма редагування особистого профілю користувача

2) В віконці, що з'явилося натиснути “Ok”

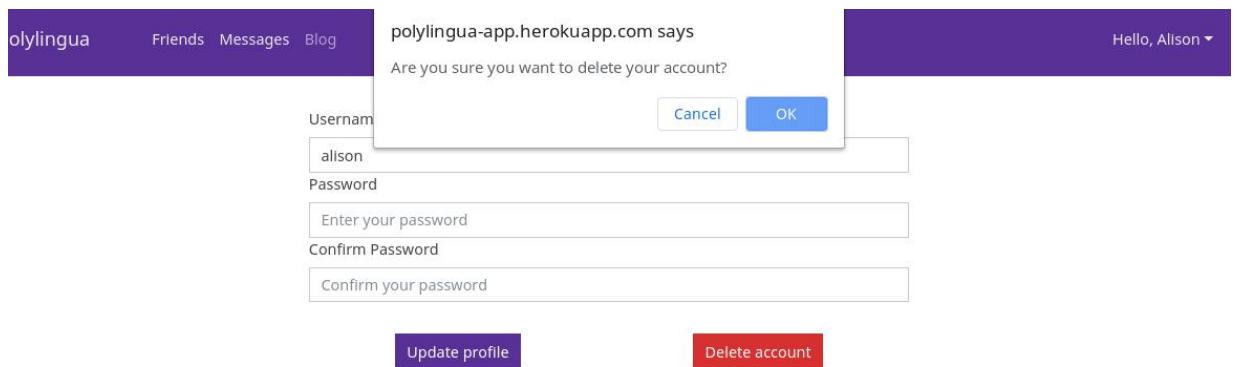


Рис. 6.2.20 Вікно з повідомлення про підтвердження видалення аккаунту користувача

8. Вихід з аккаунту

1) Натиснути на надпис “Hello, username” та натиснути кнопку “Logout”, після чого Ви повернетеся на вкладку із авторизацією.

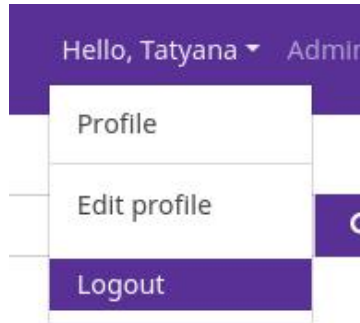


Рис. 6.2.21 Контекстне меню для вибору виходу з аккаунту

9. Адміністрування

1) Якщо ви адміністратор ПЗ – ви можете видаляти користувачів, для цього потрібно во вкладці “Admin” обрати “Users”, після чого обрати користувача, якого хочете видалити та натиснути кнопку ”Delete account”, в віконці, що з’явилось натиснути ”Ok”.

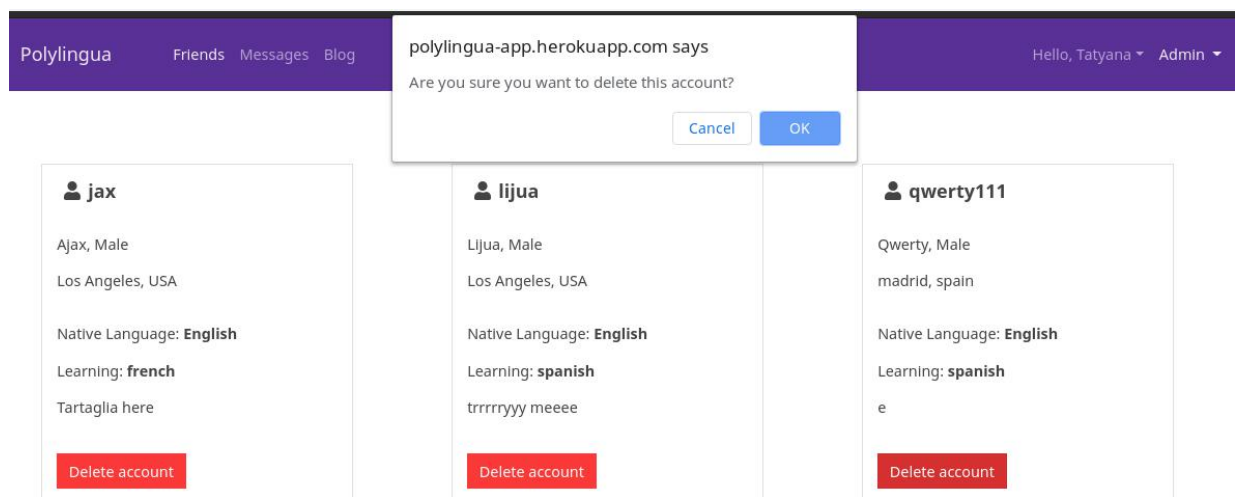


Рис. 6.2.22 Сторінка профілів користувачів ПЗ та вікно з повідомленням про підтвердження видалення аккаунту користувача зі сторони адміністратора

6.3 Результати валідації програмного продукту

Метою створення програмного продукту Polylingua є збільшення доступності ефективного та легкого отримання знань в області іноземних мов шляхом обміну знань із іншими користувачами продукту.

За допомогою даного програмного продукту користувачі будуть мати змогу розмовляти один із одним, тим самим отримують корисні навички щодо вивчення іноземних мов, ділитися своїми знаннями із іншими людьми.

Точну валідацію виконання мети даного ПП уявити складно, тому що користувач сам визначає свій напрямок у вивченні мов (за допомогою даного ПП) та вибирає зручний для себе спосіб.

Але завдяки раніше проведеним опитуванням потенційних користувачів ПП (17 відповідей), можна розрахувати рівень доступності AL (AL – Access Level) можна визначити як $AL = NA / N$,

Де NA – кількість людей, які мають достатньо доступності практикувати іноземну мову;

N – загальна кількість людей, які навчатимуть іноземну мову

$$AL = 10/17 = 0,59$$

Відгуки користувачів:

1) Я вивчала англійську мову в школі з першого класу, але вона ніколи не була моїм улюбленим предметом, тому що мені завжди було важко вивчати всі ці правила та слова без контексту. Також я зовсім себе не вважаю гуманітарієм, тому не дуже бачу сенсу багато часу приділяти навчанню англійської. Коли мені розповіли про веб-сайт «Polylingua» я не була у захваті, бо мені не дуже хотілося тестувати нічого, але я погодилася із ввічливості та протестувала цей сайт.

Наскільки я зрозуміла, цей сайт — це соціальна мережа, але через те, що в сайті немає постійних користувачів він не має сенсу, бо ви не можете розмовляти з користувачами — їх немає. Але це не є проблемою веб-сайта, усі з чогось починали.

На мою думку, якщо в сайту буде багато постійних користувачів він буде дуже корисним для вивчення мов.

2) Я навчаюся на лінгвістиці в інституті і мені дуже не вистачає живого спілкування з носіями, але мені моя подруга з іншого міста порадила сайт Polylingua, я зареєструвалася там і мені дуже сподобалося, правда шкода, що мало активних користувачів.

Я сподіваюся, що ця соціальна мережа розвиватиметься і набере свою аудиторію!

З отриманих відгуків можна побачити, що програмний продукт сподобався користувачам, але вони не можуть активно їм користуватися через відсутність спільноти в ПП. Але не дивлячись на це, можна зробити висновок, що продукт пройшов валідацію.

Висновки

В результаті виконання етапів курсової роботи було створено програмне забезпечення процесу «вивчення іноземної мови».

Приклад використання програмного забезпечення у відповідності з інструкцією користувача представлено у відеозапису, доступним за посиланням:

<https://drive.google.com/file/d/1fE4UpviJKk0iymtf5l1QssT51NGNLv3Y/view?usp=sharing>

Створене програмне забезпечення досягло наступної мети його споживача: «збільшення доступності ефективного та легкого отримання знань в області іноземних мов».

Доказом цього є наступні факти:

1) Програмний продукт забезпечує виконання прецедентів, які були визначені, як ті, які впливають на досягнення мети.

2) Програмний продукт надає користувачам можливість вивчати іноземні мови шляхом спілкування та ведення блогів.

Вказані факти перетворили програмне забезпечення на програмний продукт.

В процесі створення програмного продукту виникли такі труднощі (організаційні, проблеми відсутності досвіду, знань, потрібних в різних етапах):

1) Організаційні — в ході створення програмного продукту було важко скоординувати дії учасників, тому що було важко писати програмний код не заважаючи один одному.

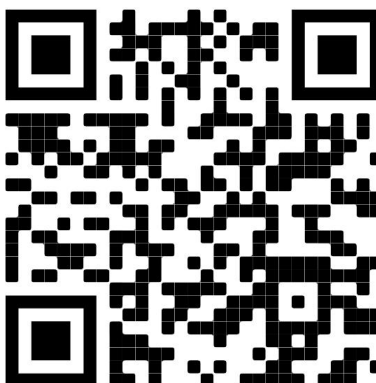
2) Проблеми відсутності знань — було витрачено багато часу на вирішення проблем, які виникали в ході написання програмного коду.

В ході виконання курсової роботи були реалізовані всі прецеденти.

Посилання на ПП: <https://polylingua-app.herokuapp.com>



Посилання на відкритий код ПП: <https://github.com/Ayashma/Polylingua>



Посилання на статтю з інструкцією розгортання серверу:
<https://devcenter.heroku.com/articles/git>