КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-застосунок для підтримки взаємодії перекладачів та клієнтів»

Виконав _____ _____
(Підпис)
Колумбет Антон Петрович
(прізвище, ім'я, по батькові)

Керівник _Сайко Володимир Григорович
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2021

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	23.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи	1
Календарний план дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	2
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	6
2	23
3	12
Висновки	1
Перелік використаних джерел	1
Додатки	4

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Колумбет А. П.			Відомість дипломної роботи	Лист	Листів
Керівн.	Сайко В. Г.				1	57
Н/контр.	Макаренко С. А.					
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

Мета роботи – розробка веб-застосунку для підвищення взаємодії перекладачів та клієнтів. Під час розробки застосунку було використано мову програмування JavaScript, веб-сервер Node.js, фреймворк для описання роботи серверу ExpressJS, бібліотеку React та модулі до неї, а також базу даних MongoDB.

Розроблений застосунок надає змогу перекладачам та клієнтам за допомогою зручного інтерфейсу робити та обробляти замовлення, також надає змогу спілкування між замовником та перекладачем стосовно конкретного замовлення та відслідковування прогресу його виконання.

Записка містить 55 сторінок, 27 картинок, 3 таблиці, 1 додаток, 8 посилань.

Ключові слова: веб-застосунок, замовлення, взаємодія, переклад.

ABSTRACT

The purpose of the work is to develop a web application to increase the interaction between translators and clients. The application was developed using the JavaScript programming language, the Node.js web server, a framework for describing the operations of the ExpressJS server, the React library and its modules, as well as the MongoDB database.

The developed application allows translators and the client to make and process orders through a user-friendly interface, as well as allows communication between the customer and the translator regarding a specific order and tracking the progress of its execution.

The note contains 55 pages, 27 pictures, 3 tabels, 1 appendix, 8 links.

Keywords: web application, order, interaction, translation.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Скорочення

CSS – Cascading Style Sheets – каскадні таблиці стилів

HTML – HyperText Markup Language – мова гіпертекстової розмітки

СУБД – Система управління базами даних

API – Application Programming Interface – інтерфейс програмування.

Терміни

Акаунт – обліковий запис користувача, що знаходиться в базі даних та зберігає його особисті дані для входу.

NodeJS – вебсервер для запуску веб-застосунків розроблених мовою JavaScript

MongoDB – система управління базами даних (СУБД)

Ендпоінт – кінцева точка. Це звернення до маршруту віддаленим HTTP методом.

Фреймворк – це програмне оточення спеціального призначення, каркас, який полегшує процес об’єднання компонентів при створенні програми.

Утиліта – невеличка прикладна программа.

ВСТУП

Веб-застосунок – це клієнт-серверний застосунок, який працює завдяки взаємодії клієнту з веб-сервером за допомогою браузера. Перевага розробки таких застосунків полягає у тому, що користувачу немає необхідності встановлювати додаткове програмне забезпечення, тому що всі операції виконуються за допомогою будь-якого браузеру.

Розробка веб-застосунків на даний момент є актуальною галуззю розробки тому, що з розвитком технологій життя стає все динамічнішим, в людей виникає необхідність в постійному пересуванню, через це не завжди є можливість встановлювати застосунки на свій портативний комп'ютер, або взагалі, використовувати комп'ютер в момент необхідності. Веб-додатки дозволяють своє використання з будь-яких платформ, будь це смартфон, портативний комп'ютер чи, навіть, смарт-телевізор, без встановлення будь-якого програмного забезпечення окрім браузеру.

Ця курсова робота є актуальною темою в сучасні часи, це зумовлено важливістю сучасної людини в постійному переміщенню, і розвитком та популяризацією смартфонів, та глобалізацією багатьох процесів у житті держав, людей, та підприємств. Велика частина документообігу ведеться різними мовами світу, тому є необхідність у якісному перекладі важливих документів чи текстів. Враховуючи динамічність життя сучасної людини, не завжди знаходиться час на обговорення кожної деталі по телефону, електронній пошті та месенджером. Вирішити ці проблеми допоможе веб-застосунок, що полегшить взаємодію клієнта та перекладача.

При розробці проекту дипломної роботи, метою була розробка веб-застосунку для підтримки взаємодії перекладачів та клієнтів, для цього необхідно було вирішити наступні завдання:

- Дослідити теоретичні основи розробки веб-застосунку для полегшення взаємодії перекладачів та клієнтів;
- Проаналізувати технічні рішення для побудови веб-застосунку, розглянути подібні рішення, які є в наявності.

– Спроектувати і впровадити веб-застосунок для полегшення взаємодії перекладачів та клієнтів.

Об’єктом дослідження курсової роботи є веб-застосунок для спрощення взаємодії перекладачів та клієнтів, способи розробки подібних рішень, принципи та підходи до побудови, реалізації та організації розробки веб-застосунку з необхідним для задоволення потреб кінцевого користувача функціоналом.

Предметом дослідження курсової роботи є технічні, організаційні та концептуальні принципи побудови сучасних веб-застосунку для полегшення взаємодії викладачів та клієнтів

В якості засобів для розробки були використані сучасні методи, які дозволяють будувати динамічні веб-сторінки, та їхня документація: React та JavaScript, ExpressJS для серверної частини застосунку, програмне забезпечення Microsoft VisualStudio Code, різноманітні бібліотеки для React та Express, а також СУБД MongoDB для побудови бази даних веб-застосунку та різних бібліотек.

Бакалаврська робота складається зі вступу, трьох розділів, висновку та додатків.

РОЗДІЛ 1. АНАЛІЗ ОБЛАСТІ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПІДТРИМКИ ВЗАЄМОДІЇ ПЕРЕКЛАДАЧІВ ТА КЛІЄНТІВ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд і аналіз існуючих веб-застосунків, способів і засобів створення застосунку для підтримки взаємодії перекладачів та клієнтів

Глобалізація, що знайшла свій прояв і в бізнесі, і в освіті, і в медицині, і в соціальному житті, надала діяльності перекладача значущості, якої не могло бути в закритому просторі радянської реальності. Сьогодні жодна сфера людської діяльності не обходиться без послуг перекладу. При цьому, не применшуючи важливості машинного перекладу за допомогою веб застосунків, робота фахівця, не машини, має найбільший попит на ринку послуг перекладу. Тільки людина може передати засобами іншої мови все розмаїття семантичних, лексичних, ба навіть емоційних відтінків. Та й граматичні конструкції наразі ще не завжди під силу машині. Машинний переклад вимагає дуже ретельного редагування фахівцем.

Крім того, у сфері бізнесу досить часто є потреба нотаріального завірення перекладу. Звісно, жоден перекладач не поставить свій підпис, і жоден нотаріус не завірить такий підпис під текстом машинного перекладу.

Також послуги усного послідовного або синхронного перекладу, чи то одностороннього чи двостороннього, залишаються виключною прерогативою фахівця. Тому замовники послуг звертаються до перекладачів безпосередньо.

Комунікація між перекладачем та замовником є непростою, передбачає обговорення багатьох дрібних деталей, вимагає дуже швидкого реагування з обох сторін. Замовник спілкується з перекладачем як на етапі передзамовлення, так і на етапі підписання договору або розміщення замовлення, а також в процесі виконання замовленого перекладу врегулювання процедури та здійснення оплати за послуги.

Оптимізувати процес взаємодії між перекладачем та замовником можливо за допомогою веб-застосунку, що за рахунок зручного інтерфейсу та можливості

доступу з будь-якого девайсу надає можливість для максимальної ефективності такої взаємодії.

На даний момент існують деякі сайти та застосунки, що можуть облегшити взаємодію між перекладачем та клієнтом, вони приймають файли для перекладу, запитують ваші особисті дані, щоб з вами надалі можливо було встановити зв’язок, щодо замовлення. Але розглянуті веб-застосунки для спрощення взаємодії перекладачів та клієнтів надають лише обмежений функціонал, що складається з відправки онлайн текстового документу на переклад, додавання коментаря щодо файлу для перекладача, і подальша взаємодія вже відбувається напряму з клієнтом через телефонний зв'язок, чи електронну пошту. Також, деякі з таких застосунків, надають доволі розмите уявлення про розцінку перекладу, вказуючи загальну ціну на титульній сторінці сайту, але не розраховуючи її в залежності від розміру тексту в файлі. Для проведення аналізу сучасних застосунків для спрощення взаємодії перекладачів та клієнтів, було обрано наступні веб-застосунки, доступні в мережі інтернет:

– Gengo
– Translate.com
– MotaWord

Функціонал даних застосунків було вивчено та проаналізовано. Всі веб-застосунки були написані з використанням різних мов програмування, але веб складова кожного з них залишається незмінною – HTML, CSS, JavaScript.

Розглянемо перечисленні застосунки детальніше:

Gengo – найпопулярніший з наявних застосунків для підтримки взаємодії клієнтів та перекладачів. Базується в Великобританії, але дозволяє перекладачам з усього світу підключатися до команди та працювати над замовленнями клієнтури даного сервісу. Gengo було засновано в 2008 році програмним інженером та перекладачем що працював з компанією Sony Corporation. Раніше веб-застосунок для взаємодії перекладачів та клієнтів називався myGengo, проте в 2012 пройшов через ребрендинг, змінивши назву на Gengo.

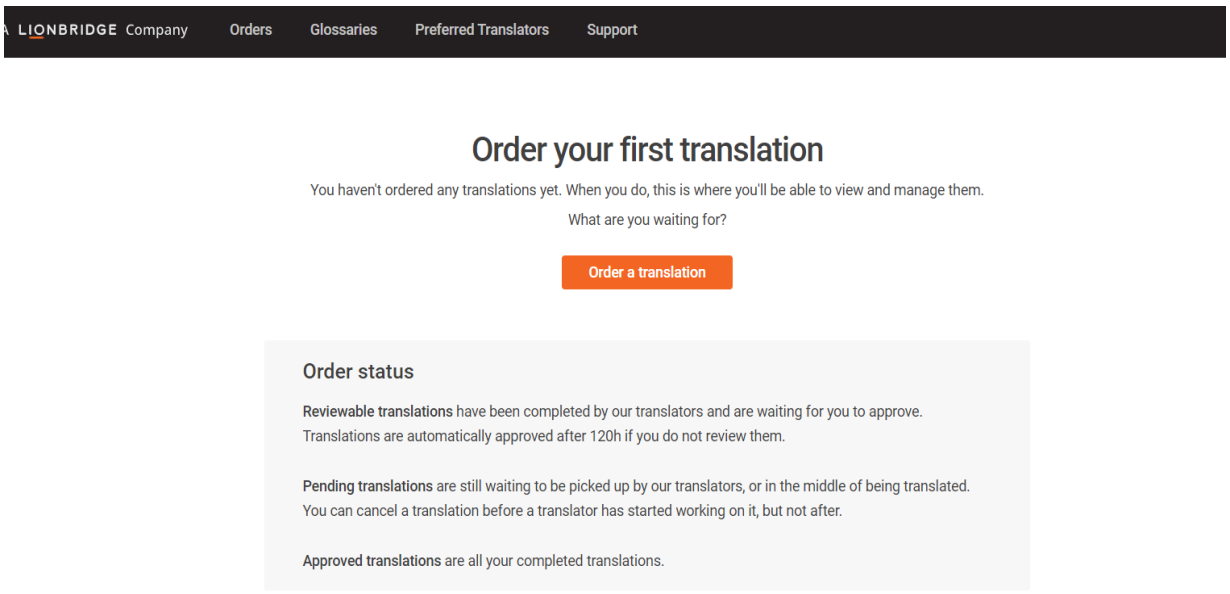

Рисунок 1.1. Головна сторінка веб-застосунку Gengo.

Цьому застосунку властивий найприємніший дизайн з усіх зазначених подібних систем. Gengo інформує користувача про тарифи зразу при оформленні замовлення, дозволяє вказувати при реєстрації, чи користувач є клієнтом, якому необхідний переклад документу, чи перекладачем, що бажає приєднатися до команди та опрацьовувати заявки.

Перевагами даної системи можна назвати:

– зручний інтерфейс веб-застосунку;

– детальне інформування користувача про наявні тарифи перекладу з різних мов;

– взаємодія між клієнтом та перекладачем в даному застосунку є біль продуманою, завдяки можливості зразу описати сутність проблеми при закріплені документу до самого замовлення.

Недоліками є:

– сам веб-застосунок існує лише англійською мовою;

– відсутність подальшої взаємодії через інтерфейс додатку після завершення оформлення замовлення, окрім як отримання інформації про його завершення.

Наступна система для аналізу – Translate.com.

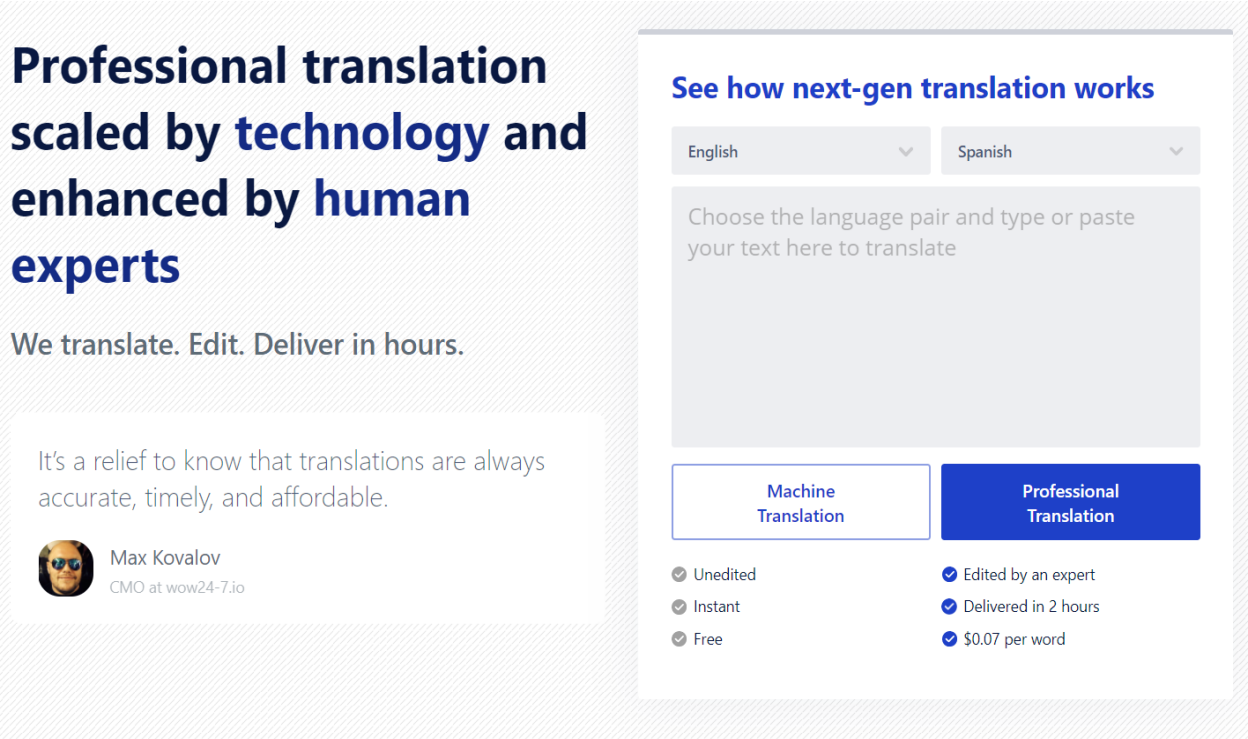

Рисунок 1.2. Головна сторінка веб-застосунку Translate.com.

Цей застосунок базується не тільки на перекладі документів, але й дозволяє замовляти переклад сайтів, даючи доступ до свого інструментарію в вигляді API, що дозволяє перекладати цілі веб-сторінки.

Перевагами даної системи можна назвати:

– простий інтерфейс веб-застосунку;
– різні можливості перекладу для різних потреб;

Недоліками є:

– сам веб-застосунок існує лише англійською мовою;
– відсутність будь-якої взаємодії між перекладачем та клієнтом через інтерфейс застосунку після завершення замовлення;
– відсутність інформування клієнта про ціну перекладу при закріпленні документу для перекладу;
– відсутня можливість додавання файлу до замовлення. Додати можна лише текст, який буде перекладено спочатку онлайн перекладачем, а надалі перевірено справжнім перекладачем-людиною. Тому даний застосунок не є гарним способом для перекладу великих документів.

Далі розглянемо систему Motaword.

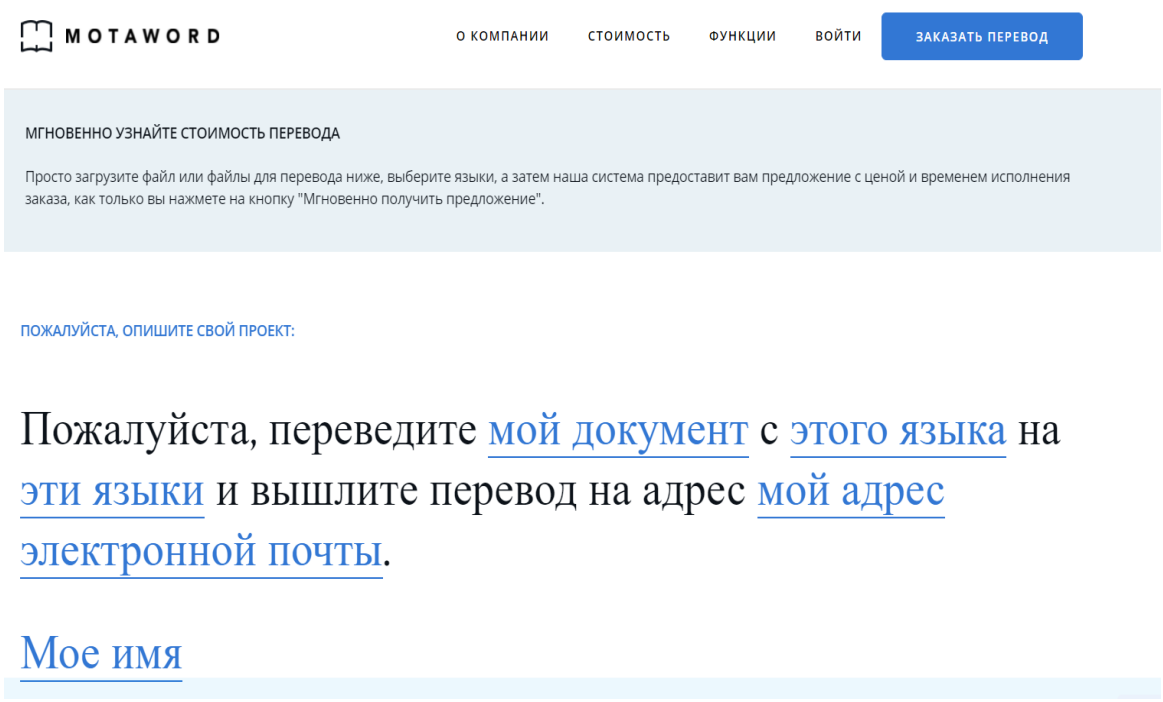

Рисунок 1.3. Головна сторінка веб-застосунку Motaword.

Motaword – сервіс перекладів, який використовує можливості зберігання даних в хмарному сервісі і об'єднує перекладачів для спільної роботи над проектами. [1]

Перевагами даної системи можна назвати:

- багатомовність сайту;
- різні можливості перекладу для різних потреб клієнтів;

Недоліками є:

- відсутність будь-якої взаємодії між перекладачем та клієнтом через інтерфейс застосунку після завершення замовлення;
- відсутність інформування клієнта про ціну перекладу при закріпленні документу для перекладу;
- відсутня можливість додавання файлу до замовлення. Додати можна лише текст, який буде перекладено спочатку онлайн перекладачем, а надалі перевірено справжнім перекладачем-людиною. Тому даний застосунок не є гарним способом для перекладу великих документів;
- не інтуїтивний інтерфейс.

У результаті проведеного аналізу існуючих веб-застосунків, найбільш зручною та бажаною у використанні є та, що має найбільш простий користувацький інтерфейс, та надає якомога більшу інформованість клієнту про наявні тарифи та прогрес роботи над перекладом.

1.2 Обґрунтування мети вирішення поставленої задачі

Відштовхуючись від висновків з проведеного аналізу наявних веб-застосунків для підтримки взаємодії перекладачів та клієнтів, а також враховуючи їх функціонал, їх переваги та недоліки, можемо дійти висновку, що:

– веб-застосунки для замовлень перекладу не надають достатнього функціоналу для взаємодії між замовником та перекладачем;

– спеціалізовані веб-застосунки не надають детальний опис ціни замовлення для окремого файлу;

– взаємодія між клієнтом та перекладачем в таких застосунках, після оформлення замовлення переходить у поштову скриньку, або телефонну розмову, а не продовжується за допомогою веб-застосунку.

Основною метою цієї курсової роботи є створення та впровадження веб-застосунку для підтримки взаємодії між клієнтом та перекладачем, який забезпечить користувача найпростішим там зрозумілим інтерфейсом, дозволить легко та швидко оформити замовлення та дізнатися його ціну, а також дозволить клієнтам дізнатися стан перекладу просто перейшовши на веб-сторінку застосунку, а перекладач зможе відслідковувати замовлення та змінювати їх стан за допомогою зручного інтерфейсу. Але основним фактором, що дає перевагу над іншими подібними веб-застосунками, є прямий зв'язок з перекладачем щодо окремого замовлення вбудований в сам веб-застосунок.

1.3 Постановка задачі. Технічне завдання на розробку

Основним завданням курсової роботи є створення веб-застосунку для підтримки взаємодії між перекладачем та клієнтом, який матиме основний функціонал необхідний як для замовника (клієнта), який зможе оформити замовлення та бути проінформованим про його стан, налагодити прямий зв’язок

з перекладачем за допомогою зручної системи коментування всередині замовлення, та дізнатися його ціну, в залежності від розміру тексту в замовленому файлі для перекладу. Для виконання заданого завдання необхідно ретельно дослідити та вирішити наступні задачі:

- проаналізувати існуючі додатки, які мають функціонал для підтримки взаємодії між перекладачем та клієнтом;
- дослідити основний функціонал, який необхідно реалізувати у веб-застосунку;
- по результату аналізу інших схожих застосунків, знайти їхні недоліки, та виправити або прибрати їх з веб-застосунку;
- створення UML–діаграми з готовим функціоналом веб−застосунку;
- створення веб−застосунку зі зручним та простим інтерфейсом;
- створити сторінку з HTML розміткою для веб−застосунку;
- стилізувати застосунок по сучасним заповітам веб-дизайну;
- забезпечити кроссбраузерну підтримку роботи застосунку для всіх браузерів на всіх пристроях;
- розробити адаптивний дизайн застосунку для відображення на мобільних телефонах;
- розробити модель бази даних, для збереження даних;
- реалізувати в застосунку всі задумки щодо функціоналу;
- протестувати веб-застосунок на продуктивність, правильну роботу функціоналу, та адаптивність.

РОЗДІЛ 2. ПРОЕКТНІ ТА ТЕХНІЧНІ РІШЕННЯ. ЗАБЕЗПЕЧЕННЯ ПРОЕКТОВАНОЇ СИСТЕМИ ЗАСТОСУНКУ

2.1 Мова програмування та допоміжні засоби проектованої системи

Для реалізації системи проекту, було використано технології та бібліотеки для створення веб−застосунків: HTML, CSS, JavaScript, React, ExpressJS для реалізації серверного спілкування з базою даних MongoDB, яка використовується для ведення бази даних. За допомогою даних технологій можна виконати поставлені задачі технічного завдання роботи. Для ознайомлення з використаними технологіями у розробці застосунку, будуть надані відповіді на питання як, і для чого була використана та чи інша технологія, фреймворк, мова, бібліотека.

2.1.1 Розмітка гіпертекстових документів за допомогою HTML

HTML (Hyper Text Markup Language) – являє собою мову розмітки сторінки, яка містить інструкції щодо розміщення її елементів, які називаються тегами (tags). Процес створення HTML-документа полягає у включенні тексту всередину тегів, що дозволяє здійснювати з ним деякі маніпуляції.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5      <title>Title goes here</title>
6    </head>
7    <body>
8
9    </body>
10 </html>
```

Рисунок 2.1. Вигляд базового документу HTML.

Теги – це послідовність символів, які є контейнером для тексту або таких самих тегів, які в собі можуть вміщувати також текст або теги. Вони починаються зі знаку «менше» (<) та закриваються знаком «більше» (>).

Завдяки тегам HTML можна вказати розмір літер, що знаходяться всередині, для прикладу можна задати заголовки – H1 – для головного заголовку, H2, H3 – для індексації заголовків поменше. Браузер (який виконує роль клієнтської програми) вибирає розмір тексту заголовків, який перевищує звичайний. Саме завдяки браузеру, документ написаний за допомогою HTML здатен відображатися на будь-якій платформі, необхідно мати лише браузер. Недоліком HTML є факт, що на браузерах та екранах документ виглядає по-різному.

Неможливо знайти дві однакові сторінки HTML - всі вони включають унікальні комбінації тегів і вмісту. У той же час будь-яка правильно спроектована сторінка HTML вимагає наявності однієї і тієї ж базової структури, що включає в себе:

– пропозиція, що ідентифікує документ як документ HTML;

– заголовок документа;

– тіло документа.

При створенні будь-якого документа HTML потрібно починати саме з цих трьох конструкцій - наповнити їх вмістом і розміткою можна в будь-який момент. [2]

Браузери читають код HTML за наступними правилами правилами:

– теги форматування можуть бути написані малими і / або прописними буквами;

– більшість тегів необхідно після оголошення закривати (тобто – парами);

– пропуски, пробіли та все, що не відображається на сторінці ігнорується;

– всі теги розділяються на: базові теги, теги форматування, структурні теги, покажчики і тому подібне.

Команда, яка відкриває тег активує даний ефект, а та що закриває – вимикає його. Пара тегів називається контейнером, оскільки відкриваючи тег ефект застосовується до елементів всередині тегу до тих пір, поки цей тег не закриється.

2.1.2 Каскадні таблиці стилів CSS

Стилізуючи HTML документи зазвичай використовують CSS − це технологія, яка здатна змінювати, розширювати та розробляти зовнішній вигляд документа.

CSS являє собою каскадні таблиці стилів, які є окремим кодом, що розширює можливості HTML, які дозволяють брати існуючі HTML-теги та перевизначити їх. HTML існував раніше CSS – який було створено в 1996 році.

В першу чергу HTML слугував як засіб створення текстових документів. Для користувачів, якими були в першу чергу військові і вчені головним був вміст документу, а не його оформлення. В перших версіях була відсутня будь-яка можливість складного форматування тексту, або графічного оформлення картинками. Але через деякий час користувачі мережі інтернет помітили, що сторінки виглядають негарно, тому з’явилася потреба в стилізації.

Так виникли каскадні таблиці стилів, створені для полегшення роботи веб-дизайнерів. Останньою версією є CSS3. Її головна особливість – масштабована векторна графіка

Каскадні (багаторівневі) таблиці стилів - cascading style sheets (CSS) - це потужний стандарт на основі текстового формату, який визначає розміщення елементів сторінка в браузері.

CSS розроблений таким чином, щоб забезпечити максимальний рівень контролю над розміщенням тексту і графічних елементів. Він забезпечує належний рівень оформлення, організації і контролю під час розробки сторінки, що не дозволяє HTML в чистому вигляді.

Якщо формат HTML надає інформацію про склад документа, то таблиці стилів повідомляють як він повинен виглядати. Стиль CSS включає в себе всі

типи елементів дизайну: шрифт, фон, текст, кольори посилань, поля і розташування об'єктів на сторінці. [3]

«Каскадний» CSS означає, що на одній сторінці HTML можуть використовуватися різні стилі. Браузер, який підтримує таблиці стилів, буде слідувати їх порядку (як по каскаду), інтерпретуючи інформацію стилів один за одним. Це означає, що ви можете використовувати три типи стилів (впроваджений, пов'язаний, вбудований) і браузер буде інтерпретувати спочатку пов'язані, потім впроваджені і, нарешті, вбудовані стилі. Навіть якщо до всього вузла будуть застосовані зразки стилів, можна буде управляти окремими аспектами сторінок за допомогою впроваджених стилів, а окремими областями всередині цих сторінок - за допомогою вбудованих стилів.

Значним написання CSS є успадкування (inheritance). Це означає, якщо у елемента не вказано конкретний стиль, певний стиль буде успадкований цими елементами сторінки HTML. Наприклад, якщо ви застосуєте певний колір тексту в тезі <p>, то всі теги всередині цього абзацу успадковують цей колір, за відсутності іншої домовленості.

Існує три методи для застосування таблиці стилів до документа HTML:

Вбудований (Inline). Метод Inline дозволяє додати стиль до будь-якого тегу HTML. Вбудований метод надає можливість максимального контролю усіх властивостей певної Web-сторінки. Припустимо, необхідно задати зовнішній вигляд окремого абзацу. Можна просто додати атрибут style до тегу абзацу, і браузер відобразить цей абзац за допомогою параметрів стилю, доданого в код.

Впроваджений (Embedded). Впровадження дозволяє контролювати всю сторінку HTML. При використанні тега <style>, поміщеного всередині розділу <head> сторінки HTML, в код вставляються деталізовані атрибути стилю, які будуть застосовуватися до елементів що мають необхідний атрибут.

Пов'язаний (Linked або External). Пов'язана таблиця стилів являє собою документ який дозволяє створювати зразки стилів (master styles), які можна потім застосовувати до всього вузла. Основний документ таблиці стилів (style.css) створюється Web-дизайнером. Цей документ містить стилі, які будуть єдиними

для всього Web-вузла (неважливо, містить одну сторінку або тисячі сторінок). Будь-яка сторінка, пов'язана з цим документом, буде використовувати зазначені стилі.

Рисунок 2.2. Демонстрація типів інтеграції CSS.

Таблиці стилів мають певний порядок (синтаксис), в іншому випадку вони не можуть працювати.

Синтаксис всіх методів, які використовуються для застосування стилів до документа HTML, практично однакові.

Покажчик (Selector). Покажчик є елементом, до якого будуть застосовуватися призначаються вами атрибути. Це може бути просто тег типу заголовка (H1) або абзацу (P).

Властивість (Property). Властивість визначає покажчик. Наприклад, якщо в якості покажчика обраний абзац, ви можете включити властивості, що визначають цей покажчик. У властивості входять такі елементи, як поля, шрифти і фонові зображення. У таблицях стилів існує багато властивостей, які можна використовувати для того, щоб визначити покажчик.

Значення (Value). Значення визначають властивості. Припустимо, що у вас є заголовок першого рівня H1 (покажчик) і ви включаєте властивість type family (сімейство шрифту). Шрифт, який насправді буде застосований до зазначеного фрагменту, задається значенням цієї властивості. [4]

Отже, каскадна таблиця стилів - це набір правил форматування тегів HTML.

2.1.3 Мова програмування JavaScript

JavaScript входить в трійку технологій, які повинен знати будь-який веб-розробник: мова розмітки HTML, що дозволяє визначати вміст веб-сторінок,

мова стилів CSS, що дозволяє визначати зовнішній вигляд веб-сторінок, і мова програмування JavaScript, що дозволяє визначати поведінку веб-сторінок. [5]

В момент розцвіту інтернету та популяризації веб-сторінок у розробників виникла проблема, яка заключалася в розробці архітектури спілкування веб-застосунку у вигляді «клієнт – сервер». Сторінки можливо генерувати як на стороні сервера так і на стороні клієнта. У 1995 році було створено мову програмування, стандарту ECMAScript для управління сторінками на клієнтській стороні, яка отримала назву JavaScript.

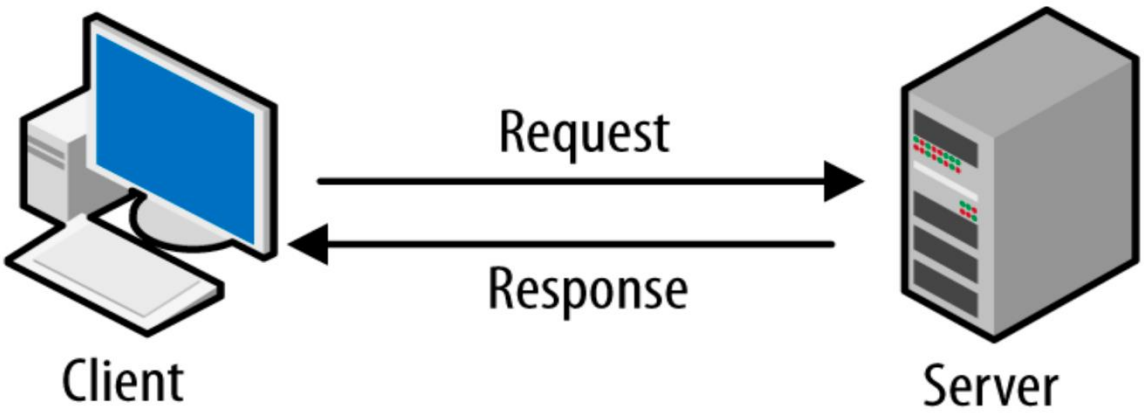

Рисунок 2.3. Демонстрація клієнт-серверної архітектури побудови веб-застосунків.

В сучасні часи, JavaScript − це мова для керування сценаріями перегляду HTML сторінок на клієнтській стороні. Ідея JavaScript полягає в можливості змінити значення атрибутів HTML−контейнерів та властивостей середовища показу в процесі самого перегляду HTML−сторінки користувачем, а також надати функціонал для їх обробки. При цьому це можна виконати без перезавантаження сторінки.

На практиці можна навести багато прикладів подібного використання мови JS, наприклад: відображення нового віконця в середині іншого, зміна картинки при натисканні кнопки, оновлення сторінки при натисканні кнопки тощо.

Фрази в мові JavaScript називаються виразами, то повні речення називаються інструкціями. У програмному коді, рядки, що знаходяться крапкою з комою, є інструкціями. Між інструкціями і виразами багато спільного. Грубо кажучи, Вираз - це конструкція, яка обчислює значення, але нічого не де гавкає: вона ніяк не змінює стан програми. Інструкції, навпаки, не мають значення (або

їх значення не представляє інтересу для нас), але вони змінюють станами програми.[5]

Для керування веб-сторінками на стороні клієнта застосовується об'єктна модель документа. Ця модель характеризується тим, що HTML−контейнер − це об'єкт, що має три основні характеристики:

- події
- властивості
- методи

Зв'язок між різними сторінками одного веб-сайту та браузером виконується за допомогою об’єктної моделі. Об’єктною моделлю вважається набір методів, об’єктів, подій та властивостей що виконуються всередині браузера у вигляді HTML коду та вихідного тексту сценарію на сторінці. Завдання розробника – задати необхідний алгоритм дій, для відображення компонентів відвідувачу сторінки. Коли JavaScript код виконається, тоді весь заданий алгоритм дій програмістом буде запущений, або буде очікуватиме на активацію користувачем.

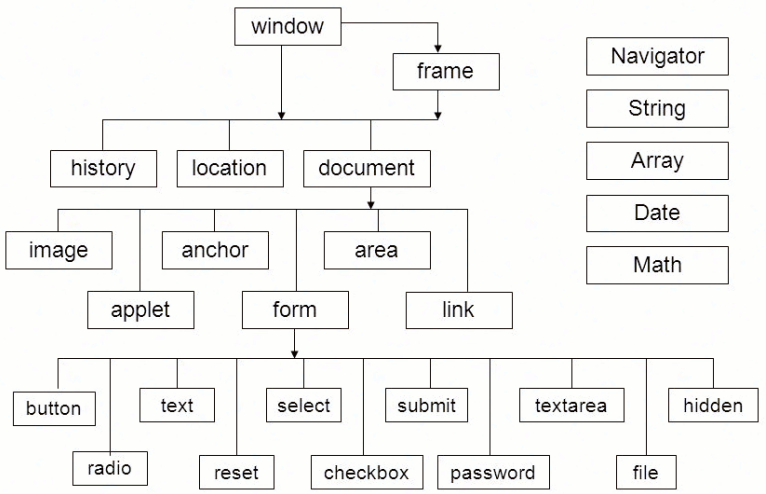

Рисунок 2.4. Об’єктна модель JavaScript

2.1.4 JavaScript-бібліотека для створення користувацьких інтерфейсів React

Для створення інтерфейсу користувача використовувалася бібліотека ReactJS − це бібліотека JavaScript, що використовується для інтерфейсу

користувача. Її розробила компанія Facebook, яка ж її і використала для розробки однойменної соцмережі. Випуск першої версії бібліотеки React приходиться на 2013 рік.

З моменту свого створення React існував в основному для розробки односторінкових динамічних веб-сайтів, проте пізніше його розширили завдяки React Native, що дозволило використовувати інструментарій React для створення окремих мобільних додатків з побудовою користувацького інтерфейсу на архітектурі React.

React є ідеальним інструментом для розробки масштабних та функціональних веб-сайтів та веб-додатків, завдяки своїй функціональності, та обширного співтовариства розробників, що розміщують свої власні додаткові бібліотеки на відкритих репозиторіях, завдяки чому будь-який програміст може спростити собі роботу не переписуючи загально використований код для тих чи інших функцій застосунку.

Вся структура веб-сторінки може бути представлена за допомогою DOM − організації HTML елементів, якими можна маніпулювати, змінювати, видаляти або додавати нові. React основується на принципі віртуального DOM, адже це дозволяє оптимізувати роботу веб-додатку.

Розмітка сторінок в React не закінчується на простій генерації HTML-коду. Для генерації розмітки сторінок використовується власне розширення мови JavaScript, що называеється – JSX.

JSX дозволяє присвобвати змінним HTML розмітку, тобто, на практиці JSX генерує елементи React.

React виходить із принципу, що логіка рендеринга нерозривно пов'язана з іншою логікою UI: з тим, як обробляються події, як стан змінюється в часі і як дані готуються до відображення. [6]

Коли розробник хоче змінити щось в документі, він вносить зміни спочатку в віртуальний DOM. Після цього новий стан віртуального DOM, який розробники React називають «стейтом», зрівнюється з поточним станом. Якщо

ці стани відрізняються, то React виконує мінімальну кількість операцій, що необхідні для відображення реального DOM.

Такий спосіб обробки елементів сторінок веб-застосунку дозволяє зробити сайт набагато динамічнішим та швидким, та змусити його працювати набагато ефективніше, аніж якою вона б була, якби застосунок розроблявся на стандартному HTML.

В розробці застосунку для підтримки взаємодії між перекладачем та клієнтом було використано нове архітектурне рішення для побудови веб-застосунків для React - хуки (Hooks).

2.1.5 Фреймворк для React Material UI

Існує багато фреймворків для React, що дозволяють спростити роботу дизайнера над будь-яким проектом, надаючи потужний інструментарій для розробки дизайну застосунку, за допомогою використання власних бібліотек. В розробці застосунку для підтримки взаємодії перекладачів та клієнтів було використано Material UI.

Material UI – це найпопулярніший в світі фреймворк для розробки користувацького інтерфейсу на React.[7]

2.1.6 Сервер для розташування застосунку

Для створення веб-серверу, на якому буде розташована клієнт частина односторінкового застосунку на React та буде відбуватися виконання її коду – використовується Node.js.

Node.js представляє середовище виконання коду написаного на JavaScript, яке побудоване на основі движка JavaScript Chrome V8, який дозволяє транслювати виклики на мові JavaScript в машинний код. Node.js перш за все призначений для створення серверних додатків на мові JavaScript. Хоча також існують проекти по написанню десктопних додатків (Electron) і навіть зі

створення коду для мікроконтролерів. Але перш за все ми говоримо про Node.js, як про платформу для створення веб-додатків. [8]

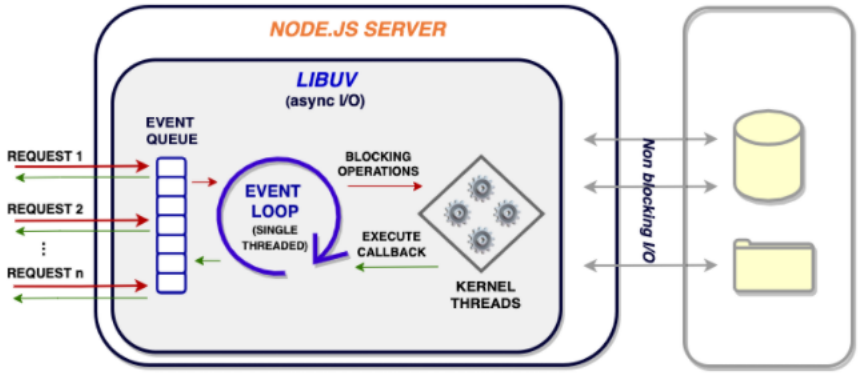

Рисунок 2.5. Архітектура роботи серверу Node.js

2.1.7 Система управління базами даних застосунку

Після закінчення створення інтерфейсу користувача, необхідно забезпечити веб-застосунок необхідним функціоналом по запису та виведенню даних. Для цього ми використаємо базу даних MongoDB та будемо взаємодіяти з нею за допомогою фреймворку веб-додатків для NodeJS Express.

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які притаманні об'єктно-реляційних баз даних.

Дані завжди зберігалися в реляційних базах даних, таких як MySQL, Oracle, PostgresSQL. При цьому було не так важливо, а чи підходять реляційні бази даних для зберігання даного типу даних чи ні.

На відміну від реляційних баз даних MongoDB пропонує документо-орієнтовану модель даних, завдяки чому MongoDB працює швидше, має кращу масштабованість, її легше використовувати. [9]

База даних – це просто набір структурованих даних. MongoDB це документно-орієнтована система для керування цими базами даних, вона дозволяє за допомогою простих для вивчення запитів діставати, змінювати, додавати та видаляти дані з бази даних.

2.1.8 Фреймворк Express для NodeJS на моделі MVC

Express – це легка платформа веб-додатків, яка допоможе організувати ваш веб-додаток на архітектурі MVC на стороні сервера.

Його можна використовувати з базою даних, наприклад MongoDB, щоб забезпечити бекенд для вашого застосунку на Node.js. Express.js в основному допомагає вам управляти всім, від маршрутизації, до обробки запитів і відображення сторінки.

MVC розшифровується як модель-відображення-контролер (від англ. Model-view-controller). Це спосіб організації коду, який передбачає виділення блоків, що відповідають за вирішення різних завдань. Один блок відповідає за дані додатки, інший відповідає за зовнішній вигляд, а третій контролює роботу додатка. Компоненти MVC:

Модель – це компонент, що відповідає за дані, а також визначає структуру програми. Наприклад, якщо ви створюєте додаток зі списком завдань для роботи, код компонента model визначатиме список завдань і окремі завдання.

Відображення – це компонент відповідає за взаємодію з користувачем. Тобто код компонента view визначає зовнішній вигляд програми і способи його використання.

Контролер – цей компонент відповідає за зв'язок між model і view. Код компонента controller визначає, як сайт реагує на дії користувача. По суті, це мозок MVC-додатку. [10]

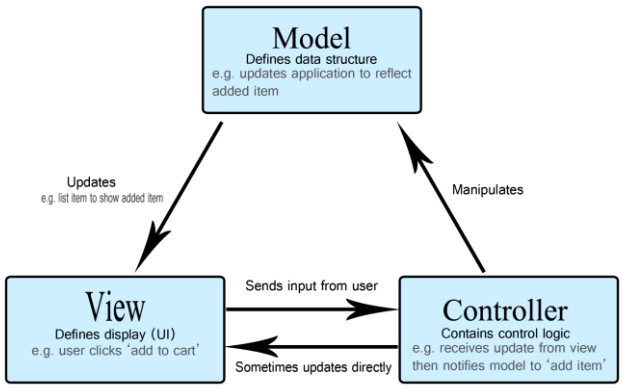

Рисунок 2.6. Модель MVC

2.2 Структурний вигляд сторінок веб-застосунку

Структура сторінок веб–застосунку та їх зміст повинен задовольняти умови бакалаврської роботи, тому було створено основні сторінки з реалізованим функціоналом, який буде вирішувати такі задачі як: реєстрація та авторизація користувачів, вибір роли перекладача чи клієнта, оформлення заявки щодо перекладу, прорахунок ціни перекладу в залежності від розміру файлу, перегляд статусу та замовлень, та зручний спосіб коментування свої замовлень для зв’язку з перекладачем та контролем над ходом перекладу.

Структура сайту складається з таких сторінок:

- головна презентаційна сторінка сайту;
- сторінка для реєстрації нового користувача;
- сторінка для авторизації користувача;
- сторінка для перегляду замовлених перекладів для клієнта;
- сторінка з усіма перекладами в черзі для перекладача;
- сторінка профілю користувача;
- сторінка для оформлення заявки на переклад;
- сторінка подяки;
- сторінка опису додатку

Для зручності користувача ці сторінки було розроблено за принципом розробки веб-застосунків React, що дозволило набагато більш плавну роботу застосунку, та відчуття використання повноцінного застосунку поза браузером. Такий результат досягається за допомогою динамічного рендеру сторінок – елементи сторінки кожен раз не перезавантажуються, а прибираються частково або додаються нові. Це дає можливість отримувати інформацію з вказаних сторінок без перезавантаження сторінки, а лише заміною її значних елементів. Розглянемо зовнішню схему сторінок застосунку для нового клієнта, якого цікавить переклад його документу (рис 2.7).

Рисунок 2.7. Структура веб-сторінок застосунку для під

2.3 Можливості та функціонал розробленого веб-застосунку

Для вирішення завдань проектованої системи було проаналізовано декілька існуючих веб-застосунків які надавали сервіс для взаємодії перекладачів та клієнтів, серед них виділено головний функціонал необхідний застосунку, що буде забезпечувати необхідний рівень взаємодії та зручності у використанні, та не буде навантажувати систему зашкоджуючи продуктивності веб-застосунку. Серед основного функціоналу розроблено для веб-застосунку варто виділити наступне: система коментування замовлень для спілкування між перекладачем та клієнтом, список власних замовлень та наявних замовлень для перекладача, та сторінку оформлення замовлення з вбудованим калькулятором розрахунку ціни замовлення, що дозволить клієнту ознайомитися з наявним тарифом ще до оформлення замовлення. Детальніше про кожен елемент функціоналу застосунку можна дізнатися в таблиці 2.1.

Таблиця 2.1

Функціонал веб−застосунку для підтримки взаємодії перекладачів та клієнтів

Назва функціоналу	Опис функціоналу
Авторизація користувача	Ідентифікація користувача шляхом порівняння введених даних з даними в БД. Авторизація необхідна для отримання повного доступу до сайту
Реєстрація користувача	Функція, яка дозволяє створювати обліковий запис користувача та надає доступ до сайту авторизованому користувачу
Створення замовлення	Функція створення замовлення дозволяє будь-якому користувачу-клієнту додати своє замовлення в базу даних, обравши необхідні послуги.
Коментування замовлення	Функція коментування замовлення надає можливість клієнтам та перекладачам спілкуватися, щодо конкретного замовлення всередині додатку, без необхідності користуватися додатковими засобами зв’язку

Продовження таблиці 2.1

Зміна статусу замовлення	Функція, що надає можливість перекладачу змінювати статус виконання замовлення, для інформування клієнтів.
Розрахунок ціни замовлення	Після загрузки документу для перекладу при оформленні замовлення, користувач інформується про ціну в залежності від кількості символів
Видалення коментарів	Замовник може видалити коментарі при спілкуванні з перекладачем
Прикріплення файлу до коментаря	До повідомлення в коментарях до замовлення можна прикріпити документ для кращої взаємодії.

Кінець таблиці 2.1

Це є головний функціонал, що вимагається від будь-якого веб-додатку, що підтримує взаємодію перекладачів та клієнтів. Він дозволить без труднощів переглядати список замовлень, надасть перекладачам доступ до повного списку замовлень, коментувати світлини інших користувачів, залишати свої оцінки до світлин. Забезпечується ж цей функціонал завдяки алгоритму спілкування клієнт-сервер, де з боку клієнта на сервер передаються дані, які вводить користувач, а з серверу приходить відповідь з отриманим результатом з бази даних (див. рис. 2.8.).

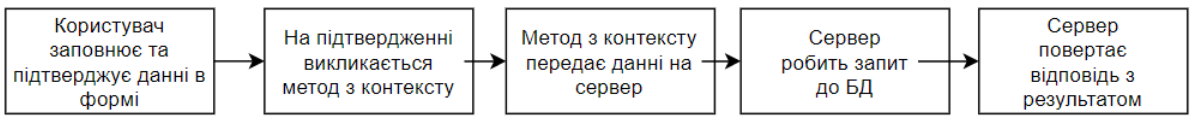

Рисунок 2.8. Схема алгоритму роботи запитів веб-застосунку для підтримки взаємодії перекладачів та клієнтів

2.4 Схематична модель побудованої бази даних

База даних − це сукупність простих структур, що є організовані за відповідними правилами, концепціями які дозволяють зберігати та здійснювати різноманітні операції з великими обсягами даних, здійснювати вибірку, управляти ними, сортувати, видаляти, редагувати та інші дії з ними. Для розробки дипломної роботи були створені наступні таблиці:

– Користувачі
– Замовлення
– Коментарі

Кожна з таблиць має ключі завдяки яким можна показати всі коментарі одного замовлення, або всі замовлення окремих користувачів. Всі дані зберігаються в вигляді записів різних типів, створення та дії з якими здійснюються за допомогою ExpressJS. З даними в таблицях можна здійснювати різноманітний набір дій: редагувати, видаляти та інші. Також в кожній таблиці є набір іменованих полів, які зберігають корисну інформацію починаючи з імені користувача закінчуючи оцінкою замовлення від клієнтів. Детальна схема побудованої бази даних зображена на рис. 2.9

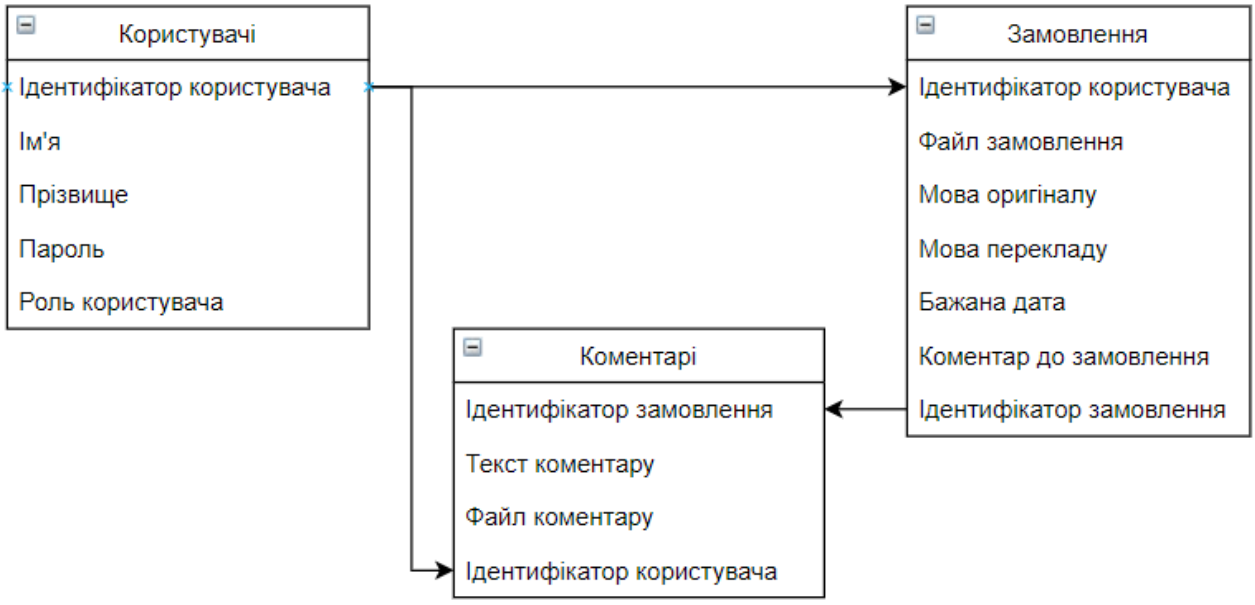

Рисунок 2.9. Схема бази даних веб-застосунку для підтримки взаємодії перекладачів та клієнтів

2.5 Опис бази даних

Для заповнення та взаємодії з базою даних було використано фреймворк ExpressJS. Було вирішено використати Express тому, що це є одним з сучасних найпопулярніших фреймворків веб-застосунків для взаємодії з базами даних, який використовується по всьому світу в багатьох проектах. На Express були написані кінцеві точки (API endpoint), до яких React звертається, при відправленні користувачем будь якого запиту. Наприклад: при створенні нового замовлення, дані відправляються з форми веб-документу через функцію fetch() до серверу, написаного на мові JavaScript з використанням Express, де передана інформація з форм обробляється та перетворюється в запит до бази даних, який вже виконує деякі маніпуляції зі структурами даних всередині бази даних. React дозволяє динамічно отримувати відповідь з серверу, та оновлювати стани яких-небудь елементів сторінки, не оновлюючи саму сторінку. Тому обробка запитів відбувається плавно і непомітно для користувача. Динамічне відслідковування стейтів забезпечують функції-хуки, які були введені в React в пакетах останніх років. А саме хук стану (useState).

Саме завдяки хукам стану (useState), ми можемо змінювати стан і ділити та динамічно оновлювати елементи сторінки, яким присвоюється значення стану.

Всі операції з даними виконуються через контекст, в якому знаходяться все інші компоненти веб-додатку. Наприклад, через компонент який відповідає за додавання світлини, виконується виклик асинхронної функції, яка знаходиться в контексті, в самій функції вказується ендпоїнт (ссилка до серверу, по якій здійснюється запит) та викликається запит до нього, звернувшись до ендпоїнта та передавши йому значення з компоненту, сервер перевіряє тип запиту, а потім що це за запит, функція яка далі обробляє дані до форми запитів до БД, і зразу ж повертає результат запиту, який записується в useState(). Завдяки такій асинхронній архітектурі побудови веб-додатку, надається можливість динамічно працювати зі структурами БД без перезавантажень сторінки при запитах.

Компоненти React отримують дані завдяки функціям map, які проходяться по записаним в стан (state) масивам, та виводять дані з них в елементи компоненту. Надалі, користувач може здійснювати вибірку по отриманим даним або додати нові, видалити або редагувати чинні.

```
    });
    try {
        const savedUser = await user.save();
        res.send({ user: user._id });
    } catch (err) {
        res.status(400).send(err);
    }
});

//LOGIN
router.post('/login', (req, res) => {
    //LETS VALIDATE THE DATA BEFORE WE A USER
    const { error } = loginValidation(req.body);
    if (error) return res.status(400).send(error.details[0].message);
        //Checking if the email exists
        const user = await User.findOne({ email: req.body.email });
        if (!user) return res.status(400).send('Email or password is wrong');
        //PASSWORD IS CORRECT
        user.password

});
```

Рисунок 2.10. Файл з перевірками запиту при зверненні до сервера на прикладі авторизації користувача

2.6 Елементи побудови системи

Основними елементами для побудови системи було використано компоненти бібліотеки React. За допомогою компонентів бібліотеки React можна розділити частини інтефейсу застосунку на окремі, повторно використовувані частини та використовувати кожен окремо на різних сторінках. Концепція компонентів React схожа на функції JavaScript. Вони отримують вхідні дані та повертають React-елементи, в яких описано, що браузер має показати.

Всі компоненти можуть посилатися на інші компоненти у своєму виводі. Завдяки такій системі, можна використовувати компонент абстрагуючи його від даних, які він відображає, та відображати на ньому все, що необхідно в момент виводу компонента на екран, чи отриманні відповіді з веб-серверу застосунку.

До компонентів можна віднести такі елементи веб−сторінки як: верхнє меню, форма, списки, віконця, спливаючі віконця.

В спроектованому застосунку використовується близько 20 різних компонентів (таб. 2.2), але найголовніший з них всіх, це − App. В нього повертаються всі компоненти які будуть показані на веб-сторінці, також за допомоогою бібліотеки React-router-dom забезпечується розподілення рендерингу різноманітних компонентів сторінки в залежності від адреси, на якій знаходиться користувач. Це надає можливість для роботи з одним і тим же компонентом наповненим різними даними, які дістаються за допомогою запитів до серверу на основі GET запитів до серверу та відповіді на ці запити. Таким чином, наприклад, можна ітеруватися по об’єктам даних які дістаються з серверу не перезавантажуючи сторінку браузеру.

Таблиця 2.2

Компоненти React веб-застосунку

Назва компонента	Опис компонента
App	Головний компонент в який повертаються всі інші компоненти. В цьому компоненті відбувається підключення компонентів створених розробником для їхнього показу на головному екрані веб-застосунку та додаються дані із глобального середовища. Також там розробляються дороги (тобто адреси), по яким користувач переходить при навігації по застосунку.
Order	Компонент, що відображає форму для створення нового замовлення клієнтом, в нього входить кілька інших компонентів. Наприклад: Price, Form.

Продовження таблиці 2.2

AuthContext	Компонент, який є контекстом, що обгортає інші компоненти та контейнери. В ньому обробляються всі функції та стани, які пов’язані з авторизацією користувача
OrderContext	Компонент, який є контекстом, що обгортає інші компоненти та контейнери. В ньому обробляються всі функції та стани, які пов’язані з додаванням, зміною, видаленням та ін. діями з світлинами.
Header	Компонент, за відображення верхньої панелі на екрані додатку, яка надає можливість відстежувати, де саме зараз знаходиться користувач користуючись навігацією по додатку.
Message	Компонент, який відповідає за показ доступних сторінок додатку, показує де саме користувач знаходиться, та дозволяє вийти зі свого облікового запису.
PrivateRoute	Системний компонент, який обгортає всі інші компоненти, та робить їх приватними, що означає, що для не авторизованого користувача вони відмальовуватися не будуть, а самого юзера перешле на сторінку з авторизацією.
OrderComment	Компонент, який відповідає за оформлення та функціонал полів для вводу на сторінці з авторизацією та реєстрацією
Form	Компонент, який виводить кнопки та форму на екранах реєстрації та авторизації

Продовження таблиці 2.2

Alert	Компонент, який відповідає за показ повідомлень з помилками. Наприклад, коли користувач ввів ссилку на відео в невірному форматі.
Api	Системний компонент, в якому зберігаються дані для підключення до серверу.
Price	Компонент, в якому користувач задає дані для створення своєї світлини, такі як: відео, назва, опис.
Status	Компонент з відео, назвою та описом світлини, з можливістю коментування та редагування самого себе.
UserStatus	Компонент, який надає список світлин, які створив авторизований користувач.
OrderList	Компонент, який надає список світлин усіх користувачів.
Title	Компонент який є домашньою сторінкою сайту, не виконує ніякого функціоналу, лише вітає користувача по його нікнейму.
Thanks	Системний компонент, що відповідає за показ зображення або напису під час завантаження даних з глобального середовища.
SignUp	Компонент з формою для реєстрації.
SignIn	Компонент для авторизації та необхідним для цього функціоналом.

Саме з цих компонентів складається веб−сайт по веденню свого відеоблога. Також з компонентами використовуються серверні ExpressJS файли для обробки запитів, такі як:

– **app.js** – головний файл що слугує вхідною точкою для серверу, тут вказуються параметри серверу, та підключається база даних;

В папці config зберігаються налаштування для веб-серверу, там знаходяться наступні файли:

– **index.js** – конфігурація для зв’язку з базою даних;

– **server.js** – містить в собі функції необхідні для обробки дій пов’язаних з коментарями.

В папці routes знаходяться файли:

– **users.js** − містить в собі функції необхідні для обробки дій пов’язаних з авторизацією користувачів.

– **comments.js** − містить в собі функції необхідні для обробки дій пов’язаних із коментарями до замовлень.

– **orders.js –** містить в собі функції необхідні для обробки дій пов’язаних із замовленнями

В папці models знаходяться:

– **user.js** – визначає структуру організації даних користувача

– **comment.js** − визначає структуру організації коментарів до замовлень

– **order.js** – визначає структуру даних замовлень

2.7 Програмне забезпечення

Проектована система присвячена розробці веб-застосунку з використанням мов HTML, CSS, JavaScript та бібліотеки React, серверна частина була розроблена за допомогою фреймворку ExpressJS та базою даних MongoDB. Розроблявся веб-застосунок на операційній системі Windows 10, а саме за допомогою інтегрованого середовища розробки Visual Studio Code. Інтерфейс веб-застосунку було спроектовано в графічному редакторі Figma та реалізовано за допомогою середовища розробки.

Сама бібліотека React є потужним та досить складним інструментом для побудови веб-застосунків під будь-якої систему. Цей інструмент надає

можливість створювати інтерактивні сторінки, налагодити спілкування з сервером та покращити швидкість завантаження веб-сайту. Не дивлячись на те, що вона досить складна, для її використання програмісту потрібно набагато менше зусиль, якщо він вміє правильно працювати з цим інструментарієм.

Завдяки тому, що проектована робота являє собою веб-застосунок вона має мінімальні рекомендації до програмного забезпечення користувача. Для повноцінного користування цим веб-застосунком досить будь-якого комп'ютера, планшета або смартфона, зі встановленим браузером на ньому, та підключенням до мережі інтернет.

2.8 Структурний опис веб-застосунку

У спроектованому веб-застосунку було використано певні компоненти, які спілкуються між собою. Завдяки схемі, що можна побачити нижче на рис. 2.11, можно побачити розроблену структуру веб-застосунку – в момент до їх компіляції та зтиснення.

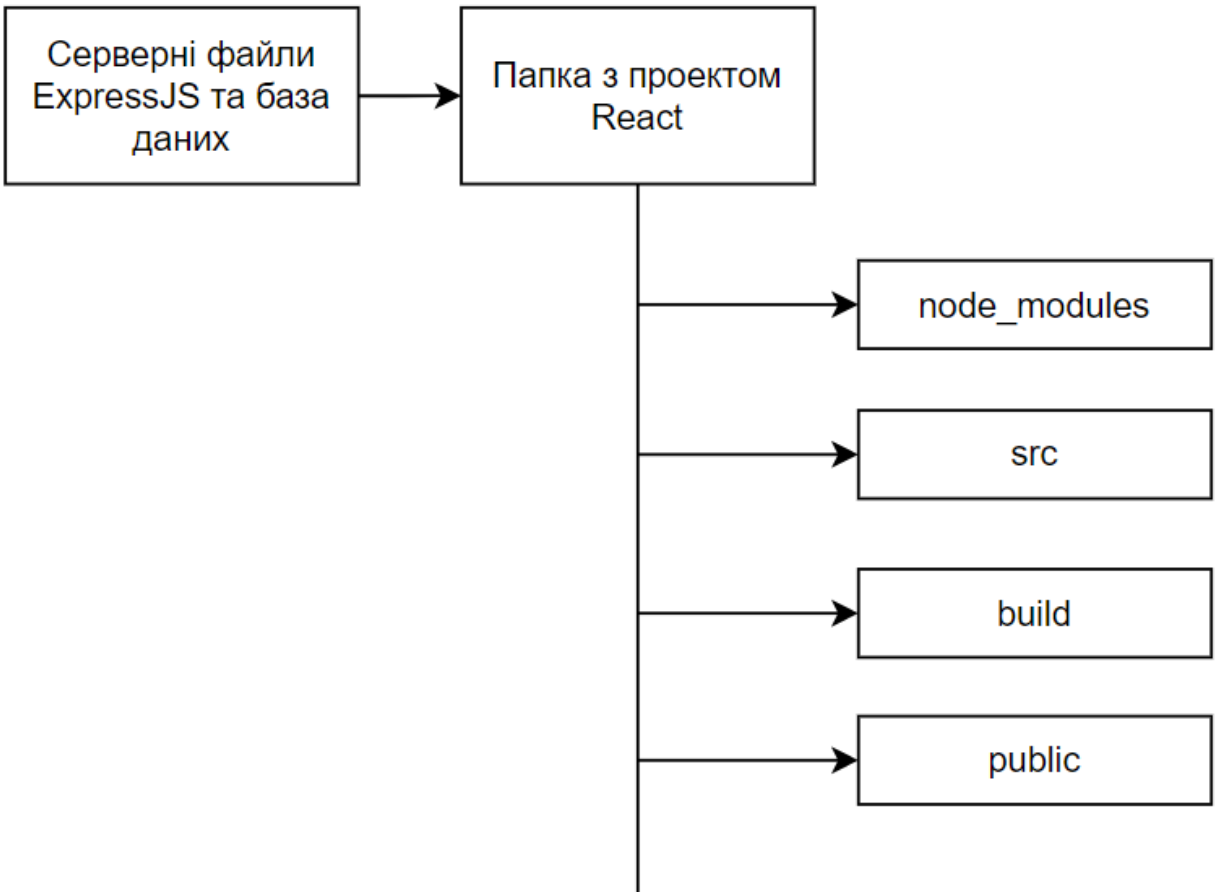

Рисунок 2.11. Файлова система з глобальними даними веб-застосунку

Детальний опис папок можливо побачити в таб. 2.3.

Таблиця 2.3

Опис системних папок веб-застосунку для підтримки взаємодії

Назва папки	Опис
Коренева папка	Головна папка, яка в собі зберігає всі основні файли для роботи веб-застосунку, та для запуску серверу з ним.
node_modules	Папка node.js, яка слугує для зберігання усіх модулів, пакетів та бібліотек доданих з репозиторію npm чи іншим способом
public	Папка з html розміткою застосунку, яка слугує місцем для підключення всіляких бібліотек, мета тегів, шрифтів за допомогою CDN.
src	Папка яка вміщає в собі основу веб-застосукнку, його контексти, компоненти, контейнери та іншими файли пов’язаними з написанням коду на React
build	Папка з веб-застосунком після проходження упаковки за допомогою утиліти Webpack.
Серверні файли ExpressJS та база даних	Папка з серверними файлами розробленими за допомогою фреймворку ExpressJS, з яких отримуються кінцеві точки API та реалізовується робота з БД.

Дана побудова файлової системи додатку є спільною для всіх проектів написаних за допомогою бібліотеки React, та будь-яких проектів з використанням NodeJS. Таке розподілення дозволяє відрізнити папки які мають різне призначення в даних проектах.

РОЗДІЛ 3. ПОКРОКОВИЙ ОПИС РОБОТИ ВЕБ-ЗАСТОСУНКУ ТА ТЕСТУВАННЯ

3.1 Інструкція веб-застосунку для комп'ютерів

Даний веб-застосунок надає змогу клієнтам та перекладачам більш вдало підтримувати зв'язок без переходу на інші види зв'язку, надаючи всі необхідні функції для взаємодії через власний інтерфейс веб-застосунку. Таку змогу надають компоненти застосунку, які дозволяють: оформляти замовлення користувачу, переглядати його стан, коментувати його для зв'язку з перекладачем. Перекладачу ж надана можливість переглядати замовлення клієнтів, змінювати їх стан, прикріпляти файли до повідомлень.

Основною ідеєю при розробці інтерфейсу було зробити його максимально простим і зрозумілим, основи дизайну та стилізування його елементів були зумовлені стильовими преференціями та рекомендаціями сучасного стандарту дизайну та написання користувацьких інтерфейсів.

Вперше зайшовши на сайт, користувач потрапляє на титульну сторінку сайту. Звідти йому пропонується кілька варіантів розвитку подій: перейти на сторінку опису застосунку, зареєструватися, увійти в існуючий обліковий запис, перейти до замовлення. Слід зазначити, що при виборі переходу до замовлення користувача перенаправить на сторінку авторизації, адже замовлення доступне лише авторизованим користувачам(рис. 3.1).

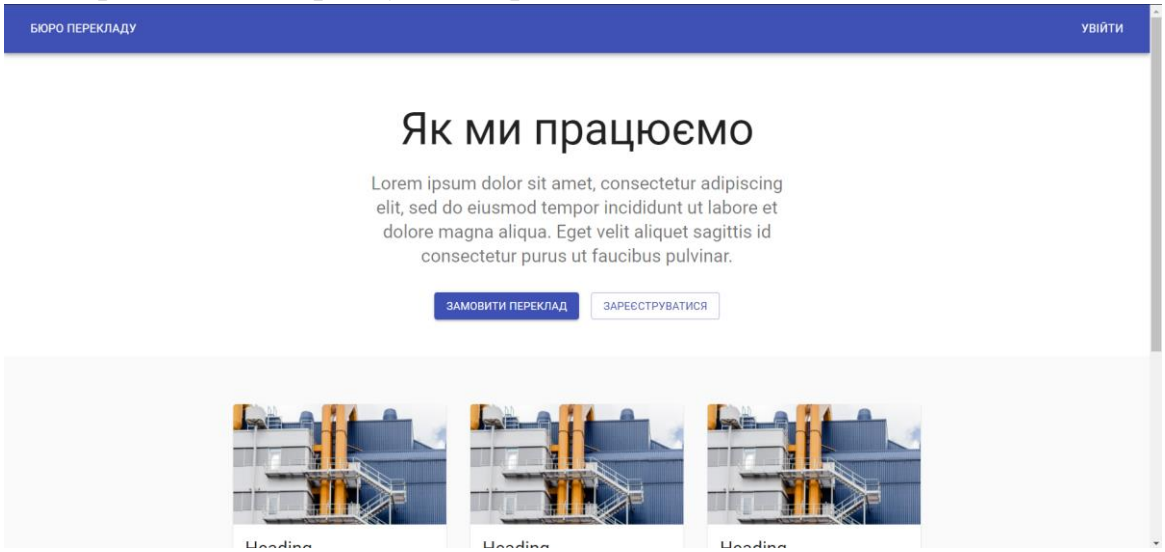

Рисунок 3.1. Головна сторінка застосунку

Обравши перейти на сторінку авторизації користувачу пропонується два поля для того, щоб авторизуватися в свій обліковий запис, або посилання на реєстрацію, якщо такого облікового запису не існує (рис 3.2).

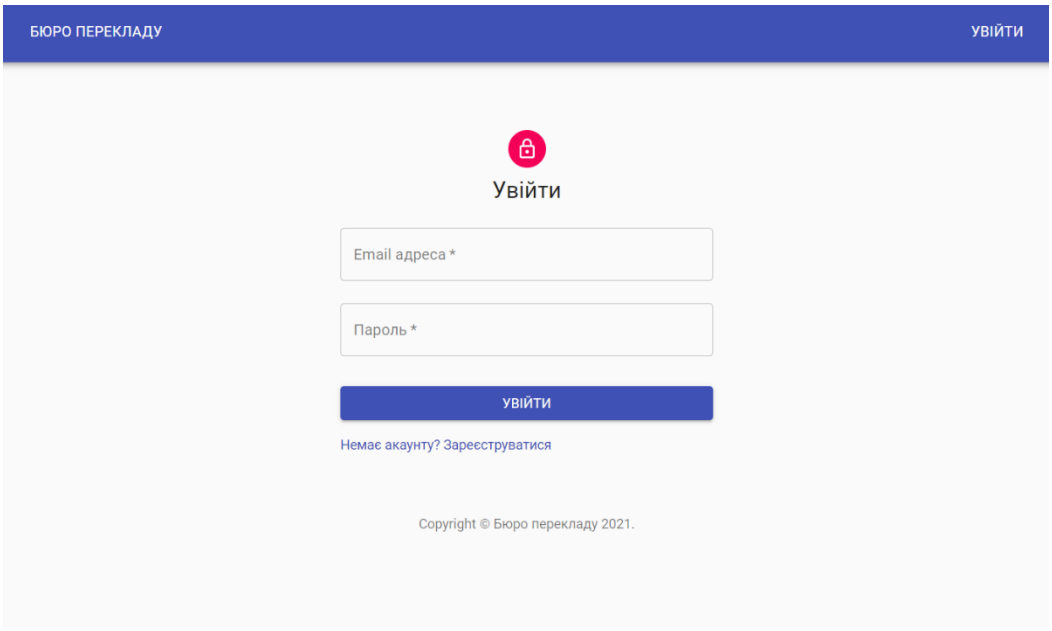

Рисунок 3.2. Сторінка для авторизації в обліковий запис користувача

Якщо ж такого облікового запису в системі не зареєстровано, необхідно зареєструвати нового користувача. Сторінка реєстрації має наступні поля: Ім’я, прізвище, поштова скринька та пароль. Також можна обрати, чи хочете ви отримувати розсилку про оновлення застосунку на свою поштову скриньку.

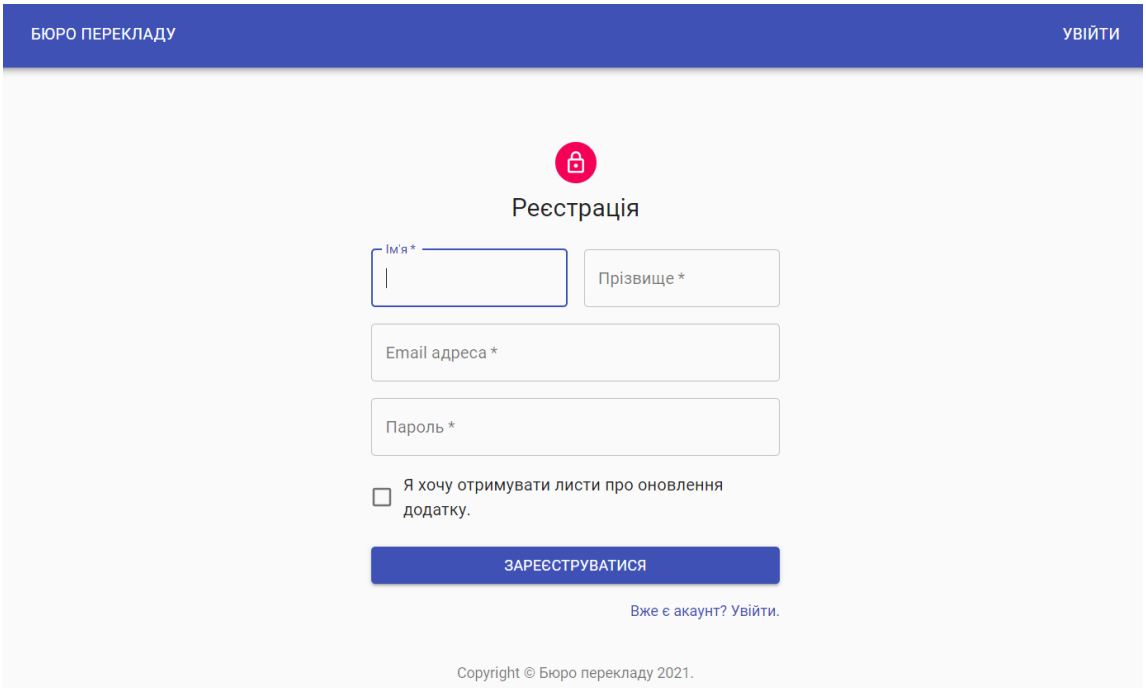

Рисунок 3.3. Форма для реєстрації нового користувача веб-застосунку

Одразу після реєстрації нового облікового запису, або після входу в вже існуючий, користувача перенаправляє знову на головну сторінку. Там в компоненті додатку «Header» з’являються нові поля для навігації по застосунку (рис 3.4).

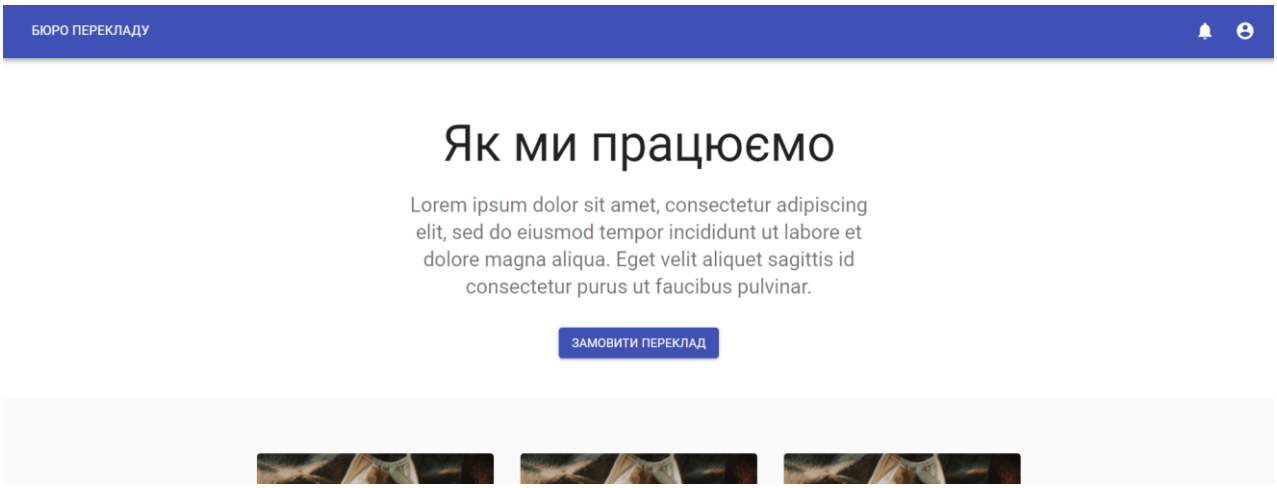

Рисунок 3.4. Головна сторінка застосунку для авторизованого користувача

Наступним кроком користувач може забажати створити нове замовлення. Для створення замовлення на головній сторінці застосунку необхідно натиснути кнопку «Замовити переклад». На сторінці замовлення перекладу користувача очікує форма з декількома полями: вибір мови оригіналу, вибір мови на яку потрібно перекласти, кнопка для закріплення документу, та опціональне поле для вибору бажаної дати завершення перекладу. (рис 3.5).

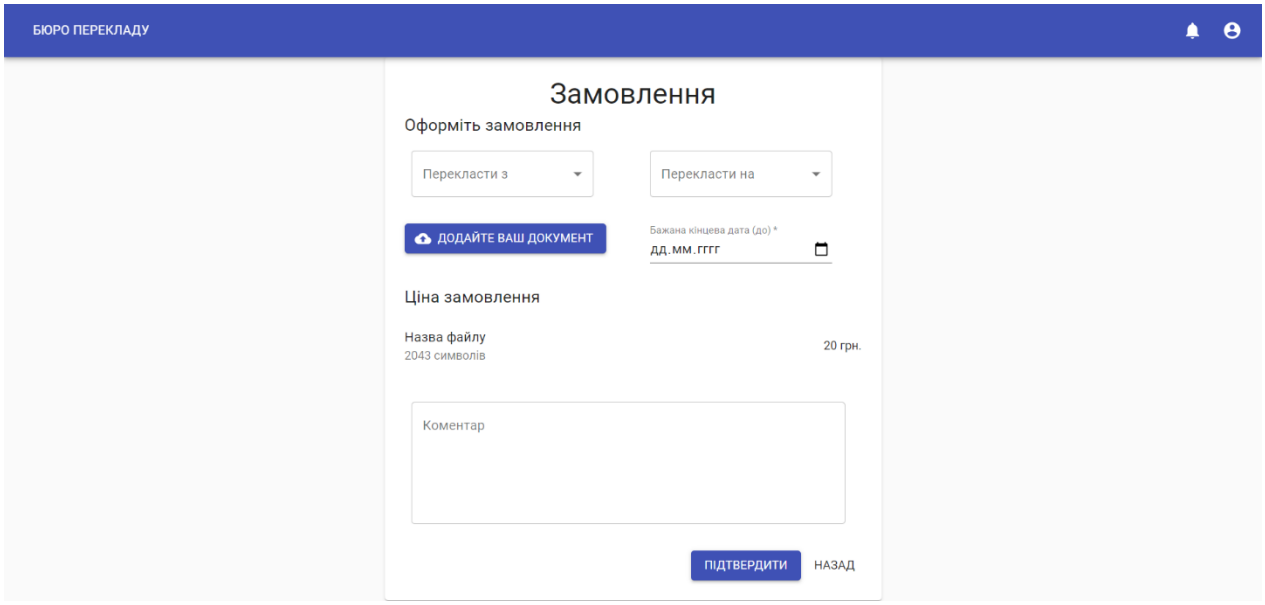

Рисунок 3.5. Форма створення нового замовлення для перекладу.

Після заповнення усіх даних щодо замовлення, та погодженням з ціною перекладу, користувач повинен натиснути «Підтвердити» для переходу до наступного этапу зіставлення замовлення на переклад документу. Наступним етапом є «Оплата». Для оплати буде запропоновано кілька варіантів сервісів онлайн розрахунків по типу LiqPay.

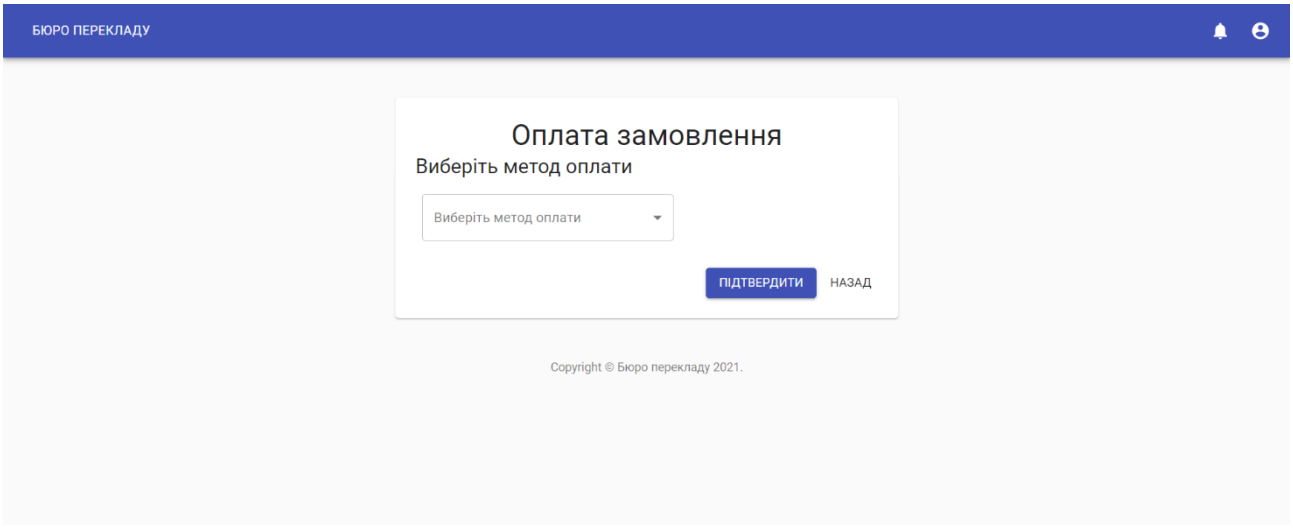

Рисунок 3.6. Форма вибору метода оплати замовлення

Після підтвердження оплати користувача переправляє на сторінку подяки. Такі сторінки використовуються в багатьох веб-застосунках, що надають які б то не було онлайн послуги для аналітики та збору даних клієнтів, що користуються застосунком. Прикладом підключення такої аналітики Google Analytics.

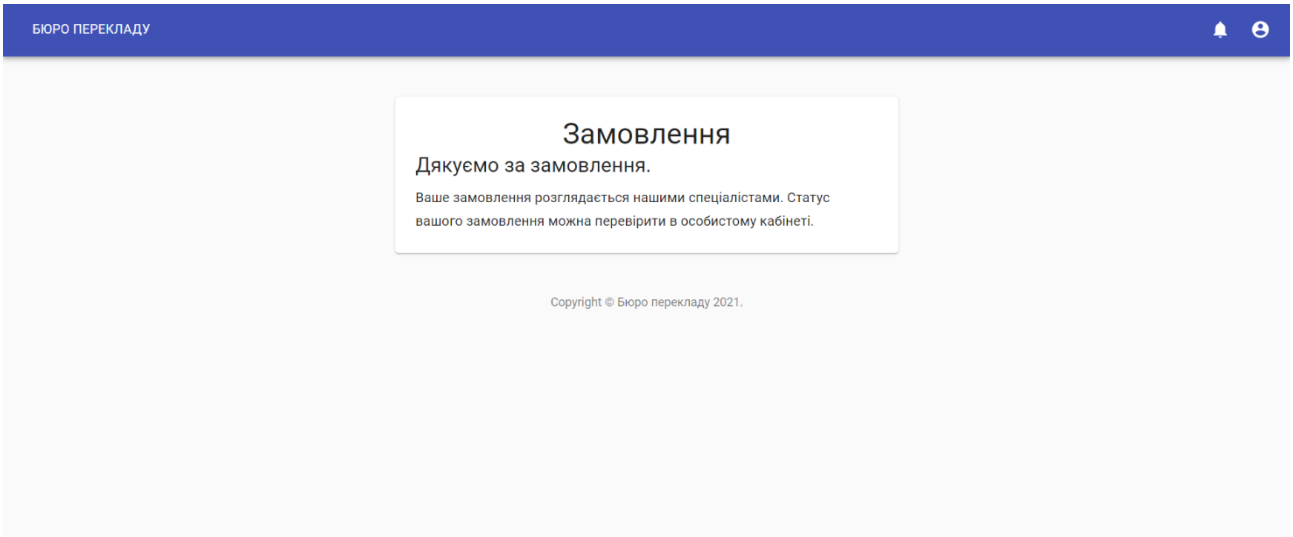

Рисунок 3.7. Сторінка подяки

За допомогою навігації через компонент «Header» зверху сторінки можна перейти до списку замовлень користувача. Для викладачів будуть відображатися

всі доступні замовлення, для замовників тільки їх власні. Зліва відображаються пронумеровані ідентифікатори замовлень, справа знаходиться хмарка для переходу до чату з замовником чи перекладачем, у випадку, якщо обліковий запис має роль замовника.

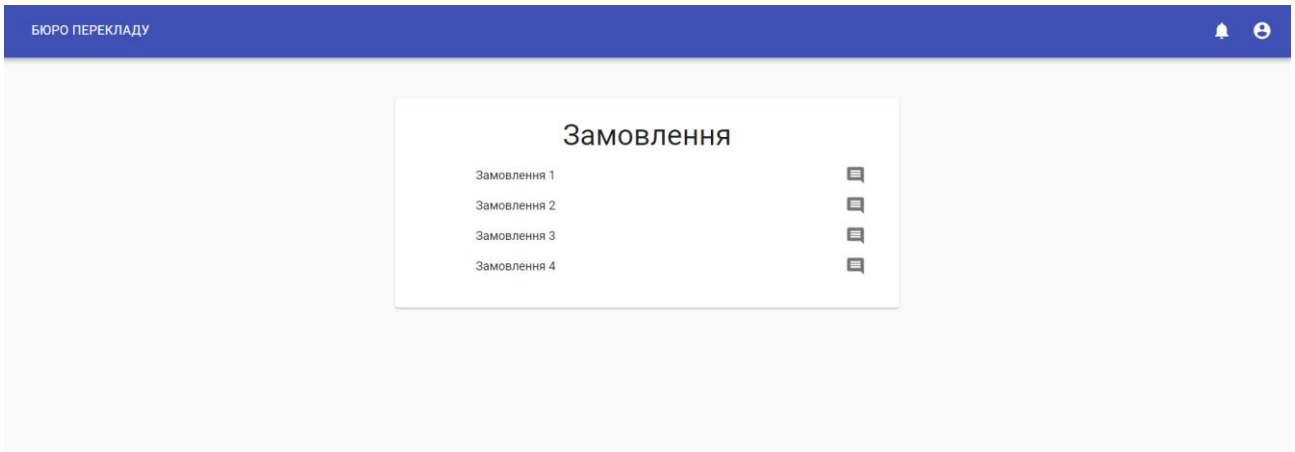

Рисунок 3.8. Сторінка зі списком замовлень (перекладач)

Користувач може обрати одне з замовлень та натиснути на кнопку в вигляді хмарки. Натиснувши на неї, користувач перейде до замовлення навпроти якого вона стояла. Всередині замовлення буде відкрито коментарі. Де перше повідомлення – коментар замовника, а також файл закріплений ним при створенні замовлення. Надалі сектор коментарів може вільно використовуватись як користувачем, так і перекладачем для встановлення взаємодії між обидвома. В той час як замовник має лише кнопку для надсилання коментаря, перекладач має додаткові кнопки для взаємодії з замовленням. В той час як у перекладача в наявності додаткові кнопки «file» – для закріплення файлу в повідомлення, «done» і «in progress», для зміни стану перекладу. Така архітектура інтерфейсу, де всі необхідні функції для взаємодії з клієнтом зручно розміщені в одному місці віконця для замовлення дозволяє максимізувати задоволеність користувацьким досвідом з використання веб-застосунку. Це є дуже важливим аспектом побудови UI – щоб все було в зрозумілих, помітних місцях, та не вимагало від користувачів додаткових натискань (рис 3.9).

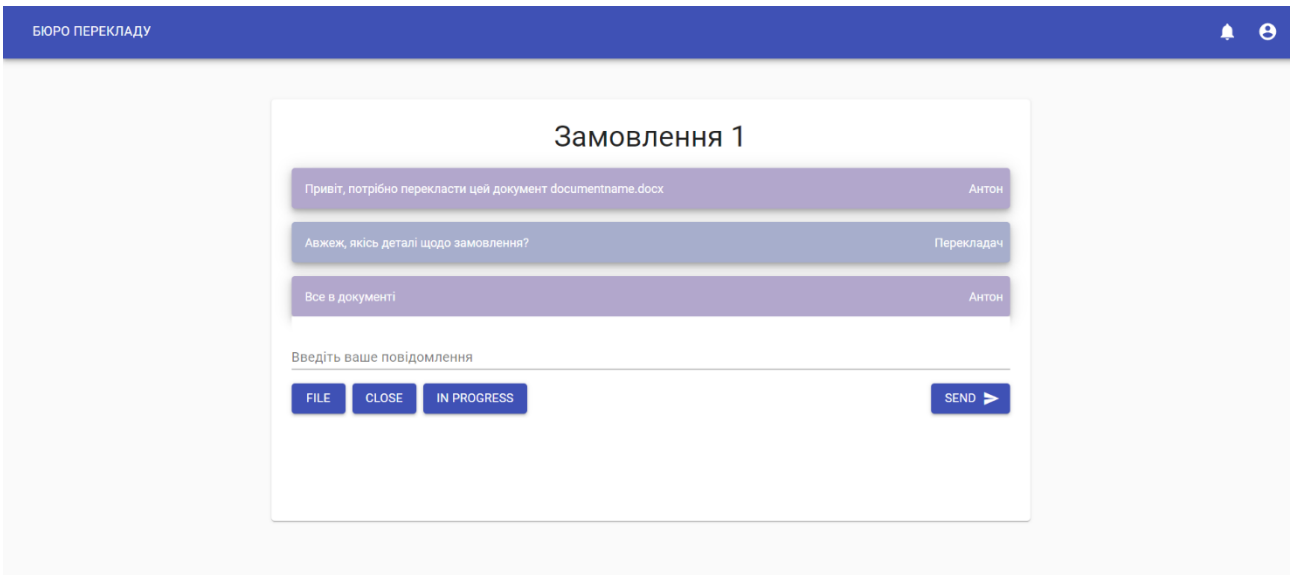

Рисунок 3.9. Сторінка коментування для взаємодії з замовником/перекладачем

На головній сторінці знизу також можна знайти картки з додатковою інформацією про застосунок, перейшовши по посиланню однієї з яких користувач зможе перейти на сторінку «Про застосунок», на якій буде описано основну інформацію про автора та ціль створення застосунку.

За допомогою навігації через компонент «Header» зверху сторінки можна перейти до списку замовлень користувача. Для викладачів будуть відображатися всі доступні замовлення, для замовників тільки їх власні.

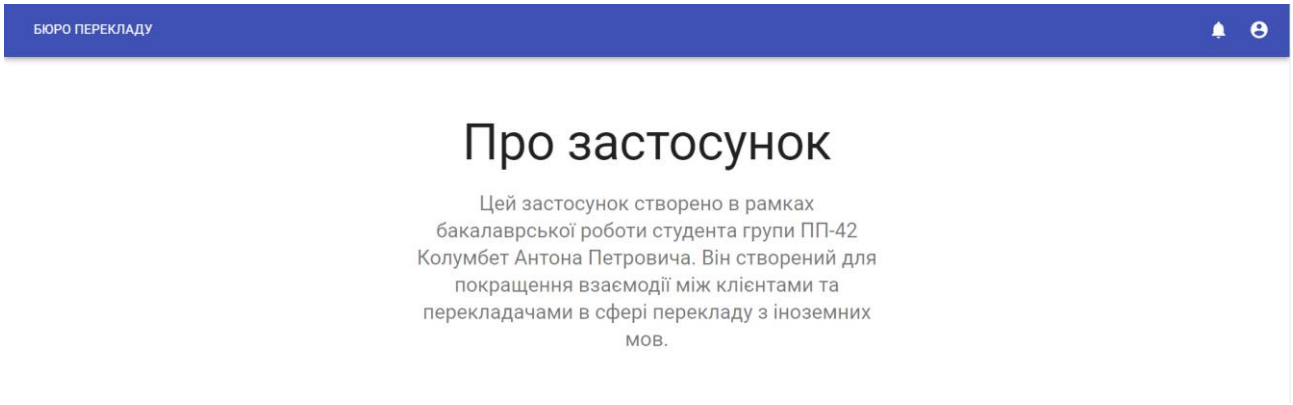

Рисунок 3.10. Сторінка опису деталей про цілі та творця додатку

Титульні веб-сторінки дозволяють ознайомити користувача з функціоналом, історією, описом додатку.

3.2 Опис мобільної версії веб-додатку

Веб застосунок є адаптованим для роботи на смартфонах, має зручний та зрозумілий інтерфейс. Карта сайту незалежно від версії на яку зайшов користувач, комп’ютерну чи мобільну не змінюється, загальне розміщення елементів залишається тим же, що і на комп’ютерній версії (див. рисунок 3.11).

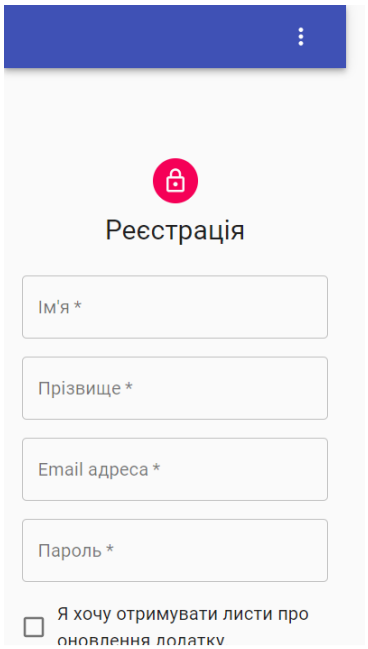

Рисунок 3.11. Вигляд віконця реєстрації в мобільній версії додатку

Для розробки інтерфейсу для смартфонів не використовувались додаткові бібліотеки. Інтерфейс є повністю адаптивним та змінним завдяки відлагодженної роботи CSS в фреймворку Material UI, що надає можливість розробляти адаптивний для смартфонів інтерфейс одразу без написання додаткового коду. Всі компоненти цього фреймворку одразу налаштовані для використання на засобах з різною розмірністю екрана. Зміна версії додатку, до версії для мобільних пристроїв відбувається при зменшенні ширини екрана до 600 пікселів.

Компонент Header в результаті згортання розміру екрана змінює свій вигляд. Кнопки, що раніше були доступні зразу при загрузці сторінки в верхньому меню перенеслися у «бургер» - кнопка при натисканні якої розгортається додаткове меню.

3.3 Тестування застосунку на швидкість та доступність

Не менш важливим фактором для успішності роботи веб-застосунку є доступність та швидкість для доступу для нього. Багато веб-ресурсів в мережі інтернет не задумаються над тим, наскільки ці веб-ресурси швидко працюють на смартфонах, адже при використанні домашнього інтернету, чи з власного комп’ютеру навіть сайти з доволі важким наповнення відображаються доволі швидко. В такому підході є велика проблема, адже більшість людей, які намагаються отримати доступ до ресурсу за допомогою свого смартфону скоріш за все використовують мобільний інтернет, що є не дуже швидким загалом. Ще більше ця ситуація ускладнюється, коли користувач виходить за межі якісного покриття інтернету його оператору. В таких випадках швидкість інтернету може впасти до 50-100к/б, через що загрузка навіть простого не зжатого зображення може зайняти біля 2 секунда, а це лише найменша частина всього сайту. Так загрузка всіх компонентів може доходити до кількох секунд, що є дуже шкідливим для досвіду користувача.

Для тестування веб-застосунку для підтримки взаємодії користувачів та клієнтів на швидкість, доступність та інших критеріїв було використано вбудовану в веб-браузер «Chrome» утиліту Lighthouse.

Lighthouse – це зручна утиліта, що використовується багатьма розробниками веб-застосунків для перевірки оптимізації їх проектів. Ця утиліта має відкритий вихідний код, та допомагає підвищувати продуктивність веб-сайтів. Завдяки зручному інструментарію Lighthouse дозволяє провести аналіз свого додатку всього за одне натискання кнопки у вбудованому інструментарії

для розробників браузеру Chrome. На рисунку 3.12 можна побачити результат тестування додатку.

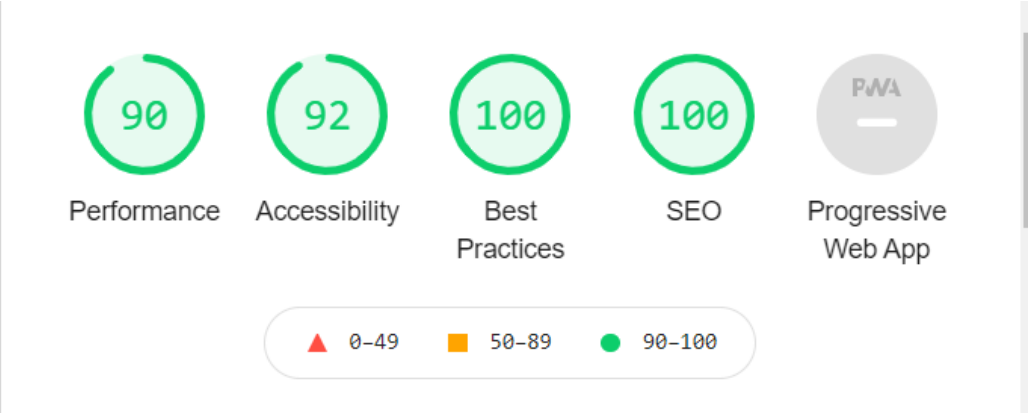

Рисунок 3.12. Результат тестування

Як можна бачити на рисунку 3.12 – веб-застосунок має достатній рівень доступності та швидкості.

ВИСНОВОК

В результаті виконання бакалаврської було розроблено веб-застосунок для підтримки взаємодії перекладачів та клієнтів. Цей додаток надає можливість створення замовлення, коментування цього замовлення замовником та перекладачем, відслідковування статусу замовлення. В ньому реалізована реєстрація та авторизація користувача.

За час виконання даної бакалаврської роботи були досліджені теоретичні та практичні основи та засоби для побудови веб-застосунку для підтримки взаємодії перекладачів та клієнтів. Завдяки різноманітним фреймворкам, бібліотекам та архітектурним рішенням застосованим та дослідженим в процесі виконання роботи було розроблено весь необхідний функціонал застосунку. Для дослідження теоретичних основ побудови веб-застосунку було взято існуючу документацію, книги та інформацію про роботу подібних застосунків на інших сервісах в мережі інтернет.

Інтерфейс веб-застосунку було розроблено в графічному онлайн редакторі Figma, після чого його було впроваджено в якості веб-застосунку за допомогою мов HTML5, CSS3, JavaScript, серверу node.js та фреймворку для нього Express, а також бібліотеки React та модулів до неї. Застосунок працює за допомогою серверу Node.js та має відкритий доступ для всіх локальних користувачів, або тунелюється в мережу інтернет, за допомогою програмної системи ngrok для тунелювання локальних серверів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Motaword – про застосунок. URL: https://www.motaword.com/ru/about
2. Ед Тіттел, Джеф Ноубл. HTML, XHTML і CSS для чайників, 7 видання/ HTML, XHTML & CSS For Dummies, 7th Edition, 2011. 17 с.
3. Введення в HTML5: що таке HTML. URL: https://metanit.com/web/html5/1.1.php.
4. Стилі HTML – CSS. URL: http://www.w3bai.com/ru/html/html_css.html.
5. Девід Фленаган. JavaScript – детальний посібник, 5 видання, 2008. С. 21-25.
6. React – документація. URL: https://ru.reactjs.org/docs/.
7. Material UI – документація. URL: https://material-ui.com/getting-started/learn/.
8. Введення в Node JS: Що таке Node JS. Початок роботи. URL: https://metanit.com/web/nodejs/1.1.php.
9. Введення в MongoDB. Що таке MongoDB? URL: https://metanit.com/nosql/mongodb/1.1.php.
10. Що таке MVC. URL: hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami.
11. Ерік Мейер. CSS – Каскадні таблиці стилів, 2008. С. 102-213.
12. Специфікація HTML та CSS. URL: https://www.w3schools.com.
13. Моррісон М. Вивчаємо JavaScript, 2008. *Реакції віртуального світу*. С. 45 – 65.
14. Нікольский О. Javascript на прикладах, 2016. С. 37-48.
15. Мальчук К. HTML та CSS. Самовчитель. С. 106-183.
16. Кайл Бенкер. MongoDB в дії, 2012. С. 38-233.

Додаток Е
Приклад опису користувацького інтерфейсу на React

```
import React from "react"
import { useHistory } from "react-router-dom"
import Button from "@material-ui/core/Button"
import Card from "@material-ui/core/Card"
import CardActions from "@material-ui/core/CardActions"
import CardContent from "@material-ui/core/CardContent"
import CardMedia from "@material-ui/core/CardMedia"
import CssBaseline from "@material-ui/core/CssBaseline"
import Grid from "@material-ui/core/Grid"
import Typography from "@material-ui/core/Typography"
import { makeStyles } from "@material-ui/core/styles"
import Container from "@material-ui/core/Container"
import Link from "@material-ui/core/Link"

function Copyright() {
  return (
    <Typography variant="body2" color="textSecondary" align="center">
      {"Copyright © "}
      <Link color="inherit" href="https://material-ui.com/">
        Your Website
      </Link>{" "}
      {new Date().getFullYear()}
      {"."}
    </Typography>
  )
}

const useStyles = makeStyles((theme) => ({
  icon: {
    marginRight: theme.spacing(2),
  },
  heroContent: {
    backgroundColor: theme.palette.background.paper,
    padding: theme.spacing(8, 0, 6),
  },
  heroButtons: {
    marginTop: theme.spacing(4),
  },
  cardGrid: {
    paddingTop: theme.spacing(8),
    paddingBottom: theme.spacing(8),
  },
  card: {
    height: "100%",
    display: "flex",
    flexDirection: "column",
  },
  cardMedia: {
```

```
    paddingTop: "56.25%", // 16:9
  },
  cardContent: {
    flexGrow: 1,
  },
  footer: {
    backgroundColor: theme.palette.background.paper,
    padding: theme.spacing(6),
  },
}))

const cards = [1, 2, 3]

export const Title = () => {
  let history = useHistory()

  const handleOrder = () => {
    history.push("/order")
  }

  const handleSignup = () => {
    history.push("/signup")
  }
  const classes = useStyles()
  return (
    <React.Fragment>
      <CssBaseline />
      <main>
        {/* Hero unit */}
        <div className={classes.heroContent}>
          <Container maxWidth="sm">
            <Typography
              component="h1"
              variant="h2"
              align="center"
              color="textPrimary"
              gutterBottom
            >
              Про застосунок
            </Typography>
            <Typography
              variant="h5"
              align="center"
              color="textSecondary"
              paragraph
            >
              Цей застосунок створено в рамках бакалаврської роботи студента
              групи ПП-42 Колумбет Антона Петровича. Він створений для
              покращення взаємодії між клієнтами та перекладачами в сфері
              перекладу з іноземних мов.
            </Typography>
```

```
          <div className={classes.heroButtons}>
            <Grid container spacing={2} justify="center">
            </Grid>
          </div>
        </Container>
      </div>
      <Container className={classes.cardGrid} maxWidth="md">
        {/* End hero unit */}
        <Grid container spacing={4}>
          {cards.map((card) => (
            <Grid item key={card} xs={12} sm={6} md={4}>
              <Card className={classes.card}>
                <CardMedia
                  className={classes.cardMedia}
                  image="https://source.unsplash.com/random"
                  title="Image title"
                />
                <CardContent className={classes.cardContent}>
                  <Typography gutterBottom variant="h5" component="h2">
                    Heading
                  </Typography>
                  <Typography>
                    This is a media card. You can use this section to describe
                    the content.
                  </Typography>
                </CardContent>
                <CardActions>
                  <Button size="small" color="primary">
                    View
                  </Button>
                  <Button size="small" color="primary">
                    Edit
                  </Button>
                </CardActions>
              </Card>
            </Grid>
          ))}
        </Grid>
      </Container>
    </main>
    {/* Footer */}
    <footer className={classes.footer}>
      <Typography variant="h6" align="center" gutterBottom>
        Footer
      </Typography>
      <Typography
        variant="subtitle1"
        align="center"
        color="textSecondary"
        component="p"
      >
```

```
          Something here to give the footer a purpose!
        </Typography>
        <Copyright />
      </footer>
      {/* End footer */}
    </React.Fragment>
  )
}
```