

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ ФОРМУВАННЯ
ІНДИВІДУАЛЬНОЇ ТРАЄКТОРІЇ НАВЧАННЯ СТУДЕНТІВ УЧАСНИКІВ
МІЖНАРОДНОЇ МОБІЛЬНОСТІ

WEB APPLICATION DEVELOPMENT FOR DESIGNING AN INDIVIDUAL
LEARNING TRAJECTORY FOR THE STUDENTS PARTICIPATING IN
INTERNATIONAL MOBILITY PROGRAMMES

Виконав: студент(ка) 4 курсу, групи КНТ-117
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Джрагацпанян Н.В.

(прізвище та ініціали)

Керівник Табунщик Г.В.

(прізвище та ініціали)

Рецензент Шитікова Л.В.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
 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.
С.О. Субботін
 “ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТКИ

Джрагацпанян Нелі Варданівни
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення вебзастосунку для формування
індивідуальної траєкторії навчання студентів учасників міжнародної мобільності.
Web Application Development for Designing an Individual Learning Trajectory for the
Students Participating in International Mobility Programmes

керівник проєкту (роботи) Табунщик Галина Володимирівна, професор,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 5 травня 2020 року

3. Вихідні дані до проєкту (роботи) технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення з реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Табунщик Г.В., професор		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 26 ” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області, розробка та затвердження технічного завдання	1 тиждень	Технічне завдання
2	Проектування структури системи	2 тиждень	Структурна схема
3	Розробка схеми функціонування системи та модулів програми	3 тиждень	Функціональна схема. Модулі
4	Створення програмного коду системи	4-5 тиждень	Текст програми
5	Тестування та відналагодження програми	6 тиждень	Працездатна система
6	Розробка розділів пояснювальної записки	7 тиждень	Пояснювальна записка
7	Розробка графічного матеріалу та захист дипломної роботи	8 тиждень	Графічний матеріал та дипломна робота

Студент(ка)

_____ Джрагацпанян Н.В.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Табунщик Г.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
113с., 4 табл., 35 рис., 2 дод., 5 джерел.

ВЕБЗАСТОСУНОК, МІЖНАРОДНА МОБІЛЬНІСТЬ, JAVASCRIPT,
VISUAL STUDIO CODE, VUE.

Об'єкт дослідження – процес організації міжнародної мобільності.

Предмет дослідження – програмні засоби реалізації вебзастосунку для міжнародної мобільності.

Мета роботи – створення вебзастосунку для формування індивідуальної траєкторії навчання студентів учасників міжнародної мобільності.

Матеріали, методи та технічні засоби: вебпрограмування, мова програмування JavaScript, фреймворк Vue.js, персональний комп'ютер з процесором Intel Core 5 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено вебзастосунок з адміністративною панеллю для подачі та обробки заявок на участь в академічній мобільності.

Висновки. Розроблено програмну систему для студентів, які хочуть взяти участь в академічній мобільності, яка автоматизує та прискорює процес подачі та обробки заявок на участь в академічній мобільності.

Галузь використання – навчальні заклади, які пропонують академічну мобільність.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 113 pages, 35 figures, 2 tables, 2 appendixes, 5 sources.

WEB APPLICATION, INTERNATIONAL MOBILITY, JAVASCRIPT, VISUAL STUDIO CODE, VUE.

Object of study: the process of organizing international mobility.

Subject of study: software tools for implementing a web application for international mobility.

The purpose of the work is to create a web application for designing an individual learning trajectory for the students participating in international mobility programmes.

Materials, methods, and tools: web programming, JavaScript programming language, Vue.js framework, personal computer with Intel Core 5 processor running Microsoft Windows 10 operating system.

Results. A web application with an administrative panel for submitting and processing applications for participation in academic mobility has been created.

Conclusions. A software system for students who want to take part in academic mobility, which automates and speeds up the process of submitting and processing applications for participation in academic mobility, has been developed.

The field of use: educational institutions that offer students academic mobility.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Аналіз предметної області.....	11
1.2 Огляд існуючих методів вирішення завдання	13
1.3 Технічне завдання	19
1.4 Висновок до розділу 1.....	20
2 Вибір та обґрунтування структури системи, яка проектується	21
2.1 Вибір інструментарію розробки програмного забезпечення.....	21
2.2 Структурна схема програми.....	27
2.3 Висновок до розділу 2.....	28
3 Основні рішення з реалізації компонентів системи	29
3.1 Основні розроблені алгоритми	29
3.2 Основні рішення щодо модульного уявлення системи.....	34
3.3 Організація даних.....	36
3.4 Користувачський інтерфейс GUI	39
3.5 Захищення даних	39
3.6 Адаптивність до змін та оновлень додатку	39
3.7 Висновок до розділу 3.....	39
4 Керівництво програміста.....	41
4.1 Призначення й умови застосування програми.....	41
4.2 Характеристики програми.....	41
4.3 Звернення до програми.....	42
4.4 Виняткові ситуації.....	42
4.5 Висновок до розділу 4.....	46
5 Керівництво оператора	47
5.1 Призначення та умови застосування програми	47

	7
5.2 Умови виконання програми	47
5.3 Як запустити програму	48
5.4 Виконання програми	48
5.5 Висновок до розділу 5.....	54
Висновки	55
Перелік джерел посилання	57
Додаток А Текст програми.....	58
Додаток Б Слайди презентації	98

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API – Application Programming Interface;
IT – Information Technology;
HTTP – HyperText Transfer Protocol;
ПЗ – програмне забезпечення;
ПП – програмний продукт.

ВСТУП

Впровадження комп'ютерів в соціальне життя значно спрощує роботу людини. Нові інформаційні технології (ІТ) впливають на створення середовища телекомунікаційної та комп'ютерної підтримки організації та управління в різноманітних видах діяльності, в тому числі і в освіті. Інтеграція ІТ в програми освіти здійснюється на всіх рівнях: шкільному, вузівському і післявузівську навчанні. Правильна автоматизація процесів освіти значно полегшує та оптимізує використання освітніх систем. ІТ в освіті зараз є важливою умовою переходу суспільства до інформаційної цивілізації. У наш час інформаційні технології дозволяють удосконалити навчально-виховний процес, повністю занурюють учня в інформаційно-освітнє середовище, підвищують якість освіти, мотивують процеси сприйняття інформації і отримання знань.

Основними напрямками застосування ІТ в навчальному процесі є:

- надання педагогічних програмних засобів різного призначення;
- веб-сайти навчального призначення;
- надання дидактичних та методичних матеріалів;
- здійснення управління навчальними ботами;
- комп'ютерні експерименти з віртуальними моделями;
- цілеспрямований пошук інформації різних типів в мережі, збір інформації, накопичення, зберігання, обробка та передача;
- інтелектуальне дозвілля студентів [1].

Необхідно розробити систему для організації мобільності студентів та надати можливість роботи з базою даних, можливість аналізу варіантів мобільностей, проведення процесу подачі заявки на участь та обробки заявок адміністратором.

Дана задача буде вирішуватися шляхом аналізу предметної області та технологій, які будуть використані для розробки застосунку. Завдяки цьому у студентів буде можливість користуватись сучасною системою для участі у програмі обміну. Отже, передбачається, що продукт, який буде проектуватися,

матиме соціальну значимість, в першу чергу для студентів та міжнародних координаторів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Зараз інформаційні технології грають важливу роль в сфері освіти. Технологія відкриває фундаментальні структурні зміни, які можуть бути невід'ємною частиною досягнення значних поліпшень продуктивності. Використовувані для підтримки як викладання, так і навчання, технології надають цифрові засоби навчання, навчальні матеріали, підтримують навчання 24 години на добу, 7 днів на тиждень, формують навички 21 століття, підвищують зацікавленість та мотивацію учнів, прискорюють навчання. Технології також мають силу трансформувати навчання шляхом впровадження нової моделі: ця модель пов'язує викладачів зі своїми студентами та професійним змістом, ресурсами та системами, що допомагає їм вдосконалити власні інструкції та персоналізувати навчання.

Інтернет-можливості навчання та використання відкритих освітніх ресурсів та інших технологій можуть підвищити продуктивність освіти за рахунок прискорення темпів навчання; зменшення витрат, пов'язаних з навчальними матеріалами або програмою; та краще використовувати час викладача.

Існує безліч сайтів для користування навчальними матеріалами, проведення онлайн-занять, що дуже допомагають у сучасному світі. Як було виявлено, бракує систем для формування індивідуальної траєкторії навчання студентів учасників міжнародної мобільності, яка стає більш популярною з кожним роком, бо надає багато переваг: особистісний ріст, впевненість, незалежність, знайомство з новою культурою, лідерські навички, унікальний освітній досвід, занурення в іноземну мову, нові знайомства, нетворкінг та можливості працевлаштування, корисні подорожі. Така система дозволила б студентам не витрачати багато часу на те, щоб проаналізувати існуючі можливості для міжнародної мобільності та подати заявку на участь, щоб було легше порівнювати пропозиції та умови різних навчальних закладів: необхідні для

участі документи, допустимий вік учасників, мови навчання, варіанти навчальних дисциплін, пропозиції щодо проживання.

Зараз, щоб пройти цей етап, потрібно перевіряти інформацію на різних сайтах університетів, що потребує немало часу. Сайти окремих університетів мають різний інтерфейс, який не завжди є ергономічним, а це ускладнює задачу ще більше.

Отже, вебзастосунок для формування індивідуальної траєкторії навчання студентів учасників міжнародної мобільності значно оптимізує процес аналізу пропозицій для мобільності та подачі заявки на участь наступним чином:

- ергономічність інтерфейсу;
- економія часу, бо всі дії для досягнення мети можна виконати перебуваючи на одному сайті;
- реєстрація, авторизація, користування власним обліковим записом, у приватному кабінеті зберігається вся особиста інформація, необхідна для участі у програмі обміну студентів;
- є можливість порівнювати всі відкриті для студентів програми мобільності, для оптимізації цієї дії є фільтрація;
- можливість подавати заявки;
- система розроблена і для студентів, і для координаторів (для них є адміністративна панель).

На сьогоднішній день існує невелика кількість аналогічних систем. Однією з них є система Дортмундського університету для студентської мобільності, яка буде розглянута у наступному розділі. А більшість українських університетів мають лише інформацію про навчання за кордоном, але не дозволяють студентами подавати онлайн-заявки на участь.

1.2 Огляд існуючих методів вирішення завдання

1.2.1 Веб-сайт від Дортмундського університету

Вебзастосунок для міжнародної мобільності від Дортмундського університету зображено на рис. 1.1.

До переваг цієї програми можна віднести:

- локалізація;
- ненагромаджений дизайн.

Із недоліків було виявлено те, що при створенні заявки на участь кожного разу потрібно вводити особисту інформацію, а це потребує більше часу. Це може бути оптимізовано наступним чином: зберігати всю особисту інформацію у приватному кабінеті, отже, її треба буде ввести лише 1 раз, а потім ці дані автоматично будуть додаватись до заявки.

Також локалізація працює некоректно. Якщо обрати у меню англійську мову, можна помітити, що не всі німецькі слова перекладені на англійську.

The screenshot shows the 'MOBILITY-ONLINE' application form. The header includes the university name and a language dropdown set to 'English'. The user is logged in as 'Galyna Tabunshchuk'. The left sidebar contains links like 'My institution', 'My addresses', and 'Create new application'. The main form area has two sections:

- Details of activities at Home Institution:**
 - Country of home institution: <-- Please select -->
 - Home Institution: <-- Please select -->
 - Studienniveau: ☐ Bachelor ☐ Master
- Daten zum geplanten Studium:**
 - Host country: Germany
 - Host institution: DORTMUN02 - DORTMUND UNIVERSITY ...
 - Faculty: <-- Please select --> (dropdown menu is open showing options like Faculty of Informationtechnology, Faculty of Architecture, etc.)
 - Study field: (dropdown menu is open showing options like Faculty of Design, Faculty of electrical engineering, etc.)
 - Double Degree: (checkbox)

At the bottom, there are buttons for 'Cancel application' and 'Send application'.

Рисунок 1.1 — Сторінка сайту для міжнародної мобільності від Дортмундського університету

1.2.2 Веб-сайт від Київського політехнічного інституту імені Ігоря Сікорського

Як було зазначено, майже всі українські навчальні заклади не надають студентам єдиний веб-сайт для формування індивідуальної траєкторії навчання студентів учасників міжнародної мобільності, хоча більшість з них кожного року відправляють багато студентів на навчання за кордоном.

Наприклад, Київський політехнічний інститут імені Ігоря Сікорського має окремий сайт для академічної мобільності <http://mobilnist.kpi.ua> (рис. 1.2). Перевагою сайту є присутність локалізації: англійська та українська мови.

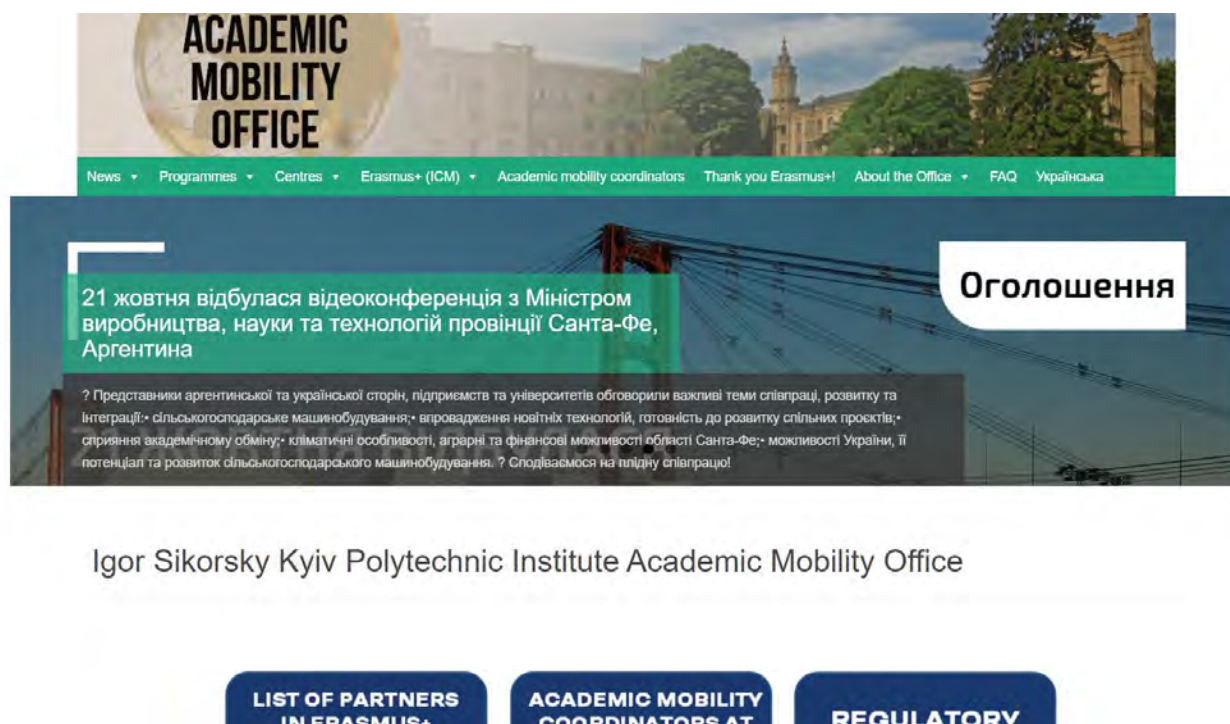


Рисунок 1.2 – Головна сторінка сайту для академічної мобільності від політехнічного інституту

Недоліки сайту:

- відсутність облікового запису;
- відсутність можливості порівнювання навчальних планів свого семестру та семестру університета-партнера на одній сторінці;
- відсутні необхідні дані для участі у мобільності, є посилання на сайти окремих навчальних закладів, для роботи з якими треба виділити час;

- є частини тексту дуже світлого кольору, що ускладнює сприйняття інформації, отже, дизайн не є ергономічним (рис. 1.3);
- некоректне відображення рисунків (рис. 1.4).

Exchange programme with University of Tohoku, Japan



I. Proposed programmes:

1. Junior Year Program in English (JYPE)

JYPE – це програма обміну, яка надає можливість іноземним студентам бакалаврату вивчати природничі науки, інженерію і сільськогосподарські курси в Університеті Тохоку. Всі лекції начитуються англійською мовою.

Програма починається в жовтні (осінній семестр). Тривалість в основному 1 рік.

Fall Semester: October 1 to March 31

Додаткова інформація: <http://www.insc.tohoku.ac.jp/english/exchange/jype/>

Рисунок 1.3 – Неергономічний дизайн

Student could conduct his researches in English if it was agreed with his scientific advisor.

Art and calligraphy: <http://www.sal.tohoku.ac.jp/index.html>

Education: <http://www.sed.tohoku.ac.jp/~english/>

Law: <http://www.law.tohoku.ac.jp/english/>

Economy: <http://www.econ.tohoku.ac.jp/econ/english/>

International culturology: <http://www.intcul.tohoku.ac.jp/english/>

Except for Faculty/Graduate school of Arts and Letters!



1

Рисунок 1.4 – Некоректне відображення рисунків

1.2.4 Веб-сайт від Львівського національного університету імені Івана Франка

Львівський національний університет імені Івана Франка теж має окремий веб-сайт для академічної мобільності <https://international.lnu.edu.ua/> (рис. 1.5). Його головною перевагою є інформативність, бо сайт дійсно насичений корисною інформацією для студентів, які бажають взяти участь у мобільності.

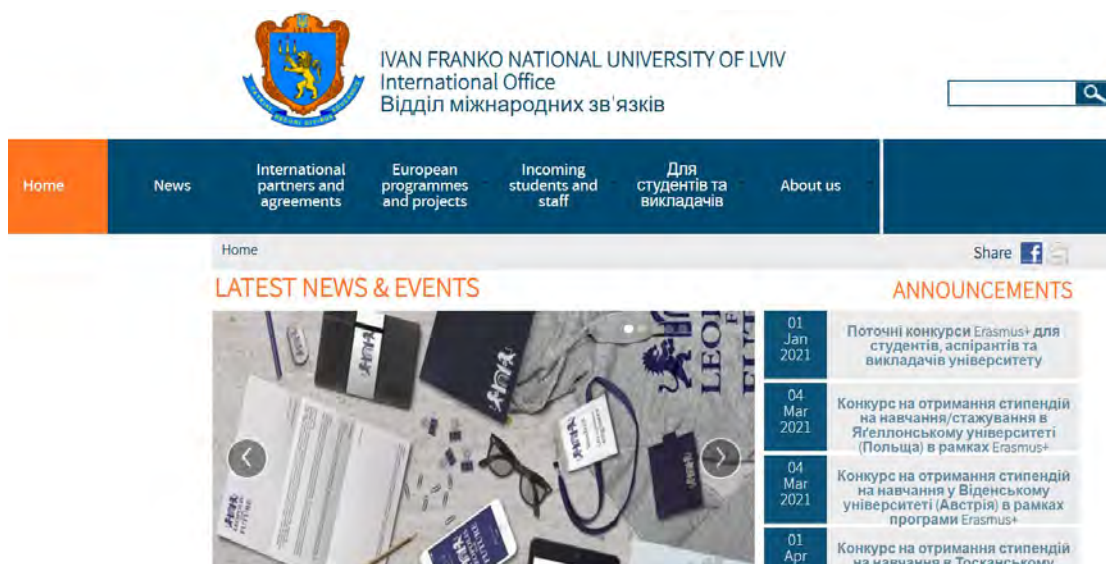


Рисунок 1.5 – Головна сторінка сайту для академічної мобільності від університету імені Івана Франка

Але розробники сайту не приділили увагу на деякі деталі, які заважають комфортному користуванню вебзастосунком або призводять до помилок. Наприклад, поле з повідомленням від користувача не має бути змінюваним (рис. 1.6).



Рисунок 1.6 – Необроблена властивість поля для повідомлення

Також відсутнє реагування адрес, номерів та поштових адрес на натискання, навіть якщо відкривати веб-сайт з мобільного пристрою (рис. 1.7). Наявність функцій для роботи з перерахованими елементами прискорила б роботу. Ще на сайті присутні елементи, що відображаються некоректно (рис. 1.8).

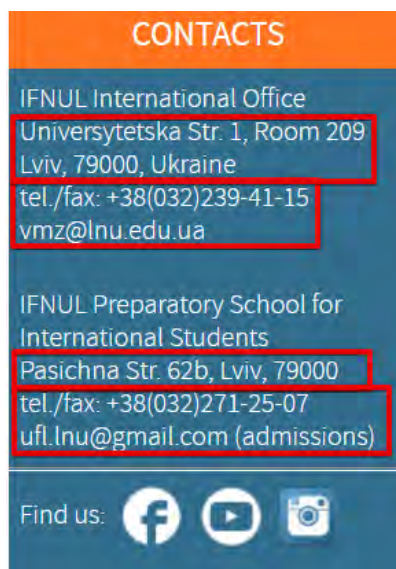


Рисунок 1.7 – Контактні дані

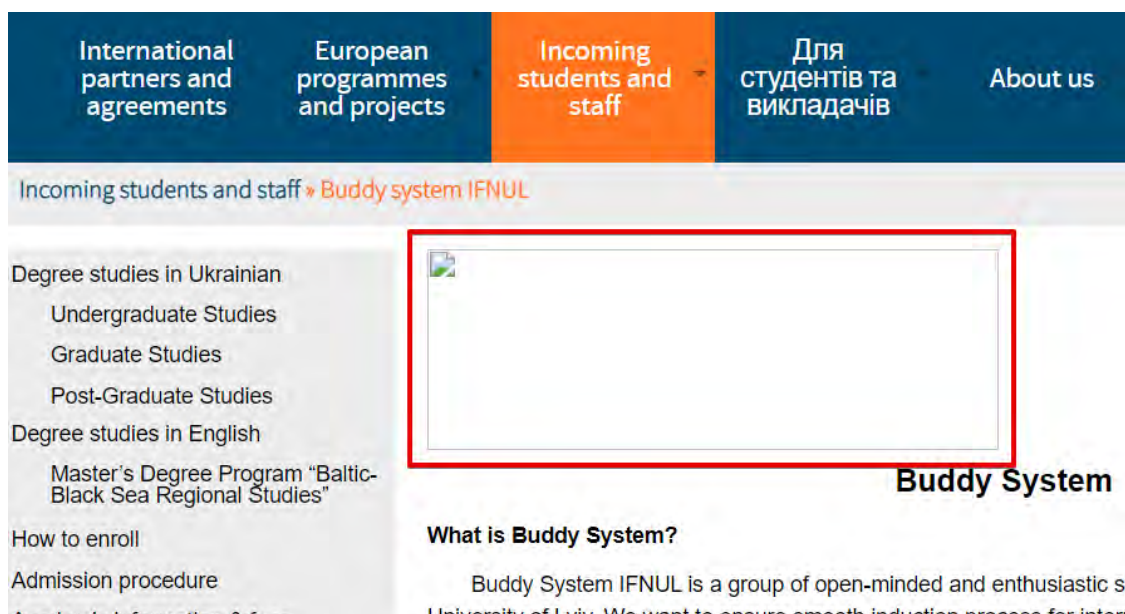
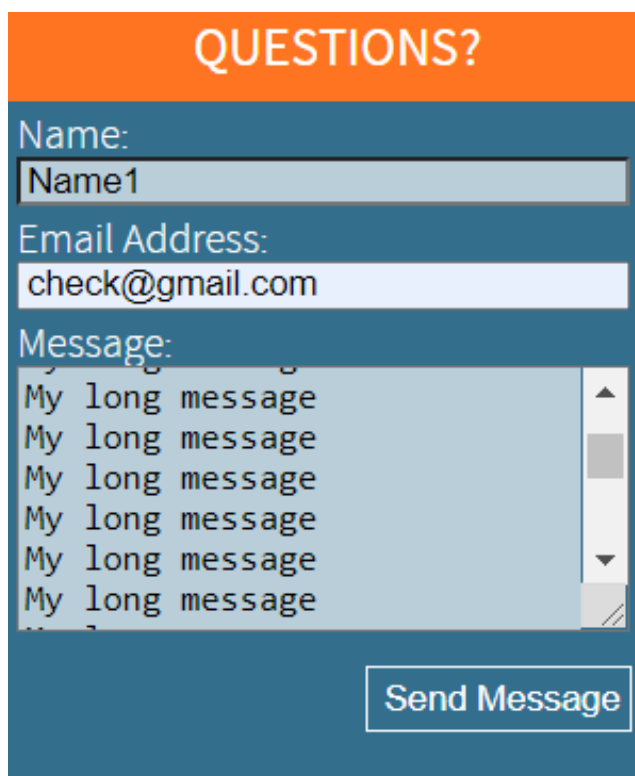


Рисунок 1.8 – Некоректне відображення елемента

Іншим недоліком є відсутність валідації, яка може стати причиною втрати повідомлення, яке користувач намагався відправити адміністратору. На рис. 1.9

видно, що у полі «Name» є помилка. При натисканні на «Send message» повідомлення буде втрачено (рис. 1.10), бо у полі Name відсутня валідація.



The screenshot shows a web form with an orange header containing the text "QUESTIONS?". Below the header, there are three input fields on a blue background. The first field is labeled "Name:" and contains the text "Name1". The second field is labeled "Email Address:" and contains the text "check@gmail.com". The third field is labeled "Message:" and contains the text "My long message" repeated six times. To the right of the message field is a vertical scrollbar. At the bottom right of the form is a button labeled "Send Message".

Рисунок 1.9 – Спроба відправити повідомлення

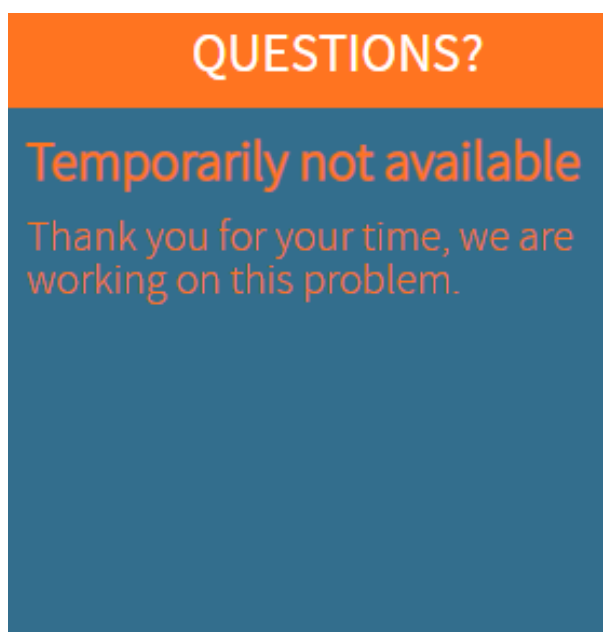


Рисунок 1.10 – Помилка при відправленні повідомлення

Як і на сайті Київського політехнічного інституту, на сайті університету імені Івана Франка немає облікових записів для збереження особистої інформації

та постійного користування нею, також немає можливості порівнювання навчальних планів свого семестру та семестру університета-партнера.

1.3 Технічне завдання

Програма матиме свою архітектуру та клієнт-сервер, тобто буде можливий віддалений доступ до бази даних (БД).

Програма призначена для створення та маніпулювання змістом бази даних. Функції вебзастосунку:

- реєстрація користувача з обов'язковим введенням персональних даних, потрібних для участі у мобільності;
- відправлення повідомлення на пошту для верифікації облікового запису користувача;
- можливість відновлення пароля;
- можливість редагувати дані користувача і перегляд поточних і минулих програм мобільності;
- пошук університетів для мобільності з використанням фільтрації та пагінації;
- відображення сторінки університету з факультетами, спеціальностями та програмами мобільності;
- можливість порівнювати програму мобільності і поточну програму користувача;
- можливість відправляти заявку на мобільність;
- адміністративна панель для маніпуляції усіма даними сайту: створення, видалення та редагування даних спеціальностей, факультетів, університетів, обробка заявок студентів (зміна статусу), обробка даних студентів.

1.4 Висновок до розділу 1

Проаналізувавши методи та програмні засоби реалізації систем для формування індивідуальної траєкторії навчання студентів можна сказати, що наразі не існує гнучкого та зручного програмного засобу для автоматизації мобільності, який задовільняв би усім необхідним вимогам, що висувуються до такого типу програмного забезпечення. Саме цьому програмний продукт дипломної роботи має поєднати в собі функціонал, який є у існуючих альтернативах та стати новим, повноцінним інструментом для організації мобільності, усунувши ті недоліки, які є в існуючих системах.

2 ВІБІР ТА ОБГРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЕКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного забезпечення

2.1.1 Вибір фреймворку

Для розробки вебзастосунку було обрано Vue.js – веб-фреймворк для розробки призначених для користувача інтерфейсів на мові програмування JavaScript. JavaScript має багато переваг, але, звісно, для оптимізації процесу створення системи та більш якісної роботи системи необхідно використовувати фреймворки та бібліотеки. Було обрано саме фреймворк JavaScript, бо JS незамінний для веб-розробки: є підтримка скриптів усіма популярними браузерами, повна інтеграція з версткою сторінок і серверною частиною. Іншою перевагою JS є швидкість роботи і продуктивність. Javascript дозволяє частково обробляти веб-сторінки на комп'ютерах користувача без запитів до сервера. Це економить час і трафік, знижує навантаження на сервер. Також в JS зручні користувацькі інтерфейси: заповнення форм, вибір дій, активація кнопок, перевірка введення, реагування на наведення/кліки миші і т.п, це дає приголомшливий рівень юзабіліті. Порівняння JS з іншими мовами програмування наведено в таблиці 2.1.

Із існуючих фреймворків та бібліотек JS було обрано саме Vue.js завдяки його багатофункціональності. Vue.js набув великої поваги і популярності серед розробників веб-продуктів. Зараз вже є версія Vue.js 3.0, але для розробки було обрано версію Vue.js 2.6.11, бо саме ця версія оптимізована для роботи багатьох пакетів, які також були використані для роботи системи.

Vue.JS було створено з оглядкою на кращі практики більш старих технологій. З React.JS була запозичена ідея віртуального DOM. Такий підхід виключає пряму взаємодію з вузлами інтерфейсу. Початкова робота ведеться з його простою копією, яка називається virtual DOM. А після цього вже застосовуються зміни до реальних вузлів інтерфейсу. Паралельно відбувається

порівняння віртуальної копії DOM дерева з реальним. Завдяки цьому виявляється різниця та перемальовується тільки те, що було змінено.

З Angular Vue.JS було запозичено two-way data binding, що дозволяє проектувати інтерфейси (по-перше, декларативно; по-друге, з використанням Vue в шаблонизаторах, таких як Haml або Pug).

Ядро Vue.JS, як і ядро React, містить тільки необхідний функціонал для взаємодії з інтерфейсом. Завдяки цьому воно компактне та є можливість його легкої інтеграції з іншими технологіями.

Також Vue має багато модулів, які реалізують гнучкий підхід для створення веб-додатків. Наприклад, майже всі React додатки проектуються разом із Redux – технологією контролювання станів. Redux – окрема бібліотека, яка реалізує flux-архітектуру. Підхід, який використовує бібліотека Redux, є дуже зручним і успішним. А для Vue було розроблено Vuex, який є власною технологією для контролювання стану. Vuex запозичує багато ідей Redux, але рівень інтеграції Vuex з Vue краще, ніж у React та Redux. Завдяки цьому з'являється швидкодія та зручність.

Отже, було вирішено, що Vue.js буде кращим варіантом. У табл. 2.2 наведено порівняльну характеристику JS-інструментів для створення додатків у відповідності до різних критеріїв.

Таблиця 2.1 – Порівняння мов програмування

1	Можливість	Мова					
		C#	Go	Java	JavaScript	PHP	Python
2	Імперативність	+	+	+	+	+	+
3	Об'єктно-орієнтованість	+	+/-	+	+	+	+
4	Функціональність	+/-	+/-	+/-	+/-	+/-	+
5	First class functions	+	-	-	+	+	+
6	Анонімні функції	+	-	+	+	+	+
7	Лексичні замикання	+	-	+	+	+	+
8	Часткове застосування	-	-	-	+	-	+

Продовження таблиці 2.1

9	Можливість	Мова					
		C#	Go	Java	JavaScript	PHP	Python
10	Карирування	+	-	-	+	+	+
11	Пітримка Unicode в ідентифікаторах	+	+	+	+	+/-	+
12	Значення параметрів за замовчуванням	+	+	-	+	+	+
13	Наявність бібліотек для роботи з графікою і мультимедіа	+	-	+	+	+/-	+

Таблиця 2.2 – JS-фреймворки

1	Критерії	Інструмент		
		Angular.js	React.js	Vue.js
2	Тип	Фреймворк	Бібліотека для створення користувацького інтерфейсу	Фреймворк
3	Типи додатків	Нативні, гібридні і веб-додатки	Односторінкові (SPA) і мобільні додатки	Модернізовані односторінкові додатки і підтримка нативних додатків
4	Модель	MVC	Віртуальна DOM	Віртуальна DOM
5	Переважна мова розробки	Рекомендовано використовувати TypeScript	Рекомендовано використовувати JSX – JavaScript XML	HTML-шаблони та JavaScript.
6	Ідеально підходить для	Великомасштабних додатків з багатим функціоналом	Веб-додатків і нативних додатків для iOS і Android	Веб-додатків і односторінкових (SPA) додатків
7	Спрямованість і класи залежностей	Взаємодії об'єктів відбувається в службах	Одностороння передача даних	Одностороння передача даних
8	Зворотня сумісність	Абсолютна залежність від попередніх версій	Повна сумісність між версіями	Повна сумісність між версіями
9	Розмір бібліотек	500+ KB	100 KB	80 KB

Vue.JS реалізує всі сучасні підходи до розробки Web-UI, це гнучкий фреймворк, який високо інтегрований зі сторонніми технологіями [2].

Для розробки проекту було використано не лише Vue.js, а і допоміжні інструменти: Vuex, Vuetify, Vue Router, Vue i18n.

Vuex - патерн управління станом та бібліотека для додатків на Vue.js. Це централізоване сховище даних для всіх компонентів програми з правилами, що гарантують, що стан може бути змінено лише певним чином. Vuex інтегрується з офіційним розширенням vue-devtools, надаючи «з коробки» такі можливості, як «машина часу» для налагодження і експорту/імпорту зліпків стану даних.

Наступні інструменти є основними в роботі Vuex:

- стан (State) - це дані, що зберігаються в додатку. Компоненти можуть «отримувати» ці дані, а при їх зміні компонент буде оновлюватись сам;

- сховище (Store) управляє станом. Воно повинно змінюватись тільки зсередини, тому зовнішні компоненти не можуть безпосередньо змінювати стан. Щоб змінити стан, компоненти можуть відправити дію (dispatch) або зафіксувати мутацію (commit);

- геттери (Getters) обчислюють властивості на основі стану сховища. Їх можна використовувати в діях (actions) або безпосередньо в компоненті. Це корисно, якщо кільком компонентам необхідно розрахувати одну і ту ж річ на основі даних сховища, це можна зробити в одному місці замість того, щоб робити це окремо для кожного компонента;

- дія (Екшен/Action) містить бізнес-логіку і не дбає про оновлення стану безпосередньо. Причина в тому, що дії є асинхронними (код може продовжувати виконуватися, навіть якщо дія ще не завершена), це корисно, якщо потрібно почекати, щоб отримати дані з Application Programming Interface (API), наприклад. Дія викликає мутацію, яка буде безпосередньо постійно оновлюватися;

- мутації (Mutations): щоб оновити стан, потрібно зробити мутацію. Мутація не дбає про бізнес-логіку, її єдиною метою є оновлення стану. Мутація є синхронною (код повинен чекати, поки мутація не буде завершена). Мутації повинні бути єдиним способом оновлення стану, щоб забезпечити управління станом передбачуваним [3].

У проєкті стилі були розроблені за допомогою Vuetify. Vuetify - це семантична структура компонентів для Vue. Він націлений на створення чистих, семантичних і багаторазово використовуваних компонентів, які роблять створення вашого сайту швидким. Vuetify надає розробникам можливість створення користувацьких інтерфейсів з використанням принципів Google Material Design.

Vue Router - офіційна бібліотека маршрутизації для Vue.js. Вона глибоко інтегрується з Vue.js і дозволяє легко створювати SPA-додатки.

Отже, завдяки використанню Vue.js та всіх перерахованих інструментів можна якісно створити вебзастосунок для студентів.

2.1.2 Вибір середовища розробки

В якості середовища розробки обрано середовище Visual Studio Code.

Visual Studio Code - це сервіс, який позиціонується як «легкий» редактор коду для кроссплатформної розробки веб- і хмарних додатків.

Особливості Visual Studio Code:

- VS Code дозволяє розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ;

- у редакторі присутні вбудований відладчик, інструменти для роботи з Git і засоби рефакторинга, навігації по коду, автодоповнення типових конструкцій і контекстної підказки;

- продукт підтримує розробку для платформ ASP.NET і Node.js, і вважається легким рішенням, яке дозволяє обійтися без повністю інтегрованого середовища розробки;

- великим плюсом редактора є підтримка великої кількості мов, таких як C ++, C #, Python, PHP, JavaScript та інших.

Можливості Visual Studio Code:

- вбудовані інструменти інтеграції з GitHub, GIT, а також Visual Studio Team Services для швидкого тестування, складання, упаковки та розгортання різних типів додатків;
- зручність роботи з Unity-проектами;
- робота з Mono і Node.js за допомогою вбудованого відладчика;
- підтримка TypeScript і JavaScript;
- публікація створених додатків в Microsoft Azure через сервіс Visual Studio Team Services;
- підтримка практично всіх мов програмування;
- написання коду для конкретного завдання з його подальшою інтеграцією в проект (з надбудовою або безпосередньо);
- велика бібліотека шаблонів, готових фрагментів коду і сніпетів з можливістю додавання своїх елементів;
- одночасна робота з декількома проектами (в декількох вікнах);
- інтерфейс можна розділити на дві панелі для порівняння коду;
- функція налагодження [4].

Отже, Visual Studio Code має потужні інструменти для обслуговування коду, виявлення і усуненні дефектів.

2.1.3 Вибір інструменту для тестування API

Для тестування API було обрано Postman, бо дана платформа - це потужний інструмент, який при цьому має простий і інтуїтивно зрозумілий інтерфейс. Можливості графічного інтерфейсу обраної платформи:

- зберігати запити для легкого запуску в майбутньому;
- легке введення інформації в правильному форматі;
- перегляд відповіді в форматі JSON або в необробленому вигляді.

Отже, Postman має графічний інтерфейс і тому він набагато зручніше, ніж консоль. Можна дуже просто налаштовувати передані Cookie, HTTP заголовки, а

також параметри форм. Все це дозволяє зручно перевірити роботу будь-якого API.

2.1.4 Огляд особливостей обраної системи управління базами даних

База даних міститься на MongoDB 4.4.4 сервері у вигляді набору колекцій, що пов'язані між собою. MongoDB – найпопулярніша нереляційна СУБД. Перевагами MongoDB є:

- динамічна схема: ця СУБД дозволяє гнучко працювати зі схемою даних без необхідності змінювати самі дані;
- масштабованість: MongoDB горизонтально масштабована, що дозволяє легко зменшити навантаження на сервер при великих обсягах даних;
- зручність в управлінні: СУБД не потребує окремого адміністратора бази даних. Завдяки достатній зручності у використанні, системою легко можуть користуватися як розробники, так і системні адміністратори;
- швидкість: висока продуктивність при виконанні простих запитів;
- гнучкість: в MongoDB можна без шкоди для існуючих даних, їх структури і продуктивності СУБД додавати поля або колонки.

Для керування MongoDB було обрано крос-платформний інструмент Robo 3T. На відміну від більшості інших інструментів користувацького інтерфейсу адміністратора MongoDB, Robo 3T вбудовує фактичну оболонку Mongo в інтерфейс з вкладками з доступом до командного рядка оболонки і взаємодією з графічним інтерфейсом.

2.2 Структурна схема програми

На (рис. 2.1) зображено структурну схему програмного забезпечення.

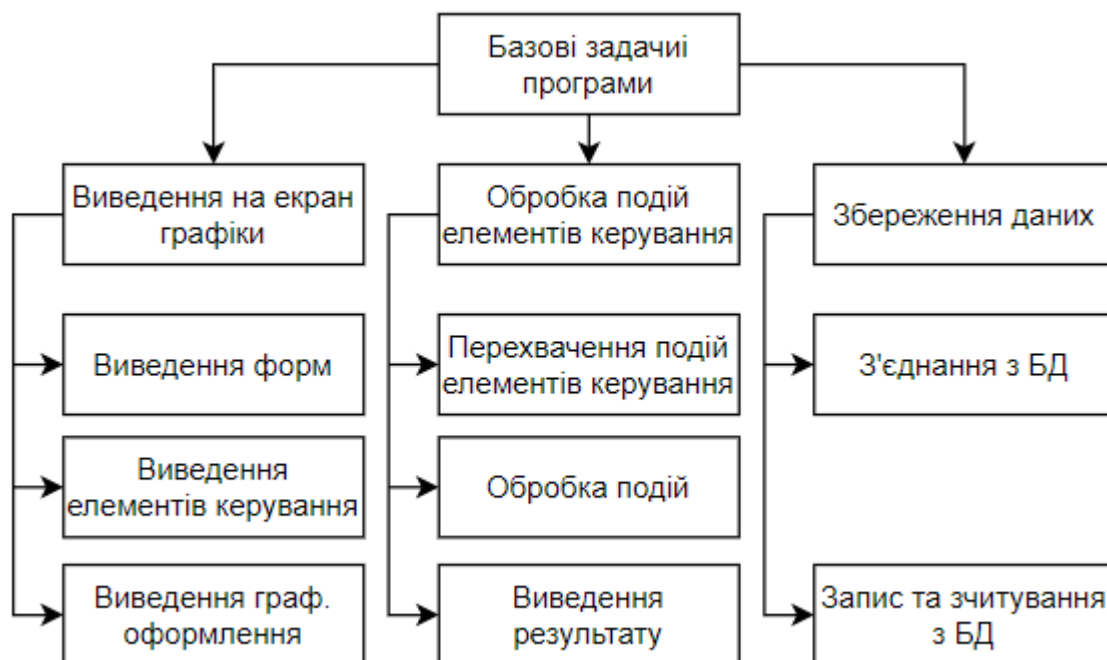


Рисунок 2.1 – Структурна схема програмного забезпечення

2.3 Висновок до розділу 2

Для створення вебзастосунку для формування індивідуальної траєкторії навчання студентів учасників міжнародної мобільності було обрано ефективний та сучасний набір інструментів. Фреймворком було обрано Vue.js, середовищем розробки Visual Studio Code. Додаткові інструменти: Vuex, Vuetify, Vue Router, Vue i18n. Також було описано обрану систему управління базами даних – MongoDB 4.4.4.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Основні розроблені алгоритми

Основні алгоритми розробленої системи:

- вхід в адміністративну панель;
- маніпуляції даними через адміністративну панель;
- реєстрація;
- авторизація;
- відновлення пароля;
- пошук університетів з фільтрацією;
- пагінація на сторінці зі списком університетів;
- подача заявок (з вибором предметів та доданням мотиваційного листа).

Щодо основного алгоритму програми, який зображений на (рис 3.1 – 3.4) [5]. Користувач (студент), запустивши систему, повинен зареєструватися. Після підтвердження електронної пошти реєстрація буде завершена. Потім користувач переходить на сторінку авторизації, де в нього завжди буде можливість відновити пароль при необхідності. Після авторизації відкривається особистий профіль студента, де в нього є можливість редагувати особисті дані в налаштуваннях та перевіряти свої заявки на участь.

Для приближення до досягнення своєї мети – взяти участь в академічній мобільності, студент повинен перейти на сторінку зі списком університетів. Для оптимізації процесу пошуку у нього є можливість користуватись пошуковим полем та також фільтрацією (локація, мова, період, тип, сезон, факультет, спеціальність). Потім студент може перейти на сторінку університету та застосувати інші фільтри для того, щоб продивитись програму навчання, яка зацікавить. Потім для порівняння предметів, які є у студента в поточному університеті, з предметами університета-партнера треба відкрити модальне вікно, де у формі таблиці є необхідні дані. Якщо студент хоче спробувати взяти участь у програмі, йому необхідно відправити заявку на участь. Для цього розроблена окрема сторінка, на якій студент обирає предмети, які зацікавили його, та

прикріплює мотиваційний лист у форматі .pdf або пише його у спеціальному текстовому полі.

Адміністратор може виконувати наступні функції через адміністративну панель: створення, видалення та редагування даних спеціальностей, факультетів, університетів, обробка заявок студентів (зміна статусу), обробка даних студентів.

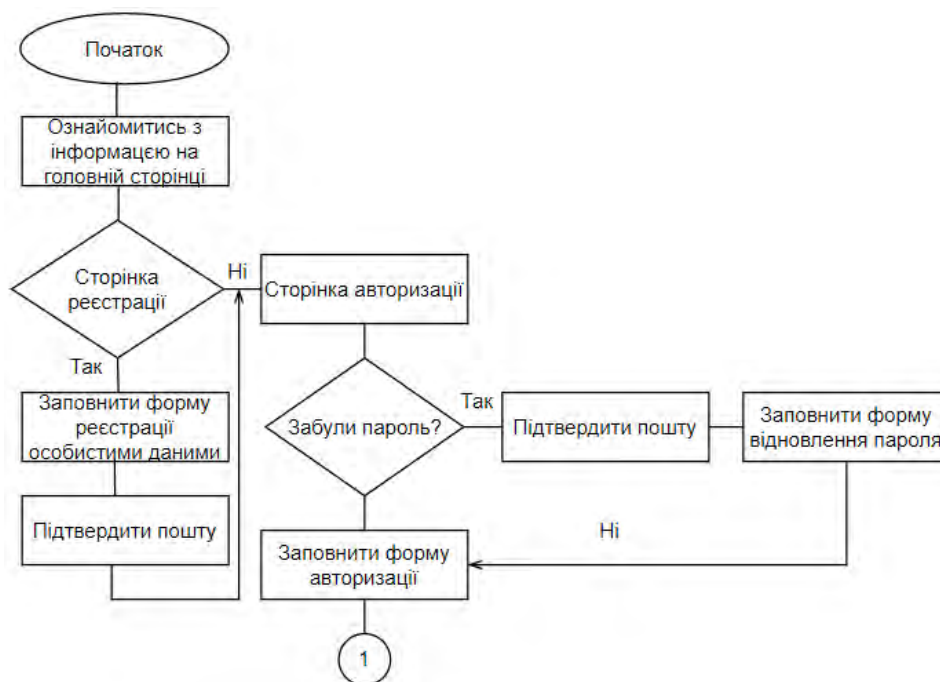


Рисунок 3.1 – Блок-схема програми, частина 1

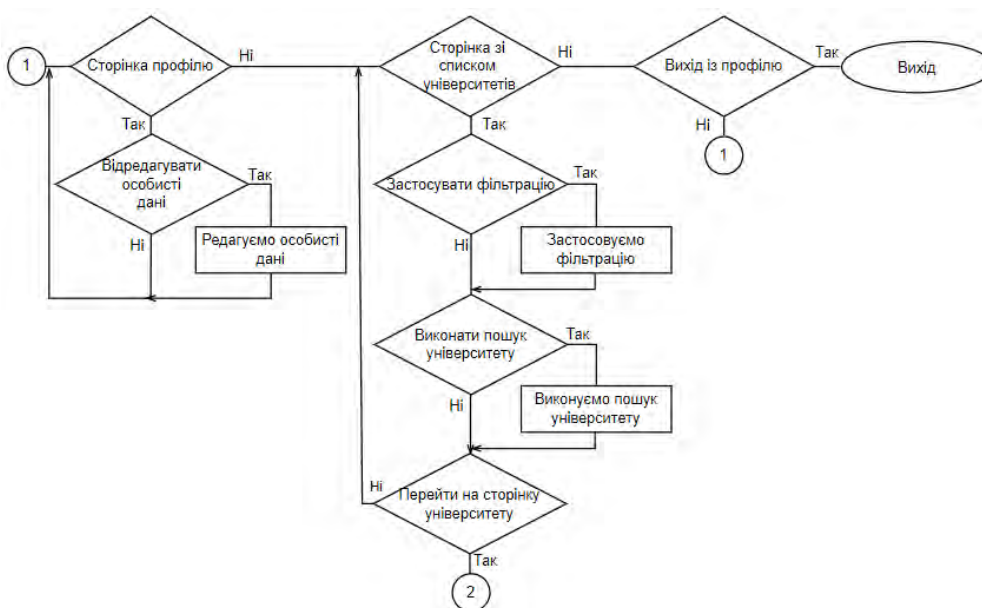


Рисунок 3.2 – Блок-схема програми, частина 2

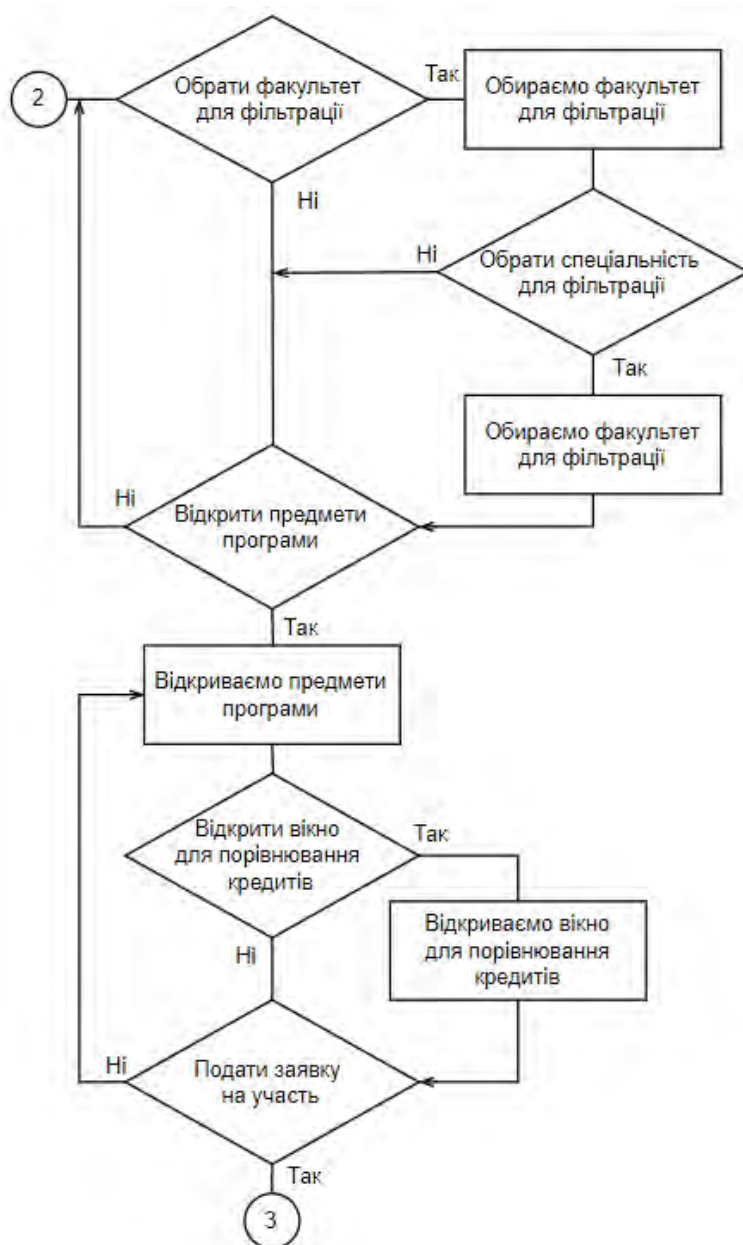


Рисунок 3.3 – Блок-схема програми, частина 3

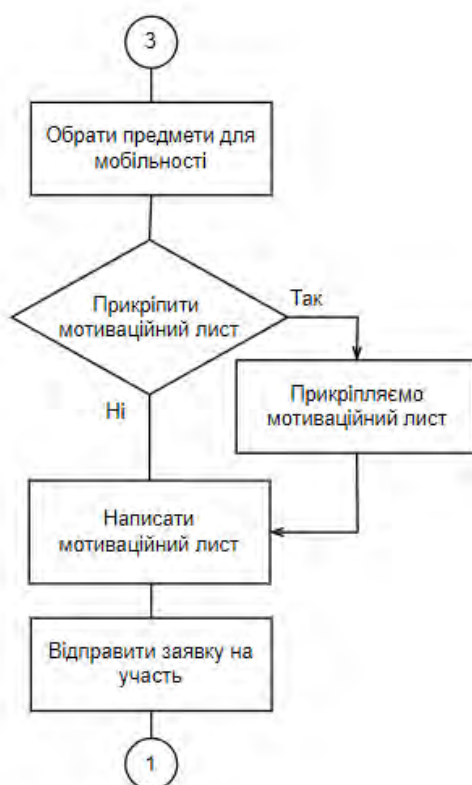


Рисунок 3.4 – Блок-схема програми, частина 4

3.1.1 Авторизація до системи

Для того, щоб користуватись розробленою системою в обмеженому доступі, достаньо перейти за посиланням на сайт. Для того, щоб мати доступ до всіх функцій, користувачу необхідно створити акаунт та ввести особисті дані. Після успішної реєстрації користувач може легко авторизуватись, використовуючи логін та пароль.

3.1.2 Вхід до системи як студент

Зайшовши до системи, як студент, користувач отримує доступ до таких функцій:

- доступ до головної сторінки з описом системи;
- авторизація;

- реєстрація з доданням особистих даних, потрібних для участі у мобільності;
- відновлення пароля;
- редагування налаштувань в особистому акаунті;
- доступ до сторінки зі списком всіх навчальних закладів;
- пошук університетів з фільтрацією (локація, мова, період, тип, сезон, факультет, спеціальність);
- доступ до сторінки окремого навчального закладу;
- можливість порівнювання навчальних планів свого університету та університету-партнера;
- подача заявки на участь у певній програмі мобільності.

3.1.3 Вхід до адміністративної панелі

Адміністратор має наступні привілеї в адміністративній панелі:

- доступ до списку користувачів та їх заявок з мотиваційними листами;
- створення даних спеціальностей, факультетів, університетів;
- видалення даних спеціальностей, факультетів, університетів;
- редагування даних спеціальностей, факультетів, університетів;
- обробка заявок студентів (зміна статусу);
- обробка даних студентів;
- пошук навчальних закладів;
- застосування фільтрації для пошуку навчальних закладів.

Сторінка авторизації в адміністративну панель зображена на рис. 3.5. На рис. 3.6 зображена панель адміністратора, за допомогою якої він може маніпулювати даними навчальних закладів.

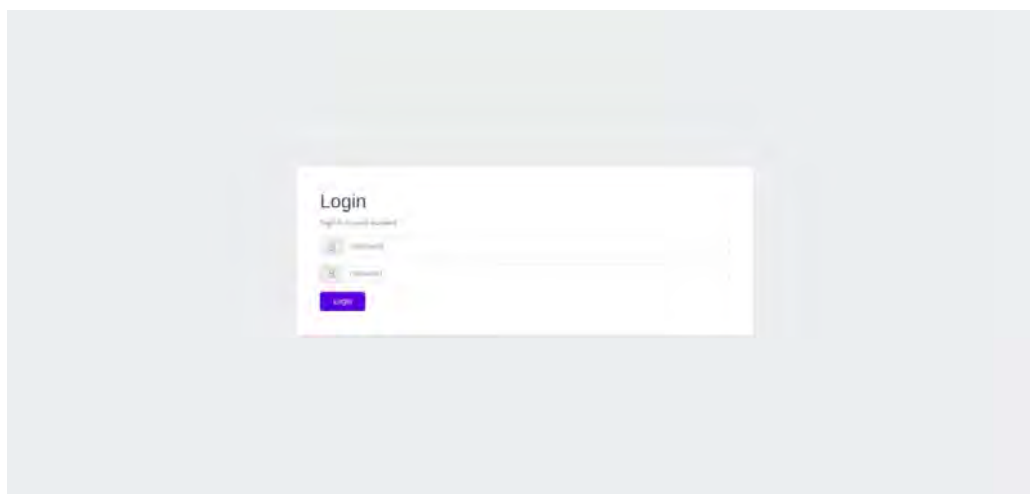


Рисунок 3.5. – Авторизація в адміністративну панель

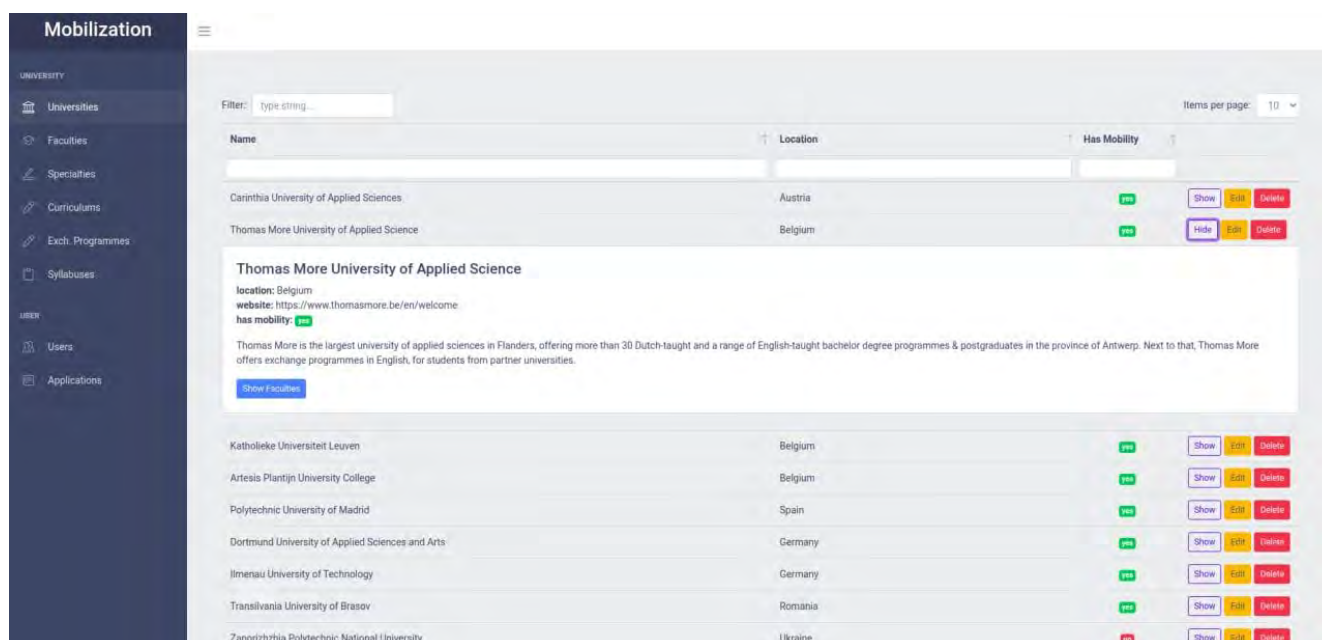


Рисунок 3.6 – Сторінка з навчальними закладами в адміністративній панелі

3.2 Основні рішення щодо модульного уявлення системи

Проект включає в себе такі частини:

Папка localization:

- en.js – словник з англійським перекладом;
- ua.js – словник з українським перекладом.

Папка public містить папку «images», в якій зберігаються картинки для проекту та файл index.html для відображення сайту.

Папка `src` містить наступні папки: `components`, `data-layer`, `pages`, `plugins`, `services`, `store`, `utils`; та файли:

- `App.vue` – головний компонент для відображення дочірніх компонентів проекту;
- `main.js` – скрипт, який зв'язує всі компоненти та плагіни воедино та підключає його до `index.html`;
- `router.js` – зберігає в собі URL-шляхи до сторінок.

Папка `components`:

- `/modal/Compare.vue` – модальне вікно для відображення даних для порівняння планів університетів;
- `/profile/Applications.vue` – відображає перелік навчальних планів, в яких ми вже брали участь або беремо участь;
- `/profile/Index.vue` – відображає дані користувача;
- `/profile/Settings.vue` – компонент для редагування даних користувача;
- `Select.vue` – кастомний компонент для поля «select»;
- `Skeleton.vue` – компонент для стартового шаблону при завантаженні сторінки;
- `SkeletonCard.vue` – компонент для стартового шаблону при завантаженні університетів в вигляді карток;
- `TranslationButton.vue` – компонент для кнопки зміни локалізації;
- `UniversityCard.vue` – компонент для відображення університетів у вигляді карток.

Папка `data-layer`:

- `HttpClient.js` – клас, який реалізує методи для запитів до API.

Папка `pages`:

- `Application.vue` – сторінка для подачі заявки на участь у мобільності;
- `Forgot.vue` – сторінка «Забув пароль», де потрібно ввести свою поштову адресу;
- `Home.vue` – сторінка-лендінг для ознайомлення з сайтом;
- `NotFound.vue` – сторінка, яка відкривається у випадку переходу на

неіснуючу сторінку;

- Profile.vue – сторінка для відображення профілю користувача;
- Reset.vue – сторінка для відновлення пароля;
- SignIn.vue – сторінка для авторизації користувача;
- SignUp.vue – сторінка для реєстрації користувача;
- Universities.vue – сторінка зі списком університетів;
- University.vue – сторінка з окремим університетом;
- Verify.vue – сторінка для верифікації пошти.

Папка plugins:

- i18n.js – плагін для підключення локалізації;
- vuetify.js – плагін для підключенні стилів.

Папка services:

- httpClient.js – скрипт для ініціалізації класів роботи з API.

Папка store:

- index.js – скрипт, який реалізує глобальне сховище.

Папка utils:

- qs.js – скрипт для парсінгу URL-запитів;
- validation.js – скрипт, який реалізує методи валідації даних з

локалізацією.

3.3 Організація даних

Для маніпулювання даними використовується сторонній API, в якому реалізовані всі необхідні ендпоінти для роботи з даними. Ендпоінт (Endpoint - кінцева точка) - це саме звернення до маршруту окремим HTTP методом. Ендпоінти виконують конкретне завдання, приймають параметри і повертають дані клієнту.

Для маніпуляції даними виконувались наступні методи HyperText Transfer Protocol (HTTP) запиту:

- GET – запитує уявлення ресурсу. Запити з використанням цього

методу можуть тільки отримувати дані. Цей метод не має тіла.

— POST – використовується для відправки сутностей до певного ресурсу. Цей метод має тіло.

— PUT – замінює всі поточні уявлення ресурсу даними запиту;

— DELETE – видаляє вказаний ресурс;

— PATCH – використовується для часткової зміни ресурсу.

Більш детальна інформація про запити до API описана в таблиці 3.1.

Таблиця 3.1 — Запити до API

1	Тип запиту	Шлях запиту	Опис запиту – що він приймає	Опис відповіді – що повертається
2	post	/auth/register	Дані для реєстрації	Повідомлення про успішну реєстрацію
3	put	/auth/verify/:token	Токен для верифікації пошти	Токен користувача
4	post	/auth/login	Логін та пароль для авторизації користувача	Токен користувача
5	post	/auth/forgot	Адреса пошти для відправлення повідомлення про відновлення пароля	Повідомлення про успішне відправлення повідомлення на пошту
6	post	/auth/reset/:token	Токен та новий пароль	Повідомлення про успішне відновлення пароля
7	get	/auth/me	Токен користувача	Дані користувача
8	post	/application	Дані для подачі заявки на мобільність	Повідомлення про успішну подачу заявки
9	get	/university	Номер поточної сторінки, кількість елементів на одній сторінці та необов'язковий параметр для пошуку університету за назвою	Вказана кількість університетів відповідно до вказаного фільтру пошуку
10	get	/university/:universityId	ID університету	Університет за вказаним ID

Продовження таблиці 3.1

11	Тип запиту	Шлях запиту	Опис запиту – що він приймає	Опис відповіді – що повертається
12	get	/university-of-mobility	Номер поточної сторінки, кількість елементів на одній сторінці, необов'язковий параметр для пошуку університету за назвою та параметри фільтрації	Вказана кількість університетів відповідно до параметрів пошуку та фільтрації
13	get	/university/:universityId/faculty	ID університету	Список факультетів вказаного університету
14	get	/university/:universityId/faculty/:facultyId/specialty	ID університету та факультету	Список спеціальностей вказаного факультету та університету
15	get	/university/:universityId/faculty/:facultyId/specialty/:specialtyId/curriculum	ID університету, факультету та спеціальності	Список навчальних планів вказаної спеціальності факультету університету
16	get	/filter-fields/university	–	Список полів для фільтрації університетів мобільності

3.3.2 Введення та виведення даних

Введення даних виконується за допомогою загальних елементів інтерфейсу, комп'ютерної миші та клавіатури. Звісно, розробленим сайтом можна користуватись через будь-який пристрій з доступом до Інтернету. Виведення даних виконується за допомогою монітору комп'ютера/екрана телефона.

3.4 Користувацький інтерфейс GUI

Архітектура користувацького інтерфейсу є компонентною. Компоненти Vue - це блоки коду, які підходять для багаторазового використання. Кожний компонент оформлюється завдяки Vuetify. Для зрозумілого відображення програми використовуються загальні елементи інтерфейсу, наприклад: Dialog, Footer, Form inputs and controls (Checkbox, File input, Form, Input, Radio button, Slider, Textarea, Text field), Table, Tooltip та інші.

3.5 Захищення даних

Для захищення даних користувач матиме свій логін і пароль. Пароль має бути не менше 6 символів та містити як цифри, так і літери. Усі паролі зберігаються у базі даних, до якої має доступ тільки адміністратор системи.

Неzareєстрований користувач має обмежений доступ до системи.

3.6 Адаптивність до змін та оновлень додатку

Оскільки програма Visual Studio Code дуже гнучка, то доповнювати код дуже легко, та цей процес ніяк не відобразиться на роботоздатності старого коду. Програміст може легко додавати необхідні пакети для оптимізації процесу розробки. Це дозволяє максимально швидко та якісно розробляти програмний продукт.

3.7 Висновок до розділу 3

Було створено основні алгоритми програми, які нададуть змогу для розширення функціоналу та реалізації усіх поставлених задач. Основними з яких є авторизація до системи, взаємодія з програмами академічної мобільності, подача заявок, маніпуляції даними через адміністративну панель.

Також було обрано механізм реалізації цих алгоритмів, елементи керування, з якими користувачу буде простіше взаємодіяти із програмою.

Були описані і заповненні всі таблиці, які роблять систему працездатною.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Розроблена система призначена для забезпечення гнучкого та простого механізму аналізу існуючих можливостей академічної мобільності та подачі заявок на участь.

За допомогою розробленої адміністративної панелі можна проводити маніпуляції (створення, видалення та редагування) з даними навчальних закладів, факультетів, спеціальностей, навчальних планів, предметів, заявок (змінювати статус заявок студентів) та також з особистими даними студентів.

Вебзастосунок надає користувачу наступні можливості:

- доступ до головної сторінки з описом вебзастосунку;
- авторизація;
- реєстрація з підтвердженням електронної пошти;
- відновлення пароля;
- редагування налаштувань в особистому акаунті;
- доступ до сторінки зі списком всіх навчальних закладів;
- пошук університетів з фільтрацією (локація, мова, період, тип, сезон, факультет, спеціальність);
- доступ до сторінки окремого навчального закладу;
- можливість порівнювання навчальних планів свого університету та університету-партнера;
- подача заявки на участь у певній програмі мобільності.

4.2 Характеристики програми

До складу технічних засобів повинна входити персональна електронна обчислювальна машина (ПЕОМ), що виконує роль сервера, що включає в себе:

- процесор Intel Core i5 3.4Hz, не менше;
- оперативну пам'ять об'ємом 16 Гігабайт, не менше;

- HDD 100 Гігабайт, не менше;
- операційну систему Ubuntu 20.04;
- Node 14.x;
- Npm 6.14.x;
- платформу для розгортання проекту Docker 19.3.x;
- базу даних MongoDB 4.4.x.

4.3 Звернення до програми

Звернення до програми виконується за допомогою наступних кроків. Із кореневої папки потрібно виконати команду `npm install`, щоб встановити необхідні пакети бекенду. Далі необхідно заповнити файл `«.env»` даними для налаштувань. Потім необхідно запустити Docker Container для ініціалізації бази даних. Потім за допомогою консолі запустити бекенд.

Далі треба перейти в папку `client`. Встановити пакети для фронтенду за допомогою команди `npm install`. Потім треба виконати команду `npm run serve` для запуску проекту.

4.4 Виняткові ситуації

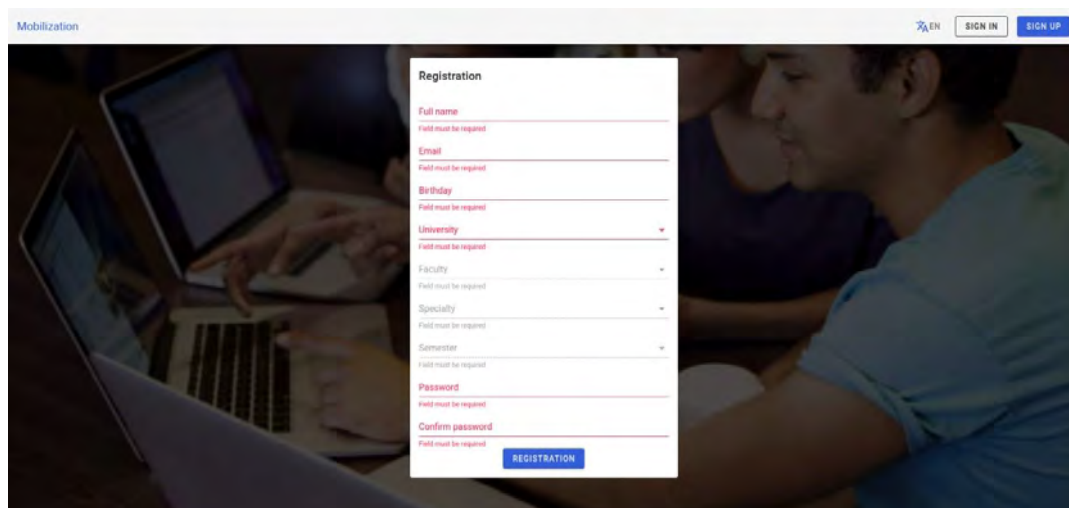
Обробка помилок поділяється на 2 типи, що детальніше описані у (табл. 4.1).

Таблиця 4.1 – Список виняткових ситуацій

Список помилок	Тип помилок	
	Помилки при заповненні форми	Помилки у середині програми
Невалідні дані при реєстрації	+	-
Авторизація без підтвердження акаунта	+	-
Невалідні дані при авторизації	+	-
Відсутність результатів при пошуку навчальних закладів	-	+
Перехід на неіснуючу сторінку	-	+

4.4.1 Помилки при заповненні форми

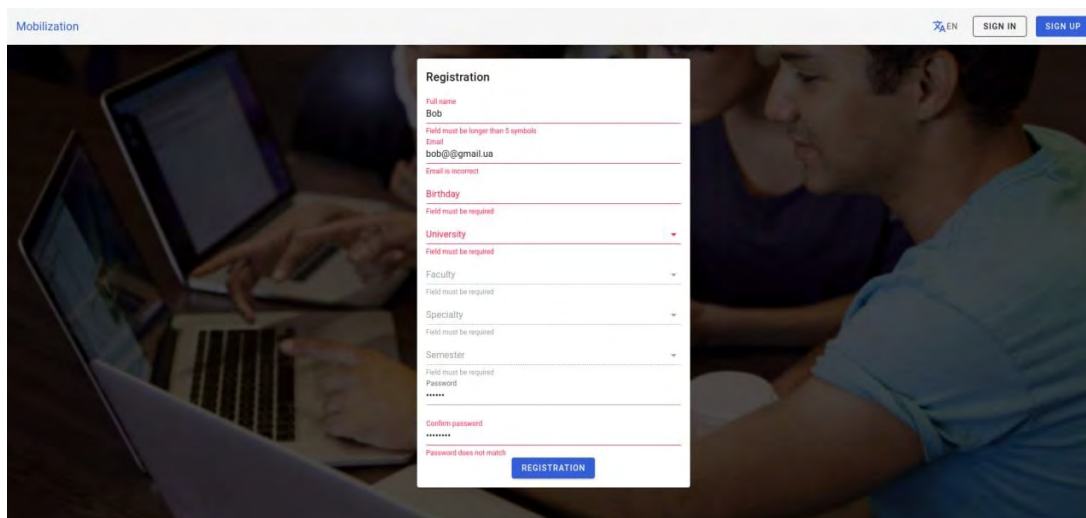
Для обробки помилок форми реєстрації використовується скрипт, який приймає поля форми та проводить аналіз за заданими правилами. При незаповнених полях користувач отримає повідомлення про помилки (рис. 4.1)



The screenshot shows a web application titled "Mobilization" with a registration form. The form fields are: Full name, Email, Birthday, University, Faculty, Specialty, Semester, Password, and Confirm password. Each field has a red error message below it: "Field must be required". A blue "REGISTRATION" button is at the bottom of the form. The background shows a person using a laptop.

Рисунок 4.1 – Повідомлення про помилки при пустих полях

Якщо користувач ввів невалідні дані (неповне ім'я, некоректний email і т.д.), то він отримає детальне повідомлення про свої помилки, щоб було легше їх виправити (рис. 4.2).



The screenshot shows the same registration form, but with specific error messages: "Full name" has "Bob" and "Field must be longer than 5 symbols"; "Email" has "bob@gmail.ua" and "Email is incorrect"; "Password" has "*****" and "Confirm password" has "*****" with the message "Password does not match". The "REGISTRATION" button is still present. The background shows a person using a laptop.

Рисунок 4.2 – Повідомлення про невалідні дані

Якщо користувач намагається авторизуватись з непідтвердженим акаунтом, то він отримає повідомлення про помилку (рис. 4.3).

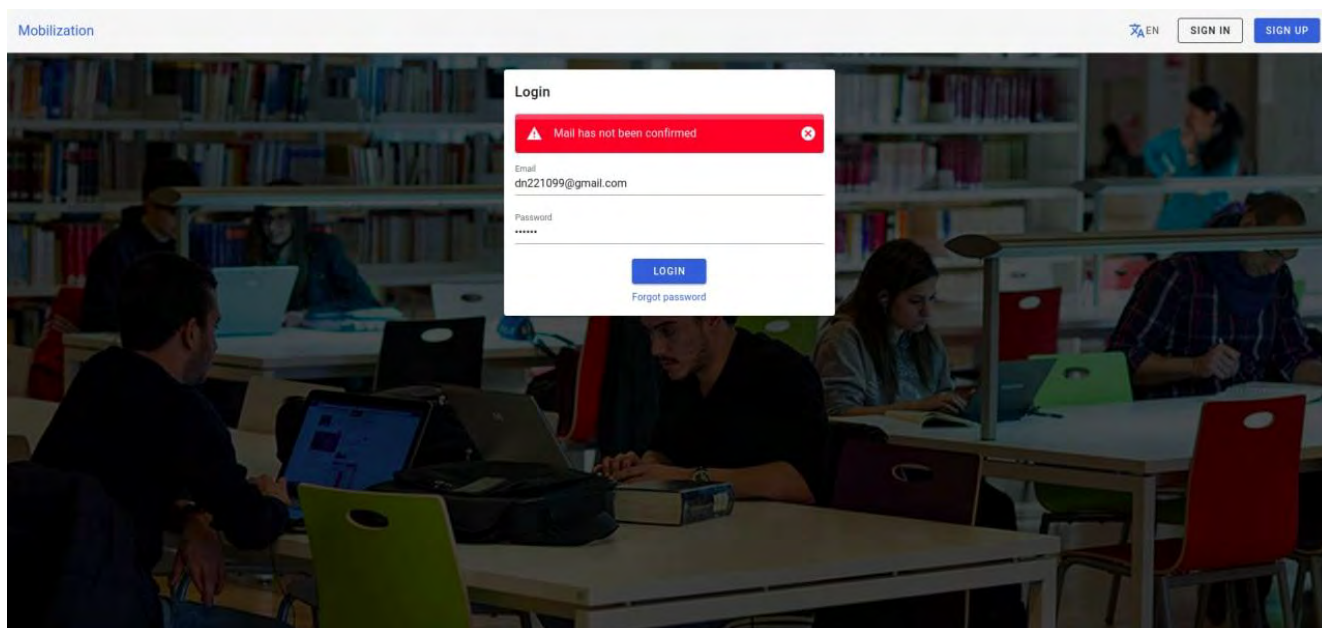


Рисунок 4.3 – Авторизація без підтвердження акаунта

Повідомлення, яке виводиться при невалідних даних при авторизації, зображено на рис. 4.4.

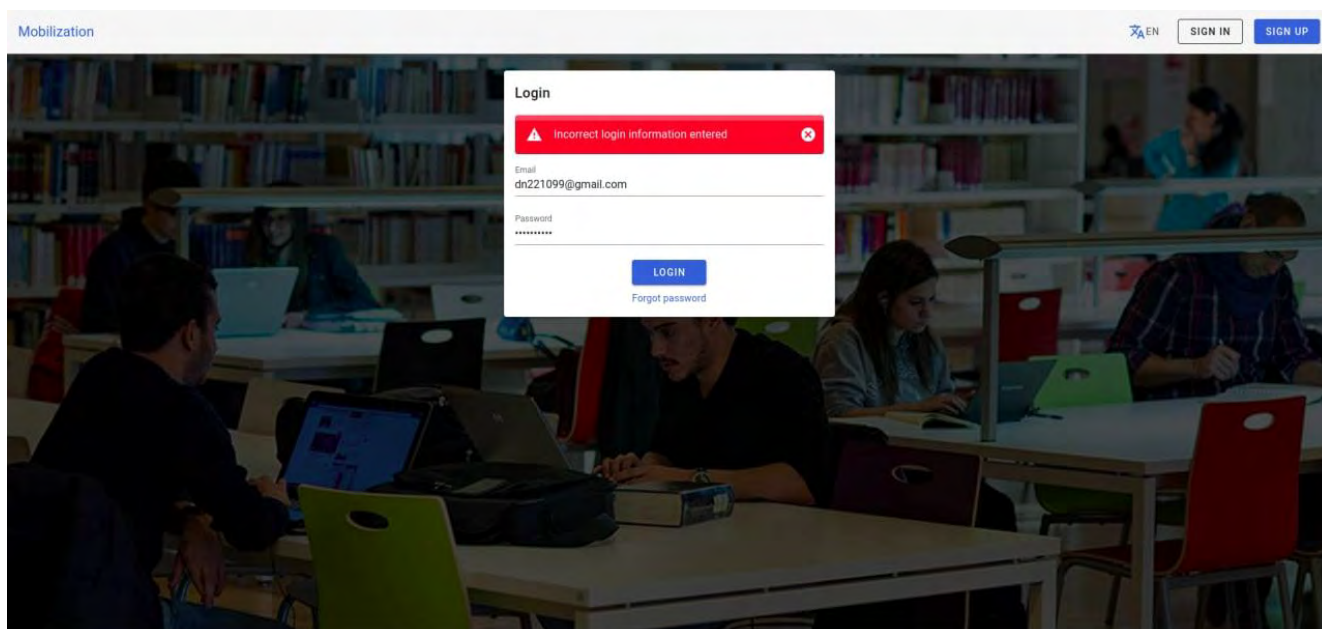


Рисунок 4.4 – Невалідні дані при авторизації

4.4.2 Обробка помилок у середині програми

Якщо при пошуку навчальних закладів нічого не було знайдено, користувач отримує відповідне повідомлення (рис. 4.5).

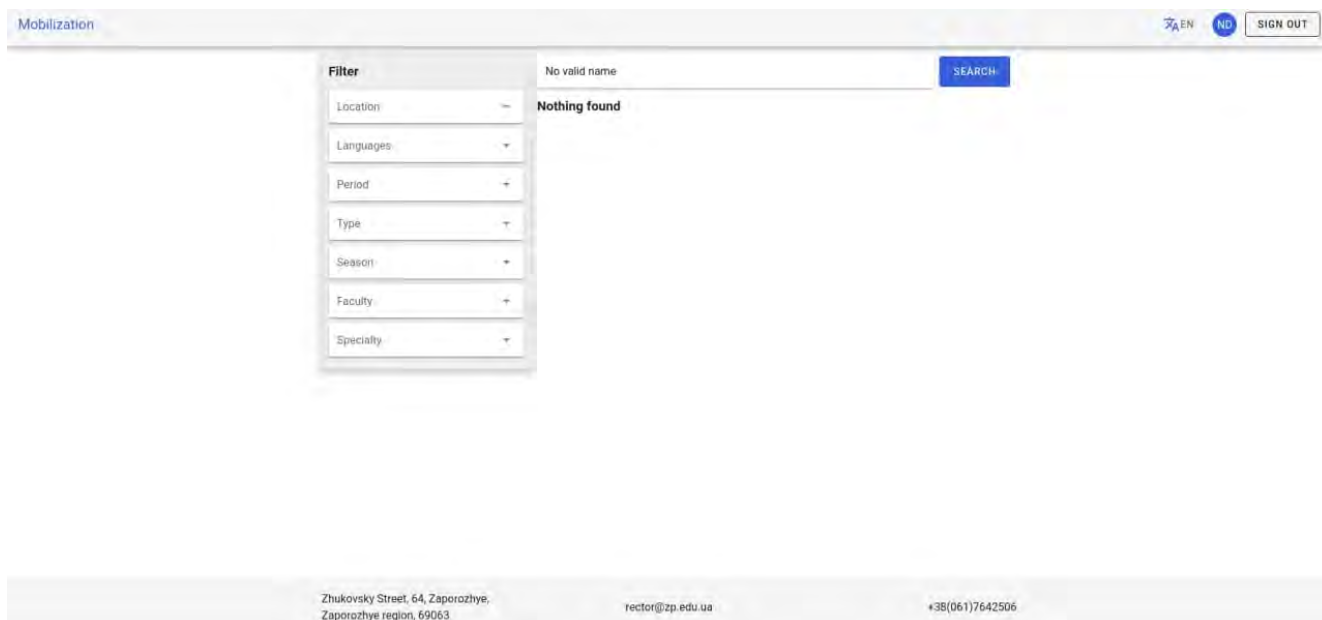


Рисунок 4.5 – Відсутність навчальних закладів за запитом

Якщо користувач ненавмисно змінив URL сайту, то замість порожньої сторінки він отримає повідомлення про помилку (рис. 4.6) та можливість перейти на необхідну сторінку.



Рисунок 4.6 – Помилка «404»

4.5 Висновок до розділу 4

У четвертому розділі було розглянуто основні призначення й умови застосування програми. Були визначені характеристики програми, яка була виконана за допомогою фреймворку Vue.js та інтегрованої середовища програмування Visual Studio Code.

Також були розглянуті виняткові ситуації які можуть виникнути як при заповненні форм, так і у середині програми.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення та умови застосування програми

Розроблений вебзастосунок призначений для забезпечення гнучкого та простого механізму аналізу існуючих можливостей академічної мобільності та подачі заявок на участь.

Програмна система надає користувачеві наступні можливості:

- доступ до головної сторінки з описом вебзастосунку;
- авторизація;
- реєстрація з доданням особистих даних, потрібних для участі у мобільності;
- відновлення пароля;
- редагування налаштувань в особистому акаунті;
- доступ до сторінки зі списком всіх навчальних закладів;
- пошук університетів з фільтрацією (локація, мова, період, тип, сезон, факультет, спеціальність);
- доступ до сторінки окремого навчального закладу;
- можливість порівнювання навчальних планів свого університету та університету-партнера;
- подача заявки на участь у певній програмі мобільності.

5.2 Умови виконання програми

Для користування розробленим веб-сайтом, треба лише мати доступ до інтернету.

5.3 Як запустити програму

Щоб запустити розроблений веб-сайт, потрібно перейти за посиланням на сайт. Користувачеві одразу відкриється головна сторінка сайту.

5.4 Виконання програми

Для запуску програми користувачу (студенту) необхідно перейти на сайт за посиланням. Усі дії виконуються за допомогою клавіатури і комп'ютерної миші сенсорного управління телефону/планшета.

При переході на веб-сайт відкривається головна сторінка (рис. 5.1), яка містить базову інформацію про академічну мобільність, хедер та футер. В хедері є кнопки для переходу на сторінки авторизації та реєстрації, також є кнопка для зміни локалізації (англійська та українська мови).

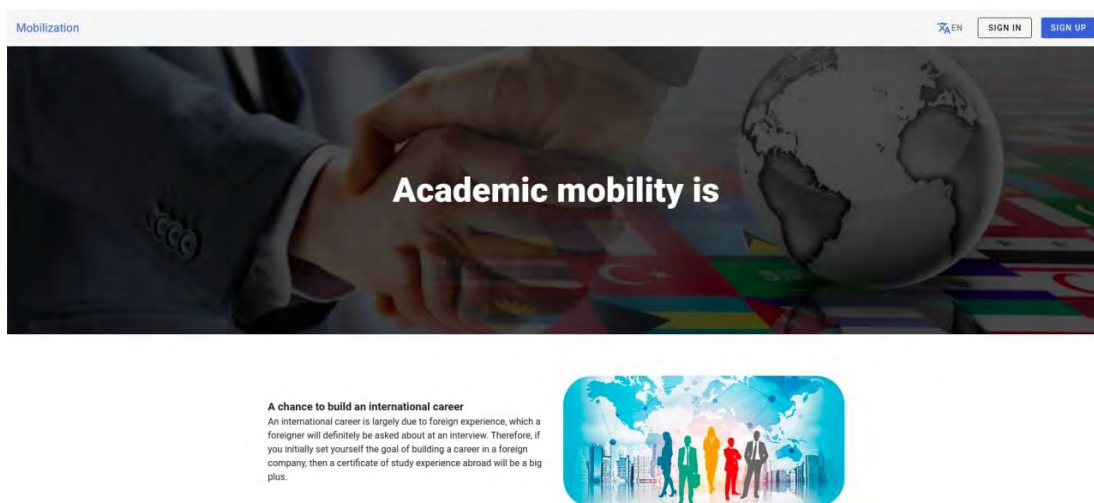


Рисунок 5.1 – Головна сторінка

Для реєстрації (рис. 5.2) на сайті студент повинен одразу ввести свої особисті дані, які будуть зберігатися в створеному акаунті та знадобляться йому при подачі заявок на участь у програмі академічної мобільності. Отже, не треба буде вводити інформацію при подачі кожної заявки. Для завершення реєстрації користувач повинен підтвердити свою електронну пошту (рис. 5.3).

Mobilization

Full name

Email

Birthday

University

Faculty

Specialty

Semester

Password

Confirm password

REGISTRATION

SIGN IN SIGN UP

Рисунок 5.2 – Реєстрація

Welcome to Mobilization

Mobilization <daniil.sokolovskiy@computools.com>

00:39 (0 минут назад)

английский русский Перевести сообщение

Отключить для языка: английский

Verify Your Email

Please verify your email to secure your account

Verify now

Yes, I confirm. Confirmed! I have not received anything yet.

Ответить Переслать

Рисунок 5.3 – Повідомлення про верифікацію

Сторінка для авторизації зображена на рис. 5.4. Якщо користувач вже має акаунт у системі, але забув пароль для входу, то він може відновити його наступним чином: у формі авторизації натиснути на «Forgot password», відкриється сторінка «Forgot password» (рис. 5.5), де потрібно ввести email. Потім на email користувача прийде повідомлення для відновлення пароля (рис. 5.6).

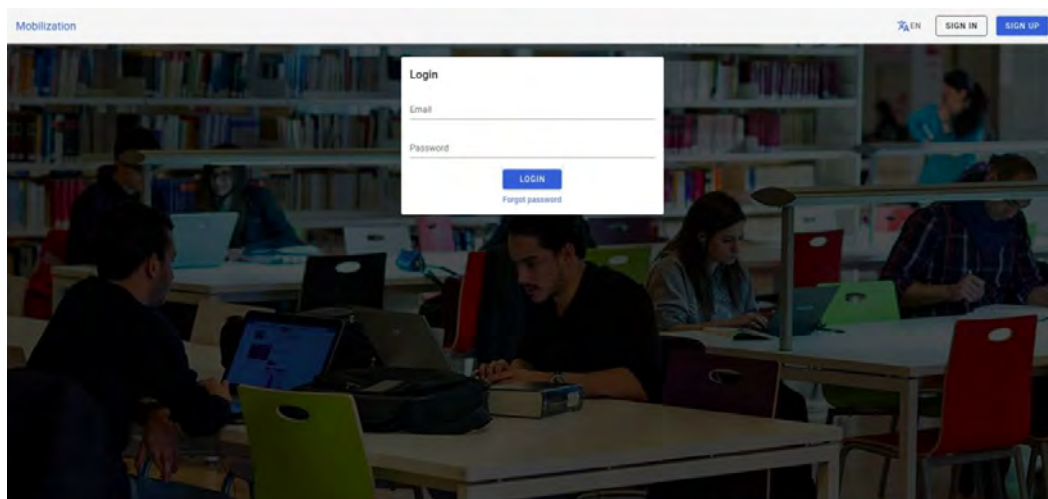


Рисунок 5.4 – Авторизація

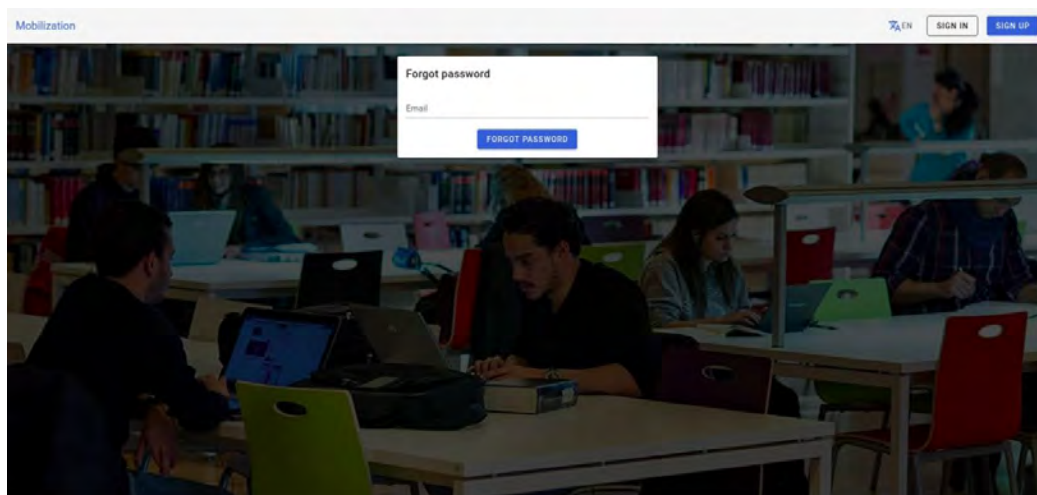


Рисунок 5.5 – Сторінка «Forgot password»



Рисунок 5.6 – Email про відновлення пароля

Після того, як користувач в отриманому повідомленні натисне «Reset your password», він опинеться на сторінці «Restore password» (рис. 5.7), де він може створити новий пароль.

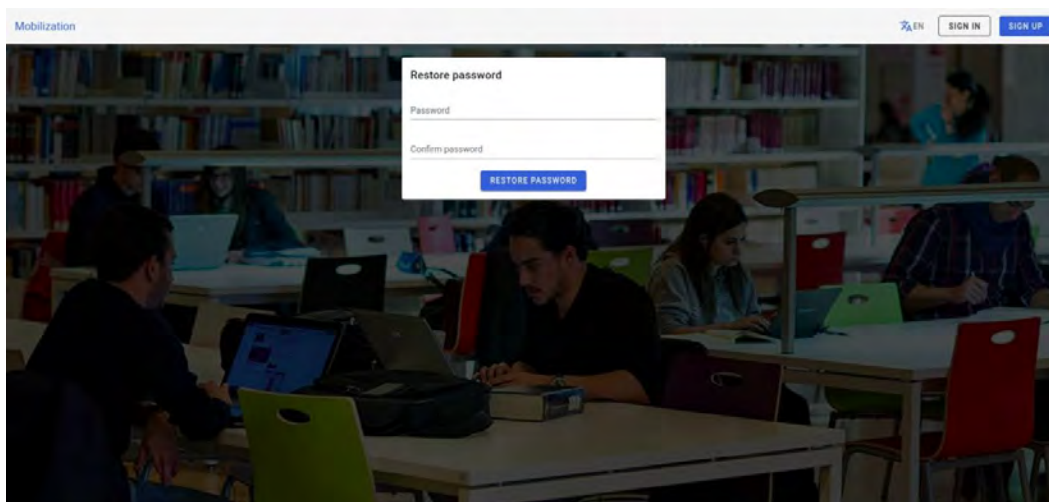


Рисунок 5.7 – Сторінка «Restore password»

Після успішної авторизації користувач переходить у свій акаунт (рис. 5.8). На особистій сторінці є інформація про студента, розділ з його заявками та розділ з налаштуваннями.

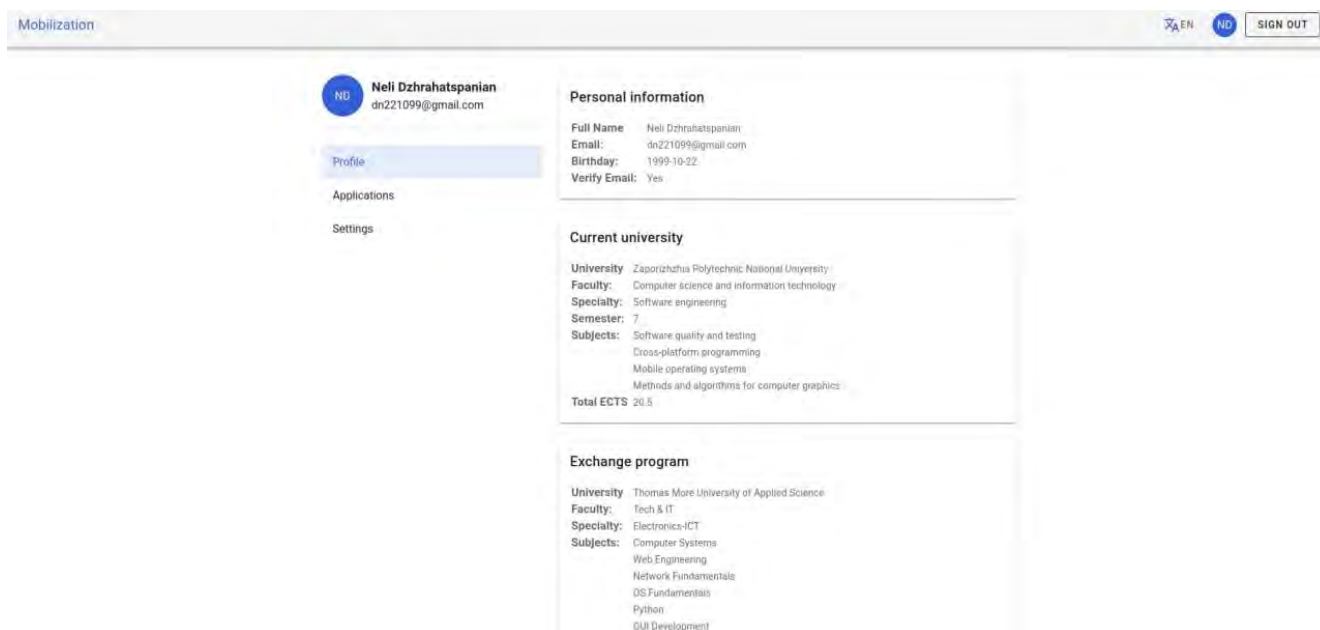


Рисунок 5.8 – Особистий профіль

Користувач за допомогою навігації може перейти до сторінки зі списком університетів (рис. 5.9). На цій сторінці для прискорення пошуку потрібного навчального закладу є пошукове поле та панель з фільтрацією.

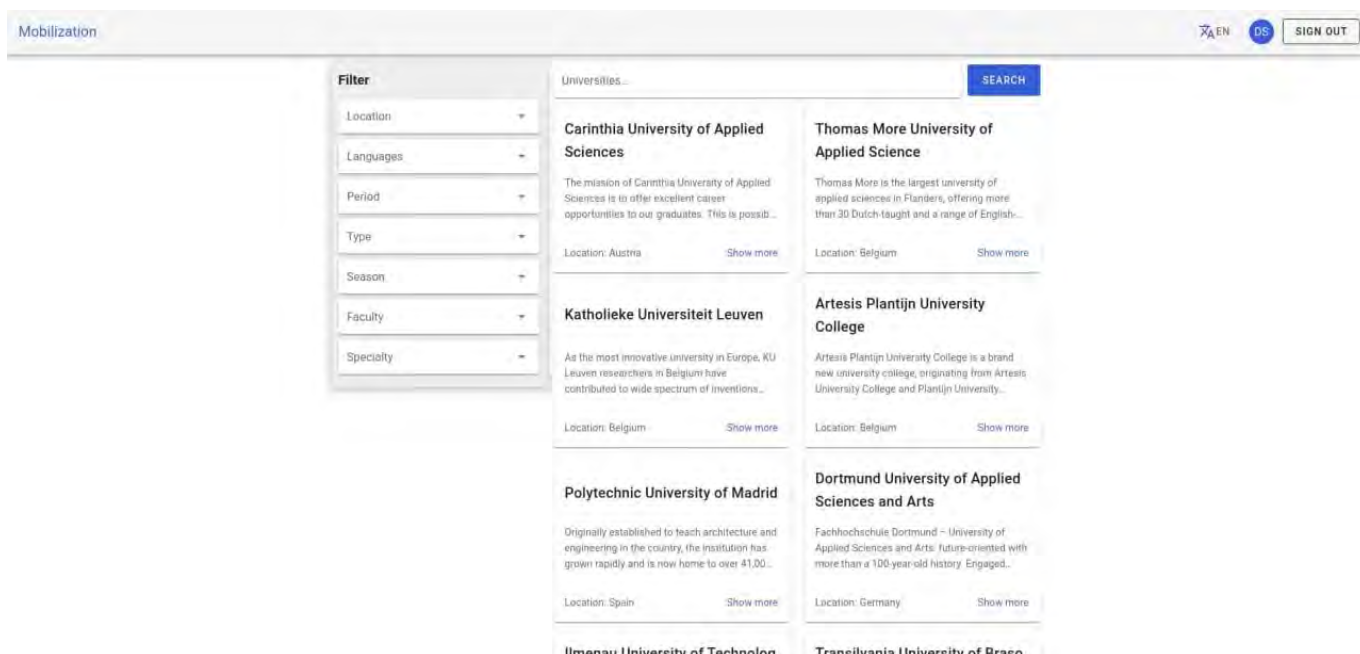


Рисунок 5.9 – Список університетів

Після натискання на обраний університет відкривається сторінка навчального закладу (рис. 5.10), яка містить основну інформацію, а саме: назва, опис, локація, веб-сайт навчального закладу; зліва розташована панель з фільтрацією, де можна обрати факультет та спеціальність; при натисканні на навчальну програму відкриваються предмети, які можна обрати для вивчення під час академічної мобільності.

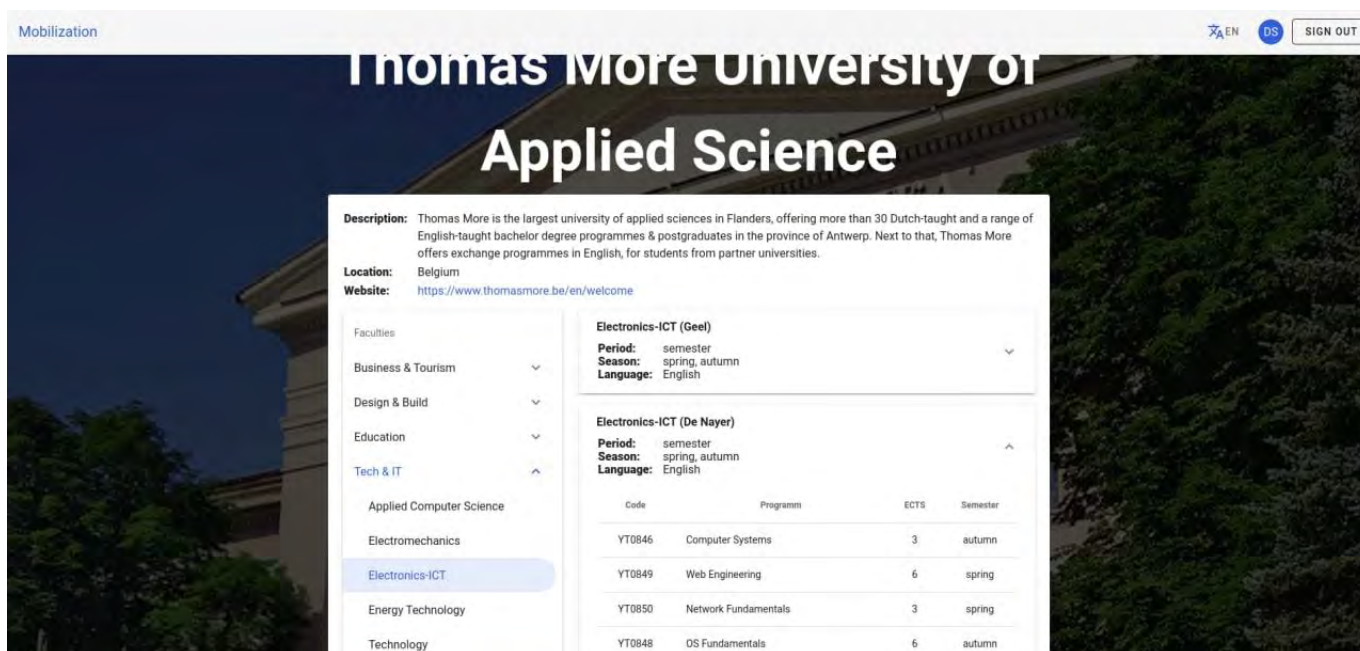


Рисунок 5.10 – Сторінка навчального закладу

Якщо студента зацікавила якась навчальна програма, він може відкрити модальне вікно (рис. 5.11) для порівняння предметів свого навчального закладу та університету-партнера.

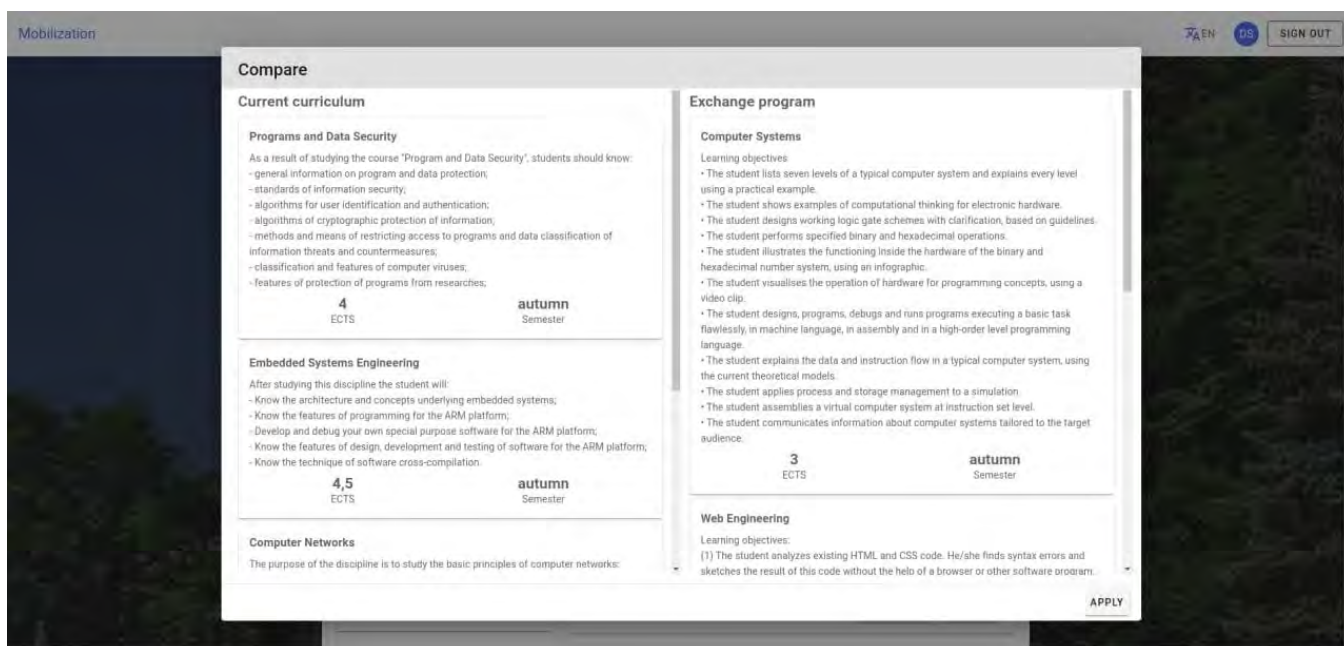


Рисунок 5.11 – Модальне вікно для порівняння предметів

Якщо студент хоче взяти участь у певній мобільності, він має натиснути на кнопку «Apply», після чого він опиниться на сторінці подачі заявки (рис. 5.12), де він повинен обрати предмети, які він хоче вивчати, та прикріпити або написати мотиваційний лист. Також на цій сторінці при виборі нових предметів автоматично підраховується кількість кредитів.

Mobilization

Current Curriculum

Course	Semester	ECTS
Programs and Data Security	autumn	4
Embedded Systems Engineering	autumn	4.5
Computer Networks	autumn	5
Software Design	autumn	4
Project Practical Work	autumn	3

Exchange Program

Course	Semester	ECTS
☒ Computer Systems	autumn	3
☐ Web Engineering	spring	6
☐ Network Fundamentals	spring	3
☒ OS Fundamentals	autumn	6
☒ Python	autumn	6
☐ GUI Development	spring	3

Select the number of ECTS so that the total number of ECTS Exchange Program is greater than or equal to the total number of ECTS Current Curriculum.

20.5 ECTS ≤ 15 ECTS

Motivation Letter

Enter text:

Рисунок 5.12 – Сторінка подачі заявки на участь

5.5 Висновок до розділу 5

Під час виконання кваліфікаційної роботи бакалавра, було розроблено вебзастосунок для формування індивідуальної траєкторії навчання студентів учасників міжнародної мобільності.

Функціонал системи досить широкий. Основними функціями розробленої системи є: присутність основної системи для студентів та окремої адміністративної панелі. Адміністративна панель дозволяє проводити маніпуляції з даними навчальних закладів, факультетів, спеціальностей, навчальних планів, предметів, заявок та також з особистими даними студентів. Студенти мають наступні можливості: реєстрація та авторизація з відновленням пароля при необхідності, редагування налаштувань в особистому акаунті, доступ до сторінки зі списком всіх навчальних закладів, пошук університетів з фільтрацією, можливість порівнювання навчальних планів свого університету та університету-партнера, подача заявки на участь у певній програмі мобільності.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра, було проведено огляд та аналіз існуючих систем для академічної мобільності. Проведений аналіз показав, що майже усі існуючі вебзастосунки для формування індивідуальної траєкторії навчання студентів учасників міжнародної мобільності, мають недоліки, які впливають на швидкість процесу обробки заявок. Тому і було розроблено вебзастосунок, яке усуває основні недоліки існуючих систем.

Складено технічне завдання, в якому зазначена мета розробки та її призначення, основні вимоги до програми, порядок її функціонування.

Розроблена система дозволяє виконувати:

- реєстрацію;
- авторизацію;
- відновлення пароля;
- редагування налаштувань в особистому акаунті;
- доступ до сторінки зі списком всіх навчальних закладів;
- пошук університетів з фільтрацією;
- доступ до сторінки окремого навчального закладу;
- можливість порівнювання навчальних планів;
- подача заявки на участь у мобільності.

А за допомогою розробленої адміністративної панелі можна проводити маніпуляції (створення, видалення та редагування) з даними навчальних закладів, факультетів, спеціальностей, навчальних планів, предметів, заявок (змінювати статус заявок студентів) та також з особистими даними студентів.

Програма була розроблена з використанням сучасних технологій, має гарний і зрозумілий інтерфейс, завдяки якому користуватись системою легко та комфортно.

Система може бути задіяна в будь-якому навчальному закладі, де активно займаються академічною мобільністю. Вебзастосунок повністю дороблено та готове до змін у будь-який час, зручне у використанні та буде зрозумілим для

кожного користувача, навіть якщо у нього немає спеціальних навичок. Програма захищена авторськими правами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гуревич, Р.С. Інформаційні технології навчання: інноваційний підхід / Р.С. Гуревич. – Вінниця: Планер, 2012. – 348 с.
2. Переваги та недоліки Vue.js. [Електрон. ресурс]. – Режим доступу: <https://jetruby.com/ru/blog/vue-js-preimuschestva-i-nedostatki/>.
3. Керівництво Vuex. [Електрон. ресурс]. – Режим доступу: <https://vuex.vuejs.org/ru/guide/>.
4. Visual Studio Code [Electronic resource]. – Access mode: <https://bizzapps.ru/p/vs-code/>.
5. Блок-схема. [Электрон. ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Блок-схема>.

ДОДАТОК А
Текст програми

A.1 Файл /src/components/modal/Compare.vue

```

<template>
  <v-dialog
    v-model="dialog"
    width="1300"
    @click:outside="closeDialog"
    @keydown.esc="closeDialog"
  >
    <v-card class="compare">
      <v-card-title class="headline grey lighten-2">
        {{ $t("compare.title") }}
      </v-card-title>

      <v-card-text class="__content">
        <v-row>
          <v-col>
            <div class="mt-2">
              <h2 class="mb-3">{{ $t("compare.currentCurriculum") }}</h2>
              <v-card
                v-for="syllabus of userSyllabuses"
                :key="syllabus._id"
                class="mb-2"
              >
                <v-card-text>
                  <h3 class="mb-2">{{ syllabus.name[local] }}</h3>
                  <div class="mb-2">
                    <span>{{ syllabus.description[local] }}</span>
                  </div>
                  <div class="d-flex justify-space-around">
                    <div class="text-center">
                      <div>
                        <h2>{{ syllabus.ects }}</h2>
                      </div>
                      <div>
                        <span>{{ $t("compare.ects") }}</span>
                      </div>
                    </div>
                    <div class="text-center">
                      <div>
                        <h2>{{ syllabus.semester[local].join("/") }}</h2>
                      </div>
                      <div>
                        <span>{{ $t("compare.semester") }}</span>
                      </div>
                    </div>
                  </div>
                </v-card-text>
              </v-card>
            </div>
          </v-col>
          <v-col>
            <div class="mt-2">
              <h2 class="mb-3">{{ $t("compare.exchangeProgram") }}</h2>
              <v-card
                v-for="syllabus of syllabuses"
                :key="syllabus._id"
                class="mb-2"
              >
                <v-card-text>
                  <h3 class="mb-2">{{ syllabus.name }}</h3>
                  <div class="mb-2">

```

```

        <span>{{ syllabus.description }}</span>
      </div>
      <div class="d-flex justify-space-around">
        <div class="text-center">
          <div>
            <h2>{{ syllabus.ects }}</h2>
          </div>
          <div>
            <span>{{ $t("compare.ects") }}</span>
          </div>
        </div>
        <div class="text-center">
          <div>
            <h2>{{ syllabus.semester.join("/") }}</h2>
          </div>
          <div>
            <span>{{ $t("compare.semester") }}</span>
          </div>
        </div>
      </div>
    </v-card-text>
  </v-card>
</div>
</v-col>
</v-row>
</v-card-text>

<v-divider></v-divider>

<v-card-actions>
  <v-spacer></v-spacer>
  <v-btn>{{ $t("compare.apply") }}</v-btn>
</v-card-actions>
</v-card>
</v-dialog>
</template>

<script>
export default {
  props: {
    dialog: Boolean,
    closeDialog: Function,
  },
  computed: {
    local() {
      return this.$store.state.locale;
    },
    userSyllabuses() {
      return this.$store.state.user.university.faculty.specialty.curriculum
        .syllabuses;
    },
    syllabuses() {
      return this.$store.state.syllabuses;
    },
  },
};
</script>

<style lang="scss">
.compare {
  .__content {
    overflow: hidden;
    height: 700px;
  }
}

```

```

.col {
  overflow-y: scroll;
  height: 700px;
}
}
}
</style>

```

A.2 Файл /src/components/Select.vue

```

<template>
  <div class="custom-select">
    <v-autocomplete
      solo
      v-model="stateValue"
      :label="label"
      :items="items"
      @change="changeValue"
      :multiple="multiple"
    ></v-autocomplete>
    <v-icon v-if="stateValue" @click="clear" class="icon-close">
      mdi-close
    </v-icon>
  </div>
</template>

<script>
export default {
  name: "Select",
  data: () => ({
    stateValue: "",
  }),
  props: {
    value: {
      type: String,
      default: "",
    },
    items: {
      type: Array,
    },
    label: {
      type: String,
      default: "Select",
    },
    changeValue: {
      type: Function,
    },
    multiple: {
      type: Boolean,
      default: false,
    }
  },
  mounted() {
    this.stateValue = this.value;
  },
  methods: {
    clear() {
      this.changeValue('');
      this.stateValue = '';
    }
  }
}

```

```

    }
  }
};
</script>

<style lang="scss">
.custom-select {
  position: relative;

  .icon-close {
    position: absolute;
    z-index: 1;
    top: 45%;
    right: 12px;
    transform: translateY(-55%);
    background-color: white;
    cursor: pointer;
  }
}
</style>

```

A.3 Файл /src/components/Skeleton.vue

```

<template>
  <v-skeleton-loader
    :style="`border-radius: ${radius};`"
    :height="height"
    :width="width"
  ></v-skeleton-loader>
</template>

<script>
export default {
  name: "Skeleton",
  props: {
    width: {
      default: "100%",
    },
    height: {
      default: "30px",
    },
    radius: {
      default: "5px",
    },
  },
};
</script>

<style lang="scss">
.v-skeleton-loader__bone {
  min-width: 100%;
  min-height: 100%;
  background-color: rgb(248, 248, 248);

  &::after {
    background: rgb(240, 240, 240) !important;
  }
}
</style>

```

A.4 Файл /src/components/SkeletonCard.vue

```

<template>
  <v-card class="skeleton-card">
    <v-card-title><Skeleton width="60%" /></v-card-title>
    <v-card-text>
      <Skeleton class="mb-2" height="15px" />
      <Skeleton class="mb-2" height="15px" />
      <Skeleton height="15px" />
    </v-card-text>
    <v-card-text class="d-flex justify-space-between">
      <Skeleton width="20%" />
      <Skeleton width="20%" />
    </v-card-text>
  </v-card>
</template>

<script>
import Skeleton from "../Skeleton";

export default {
  name: "SkeletonCard",
  components: {
    Skeleton,
  },
};
</script>

<style lang="scss">
.skeleton-card {
  width: 100%;
  max-width: 330px !important;

  @media (max-width: 1024px) {
    max-width: 100% !important;
  }
}
</style>

```

A.5 Файл /src/components/TrasnslationButton.vue

```

<template>
  <v-menu offset-y>
    <template v-slot:activator="{ on, attrs }">
      <v-btn icon v-bind="attrs" v-on="on">
        <v-icon color="blue darken-2"> mdi-translate </v-icon>
        {{ locale }}
      </v-btn>
    </template>
    <v-list>
      <v-list-item class="d-block">
        <v-list-item-title>
          <v-btn @click="() => changeLocal('en')">EN</v-btn>
        </v-list-item-title>
        <v-list-item-title>
          <v-btn @click="() => changeLocal('ua')">UA</v-btn>
        </v-list-item-title>
      </v-list-item>
    </v-list>
  </v-menu>
</template>

```

```

    </v-menu>
  </template>

  <script>
import i18n from "../utils/i18n";

export default {
  name: "TranslationButton",

  computed: {
    locale() {
      return this.$store.state.locale.toUpperCase();
    },
  },

  methods: {
    changeLocal(locale) {
      this.$router.go();
      i18n.locale = locale;
      this.$store.commit("setLocale", locale);
    },
  },
};
</script>

```

A.6 Файл /src/components/UniversityCard.vue

```

<template>
  <v-card class="university-card">
    <v-card-title>{{ item.name }}</v-card-title>
    <v-card-text class="__description">{{ item.description }}</v-card-text>
    <v-card-text class="d-flex justify-space-between">
      <div>{{ $t("filter.location") + ": " + item.location }}</div>
      <div><router-link :to="'/university/${item.id}'">{{ $t('button.showMore') }}</router-link></div>
    </v-card-text>
  </v-card>
</template>

<script>
export default {
  name: "UniversityCard",
  props: ["item"],
};
</script>

<style lang="scss">
.university-card {
  width: 100%;
  max-width: 330px !important;

  .__description {
    padding-bottom: 0 !important;
    margin-bottom: 16px !important;

    overflow: hidden;
    text-overflow: ellipsis;
    display: -webkit-box;
    -webkit-box-orient: vertical;
    -webkit-line-clamp: 3;
  }
}

```

```

}

@media (max-width: 1024px) {
  max-width: 100% !important;
}
}
</style>

```

A.7 Файл /src/components/Index.vue

```

<template>
  <div>
    <v-card>
      <v-card-title>Personal information</v-card-title>
      <v-card-text>
        <table>
          <tbody>
            <tr>
              <td><h3>Full Name</h3></td>
              <td>{{ user.fullName }}</td>
            </tr>
            <tr>
              <td><h3>Email:</h3></td>
              <td>{{ user.email }}</td>
            </tr>
            <tr>
              <td><h3>Birthday:</h3></td>
              <td>{{ getBirthdayFormat(user.birthday) }}</td>
            </tr>
            <tr>
              <td class="pr-3"><h3>Verify Email:</h3></td>
              <td>
                {{ user.isEmailVerified ? "Yes" : "No" }}
              </td>
            </tr>
          </tbody>
        </table>
      </v-card-text>
    </v-card>

    <v-card class="mt-6">
      <v-card-title>Current university</v-card-title>
      <v-card-text>
        <table>
          <tbody>
            <tr>
              <td class="pr-3"><h3>University</h3></td>
              <td>{{ user.university.name.en }}</td>
            </tr>
            <tr>
              <td><h3>Faculty:</h3></td>
              <td>{{ user.university.faculty.name.en }}</td>
            </tr>
            <tr>
              <td><h3>Specialty:</h3></td>
              <td>{{ user.university.faculty.specialty.name.en }}</td>
            </tr>
            <tr>
              <td><h3>Semester:</h3></td>
              <td>

```

```

        {{ user.university.faculty.specialty.curriculum.semester }}
      </td>
    </tr>
    <tr>
      <td><h3>Subjects:</h3></td>
      <td>
        {{
          user.university.faculty.specialty.curriculum.syllabuses[0]
            .name.en
        }}
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        {{
          user.university.faculty.specialty.curriculum.syllabuses[1]
            .name.en
        }}
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        {{
          user.university.faculty.specialty.curriculum.syllabuses[2]
            .name.en
        }}
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        {{
          user.university.faculty.specialty.curriculum.syllabuses[3]
            .name.en
        }}
      </td>
    </tr>
    <tr>
      <td><h3>Total ECTS</h3></td>
      <td>20.5</td>
    </tr>
  </tbody>
</table>
</v-card-text>
</v-card>

<v-card class="mt-6">
  <v-card-title>Exchange program</v-card-title>
  <v-card-text>
    <table>
      <tbody>
        <tr>
          <td class="pr-3"><h3>University</h3></td>
          <td>Thomas More University of Applied Science</td>
        </tr>
        <tr>
          <td><h3>Faculty:</h3></td>
          <td>Tech & IT</td>
        </tr>
        <tr>
          <td><h3>Specialty:</h3></td>
          <td>Electronics-ICT</td>
        </tr>
      </tbody>
    </table>
  </v-card-text>
</v-card>

```

```

    </tr>
    <tr>
      <td><h3>Subjects:</h3></td>
      <td>Computer Systems</td>
    </tr>
    <tr>
      <td></td>
      <td>Web Engineering</td>
    </tr>
    <tr>
      <td></td>
      <td>Network Fundamentals</td>
    </tr>
    <tr>
      <td></td>
      <td>OS Fundamentals</td>
    </tr>
    <tr>
      <td></td>
      <td>Python</td>
    </tr>
    <tr>
      <td></td>
      <td>GUI Development</td>
    </tr>
  </tbody>
</table>
</v-card-text>
</v-card>
</div>
</template>

<script>
import moment from "moment";

export default {
  name: "ProfileIndex",
  computed: {
    user() {
      return this.$store.state.user;
    },
  },
  methods: {
    getBirthdayFormat(date) {
      return moment(date).format("YYYY-MM-DD");
    },
  },
};
</script>

```

A.8 Файл /src/data-layer/HttpClient.js

```

import axios from "axios";
import set from "lodash/set";
import { stringify } from "../utils/qs";

export default class HttpClient {
  constructor({ language }) {
    const axiosConfig = {
      baseURL: "http://localhost:3000/api",

```

```

};

this.transport = axios.create(axiosConfig);

this.transport.interceptors.request.use(
  (config) => {
    set(config, "headers.content-language", language);

    return config;
  },
  (error) => Promise.reject(error)
);
}

getUniversity = (query = {}) =>
  this.transport.get(`/university?${stringify(query)}`);

getUniversityById = (id) => this.transport.get(`/university/${id}`);

getFacultyOfUniversity = (universityId, query = {}) =>
  this.transport.get(
    `/university/${universityId}/faculty?${stringify(query)}`
  );

getSpecialtyOfUniversityAndFaculty = (universityId, facultyId, query = {}) =>
  this.transport.get(
    `/university/${universityId}/faculty/${facultyId}/specialty?${stringify(
      query
    )}`
  );

getSemesterOfUFS = (universityId, facultyId, specialtyId, query = {}) =>
  this.transport.get(
    `/university/${universityId}/faculty/${facultyId}/specialty/${specialtyId}/c
urriculum?${stringify(
      query
    )}`
  );

getVerify = (token) => this.transport.put(`/auth/verify/${token}`);

signUp = (body) => this.transport.post("/auth/register", body);

signIn = (body) => this.transport.post("/auth/login", body);

forgot = (body) => this.transport.post("/auth/forgot", body);

reset = (token, body) => this.transport.post(`/auth/reset/${token}`, body);

getFilterFieldsForUniversityOfMobility = () =>
  this.transport.get("/filter-fields/university");

getUniversityOfMobility = (query = {}) =>
  this.transport.get(`/university-of-mobility?${stringify(query)}`);
}

```

A.9 Файл /src/pages/Application.vue

```

<template>
  <v-main>
    <v-container> </v-container>

```

```

    </v-main>
  </template>

  <script>
  export default {
    name: "Application",
    computed: {
      syllabuses() {
        return this.$store.state.syllabuses;
      },
    },
  };
  </script>

```

A.10 Файл /src/pages/Forgot.vue

```

<template>
  <v-main>
    <v-img
      src="../../../public/images/sign-in-bg.jpg"
      height="calc(100vh - 76px)"
      gradient="to top, rgba(0, 0, 0, .7), rgba(0, 0, 0, .7)"
    >
    <v-container>
      <v-col
        xl="6"
        offset-xl="3"
        lg="6"
        offset-lg="3"
        md="6"
        offset-md="3"
        sm="8"
        offset-sm="2"
      >
        <v-card>
          <v-card-title>{{ $t("forgot.title") }}</v-card-title>
          <v-card-text>
            <v-alert
              v-model="alert"
              border="top"
              color="red"
              dismissible
              type="error"
            >
              {{ errorMessage }}
            </v-alert>
            <v-form ref="form">
              <v-text-field
                :label="$t('form.email')"
                v-model="dataForm.email"
                :rules="[validation.required, validation.email]"
              />
              <v-main class="text-center">
                <v-btn
                  :loading="isLoading.submit"
                  color="primary"
                  @click="submit"
                >
                  {{ $t("forgot.title") }}
                </v-btn>

```

```

        </v-main>
      </v-form>
    </v-card-text>
  </v-card>
</v-col>
<v-dialog v-model="dialog" persistent max-width="500">
  <v-card>
    <v-card-title class="text-center">{{
      $t("forgot.dialogTitle")
    }}</v-card-title>
    <v-card-actions class="justify-center">
      <v-btn @click="closeDialog">{{ $t("button.home") }}</v-btn>
    </v-card-actions>
  </v-card>
</v-dialog>
</v-container>
</v-img>
</v-main>
</template>

```

```

<script>
import { required, email } from "../utils/validation";
import { Service } from "../services/httpClient";

export default {
  name: "Forgot",

  data: () => ({
    alert: false,
    dialog: false,

    errorMessages: "",

    dataForm: {
      email: "",
    },

    isLoading: {
      submit: false,
    },

    validation: {
      required,
      email,
    },
  }),

  methods: {
    submit() {
      this.alert = false;
      const isValid = this.$refs.form.validate();

      if (isValid) {
        this.isLoading.submit = true;

        Service.forgot(this.dataForm)
          .then(() => {
            this.dialog = true;
          })
          .catch((err) => {
            this.alert = true;
            const data = err.response.data;
            this.errorMessages = data.message;
          })
      }
    }
  }
}

```

```

        .finally(() => {
            this.isLoading.submit = false;
        });
    },
    },
    closeDialog() {
        this.dialog = false;
        this.$router.push("/");
    },
    },
};
</script>

<style lang="scss">
.v-card__title {
    word-break: normal;
}
</style>

```

A.11 Файл /src/pages/Home.vue

```

<template>
  <v-main class="home">
    <v-img
      src="../../../public/images/hm-top.jpg"
      height="500px"
      gradient="to top, rgba(0, 0, 0, .7), rgba(0, 0, 0, .7)"
      class="main-img"
    >
    <div class="__content">
      <span>{{ $t("homepage.title") }}</span>
    </div>
  </v-img>
  <v-container>
    <div class="content">
      <v-row class="align-center">
        <v-col md="6">
          <h3>{{ $t("homepage.content[0].title") }}</h3>
          <span>
            {{ $t("homepage.content[0].text") }}
          </span>
        </v-col>
        <v-col md="6">
          <v-img src="../../../public/images/hm-content-1.png" class="__img" />
        </v-col>
      </v-row>
      <v-row class="mt-12 align-center">
        <v-col md="6">
          <v-img src="../../../public/images/hm-content-2.jpg" class="__img" />
        </v-col>
        <v-col md="6">
          <h3>{{ $t("homepage.content[1].title") }}</h3>
          <span>
            {{ $t("homepage.content[1].text") }}
          </span>
        </v-col>
      </v-row>
      <v-row class="mt-12 align-center">
        <v-col md="6">

```

```

        <h3>{{ $t("homepage.content[2].title") }}</h3>
        <span>
            {{ $t("homepage.content[2].text") }}
        </span>
    </v-col>
    <v-col md="6">
        <v-img src="../../../public/images/hm-content-3.jpg" class="__img" />
    </v-col>
</v-row>
<v-row class="mt-12 align-center">
    <v-col md="6">
        <v-img src="../../../public/images/hm-content-4.jpg" class="__img" />
    </v-col>
    <v-col md="6">
        <h3>{{ $t("homepage.content[3].title") }}</h3>
        <span>
            {{ $t("homepage.content[3].text") }}
        </span>
    </v-col>
</v-row>
</div>
</v-container>
</v-main>
</template>

<script>
export default {
    name: "Home",
};
</script>

<style lang="scss">
.home {
    .main-img {
        position: relative;

        .__content {
            position: absolute;
            top: 50%;
            left: 50%;
            transform: translate(-50%, -50%);

            span {
                color: white;
                font-size: 60px;
                font-weight: 800;
            }
        }
    }
}

.content {
    margin: 60px 0;

    .__img {
        border-radius: 50px;
    }
}
</style>

```

A.12 Файл /src/pages/NotFound.vue

```
<template>
  <v-main>
    <v-container class="text-center">
      <h1 style="font-size: 8rem">404</h1>
      <router-link to="/">{{ $t("button.home") }}</router-link>
    </v-container>
  </v-main>
</template>

<script>
export default {
  name: "NotFound",
};
</script>
```

A.13 Файл /src/pages/Reset.vue

```
<template>
  <v-main>
    <v-img
      src="../../../public/images/
sign-in-bg.jpg"
      height="calc(100vh - 76px)"
      gradient="to top,
rgba(0, 0, 0, .7),
rgba(0, 0, 0, .7)"
    >
    <v-container>
      <v-col
        xl="6"
        offset-xl="3"
        lg="6"
        offset-lg="3"
        md="6"
        offset-md="3"
        sm="8"
        offset-sm="2"
      >
        <v-card>
          <v-card-title>{{ $t("reset.title") }}</v-card-title>
          <v-card-text>
            <v-form ref="form">
              <v-text-field
                :label="$t('form.password')"
                type="password"
                v-model="dataForm.password"
                :rules="[
                  validation.required,
                  validation.min(6),
                  validation.max(255),
                ]"
              />
              <v-text-field
                :label="$t('form.confirmPassword')"
                type="password"
                v-model="dataForm.confirmPassword"
```

```

        :rules="[
          validation.required,
          validation.confirm(dataForm.password),
        ]"
      />
      <v-main class="text-center">
        <v-btn
          :loading="isLoading.submit"
          color="primary"
          @click="submit"
        >
          {{ $t("reset.title") }}
        </v-btn>
      </v-main>
    </v-form>
  </v-card-text>
</v-card>
</v-col>
<v-dialog v-model="dialog" persistent max-width="500">
  <v-card>
    <v-card-title class="justify-center">
      {{ $t("reset.dialogTitle") }}
    </v-card-title>
    <v-card-actions class="justify-center">
      <v-btn @click="closeDialog">{{ $t("button.home") }}</v-btn>
    </v-card-actions>
  </v-card>
</v-dialog>
</v-container>
</v-img>
</v-main>
</template>

<script>
import { Service } from "../services/httpClient";
import { required, min, max, confirm } from "../utils/validation";

export default {
  name: "Reset",

  data: () => ({
    dialog: false,

    dataForm: {
      password: "",
      confirmPassword: "",
    },

    isLoading: {
      submit: false,
    },

    validation: {
      required,
      min,
      max,
      confirm,
    },
  }),

  methods: {
    submit() {
      const isValid = this.$refs.form.validate();

```

```

    if (isValid) {
      this.isLoading.submit = true;

      Service.reset(this.$route.params.token, this.dataForm)
        .then(() => {
          this.dialog = true;
        })
        .catch(() => {
          this.$router.push("/404");
        })
        .finally(() => {
          this.isLoading.submit = false;
        });
    }
  },

  closeDialog() {
    this.dialog = false;
    this.$router.push("/");
  },
};
</script>

```

A.14 Файл /src/pages/SignIn.vue

```

<template>
  <v-main>
    <v-img
      src="../../../public/images/sign-in-bg.jpg"
      height="calc(100vh - 76px)"
      gradient="to top, rgba(0, 0, 0, .7), rgba(0, 0, 0, .7)"
    >
    <v-container>
      <v-col
        xl="6"
        offset-xl="3"
        lg="6"
        offset-lg="3"
        md="6"
        offset-md="3"
        sm="8"
        offset-sm="2"
      >
        <v-card>
          <v-card-title>{{ $t("signIn.title") }}</v-card-title>
          <v-card-text>
            <v-alert
              v-model="alert"
              border="top"
              color="red"
              dismissible
              type="error"
            >
              {{ errorMessages }}
            </v-alert>
            <v-form ref="form">
              <v-text-field
                :label="$t('form.email')"
                v-model="dataForm.email"
                :rules="[validation.required, validation.email]"
              >

```

```

    />
    <v-text-field
      :label="$t('form.password')"
      type="password"
      v-model="dataForm.password"
      :rules="[
        validation.required,
        validation.min(6),
        validation.max(255),
      ]"
    />
  </v-main>
  <div class="main__wrap">
    <v-btn
      :loading="isLoading.submit"
      color="primary"
      @click="submit"
    >
      {{ $t("signIn.title") }}
    </v-btn>
    <router-link to="/forgot" class="mt-2">
      {{ $t("forgot.title") }}
    </router-link>
  </div>
</v-main>
</v-form>
</v-card-text>
</v-card>
</v-col>
</v-container>
</v-img>
</v-main>
</template>

<script>
import { required, min, max, email } from "../utils/validation";
import { Service } from "../services/httpClient";

export default {
  name: "SignIn",

  data: () => ({
    alert: false,

    errorMessages: "",

    dataForm: {
      email: "",
      password: "",
    },

    isLoading: {
      submit: false,
    },

    validation: {
      required,
      min,
      max,
      email,
    },
  }),

  methods: {

```

```

submit() {
  this.alert = false;
  const isValid = this.$refs.form.validate();

  if (isValid) {
    this.isLoading.submit = true;

    Service.signIn(this.dataForm)
      .then(({ data }) => {
        this.$store.commit("setJwtToken", data.token);
        this.$store.commit("setUser", data.user);
        this.$router.push("/");
      })
      .catch((err) => {
        this.alert = true;
        const data = err.response.data;
        this.errorMessages = data.message;
      })
      .finally(() => {
        this.isLoading.submit = false;
      });
  }
},
};
</script>

<style lang="scss">
.main__wrap {
  display: inline-grid;
  flex-direction: column;
  justify-content: center;
  width: 100%;
}
</style>

```

A.15 Файл /src/pages/SignUn.vue

```

<template>
  <v-main>
    <v-img
      src="../../../public/images/sign-up-bg.jpg"
      height="calc(100vh - 76px)"
      gradient="to top, rgba(0, 0, 0, .7), rgba(0, 0, 0, .7)"
    >
    <v-container>
      <v-col
        xl="6"
        offset-xl="3"
        lg="6"
        offset-lg="3"
        md="6"
        offset-md="3"
        sm="8"
        offset-sm="2"
      >
        <v-card>
          <v-card-title>{{ $t("signUp.title") }}</v-card-title>
          <v-card-text>
            <v-form ref="form">
              <v-text-field
                :label="$t('form.fullName')"

```

```

    v-model="dataForm.fullName"
    :rules="[
      validation.required,
      validation.min(5),
      validation.max(255),
    ]"
  />
<v-text-field
  :label="$t('form.email')"
  v-model="dataForm.email"
  :error-messages="errorMessages.email"
  :rules="[validation.required, validation.email]"
/>
<v-menu
  ref="menuDatePicker"
  v-model="menuDatePicker"
  min-width="auto"
  offset-y
  transition="scale-transition"
  :close-on-content-click="false"
>
  <template v-slot:activator="{ on, attrs }">
    <v-text-field
      :label="$t('form.birthday')"
      v-model="dataForm.birthday"
      readonly
      v-bind="attrs"
      v-on="on"
      :rules="[validation.required, validation.date]"
    />
  </template>
  <v-date-picker
    ref="picker"
    v-model="dataForm.birthday"
    :locale="$i18n.locale"
    :max="maxDate"
    min="1950-01-01"
    @change="saveBirthday"
  />
</v-menu>
<v-autocomplete
  v-model="dataForm.universityId"
  :label="$t('form.university')"
  :loading="isLoading.university"
  :search-input.sync="search.university"
  :items="items.university"
  hide-no-data
  item-text="name"
  item-value="id"
  :rules="[validation.required]"
/>
<v-autocomplete
  :disabled="!dataForm.universityId"
  v-model="dataForm.facultyId"
  :label="$t('form.faculty')"
  :loading="isLoading.faculty"
  :search-input.sync="search.faculty"
  :items="items.faculty"
  hide-no-data
  item-text="name"
  item-value="id"
  :rules="[validation.required]"
/>
<v-autocomplete

```

```

      :disabled="!dataForm.facultyId"
      v-model="dataForm.specialtyId"
      :label="$t('form.specialty')"
      :loading="isLoading.specialty"
      :search-input.sync="search.specialty"
      :items="items.specialty"
      hide-no-data
      item-text="name"
      item-value="id"
      :rules="[validation.required]"
    />
    <v-select
      :disabled="!dataForm.specialtyId"
      v-model="dataForm.semester"
      :label="$t('form.semester')"
      :loading="isLoading.semester"
      :items="items.semester"
      :rules="[validation.required]"
    />
    <v-text-field
      :label="$t('form.password')"
      type="password"
      v-model="dataForm.password"
      :rules="[
        validation.required,
        validation.min(6),
        validation.max(255),
      ]"
    />
    <v-text-field
      :label="$t('form.confirmPassword')"
      type="password"
      v-model="dataForm.confirmPassword"
      :rules="[
        validation.required,
        validation.confirm(dataForm.password),
      ]"
    />
    <v-main class="text-center">
      <v-btn
        :loading="isLoading.submit"
        color="primary"
        @click="submit"
      >
        {{ $t("signUp.title") }}
      </v-btn>
    </v-main>
  </v-form>
</v-card-text>
</v-card>
</v-col>
<v-dialog v-model="dialog" persistent max-width="500">
  <v-card>
    <v-card-title class="justify-center">{{
      $t("signUp.dialogTitle")
    }}</v-card-title>
    <v-card-text class="text-center">{{
      $t("signUp.dialogText")
    }}</v-card-text>
    <v-card-actions class="justify-center">
      <v-btn @click="closeDialog">{{ $t("button.home") }}</v-btn>
    </v-card-actions>
  </v-card>
</v-dialog>

```

```

    </v-container>
  </v-img>
</v-main>
</template>

<script>
import moment from "moment";
import { required, min, max, email, date, confirm } from "../utils/validation";
import { Service } from "../services/httpClient";

export default {
  name: "SignUp",

  data: () => ({
    menuDatePicker: false,

    dialog: false,

    dataForm: {
      fullName: "",
      email: "",
      birthday: "",
      universityId: "",
      facultyId: "",
      specialtyId: "",
      semester: "",
      password: "",
      confirmPassword: "",
    },

    errorMessages: {},

    isLoading: {
      university: false,
      faculty: false,
      specialty: false,
      semester: false,
      submit: false,
    },

    items: {
      university: [],
      faculty: [],
      specialty: [],
      semester: [],
    },

    search: {
      university: null,
      faculty: null,
      specialty: null,
    },

    validation: {
      required,
      min,
      max,
      email,
      date,
      confirm,
    },
  }),
  computed: {
    maxDate() {

```

```

    return moment().subtract(16, "years").format("YYYY-MM-DD");
  },
},
watch: {
  menuDatePicker(value) {
    value && setTimeout(() => (this.$refs.picker.activePicker = "YEAR"));
  },
  "dataForm.specialtyId"(value) {
    this.isLoading.semester = true;

    Service.getSemesterOfUFS(
      this.dataForm.universityId,
      this.dataForm.facultyId,
      value
    )
      .then(({ data }) => {
        this.items.semester = data.map((e) => e.semester);
      })
      .catch((err) => {
        console.log(err);
      })
      .finally(() => {
        this.isLoading.semester = false;
      });
  },
  "search.university"(value) {
    if (this.isLoading.university) return;

    this.isLoading.university = true;

    Service.getUniversity({ search: value })
      .then(({ data }) => {
        this.items.university = [...data];
      })
      .catch((err) => {
        console.log(err);
      })
      .finally(() => {
        this.isLoading.university = false;
      });
  },
  "search.faculty"(value) {
    if (this.isLoading.faculty) return;

    this.isLoading.faculty = true;

    Service.getFacultyOfUniversity(this.dataForm.universityId, {
      search: value,
    })
      .then(({ data }) => {
        this.items.faculty = [...data];
      })
      .catch((err) => {
        console.log(err);
      })
      .finally(() => {
        this.isLoading.faculty = false;
      });
  },
  "search.specialty"(value) {

```

```

    if (this.isLoading.specialty) return;

    this.isLoading.specialty = true;

    Service.getSpecialtyOfUniversityAndFaculty(
      this.dataForm.universityId,
      this.dataForm.facultyId,
      {
        search: value,
      }
    )
      .then(({ data }) => {
        this.items.specialty = [...data];
      })
      .catch((err) => {
        console.log(err);
      })
      .finally(() => {
        this.isLoading.specialty = false;
      });
  },
},
methods: {
  saveBirthday(date) {
    this.$refs.menuDatePicker.save(date);
  },

  submit() {
    this.errorMessages = {};
    const isValid = this.$refs.form.validate();

    if (isValid) {
      this.isLoading.submit = true;

      Service.signUp(this.dataForm)
        .then(() => {
          this.dialog = true;
        })
        .catch((err) => {
          const data = err.response.data;
          this.errorMessages = { ...data };
        })
        .finally(() => {
          this.isLoading.submit = false;
        });
    }
  },

  closeDialog() {
    this.dialog = false;
    this.$router.push("/");
  },
},
};
</script>

```

A.16 Файл /src/pages/Universities.vue

```

<template>
  <v-main>
    <v-container>
      <div class="d-sm-flex universities">
        <div class="__fields-filter">
          <h3 class="mb-3">{{ $t("filter.title") }}</h3>
          <template v-if="isLoading.filter">
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" />
          </template>
          <template v-else>
            <Select
              :value="query.location"
              :items="filter.location"
              :label="$t('filter.location')"
              :changeValue="
                (value) => {
                  query.location = value;
                }
              "
            />
            <Select
              :value="query.language"
              :items="filter.language"
              :label="$t('filter.language')"
              :changeValue="
                (value) => {
                  query.language = value;
                }
              "
            />
            <Select
              :value="query.period"
              :items="filter.period"
              :label="$t('filter.period')"
              :changeValue="
                (value) => {
                  query.period = value;
                }
              "
            />
            <Select
              :value="query.type"
              :items="filter.type"
              :label="$t('filter.type')"
              :changeValue="
                (value) => {
                  query.type = value;
                }
              "
            />
            <Select
              :value="query.season"
              :items="filter.season"
              :label="$t('filter.season')"

```

```

      :changeValue="
        (value) => {
          query.season = value;
        }
      "
    />
    <Select
      :value="query.faculty"
      :items="filter.faculty"
      :label="$t('filter.faculty')"
      :changeValue="
        (value) => {
          query.faculty = value;
        }
      "
    />
    <Select
      :value="query.specialty"
      :items="filter.specialty"
      :label="$t('filter.specialty')"
      :changeValue="
        (value) => {
          query.specialty = value;
        }
      "
    />
  </template>
</div>
<div class="__content">
  <div class="d-flex align-center mb-3">
    <v-text-field
      class="__search"
      v-model="query.search"
      :label="$t('search.university')"
      solo
    ></v-text-field>
    <v-btn class="ml-2" depressed color="primary" large @click="search">
      {{ $t("search.title") }}
    </v-btn>
  </div>
  <div class="d-sm-none __mob-fields-filter">
    <v-expansion-panels>
      <v-expansion-panel>
        <v-expansion-panel-content>
          <h3 class="mb-3">{{ $t("filter.title") }}</h3>
          <template v-if="isLoading.filter">
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" class="mb-2" />
            <Skeleton height="48px" />
          </template>
          <template v-else>
            <Select
              :value="query.location"
              :items="filter.location"
              :label="$t('filter.location')"
              :changeValue="
                (value) => {
                  query.location = value;
                }
              "
            />
          </template>
        </v-expansion-panel-content>
      </v-expansion-panel>
    </v-expansion-panels>
  </div>

```

```

/>
<Select
  :value="query.language"
  :items="filter.language"
  :label="$t('filter.language')"
  :changeValue="
    (value) => {
      query.language = value;
    }
  "
/>
<Select
  :value="query.period"
  :items="filter.period"
  :label="$t('filter.period')"
  :changeValue="
    (value) => {
      query.period = value;
    }
  "
/>
<Select
  :value="query.type"
  :items="filter.type"
  :label="$t('filter.type')"
  :changeValue="
    (value) => {
      query.type = value;
    }
  "
/>
<Select
  :value="query.season"
  :items="filter.season"
  :label="$t('filter.season')"
  :changeValue="
    (value) => {
      query.season = value;
    }
  "
/>
<Select
  :value="query.faculty"
  :items="filter.faculty"
  :label="$t('filter.faculty')"
  :changeValue="
    (value) => {
      query.faculty = value;
    }
  "
/>
<Select
  :value="query.specialty"
  :items="filter.specialty"
  :label="$t('filter.specialty')"
  :changeValue="
    (value) => {
      query.specialty = value;
    }
  "
/>
</template>
</v-expansion-panel-content>
<v-expansion-panel-header></v-expansion-panel-header>

```

```

        </v-expansion-panel>
      </v-expansion-panels>
    </div>
    <div class="d-flex flex-wrap justify-space-between mt-3 mt-sm-0">
      <template v-if="isLoading.items">
        <SkeletonCard class="mb-3" v-for="n in 10" :key="n" />
      </template>
      <template v-else-if="!isLoading.items && items.length == 0">
        <div class="text-center">
          <h3>{{ $t("message.nothingFound") }}</h3>
        </div>
      </template>
      <template v-else>
        <UniversityCard
          class="mb-3"
          v-for="item in items"
          :key="item.id"
          :item="item"
        />
      </template>
    </div>
    <div v-if="!isLoading.items && items.length != 0" class="text-center">
      <v-pagination
        v-model="query.page"
        :length="Math.ceil(count / 10)"
        :total-visible="7"
      ></v-pagination>
    </div>
  </div>
</div>
</v-container>
</v-main>
</template>

<script>
import Skeleton from "../components/Skeleton";
import SkeletonCard from "../components/SkeletonCard";
import UniversityCard from "../components/UniversityCard";
import Select from "../components/Select";
import { Service } from "../services/httpClient";

export default {
  name: "Universities",
  components: {
    Skeleton,
    SkeletonCard,
    UniversityCard,
    Select,
  },
  data: () => ({
    isLoading: {
      filter: true,
      items: true,
    },
    filter: {},
    items: [],
    count: 0,

    query: {
      search: "",
      location: "",
      language: "",
      period: "",
      type: "",
    }
  })
}

```

```

        season: "",
        faculty: "",
        specialty: "",
        page: 1,
    },
    }),
    watch: {
        ["query.page"]: function () {
            this.search();
        },
    },
    mounted() {
        this.query = {
            ...this.$route.query,
            page: this.$route.query.page
                ? Number.parseInt(this.$route.query.page)
                : 1,
        };

        Service.getFilterFieldsForUniversityOfMobility().then((res) => {
            this.filter = { ...res.data };
            this.isLoading.filter = false;
        });

        Service.getUniversityOfMobility(this.query).then((res) => {
            this.items = [...res.data.items];
            this.count = res.data.count;
            this.isLoading.items = false;
        });
    },
    methods: {
        search() {
            this.isLoading.items = true;
            this.$router.push({ query: this.query });
            Service.getUniversityOfMobility(this.query)
                .then((res) => {
                    this.items = [...res.data.items];
                })
                .finally(() => {
                    this.isLoading.items = false;
                });
        },
    },
};
</script>

<style lang="scss">
.universities {
    .v-text-field__details {
        display: none;
    }

    .__fields-filter {
        width: 30%;
        background-color: #eee;
        padding: 10px;
        box-shadow: 0px 3px 10px 3px rgba(0, 0, 0, 0.2);
        border-radius: 5px;
        height: fit-content;

        @media (max-width: 600px) {
            display: none;
        }
    }
}

```

```

. __mob-fields-filter {
  .v-expansion-panel {
    background-color: #eee !important;

    .v-expansion-panel-content__wrap {
      padding: 16px !important;
    }

    .v-expansion-panel-header {
      position: relative;
      padding: 0 !important;
      min-height: 30px !important;

      .v-expansion-panel-header__icon {
        position: absolute;
        left: 50%;
        top: 60%;
        transform: translate(-50%, -40%);
        margin-left: unset !important;
      }
    }
  }
}

. __content {
  width: 70%;
  padding: 0 10px;

  . __search {
    .v-input__slot {
      margin: 0 !important;
    }
  }
}

@media (max-width: 600px) {
  width: 100%;
}
}
</style>

```

A.17 Файл /src/pages/University.vue

```

<template>
  <v-main class="university">
    <v-img
      class="bg-image"
      src="../../../public/images/university.jpg"
      height="calc(100vh - 76px)"
      gradient="to top, rgba(0, 0, 0, .7), rgba(0, 0, 0, .7)"
    >
    <v-container class="overflow-y-auto">
      <h1 class="__name">
        <template v-if="!isLoading">
          {{ university.name }}
        </template>
        <template v-else>
          <Skeleton height="120px" class="mb-3" />
        </template>
      </h1>
    </v-container>
  </v-main>
</template>

```

```

    </template>
</h1>
<v-card class="__card">
  <table>
    <tbody>
      <tr>
        <td valign="top" class="pr-3 font-weight-black">
          {{ $t("university.description") }}:
        </td>
        <td style="width: 100%">
          <template v-if="!isLoading">
            {{ university.description }}
          </template>
          <template v-else>
            <Skeleton height="20px" class="mb-2" />
            <Skeleton height="20px" class="mb-2" />
            <Skeleton height="20px" class="mb-2" />
          </template>
        </td>
      </tr>
      <tr>
        <td valign="top" class="pr-3 font-weight-black">
          {{ $t("university.location") }}:
        </td>
        <td style="width: 100%">
          <template v-if="!isLoading">
            {{ university.location }}
          </template>
          <template v-else>
            <Skeleton height="20px" class="mb-2" />
          </template>
        </td>
      </tr>
      <tr>
        <td valign="top" class="pr-3 font-weight-black">
          {{ $t("university.website") }}:
        </td>
        <td style="width: 100%">
          <template v-if="!isLoading">
            <a :href="university.website" target="_blank">
              {{ university.website }}
            </a>
          </template>
          <template v-else>
            <Skeleton height="20px" class="mb-2" />
          </template>
        </td>
      </tr>
    </tbody>
  </table>
  <v-row class="d-block d-sm-flex mt-0">
    <v-col sm="4">
      <v-card tile>
        <v-list shaped>
          <v-subheader>{{ $t("university.faculties") }}</v-subheader>
          <template v-if="!isLoading">
            <v-list-group>
              v-for="(faculty, i) of faculties"
              :key="faculty.id"
              @click="facultyId = i"
              color="primary"
              no-action
            >
              <template v-slot:activator>

```

```

        <v-list-item-content>
          <v-list-item-title
            v-text="faculty.name"
          ></v-list-item-title>
        </v-list-item-content>
      </template>

      <v-list-item-group v-model="specialtyId">
        <v-list-item
          v-for="specialty of faculty.specialties"
          :key="specialty.id"
        >
          <v-list-item-content>
            <v-list-item-title
              v-text="specialty.name"
              style="padding-left: 20px"
            ></v-list-item-title>
          </v-list-item-content>
        </v-list-item>
      </v-list-item-group>
    </v-list-group>
  </template>
  <template v-else>
    <v-list-group v-for="index in 5" :key="index">
      <template v-slot:activator>
        <v-list-item-content>
          <v-list-item-title>
            <Skeleton />
          </v-list-item-title>
        </v-list-item-content>
      </template>
    </v-list-group>
  </template>
</v-list>
</v-card>
</v-col>
<v-col sm="8">
  <template v-if="exchProgrammes.length">
    <v-expansion-panels>
      <v-expansion-panel
        v-for="exchProgramm of exchProgrammes"
        :key="exchProgramm.id"
      >
        <v-expansion-panel-header>
          <div>
            <span class="font-weight-bold">
              {{ exchProgramm.name }}
            </span>
            <table class="mt-3">
              <tbody>
                <tr>
                  <tr>
                    <td valign="top" class="pr-3 font-weight-black">
                      {{ $t("university.period") }}:
                    </td>
                    <td>
                      {{ exchProgramm.period }}
                    </td>
                  </tr>
                  <tr>
                    <td valign="top" class="pr-3 font-weight-black">
                      {{ $t("university.season") }}:
                    </td>
                    <td>
                      {{ exchProgramm.season.join(", ") }}
                    </td>
                  </tr>
                </tbody>
              </table>
            </div>
          </v-expansion-panel-header>
        </v-expansion-panel>
      </v-expansion-panels>
    </template>
  </v-col>

```

```

        </td>
      </tr>
    <tr>
      <td valign="top" class="pr-3 font-weight-black">
        {{ $t("university.language") }}:
      </td>
      <td>
        {{ exchProgramm.language.join(", ") }}
      </td>
    </tr>
  </tbody>
</table>
</div>
</v-expansion-panel-header>
<v-expansion-panel-content>
  <div>
    <v-simple-table>
      <template v-slot:default>
        <thead>
          <tr>
            <th class="text-center">
              {{ $t("university.code") }}
            </th>
            <th class="text-center">
              {{ $t("university.programm") }}
            </th>
            <th class="text-center">
              {{ $t("university.ects") }}
            </th>
            <th class="text-center">
              {{ $t("university.semester") }}
            </th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td colspan="4">
              v-for="syllabus of exchProgramm.syllabuses"
              :key="syllabus.id"
            </td>
          </tr>
          <tr>
            <td class="text-center">{{ syllabus.code }}</td>
            <td>{{ syllabus.name }}</td>
            <td class="text-center">{{ syllabus.ects }}</td>
            <td class="text-center">
              {{ syllabus.semester.join("/") }}
            </td>
          </tr>
        </tbody>
      </template>
    </v-simple-table>

    <table class="mt-3">
      <tbody>
        <tr>
          <td valign="top" class="pr-3 font-weight-black">
            {{ $t("university.coordinator") }}:
          </td>
          <td>
            {{ exchProgramm.coordinatorFullName }}
          </td>
        </tr>
        <tr>
          <td valign="top" class="pr-3 font-weight-black">
            {{ $t("university.email") }}:
          </td>

```

```

        <td>
            {{ exchProgramm.coordinatorEmail }}
        </td>
    </tr>
    <tr>
        <td valign="top" class="pr-3 font-weight-black">
            {{ $t("university.ectsCredits") }}:
        </td>
        <td>
            {{ exchProgramm.ectsCredits }}
        </td>
    </tr>
</tbody>
</table>
<div class="text-end">
    <v-btn @click="() => handlerForCompare(exchProgramm)">
        {{ $t("university.compare") }}
    </v-btn>
    <v-btn class="ml-3">
        {{ $t("university.apply") }}
    </v-btn>
</div>
</div>
</v-expansion-panel-content>
</v-expansion-panel>
</v-expansion-panels>
</template>
<template v-else>
    <h3 class="text-center">
        {{ $t("university.chooseSpecialty") }}
    </h3>
</template>
</v-col>
</v-row>
</v-card>
</v-container>
</v-img>
<Compare :dialog="isCompare" :closeDialog="closeDialog" />
</v-main>
</template>

<script>
import Skeleton from "../components/Skeleton";
import { Service } from "../services/httpClient";
import Compare from "../components/modal/Compare";

export default {
    name: "University",
    components: {
        Skeleton,
        Compare,
    },
    data: () => ({
        university: {},
        faculties: [],
        facultyId: undefined,
        specialtyId: undefined,
        exchProgrammes: [],
        isLoading: true,
        isCompare: false,
    }),
    watch: {
        facultyId() {
            this.specialtyId = undefined;

```

```

    },
    specialtyId(value) {
      if (value !== undefined) {
        this.exchProgrammes = this.faculties[this.facultyId].specialties[
          value
        ].exchProgrammes;
      } else {
        this.exchProgrammes = [];
      }
    },
  },
  mounted() {
    Service.getUniversityById(this.$route.params.id)
      .then((res) => {
        this.university = res.data;
        this.faculties = this.university.faculties;
      })
      .catch(() => {
        this.$router.push("/404");
      })
      .finally(() => {
        this.isLoading = false;
      });
  },
  methods: {
    handlerForCompare(exchProgramm) {
      this.isCompare = true;
      this.$store.commit("setSyllabuses", exchProgramm.syllabuses);
    },
    closeDialog() {
      this.isCompare = false;
      this.$store.commit("setSyllabuses", []);
    },
  },
};
</script>

```

```

<style lang="scss">
.university {
  .v-responsive__content {
    overflow-y: auto;
  }

  .bg-image {
    background-attachment: fixed;
  }

  .v-expansion-panel-header {
    font-size: 16px !important;
  }

  .__name {
    color: #fff;
    font-size: 80px;
    text-align: center;
  }

  .v-skeleton-loader__bone {
    background-color: unset;

    &::after {
      background: linear-gradient(
        90deg,
        rgba(255, 255, 255, 0),
        rgba(255, 255, 255, 0.3),

```

```

        rgba(255, 255, 255, 0)
      ) !important;
    }
  }
}

.__card {
  padding: 20px;
}
}
</style>

```

A.18 Файл /src/pages/Verify.vue

```

<template>
  <v-main>
    <div class="container">
      <div class="progress">
        <v-progress-circular
          :size="70"
          :width="7"
          color="primary"
          indeterminate
        ></v-progress-circular>
      </div>
    </div>
  </v-main>
</template>

<script>
import { Service } from "../services/httpClient";

export default {
  name: "Verify",
  mounted() {

    Service.getVerify(this.$route.params.token)
      .then(() => {
        this.$router.push("/");
      })
      .catch(() => {
        this.$router.push("/404");
      });
  },
};
</script>

<style lang="scss">
.progress {
  position: absolute;
  top: -60px;
  width: 100%;
  height: 100vh;
  z-index: 1;

  .v-progress-circular {
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
  }
}

```

```

    background-color: rgba(0, 0, 0, 0.7);
  }
</style>

```

A.19 Файл /src/plugins/vuetify.js

```

import Vue from 'vue';
import Vuetify from 'vuetify/lib/framework';

Vue.use(Vuetify);

export default new Vuetify({
});

```

A.20 Файл /src/services/vuetify.js

```

import HttpClient from "../data-layer/HttpApiClient";
import store from "../store";

export const Service = new HttpClient({
  language: store.state.locale,
});

```

A.21 Файл /src/store/index.js

```

import Vue from "vue";
import Vuex from "vuex";
import createPersistedState from "vuex-persistedstate";

Vue.use(Vuex);

export default new Vuex.Store({
  plugins: [
    createPersistedState({
      paths: ["jwtToken", "locale", "user"],
    }),
  ],
  state: {
    jwtToken: null,
    locale: "en",
    user: {},
    syllabuses: [],
  },
  mutations: {
    setJwtToken(state, token) {
      state.jwtToken = token;
    },
    setLocale(state, locale) {
      state.locale = locale;
    },
    setUser(state, user) {
      state.user = user;
    },
  },
});

```

```

    setSyllabuses(state, syllabuses) {
      state.syllabuses = syllabuses;
    },
  },
  getters: {
    isAuthorization(state) {
      return state.jwtToken ? true : false;
    },
  },
});

```

A.22 Файл /src/utils/qs.js

```

import set from "lodash/set";
import isDate from "lodash/isDate";
import isPlainObject from "lodash/isPlainObject";
import isEmpty from "lodash/isEmpty";

export const parse = (text) => {
  const res = {};

  for (const [key, value] of new URLSearchParams(text)) {
    set(res, key, value);
  }

  return res;
};

export const stringify = (values, options = { addQueryPrefix: false }) => {
  const pairs = [];

  const addPairs = (prefix, value) => {
    if (value === undefined) {
      return;
    }

    if (Array.isArray(value)) {
      value.forEach((val, index) => addPairs(`${prefix}[${index}]`, val));
    } else if (isPlainObject(value)) {
      for (const [key, val] of Object.entries(value)) {
        addPairs(`${prefix}.${key}`, val);
      }
    } else if (isDate(value)) {
      pairs.push([prefix, value.toISOString()]);
    } else {
      pairs.push([prefix, value ? value.toString() : ""]);
    }
  };

  for (const [key, val] of Object.entries(values)) {
    addPairs(key, val);
  }

  return `${
    options.addQueryPrefix && !isEmpty(pairs) ? "?" : ""
  }${new URLSearchParams(pairs).toString()}`;
};

```

A.23 Файл /src/utils/validation.js

```
import { isEmail, isDate } from "validator";
import i18n from "../i18n";

export const required = (v) => !!v || i18n.t("validator.required");

export const min = (value) => (v) =>
  v.length >= value || i18n.t("validator.min").replace("{0}", value);

export const max = (value) => (v) =>
  v.length <= value || i18n.t("validator.max").replace("{0}", value);

export const email = (v) => isEmail(v) || i18n.t("validator.email");

export const date = (v) => isDate(v) || i18n.t("validator.date");

export const confirm = (value) => (v) =>
  v === value || i18n.t("validator.confirmPassword");
```

ДОДАТОК Б
Слайди презентації

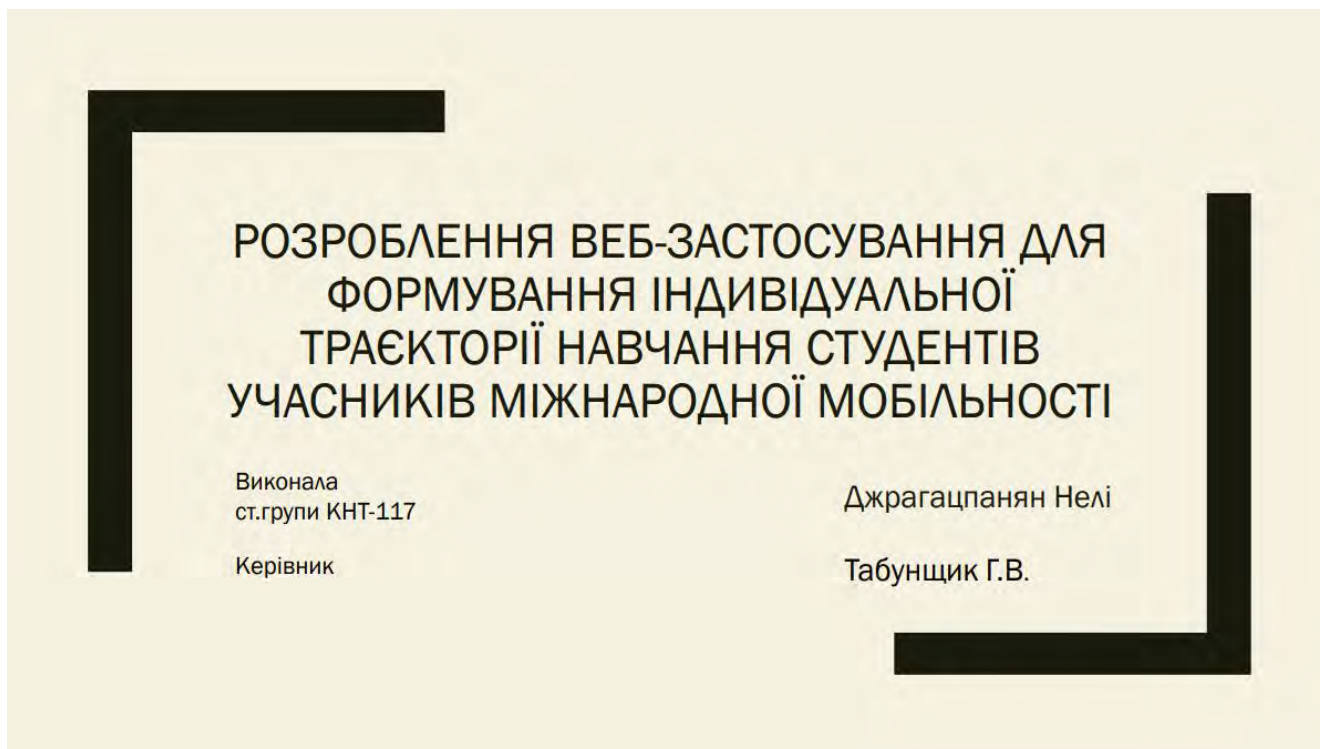


Рисунок Б.1 – Слайд 1

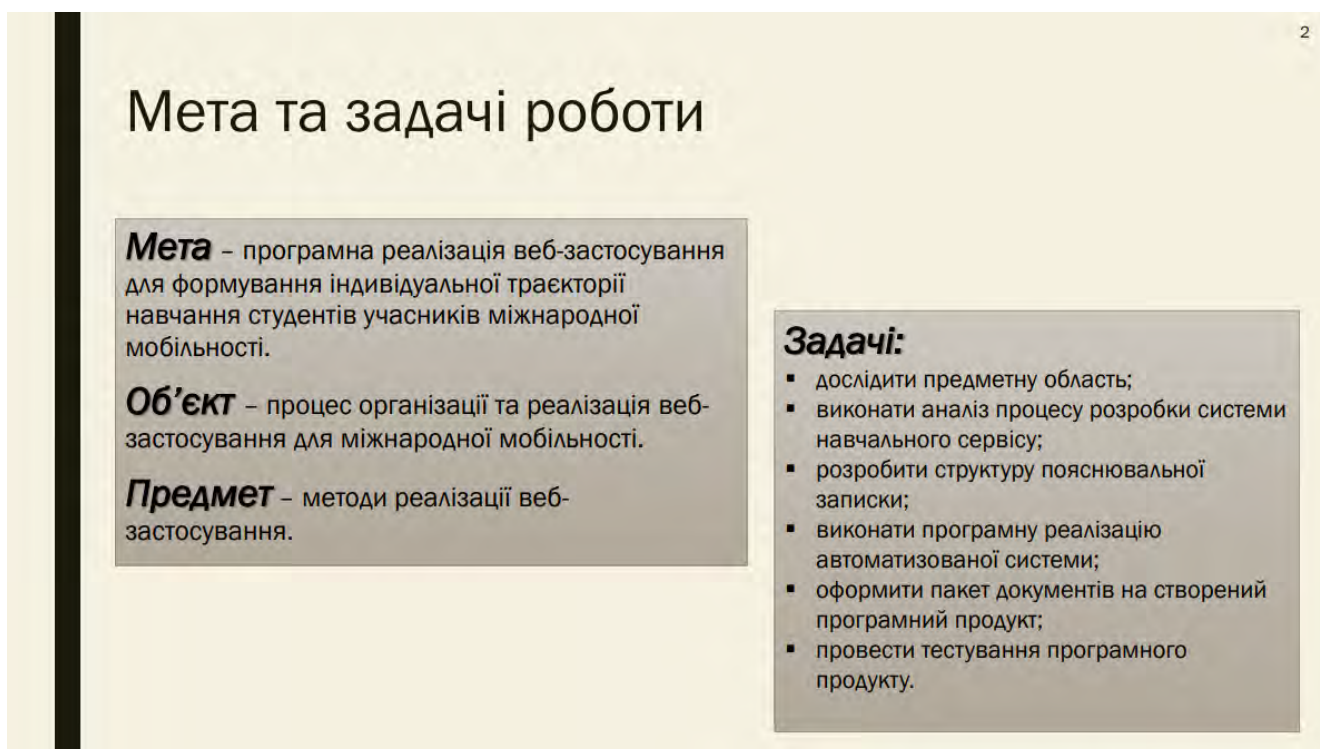


Рисунок Б.2 – Слайд 2

3

Дортмундський університет

MOBILITY-ONLINE
DORTMUND UNIVERSITY OF APPLIED SCIENCES AND ARTS

Workplace | Help | Logout

Galyna Tabunshchuk
English

My Mobility-Online
My institution
My addresses
My agreements
Create new application
Change password

Details of activities at Home Institution
Country of home institution: <-- Please select -->
Home Institution: <-- Please select -->
Studienniveau: ☐ Bachelor ☐ Master

Daten zum geplanten Studium
Host country: Germany
Host institution: DORTMUND2 - DORTMUND UNIVERSITY
Faculty: <-- Please select -->
Study field: 10 - Faculty of Informationtechnology
1 - Faculty of Architecture
2 - Faculty of Design
3 - Faculty of electrical engineering
4 - Faculty of Computer Sciences
5 - Faculty of Mechanical Engineering
8 - Faculty of Applied Social Studies
9 - Faculty of Business Studies
Verw - Administration
ZE - Central Service Units
Double Degree:

Info Box Refresh

Cancel application

Send application

Рисунок Б.3 – Слайд 3

4

Київський політехнічний інститут імені Ігоря Сікорського

ACADEMIC MOBILITY OFFICE
New! | Programmes | Centres | Erasmus+ (ICM) | Academic mobility coordinators | Thank you Erasmus+ | About the Office | FAQ | Inquiries

Оголошення
21 жовтня відбулася відеоконференція з Міністром виробництва, науки та технологій провінції Санта-Фе, Аргентина
Продовження аргентинської та української сторін, партнерства та університетів обговорили взаємні можливості, розвитку та інтеграції освітніх програм, академічних обмінів, міжнародних мовних програм, сприяють до розвитку співпраці, досліджень, інновацій, культурних обмінів, а також до фінансової міцності Інституту Санта-Фе-Україна, в перспективі та розвитку академічного співробітництва. Чи студентів та науковців?

Igor Sikorsky Kyiv Polytechnic Institute Academic Mobility Office

LIST OF PARTNERS
ACADEMIC MOBILITY
COORDINATORS AT
REGULATORY

Exchange programme with University of Tohoku, Japan



I. Proposed programmes:

1. English Language Program in English (LTPSE)
LTPSE – це програма вступу, яка надає можливість студентам Університету Київського Політехнічного Інституту (КПІ) вивчати англійську мову в Японії. Програма є обов'язковою для студентів, які вступають до університету в Японії.

2. Інформаційні технології в Японії (ITP)
Програма спрямована на розвиток знань та навичок в галузі інформаційних технологій в Японії. Студенти будуть вивчати основні принципи роботи комп'ютерних систем, мереж та баз даних.

3. Інженерні науки в Японії (ENP)
Програма спрямована на розвиток знань та навичок в галузі інженерних наук в Японії. Студенти будуть вивчати основні принципи роботи механізмів, електричних машин та електроніки.

4. Бізнес та економіка в Японії (BEP)
Програма спрямована на розвиток знань та навичок в галузі бізнесу та економіки в Японії. Студенти будуть вивчати основні принципи роботи підприємств, маркетингу та фінансів.

5. Культура та мистецтво в Японії (CPE)
Програма спрямована на розвиток знань та навичок в галузі культури та мистецтва в Японії. Студенти будуть вивчати основні принципи роботи музеїв, театрів та інших культурних закладів.

Student could conduct his researches in English if it was agreed with his scientific advisor.

Art and calligraphy: <http://www.sai.tohoku.ac.jp/index.html>
Education: <http://www.sed.tohoku.ac.jp/~english/>
Law: <http://www.law.tohoku.ac.jp/english/>
Economy: <http://www.econ.tohoku.ac.jp/econ/english/>
International cultureology: <http://www.intcul.tohoku.ac.jp/english/>
Except for Faculty/Graduate school of Arts and Letters!

1

Рисунок Б.4 – Слайд 4

Львівський національний університет імені Івана Франка

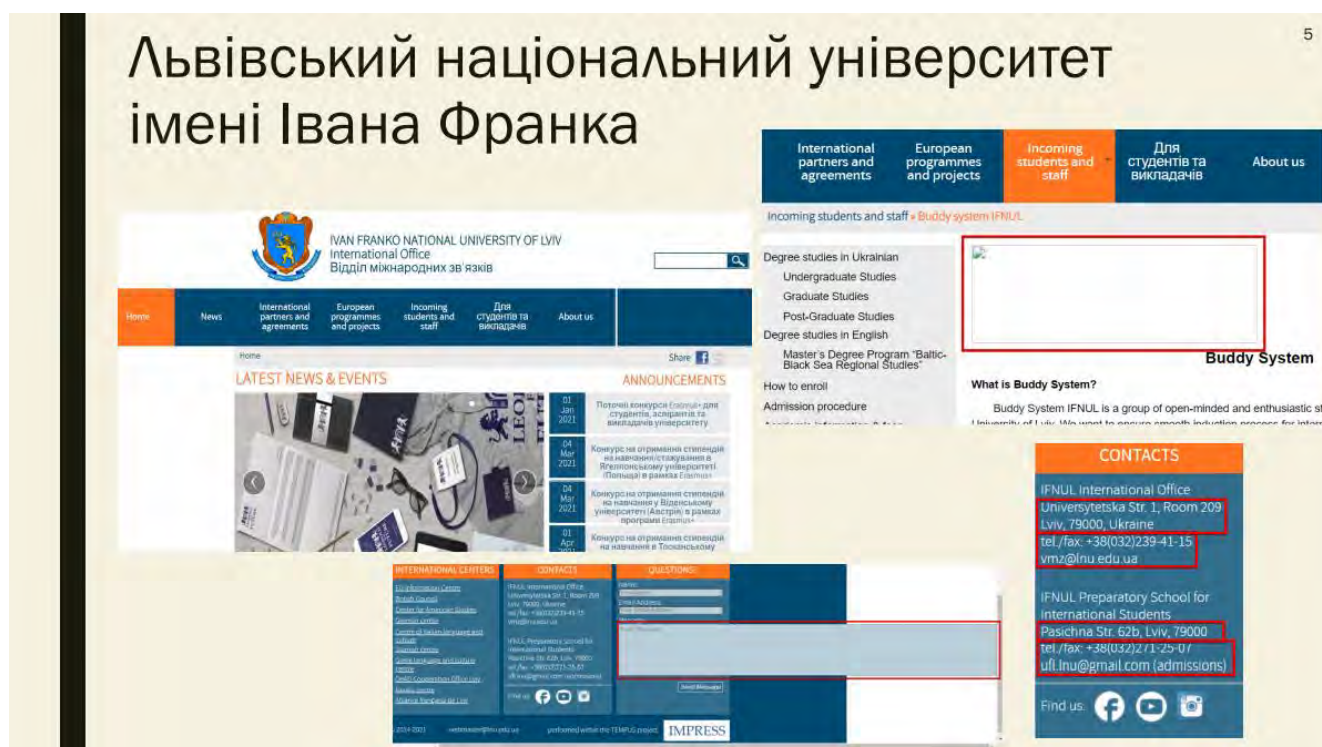


Рисунок Б.5 – Слайд 5

Львівський національний університет імені Івана Франка

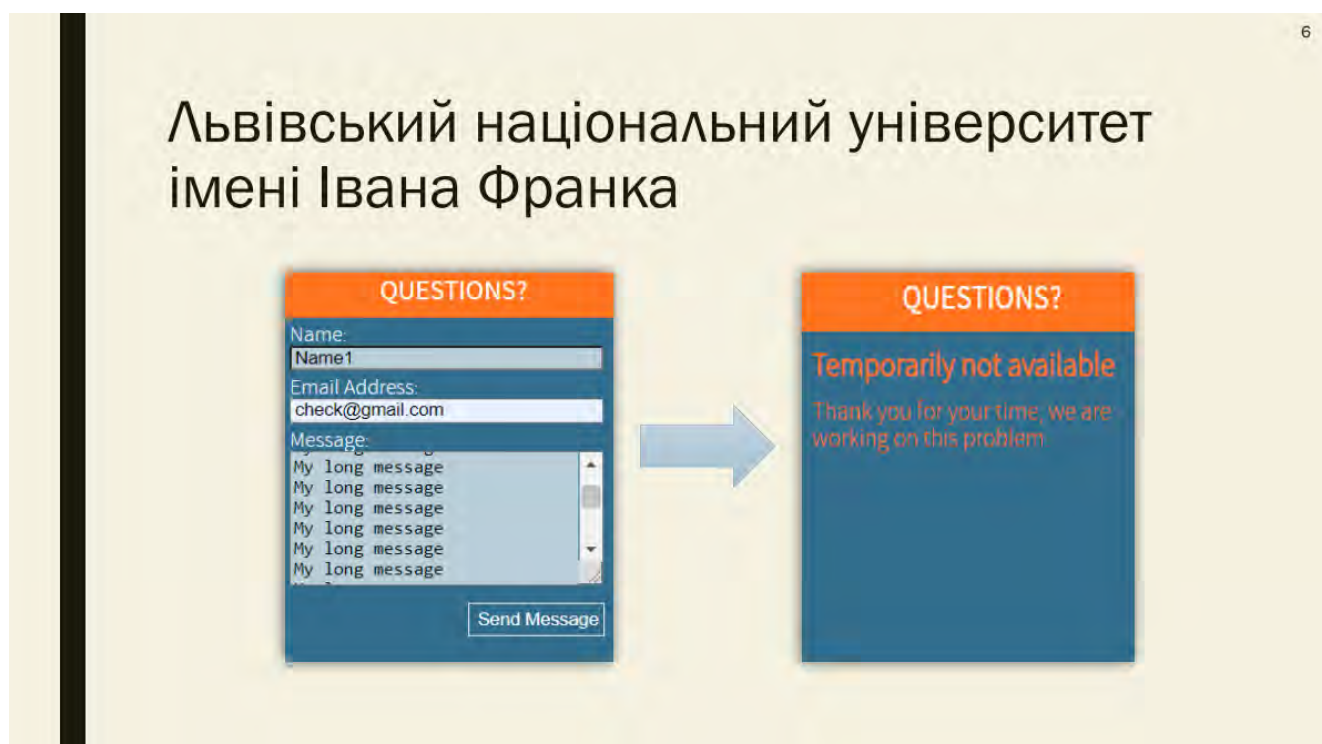


Рисунок Б.6 – Слайд 6

Порівняння JS-фреймворків

Критерії	Інструмент		
	Angular.js	React.js	Vue.js
Тип	Фреймворк	Бібліотека для створення користувацького інтерфейсу	Фреймворк
Типи додатків	Нативні, гібридні і веб-додатки	Односторінкові (SPA) і мобільні додатки	Модернізовані односторінкові додатки і підтримка нативних додатків
Модель	MVC	Віртуальна DOM	Віртуальна DOM
Переважна мова розробки	Рекомендовано використовувати TypeScript	Рекомендовано використовувати JSX – JavaScript XML	HTML-шаблони та JavaScript.
Ідеально підходить для	Великомасштабних додатків з багатим функціоналом	Веб-додатків і нативних додатків для iOS і Android	Веб-додатків і односторінкових (SPA) додатків
Спрямованість і класи залежностей	Взаємодії об'єктів відбувається в службах	Одностороння передача даних	Одностороння передача даних
Зворотня сумісність	Абсолютна залежність від попередніх версій	Повна сумісність між версіями	Повна сумісність між версіями
Розмір бібліотек	500+ KB	100 KB	80 KB

Рисунок Б.7 – Слайд 7

Vue інструменти



Рисунок Б.8 – Слайд 8

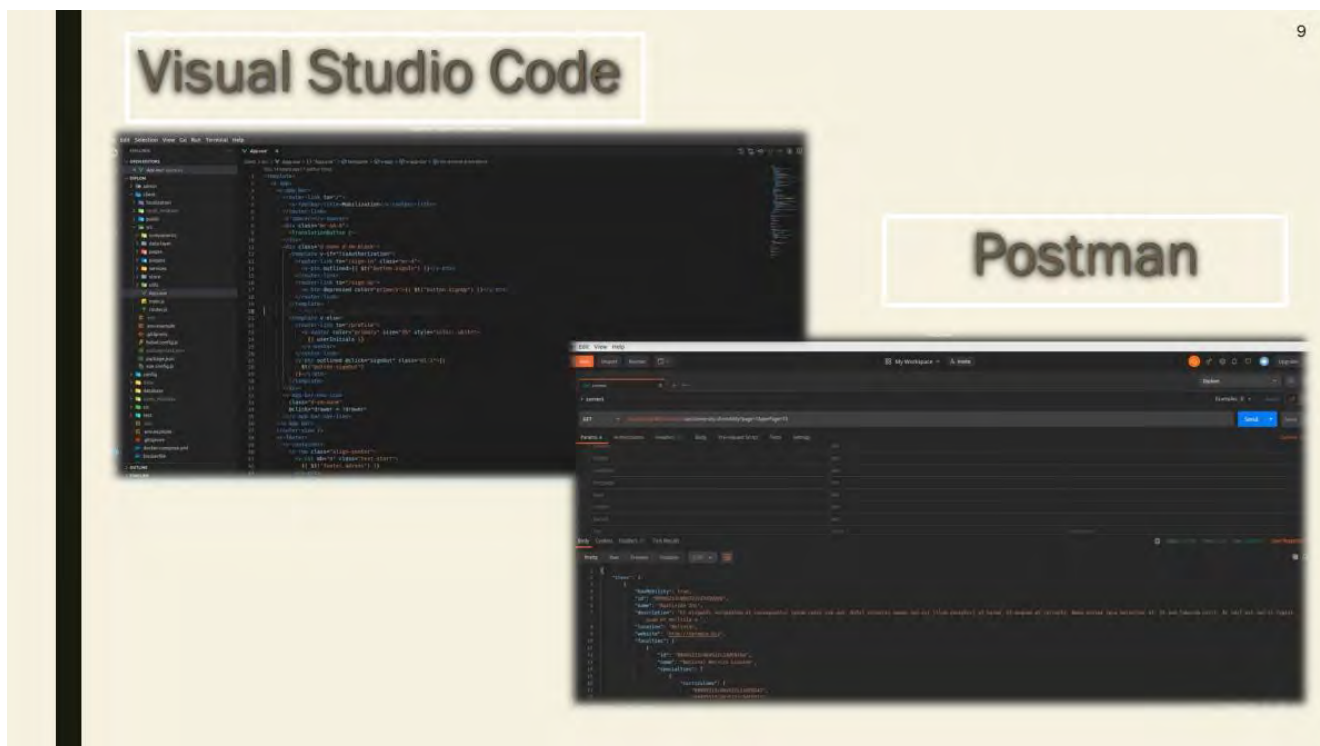


Рисунок Б.9 – Слайд 9

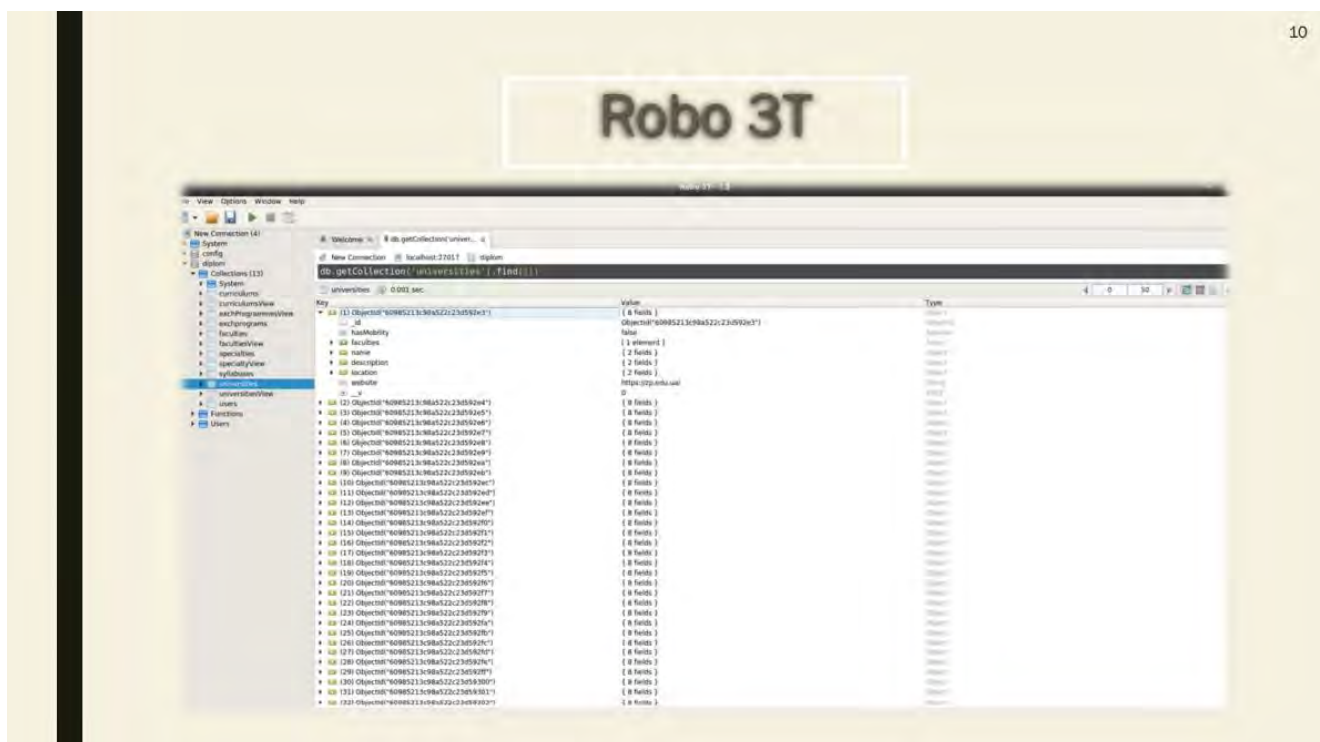


Рисунок Б.10 – Слайд 10

Структурна схема



Рисунок Б.11 – Слайд 11

Схема функціонування програми

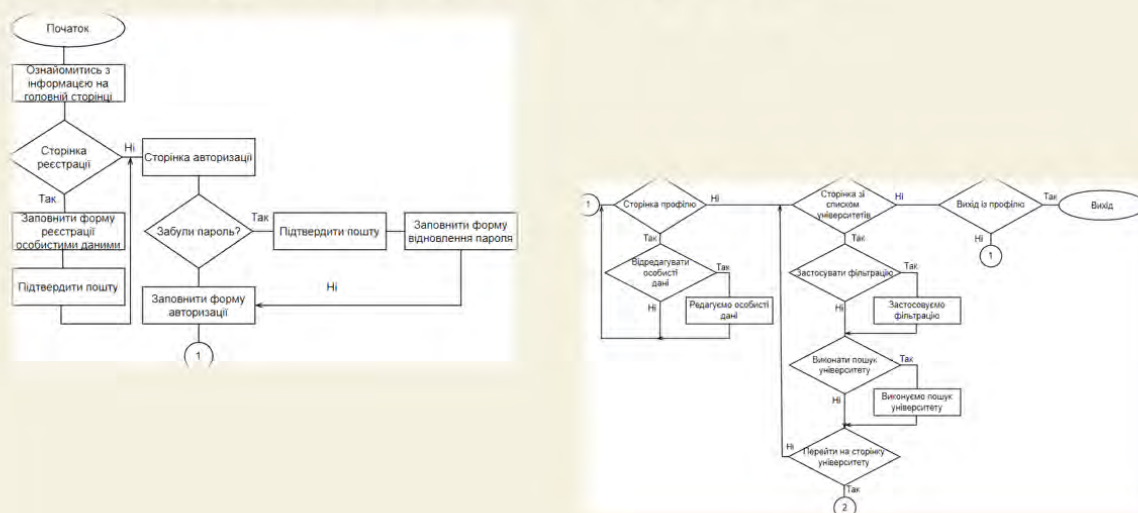


Рисунок Б.12 – Слайд 12

Схема функціонування програми

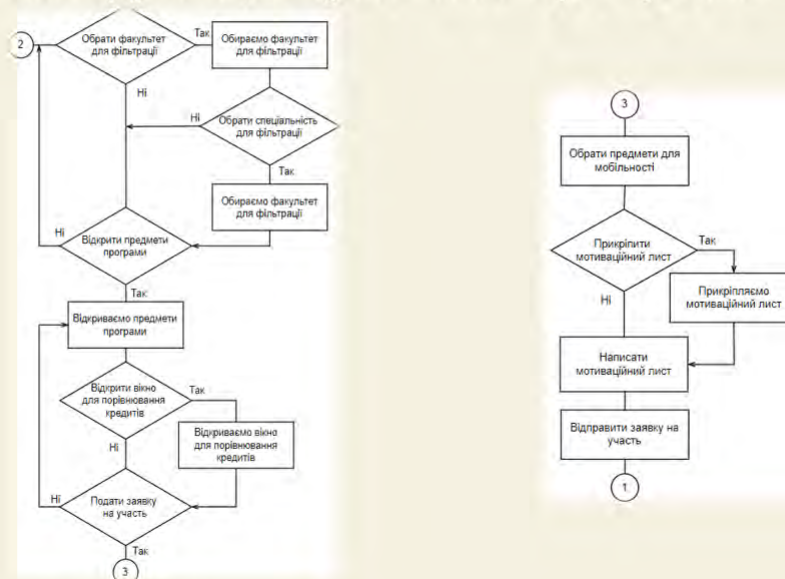


Рисунок Б.13 – Слайд 13

Тип запиту	Шлях запиту	Опис запиту (що він приймає)	Опис відповіді (що повертається)
post	/auth/register	Дані для реєстрації	Повідомлення про успішну реєстрацію
put	/auth/verify/:token	Токен для верифікації пошти	Токен користувача
post	/auth/login	Логін та пароль для авторизації користувача	Токен користувача
post	/auth/forgot	Адреса пошти для відправлення повідомлення про відновлення пароля	Повідомлення про успішне відправлення повідомлення на пошту
get	/university/:universityId	ID університету	Університет за вказаним ID
get	/university-of-mobility	Номер поточної сторінки, кількість елементів на одній сторінці, необов'язковий параметр для пошуку університету за назвою та параметри фільтрації	Вказана кількість університетів відповідно до параметрів пошуку та фільтрації
get	/university/:universityId/faculty	ID університету	Список факультетів вказаного університету

Рисунок Б.14 – Слайд 14

ЕНДПОІНТИ

Тип запиту	Шлях запиту	Опис запиту (що він приймає)	Опис відповіді (що повертається)
get	/university/:universityId/faculty/:facultyId/specialty	ID університету та факультету	Список спеціальностей вказаного факультету та університету
get	/university/:universityId/faculty/:facultyId/specialty/:specialtyId/curriculum	ID університету, факультету та спеціальності	Список навчальних планів вказаної спеціальності факультету університету
get	/filter-fields/university	—	Список полів для фільтрації університетів мобільності

Рисунок Б.15 – Слайд 15

Адміністративна панель

Mobilization
 UNIVERSITY
 Universities
 Faculties
 Specialties
 Curriculums
 Exch. Programmes
 Syllabuses
 users
 Users
 Applications

Login
 Enter your email address

 Enter your password

Filter: type string...

Name	Location	Has Mobility
Carinthia University of Applied Sciences	Austria	yes
Thomas More University of Applied Science	Belgium	yes

Thomas More University of Applied Science
 Location: Belgium
 website: <https://www.thomasmore.be/en/welcome>
 has mobility: yes
 Thomas More is the largest university of applied sciences in Flanders, offering more than 30 Dutch-taught and a range of English-taught bachelor degree programmes & postgraduates in the province of Antwerp. Next to that, Thomas More offers exchange programmes in English, for students from partner universities.

Katholieke Universiteit Leuven	Belgium	yes
Artosis Plantijn University College	Belgium	yes
Polytechnic University of Madrid	Spain	yes
Dortmund University of Applied Sciences and Arts	Germany	yes
Hamburg University of Technology	Germany	yes
Transilvania University of Brasov	Romania	yes
Zaporizhzhia Polytechnic National University	Ukraine	no

Items per page: 10

Рисунок Б.16 – Слайд 16

Головна сторінка



Рисунок Б.17 – Слайд 17

Реєстрація

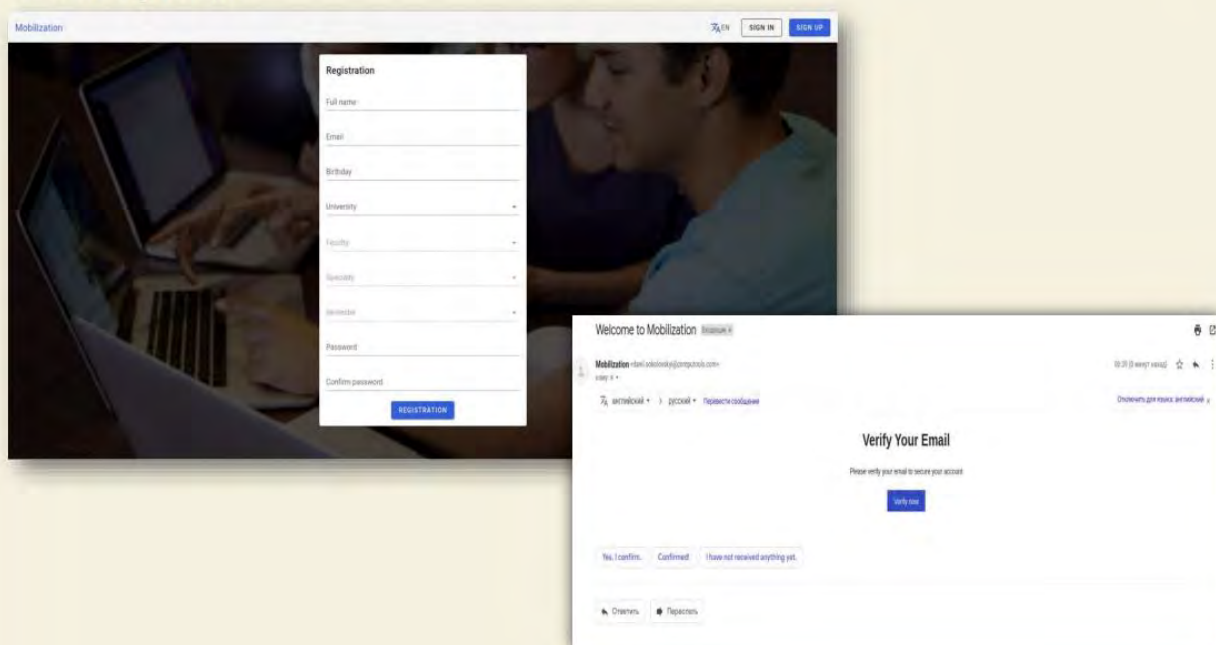
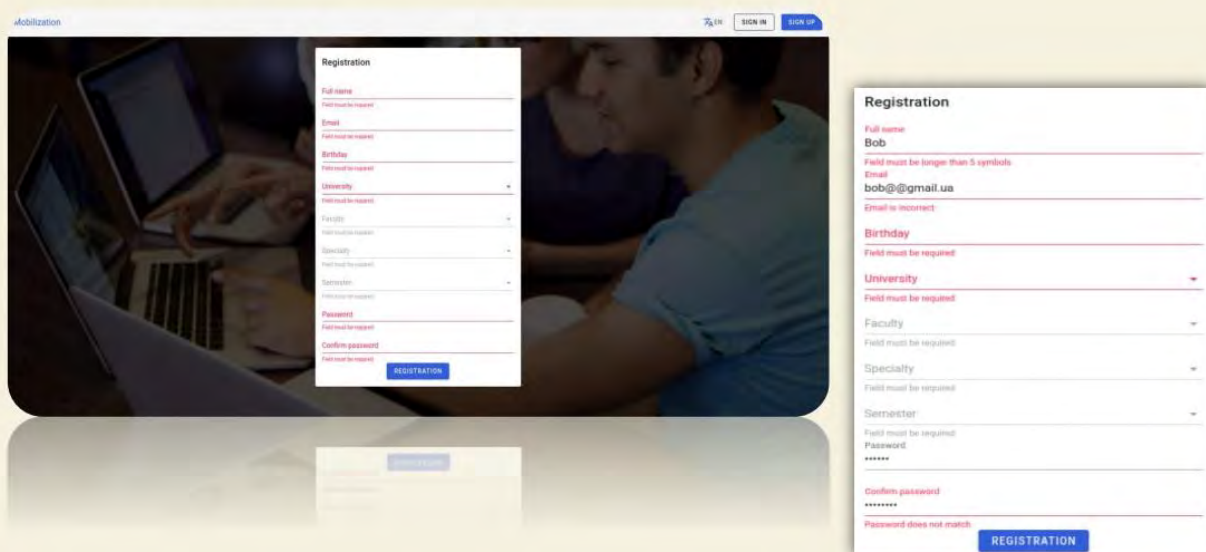


Рисунок Б.18 – Слайд 18

Реєстрація

19



Registration

Full name
Field must be required
Bob

Email
Field must be longer than 5 symbols
bob@gmail.ua

Birthday
Field must be required

University
Field must be required

Faculty
Field must be required

Specialty
Field must be required

Semester
Field must be required

Password
Field must be required

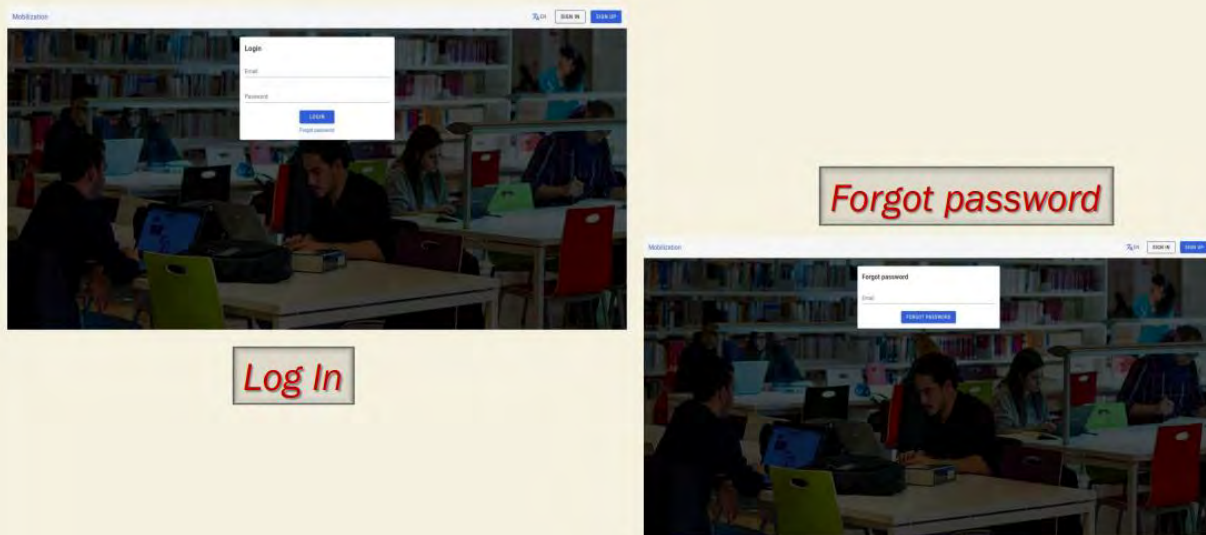
Confirm password
Field must be required

REGISTRATION

Рисунок Б.19 – Слайд 19

Авторизація

20



Login

Email

Password

Log In

Forgot password

Forgot password

Рисунок Б.20 – Слайд 20

Відновлення пароля

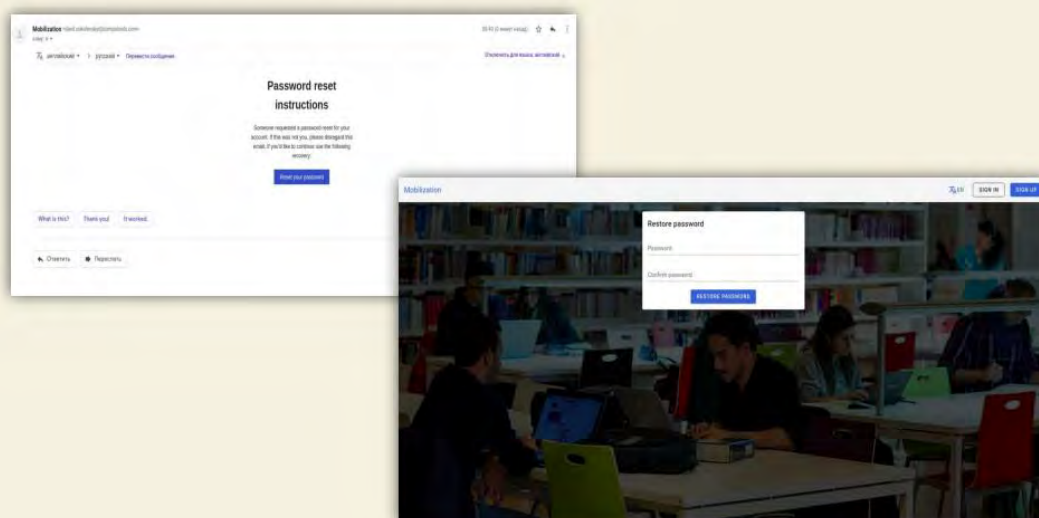


Рисунок Б.21 – Слайд 21

Обробка помилок при авторизації

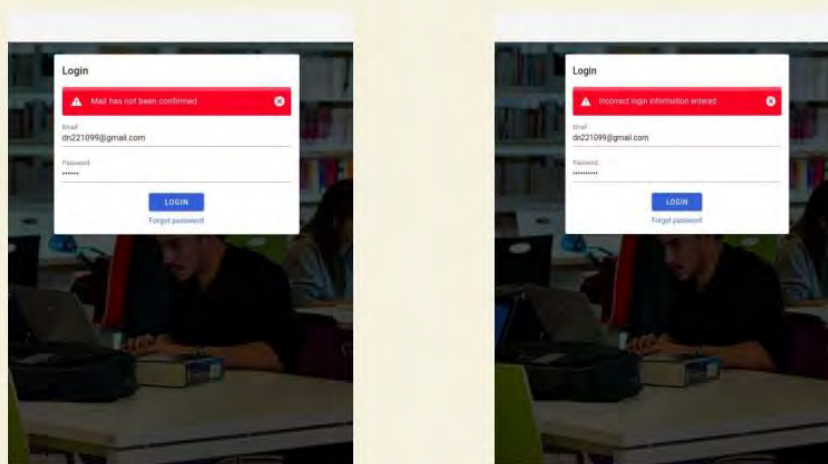


Рисунок Б.22 – Слайд 22

Особистий профіль

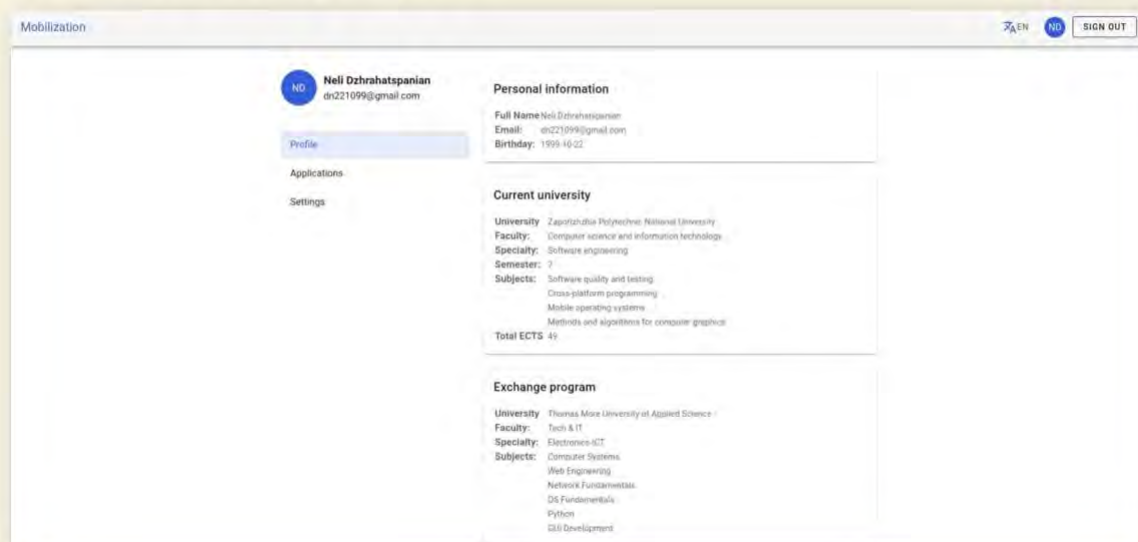


Рисунок Б.23 – Слайд 23

Список університетів

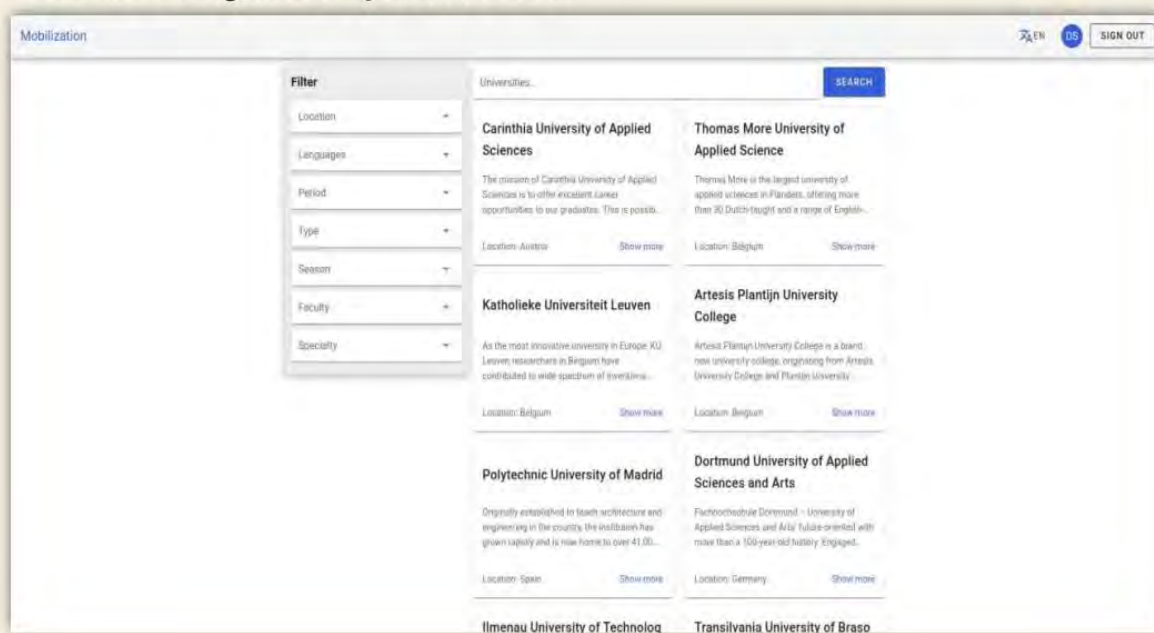


Рисунок Б.24 – Слайд 24

Обробка помилок

25

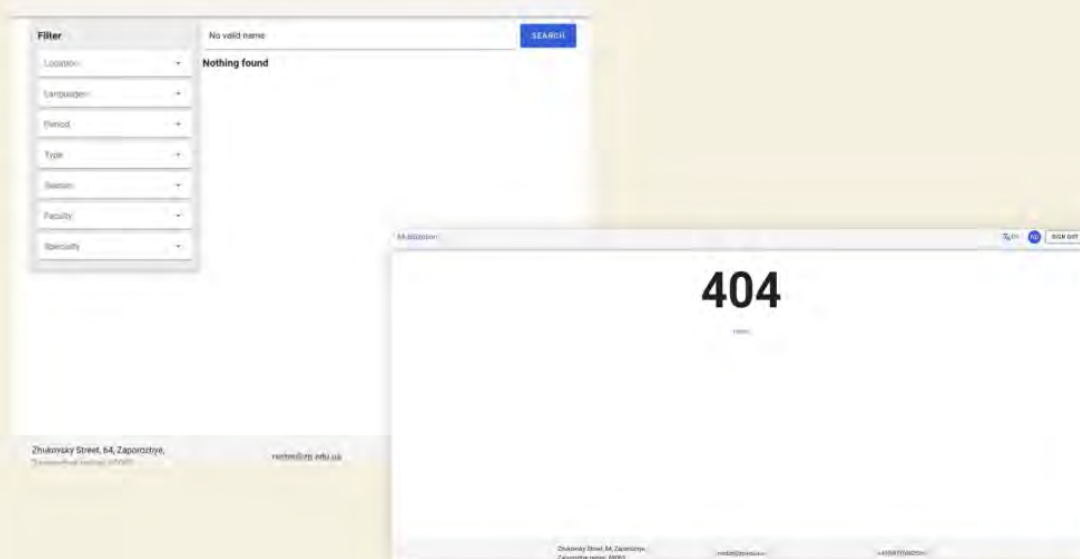


Рисунок Б.25 – Слайд 25

Сторінка навчального закладу

26

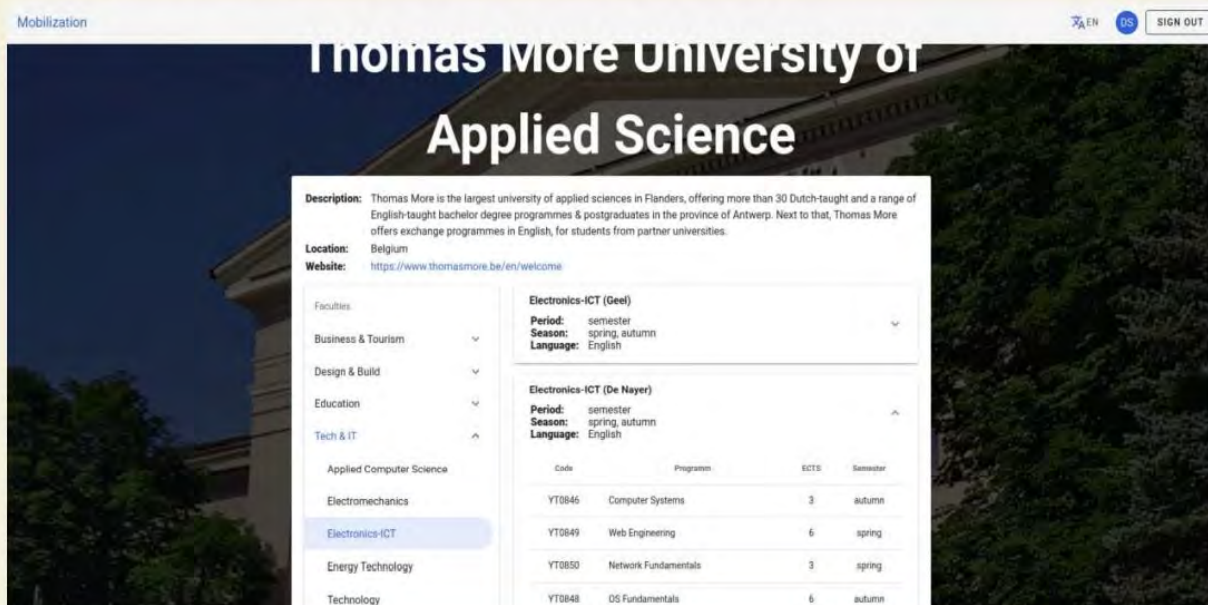


Рисунок Б.26 – Слайд 26

Модальне вікно для порівняння предметів

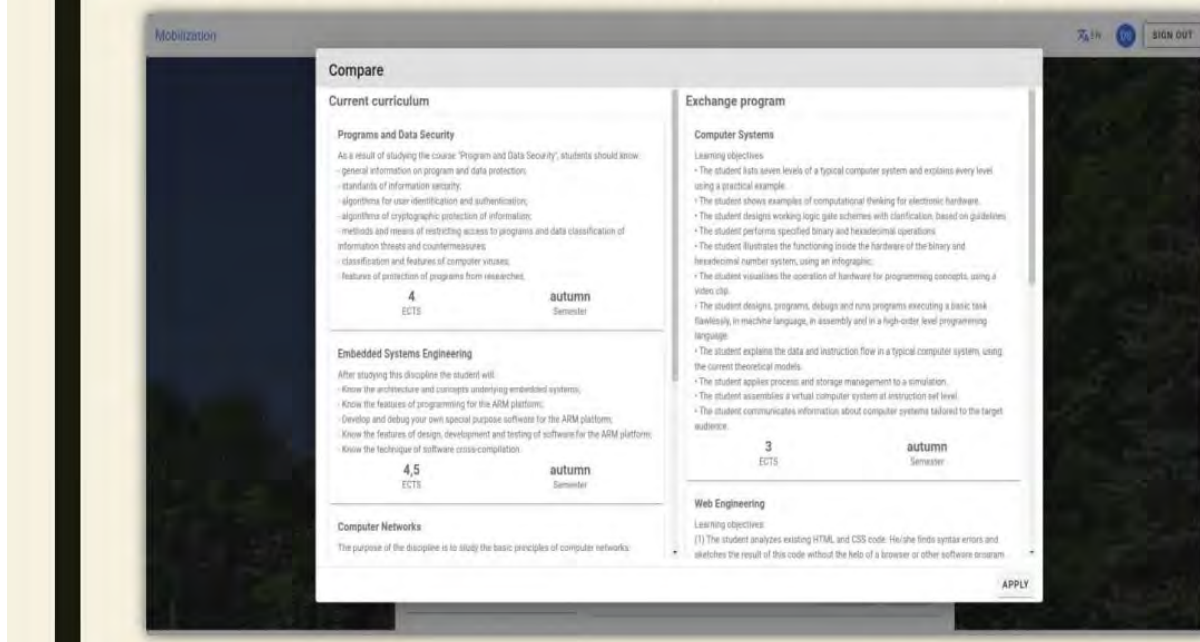


Рисунок Б.27 – Слайд 27

Подача заявки на участь

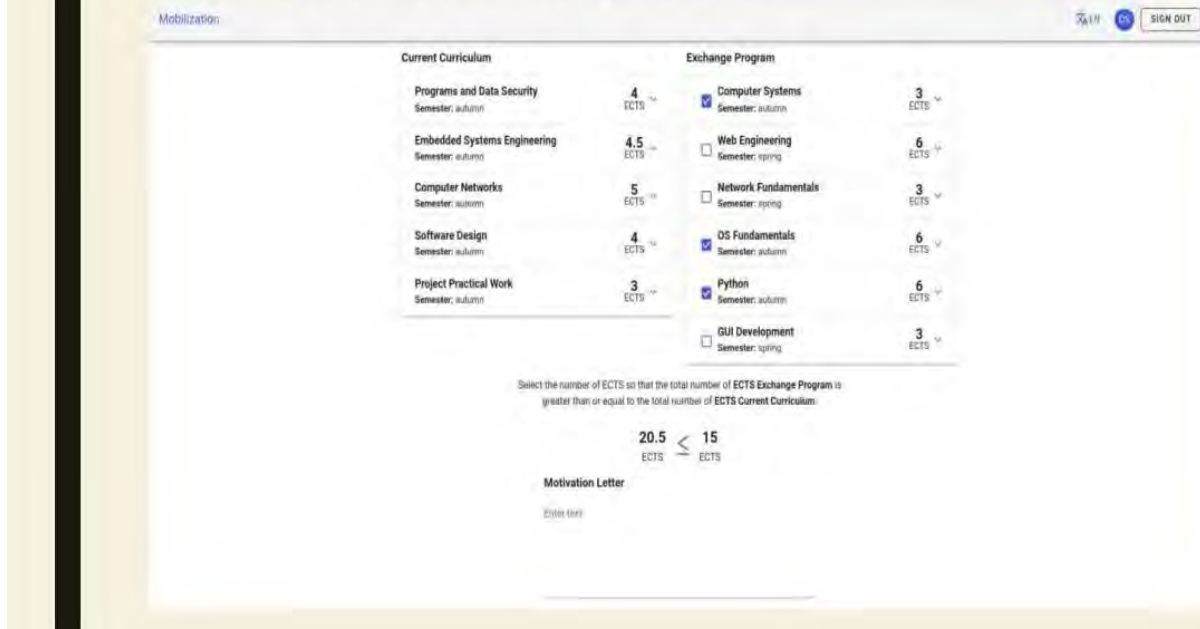


Рисунок Б.28 – Слайд 28

Висновки

- В ході виконання кваліфікаційної роботи бакалавра, було проведено огляд та аналіз існуючих систем для академічної мобільності. Проведений аналіз показав, що майже усі існуючі веб-застосування для формування індивідуальної траєкторії навчання студентів учасників міжнародної мобільності, мають недоліки, які впливають на швидкість процесу обробки заявок. Тому і було розроблено веб-застосування, яке усуває основні недоліки існуючих систем.
- Програма була розроблена з використанням сучасних технологій, має гарний і зрозумілий інтерфейс, завдяки якому користуватись системою легко та комфортно.
 - Система може бути задіяна в будь-якому навчальному закладі, де активно займаються академічною мобільністю.

Рисунок Б.29 – Слайд 29

Дякую за увагу!

Рисунок Б.30 – Слайд 30