МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розробка спеціалізованої системи для скорочення та реферування текстів

Виконав студент групи К-20і
спеціальності 122 «Комп'ютерні науки»
Потапенко Денис Валерійович

Керівник к.т.н. доцент
Терещенко Тетяна Михайлівна

Консультант

Рецензент

Одеса 2022

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ПЗ – програмне забезпечення

IDE – Integrated Development Environment – інтегроване середовище розробки

SPA – Single Page Application – односторінковий додаток

JS – Javascript

UML – Unified Modeling Language – уніфікована мова моделювання

IT – Information technologies – інформаційні технології

ВСТУП

Завдяки глобальній мережі Інтернет людство отримало змогу передавати інформацію з різних кінців планети за лічені секунди. Разом зі своєю появою у 1969 році Інтернет безумовно став найвагомішим досягненням.

Веб-технологія є зручною завдяки спеціальним програмам – браузерам. Наразі майже всі користувачі Інтернету мають браузери на своїх пристроях. Це є свідчення доступності веб-технологій. Організації та компанії ставлять за мету якнайшвидше впровадження та супровід свого сайту. Користувачам для доступу залишається тільки ввести у пошуковий рядок короткий адрес.

У веб-технології застосовується велика кількість бібліотек, які значно спрощують розробку.

Актуальність роботи полягає в тому, що користувачам часто потрібно заощаджувати обсяг інформації. Користувачі бажають заощадити місце на накопичувачах, на віртуальних сторінках та на реальних сторінках паперу після роздрукування текстів. У клієнтів є потреба отримати скорочений текст швидко, не завантажуючи програми-архіватори та не створюючи при цьому архівних файлів із складними алгоритмами шифрування.

Мета роботи – створити веб-додаток для скорочення текстів, який використовує оптимізовані алгоритми та є зручним для користувача. Додаток повинен використовувати кілька методів інтеграції: як веб-інтерфейс так і POST запити для прийому вхідного тексту. Додаток має використовувати єдину сучасну мову програмування JavaScript для написання клієнтської та серверної частин логіки для подальшого легкого розширення та можливої передачі проекту іншій команді розробників. Для досягнення поставленої мети були сформульовані наступні завдання:

- провести аналіз предметної області;
- провести порівняльний аналіз існуючих програм-аналогів;
- обґрунтувати вибір програмних засобів розробки та технологій;

- провести проектування системи з використанням мови UML;
- виконати реалізацію застосунку;
- підготувати інструкцію користувача;
- виконати тестування застосунку.

Структура кваліфікаційної роботи бакалавра складається з вступу, 5 розділів, висновків, переліку посилань на 12 найменувань, 1 додатку. Повний обсяг проекту становить 59 сторінок, містить 15 рисунків і 2 таблиці.

Результати роботи доповідалися на Студентській науковій конференції ОДЕКУ 17 травня 2022 року.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ СИСТЕМ

1.1 Аналіз предметної області

Еволюціонування у сфері веб-розробок відбувається дуже швидко у порівнянні з іншими науково-технічними галузями. За кілька років примітивні статичні веб-сторінки, написані тільки на html, перетворилися на найскладніші, багатофункціональні, інтегровані з іншими програмами веб-системи. У компаній з'явилася можливість "перемістити" до інтернет-середовища безліч бізнес-процесів. Сучасні компанії створюють не один, а безліч сайтів, вкладених у вирішення різних завдань.

Останнім часом все більше компаній стали усвідомлювати, що сайт – це не просто електронна візитка чи онлайн-каталог, а зручний та дуже ефективний бізнес-інструмент. Але, ефективним, сайт стає лише у разі якісної розробки та грамотного просування [1].

Оцінка якості сайту не така проста задача, як може здатися на перший погляд. Хтось, звичайно, згадує про зручну структуру сайту, але як визначити, чи для всіх вона зручна, вже не знаю. Про інші важливі моменти згадують дуже рідко.

А тим часом, незважаючи на те, що дизайн та структура сайту дуже важливі, існують інші важливі критерії оцінки сайту.

Якісний сайт повинен не тільки відповідати всім сучасним технічним вимогам та побажанням цільової аудиторії, але й враховувати внутрішні та зовнішні фактори ранжування (критерії якості сайту з погляду пошукових систем).

Веб-додаток – додаток, в якому клієнтом є оглядач Інтернету, а сервером – веб-сервер. Оглядач Інтернету може бути реалізацією так званих тонких клієнтів. Він відображає веб-сторінки і, як правило, входить до складу операційної системи, а його оновлення та супровід виконує постачальник операційної системи. Логіка додатка зосереджена на сервері, а оглядача Інтернету найчастіше відповідає лише за відображення інформації, завантаженої з сервера,

і за передачу на сервер даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача [2].

Веб-додаток отримує запит від клієнта і виконує обчислення, після цього формує веб-сторінку і відправляє її клієнту через мережу з використанням протоколу HTTP.

Веб-додаток може бути клієнтом інших служб, наприклад, бази даних або іншого веб-додатку, розташованого на іншому сервері.

Останнім часом набуває популярності новий підхід до розробки веб-додатків, який називається Ajax. Сторінки веб-додатка не перезавантажуються повністю, а лише завантажують з сервера зміни, що робить їх більш інтерактивними і продуктивними [11].

Веб-сайт, або сайт – сукупність веб-сторінок, доступних у мережі Інтернет, які об'єднані як за змістом, так і за навігацією під єдиним доменним ім'ям. Фізично сайт може розміщуватися як на одному, так і на кількох серверах.

Сайтом також називають вузол мережі Інтернет, комп'ютер, за яким закріплена унікальна IP-адреса, і взагалі будь-який об'єкт в Інтернеті, за яким закріплена адреса, що ідентифікує його в мережі (FTP-site, WWW-site тощо).

Набір зв'язаних між собою інформаційних онлайн ресурсів, призначених для перегляду через комп'ютерну мережу за допомогою браузерів. Веб-вузол може бути набором документів в електронному вигляді, онлайн службою.

Переваги веб-додатків:

- Немає потреби в установці. Веб-програми можна запустити, просто перейшовши за правильною URL-адресою. Завдяки цьому можна швидко й легко почати працювати з додатком, коли вам це потрібно. На вашому жорсткому диску немає великих файлів, які займають місце в пам’яті, і можна отримати до них доступ з будь-якого пристрою. З веб-програмами вам ніколи не потрібно чекати завершення завантаження та встановлення, щоб використовувати їх, і це просто полегшує ваше життя.
- Автоматичні оновлення. Регулярне завантаження та встановлення

оновлень вручну викликає клопоту. Навіть коли програма автоматично завантажить їх для вас, вам все одно потрібно це схвалити, трохи почекати та перезапустити програму. Це звучить як дрібниця, але заощаджується багато часу, коли оновлення відбуваються автоматично, і щоразу, коли відкривається програма, це завжди остання стабільна версія.

- Кросплатформеність. Для переважної більшості веб-додатків єдиною необхідною умовою є доступ до Інтернету. Вони не залежать від специфікацій обладнання та системи. В результаті можна запускати їх з будь-якого пристрою або платформи, на якій є веб-браузер. Оскільки компоненти, які відповідають за функціональність програми, знаходяться на сервері, не має значення, чи вона запускається з Windows, Mac чи Linux [3];

- Мобільний доступ. Пов’язано з попереднім пунктом, незалежність від платформи також означає мобільність. Це означає, що більшість веб-додатків також можуть працювати на мобільних пристроях. Переважна більшість корпоративних веб-рішень функціонують ідеально, незалежно від системи. Це дозволяє вам брати роботу з будь-яким місцем і продовжувати тримати руку на пульсі ваших бізнес-процесів.;

- Використання обчислювальних ресурсів. Веб-сервіси споживають значно менше процесорної потужності. Звичайно, ваш браузер все ще працює на вашому комп’ютері, і чим більше вкладок у вас відкрито, тим більше використовується пам’ять. Однак у цьому відношенні він майже не порівнюється з настільними додатками. Не у всіх є потужний ПК. Неможливість виконувати свою роботу через технічні обмеження вашої машини засмучує. Веб-додатки працюють майже однаково, незалежно від того, наскільки дорогим є ваш процесор.

Головною метою веб-додатку «Abbreviator» є зручне та швидке отримання скороченого тексту, зрозумілого для людини. Цей застосунок

допоможе користувачам опрацьовувати великі масиви інформації за короткий час, використовуючи налаштування додатку.

1.2 Аналіз існуючих програмних систем

Розроблене програмне забезпечення має такі аналоги. Для перевірки аналогів використовувався текст довжиною 1200 символів із твору відомого письменника.

1.1 Програмний аналог «ezyzip.com».

Даний веб-сайт надає послуги зі створення файлів формату .zip [4]. ezyZip — це безкоштовний онлайн-інструмент для стиснення файлів у форматі zip та розархівування, який дозволяє архівувати файли. Перевага додатку в тому, що він підтримує розархівування, що дозволяє розпаковувати заархівовані архіви zip, zipx, rar, tar, tar.gz, 7z та різні інші формати архівів.

На відміну від інших онлайн-утиліт zip та розпакування, ezyZip не вимагає від користувача завантажувати файли на сервер. Немає обмежень розміру файлу, що дозволяє створювати великі заархівовані файли. Він працює локально як додаток для браузера, що робить його набагато швидшим, ніж інші подібні онлайн-інструменти для архівування та розархівування. Це також гарантує, що ваша конфіденційність буде захищена, оскільки дані файлів не залишать ваш браузер. Недоліком є те, що стиснутий текст незрозумілий для користувача. Інтерфейс користувача сайту ezyzip.com. представлений на рисунку 1.1.

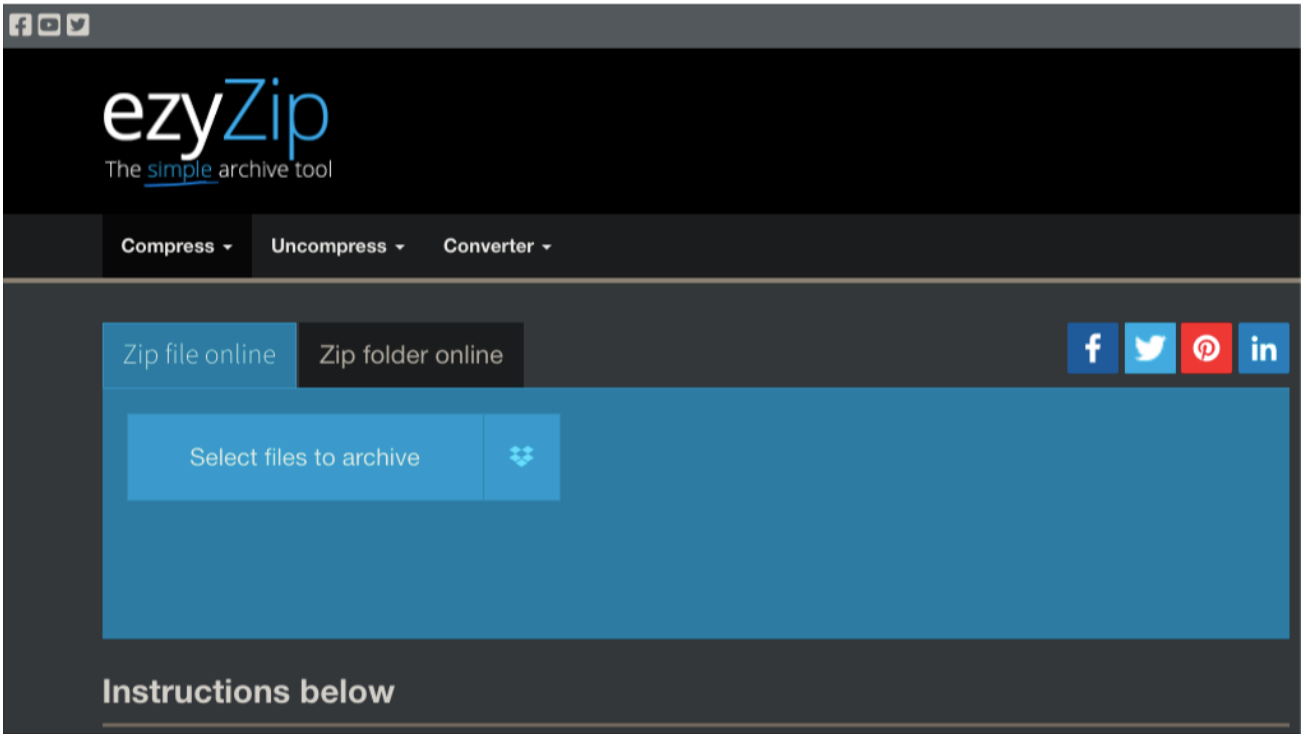

Рисунок 1.1 – Інтерфейс користувача сайту ezyzip.com

1.2 Програмний аналог «online-convert.com».

Даний веб-сайт надає послуги зі створення файлів формату .zip. Перевага додатку в тому, що користувач може завантажити свій файл або надати посилання на файл. Потім конвертер завантажить посилання та перетворить його за допомогою стиснення ZIP, щоб заощадити пропускну здатність під час завантаження.

У користувача також є можливість конвертувати архів в інший стиснений архів. Наприклад, можна конвертувати RAR в ZIP, TAR.GZ в ZIP, ISO в ZIP і багато іншого. Недоліком є те, що додаток не підтримує захищені паролем архіви та стиснутий текст незрозумілий для користувача.

Інтерфейс користувача сайту online-convert.com представлений на рисунку 1.2.

Рисунок 1.2 – Інтерфейс користувача сайту online-convert.com

1.3 Програмний аналог «files2zip.com».

Даний веб-сайт надає послуги зі створення файлів формату .zip. Цей додаток має переваги веб-сайтів ezyzip.com та online-convert.com. Ще однією перевагою files2zip.com є те, що цей додаток крім англійської підтримує ще німецьку, іспанську та голандську мови. Недоліком є те, що стиснутий текст незрозумілий для користувача. Інтерфейс користувача сайту files2zip.com представлений на рисунку 1.3.

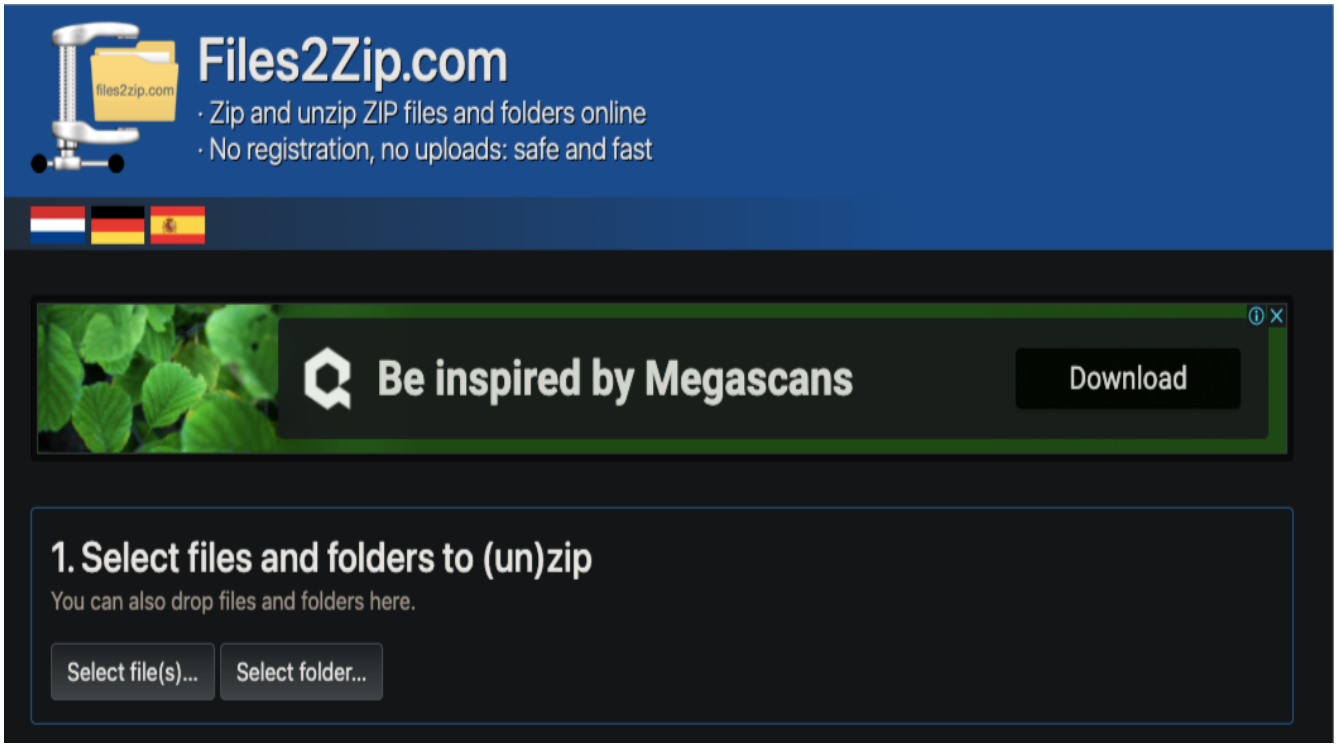

Рисунок 1.3 – Інтерфейс користувача сайту files2zip.com

Можна побачити що всі три аналоги використовують шифрування для скорочення. А одна з вимог до веб-додатку Abbreviator полягає в тому, щоб цей додаток зберігав текст зрозумілим для користувача.

2 ВИБІР ТА ОБҐРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ

2.1 Функціональні та нефункціональні вимоги

Аналіз вимог полягає в визначенні потреб та умов, які висуваються щодо нового, чи зміненого продукту, враховуючи можливо конфліктні вимоги різних замовників, таких як користувачі чи бенефіціари [5].

Аналіз вимог є критичним для успішної розробки проекту. Вимоги мають бути задокументованими, вимірними, тестовними, пов'язаними з бізнес-потребами, і описаними з рівнем деталізації достатнім для конструювання системи. Вимоги можуть бути архітектурними, структурними, поведінковими, функціональними, та не функціональними.

У мові моделювання SysML вимоги моделюють за допомогою діаграми вимог, а в UML для цього інколи пристосовують діаграму прецедентів.

Аналіз вимог включає три види діяльності:

- Виявлення вимог: задача комунікації з користувачами для визначення їх вимог. Також це називають збором вимог.
- Аналіз вимог: виявлення недоліків вимог (неточностей, неповноти, неоднозначностей чи суперечностей) і їх виправлення.
- Запис вимог: Вимоги можуть документуватись в різних формах, таких як опис звичайною мовою, прецедентами, користувацькими історіями, чи специфікаціями процесу.

Аналіз вимог може бути довгим та важким процесом що вимагає використання тонких психологічних навичок. Нові системи змінюють середовище і відношення між людьми, тому важливо розпізнати всі зацікавлені сторони, взяти до уваги всі їхні потреби, і переконатись що вони розуміють наслідки які приносить нова система. Аналітики можуть використати кілька методів щоб отримати від споживача вимоги. Історично це включає проведення інтерв'ю, чи фокус-груп (яку в цьому контексті частіше називають як майстерня вимог) і створення списків вимог. До сучасних підходів відносять прототипування, та прецеденти. За потреби аналітик використовує комбінацію цих методів щоб встановити точні вимоги зацікавлених сторін, так щоб система відповідала бізнес-потребам.

Систематичний аналіз вимог також відомий як інженерія вимог. Часом більш неформально її називають збором вимог, чи специфікацією вимог. Термін аналіз вимог також може застосовуватись до відповідного аналізу, в протилежність до, наприклад, збору чи документування вимог. Інженерія вимог може бути поділена на дискретні хронологічні кроки:

- збір (виявлення) вимог;
- аналіз вимог та їх узгодження;
- специфікація вимог;
- моделювання системи;
- перевірка (валідація) вимог;
- управління вимогами.

В деяких моделях життєвих циклів, процес аналізу вимог починається з діяльності по вивченню здійсненності, результатом якої є звіт про здійсненність. Якщо звіт припускає що продукт може бути створеним, то починається аналіз вимог. Якщо аналіз вимог передує дослідженням здійсненності, що може сприяти нестандартному мисленню, тоді здійсненність має визначитись перед завершенням аналізу вимог.

Перед початком розробки застосунку були визначені функціональні та нефункціональні вимоги до нього.

До функціональних вимог належать:

- вивід скороченого тексту у полі на веб-сторінці;
- отримання скороченого тексту після запиту на сервер.

До нефункціональних вимог належать:

- веб-браузер;
- підключення до мережі.

2.2 Обґрунтування вибору використаних технологій

Для розробки проекту була обрана архітектура клієнт-сервер, яка є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів [6].

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до

них.

Структурна схема представлена на рисунку 2.1:

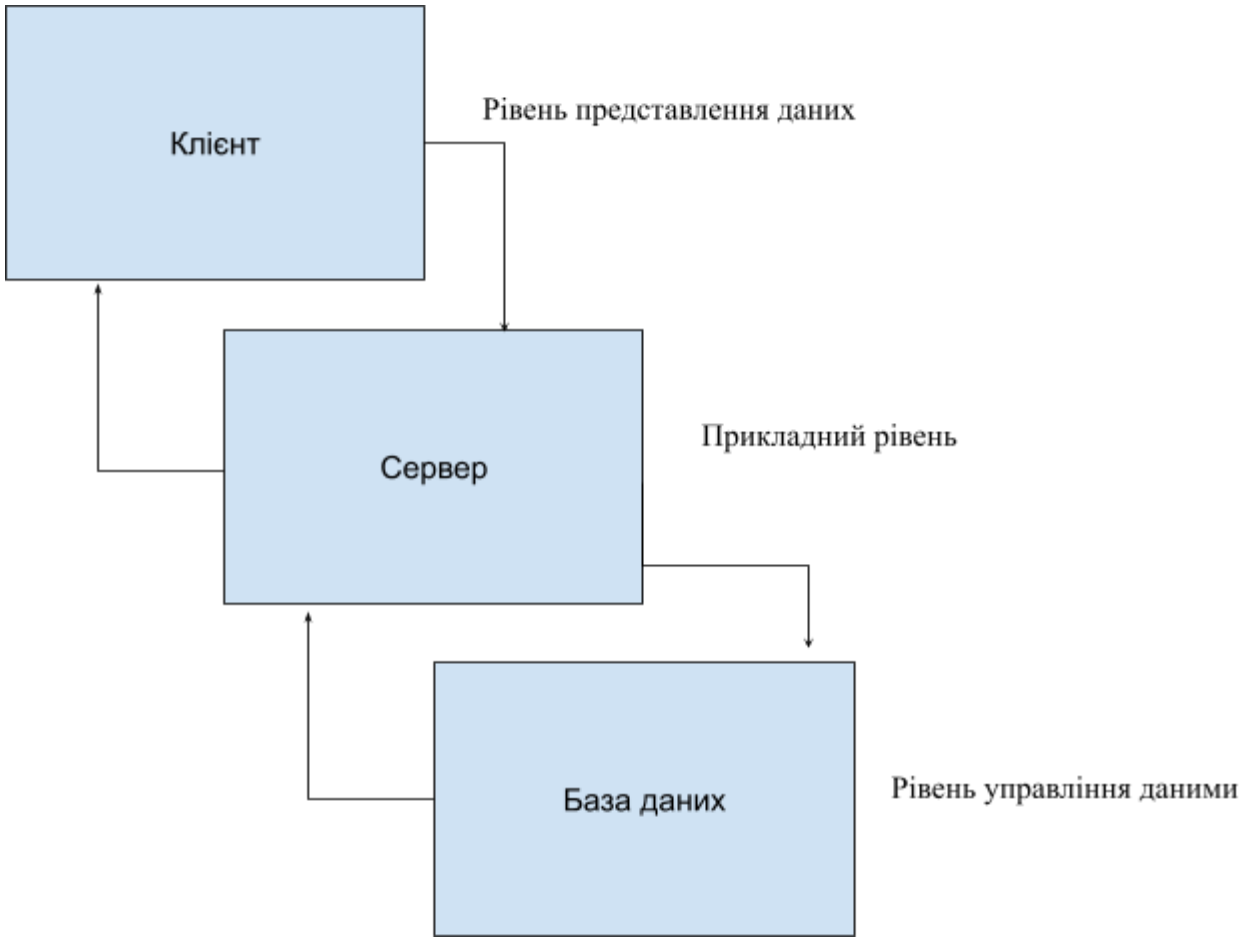

Рисунок 2.1 – Структурна схема

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Для написання веб-додатку була обрана мова JavaScript. Мова JavaScript використовується для:

– написання сценаріїв веб-сторінок для надання їм інтерактивності;
– створення односторінкових веб-додатків (React, AngularJS, Vue.js);
– програмування на стороні сервера (Node.js);
– стаціонарних додатків (Electron, NW.js);
– мобільних додатків(React Native, Cordova);
– сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
– всередині PDF-документів тощо.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme [7].

Доля JavaScript серед українського IT – 18,4 %, а серед світового – 17,8 (станом на 01.04.2022), що робить його лідером на обох ринках.

2.3 Вибір середовища розробки

Для розробки веб-додатку було обрано текстовий редактор Visual Studio Code. Visual Studio Code був обраний завдяки своїм особливостям – підсвічування синтаксису та підтримка великої кількості мов програмування (C, C++, Java, XML, HTML, PHP, JavaScript, ASCII, Visual Basic / VBScript, SQL, Ruby, CSS, Pascal, Perl і Python), багатомовна підтримка, робота з декількома документами.

Visual Studio Code — засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015. Це середовище розробки стало першим кросплатформним продуктом у лінійці Visual Studio.

За основу для Visual Studio Code використовуються напрацювання вільного проєкту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовує браузерний рушій Chromium і Node.js.

Примітно, що про використання напрацювань вільного проєкту Atom і на сайті Visual Studio Code, і в прес-релізі, і в офіційному блозі не згадується.

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціюється як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки.

Visual Studio Code – текстовий редактор, призначений для програмістів і тих, кого не влаштовує скромна функціональність програми «блокнот», що входить до складу Windows. Інтерфейс Visual Studio Code є багатомовним (українська мова присутня). Основні можливості Visual Studio Code:

– підсвітка синтаксису тексту залежно від мови програмування, режим якої може налаштувати користувач;
– можливість згортання блоків;
– WYSIWYG (друкуєш і отримуєш те, що бачиш на екрані);
– авто-завершення слова, що набирається;
– одночасна робота з безліччю документів;
– підтримка регулярних виразів для пошуку й заміни;
– повна підтримка перетягування фрагментів тексту;
– динамічна зміна вікон перегляду;
– автоматичне визначення стану файлу;
– підтримка великої кількості мов;
– нотатки і плагіни;
– запис макросу і його виконання.

При встановленні додаткових плаґінів:

– шаблони тексту (сніпети), що вводяться за допомогою скорочень (плаґін SnippetPlus);
– FTP-менеджер (плаґін NppFTP) і нех-редактор;
– автозбереження (при втраті фокусу через певний проміжок часу);
– перевірка орфографії (з використанням GNU Aspell);
– симетричне й асиметричне шифрування тексту;
– підтримка Zen Coding;

– підтримка автоматизації за допомогою скриптів мовами Python, JScript, Lua та іншими;

– підтримка збереження до OneDrive і Dropbox.

3 РОЗРОБКА ВЕБ-ДОДАТКІВ ЗА ДОПОМОГОЮ FULLSTACK JS BUNDLE

3.1 Поняття набору технологій FullStack JS Bundle

FullStack JS Bundle – це набір веб-технологій із використанням мови програмування JavaScript. До набору входять технології Node.js, Express.js, Vue.js. FullStack JS Bundle складається з технології розробки клієнтської логіки, яка поєднується з технологією розробки серверної логіки. Ця комбінація дозволяє отримати інтеграцію клієнтської програми та серверного середовища для керування налаштуваннями програми.

Щоб зрозуміти, що робить розробник FullStack, спочатку потрібно знати, що означає FullStack. По суті, технологічний стек — це комбінація інструментів і мов програмування для створення платформи, яка підтримує програми. Наприклад, стек для веб-додатка буде сформований операційною системою, веб-сервером, базою даних і принаймні однією мовою програмування.

Оскільки програмне забезпечення має клієнтську сторону (фронтенд) і серверну (серверну частину), інженери програмного забезпечення говорять про два окремих стеки, які охоплюють усі рівні, що складають весь технологічний стек цього конкретного програмного забезпечення. Таким чином, багато людей стверджують, що для успішного створення програмного забезпечення потрібно працювати з обома стеками окремо.

Однак інші люди вважають, що розробка передньої та задньої частини за допомогою одного стека не просто можлива – це найкраща альтернатива. Це розробники FullStack, інженери, які мають необхідні навички для створення або оптимізації повної інтеграції між зовнішніми та бекендовими системами за допомогою одного коду.

Коли розробники та програмісти використовують JavaScript для кодування частин свого додатка, це призводить до кращої ефективності та продуктивності команди. Це дозволяє розробникам краще зрозуміти вихідний код, який створюється в команді. Через це не існує такого поняття, як розрив між інженерами-розробниками клієнтської програми та інженерами-розробниками серверного середовища. Усі члени команди можуть функціонувати разом і

співпрацювати один з одним. Це зменшує розмір команди, оскільки не потрібно підтримувати та створювати дві окремі команди. Завдяки цьому витрати на найм та працевлаштування талантів значно зменшуються в організації чи компанії. Це стає великим надбанням, коли справа доходить до управління командою за допомогою гнучких методологій.

При використанні FullStack JS Bundle заохочується повторне використання коду, пропагуючи ефективні методи створення та використання коду. Технології розробки заохочують і сприяють цьому між розробниками програми. Це відбувається шляхом спільного використання бібліотек, шаблонів і моделей між технологіями, які використовуються для створення функціональних можливостей програми. Ви, як розробник, можете зменшити та повторно використати до 40% коду в програмі, тим самим зменшуючи загальну вартість розробки на 50%.

Використання FullStack JS Bundle сприяє високошвидкісній розробці та створенню високопродуктивних програм. Такі програми швидко масштабуються і відповідають вимогам користувачів і клієнтів, коли мова йде про продуктивність таких розроблених програм і веб-сайтів. Наприклад, Node.js просуває та забезпечує високу швидкість розробки та масштабовані додатки, які цілком здатні задовольнити запити користувачів і клієнтів.

Full-stack розробник це “майстер на всі руки” у світі веб-розробки. Йому під силу реалізувати як клієнтську, і серверну бік програми, якими, зазвичай, займаються FrontEnd і BackEnd розробники окремо друг від друга. Таким чином, Full-stack фахівець здатний самостійно вести проект від початку до кінця.

Ще в далеких нульових і раніше не існувало такого поділу обов'язків між розробниками. Відносна простота програмного забезпечення, як і технології того часу, дозволяли тримати процеси, які зараз виконують різні люди, в одних руках. Наприклад, у ті часи IT-фахівець, який називається веб-майстром, і зовнішній вигляд сайту створював, і серверну частину реалізовував, і розміщував сайт на хостингу. Тобто Full-stack розробники існували і раніше, просто ніхто їх так не називав.

Однак IT-сектор не стояв на місці. Вимоги до програмних продуктів зростали, з'являлися нові мови та технології, змінювалися підходи до розробки.

Дерево IT почало ставати все більш гіллястим, породжуючи нові спеціальності. Разом з цим, професія універсального бійця розбилася на два окремі напрямки, а потім знову відродилася з гордою назвою “Full-stack Developer”.

3.2 Сфери використання набору технологій FullStack JS Bundle

FullStack JS Bundle використали при побудові своїх додатків такі компанії як: Meta, Netflix, LinkedIn.

Meta продовжує залишатися одним із найважливіших корпоративних прихильників бібліотеки розробки JavaScript. Популярний фреймворк JavaScript React Native був розроблений і випущений Meta у 2015 році – і з тих пір він зазнав величезного зростання. React Native був створений для вирішення серйозної проблеми з мобільними додатками Meta. Кросплатформна розробка дозволяє компаніям розробляти мобільні додатки, які функціонують на кількох операційних системах. Команда розробників компанії використала React Native для створення поточних мобільних додатків як для iOS, так і для Android. Обидва мобільні програми були створені однією командою розробників з використанням єдиної мови JavaScript, тому вони мають спільні функції та працюють однаково інтуїтивно.

Коли в 2014 році для Netflix настав час переробити свою платформу онлайн-потоку, команда розробників звернулася до Node.js. До цього часу бібліотека JavaScript вже заслужила репутацію простого фреймворка з широким застосуванням. Раніше компанія використовувала Java для серверної частини та JavaScript для інтерфейсу. Бек-енд оброблявся однією великою програмою, що запускала сценарії Groovy. Це змусило розробників вільно володіти двома різними мовами програмування. Node.js усунув цю проблему, тому провідні компанії-розробники JavaScript у блозі Theymakedesign знають і справляються з цим дуже добре. Компанія також побачила покращений досвід користувача в результаті прискореного завантаження та інтуїтивно зрозумілого попереднього перегляду фільмів. Це тому, що Node.js використовує цикл подій та асинхронний і неблокуючий ввід-вивід для легкої обробки десятків тисяч запитів на секунду. Це суттєво зменшило затримку, яка гальмувала послуги Netflix.

Коли компанія LinkedIn намагалася випустити вдосконалену мобільну програму в 2012 році, вона звернулася до FullStack JS Bundle. Використовуючи комбінацію HTML та рідного коду JavaScript, команда розробників LinkedIn змогла створити добре розроблену програму з приголомшливим інтерфейсом користувача. JavaScript також суттєво вплинув на продуктивність мобільного додатка LinkedIn. У 2012 році Ars Technica провела поглиблений аналіз розвитку мобільних додатків LinkedIn. На веб-сайті зазначається, що Node.js значно покращив функціональність програми. Node.js збільшив продуктивність і використання пам'яті в порівнянні з Ruby on the Rails. Насправді в деяких випадках програма працювала в 20 разів швидше.

Компанія Microsoft тісно співпрацювала з JavaScript, щоб створити свій веб-браузер Edge. Усі браузери повинні ефективно обробляти та виконувати JavaScript, тому Microsoft розробила та підтримує власний механізм JavaScript для Edge. Насправді, були розмови про створення альтернативної версії NodeJS з движком Edge. Нещодавно Microsoft дійсно прийняла NodeJS. Вони повністю підтримують Node на хмарній платформі Azure. Це одна з основних функцій Azure, і вони інтегрували підтримку Visual Studio для Node. Microsoft також розробила версію Node для додатків Інтернету речей (IoT). NodeJS чудово підходить для IoT, оскільки він легкий та ефективний.

PayPal використовує JavaScript на клієнтській частині свого веб-сайту протягом тривалого часу, і це лише початок. Гігант онлайн-платежів був одним із перших користувачів NodeJS. Під час капітального перегляду сторінки огляду облікового запису вони вирішили спробувати створити сторінку в Node одночасно зі своєю звичайною розробкою на Java. Версія NodeJS працювала настільки добре, що вони вирішили використовувати її у виробництві та створювати всі клієнтські додатки в Node. Це означає, що більшість того, що користувач бачить у своєму обліковому записі, працює на Node. Розробка PayPal JavaScript навіть дійшла до створення та підтримки власної версії Express, яка називається KrakenJS.

3.3 Основні принципи роботи набору технологій FullStack JS Bundle

Vue – це фреймворк JavaScript для створення інтерфейсів користувача. Він побудований на основі стандартних HTML, CSS і JavaScript і надає декларативну і компонентну модель програмування, яка допомагає вам ефективно розробляти інтерфейси користувача, будь то прості чи складні. Vue розширює стандартний HTML за допомогою синтаксису шаблону, який дозволяє нам декларативно описувати вихідні дані HTML на основі стану JavaScript. Vue автоматично відстежує зміни стану JavaScript і ефективно оновлює DOM, коли відбуваються зміни.

Vue — це фреймворк та екосистема, яка охоплює більшість загальних функцій, необхідних для розробки інтерфейсу. Але Інтернет надзвичайно різноманітний – речі, які ми створюємо в Інтернеті, можуть сильно відрізнятися за формою та масштабом. З огляду на це, Vue розроблено, щоб бути гнучким і поступово адаптованим. Залежно від вашого випадку використання Vue можна використовувати різними способами:

- покращення статичного HTML без етапу збірки;
- вбудовування як веб-компоненти на будь-яку сторінку;
- односторінковий додаток;
- відтворення на стороні сервера;
- генерація статичного сайту;
- націлювання на комп’ютер, мобільний телефон, WebGL або навіть термінал.

Традиційним недоліком односторінкових додатків (SPA) є неможливість ділитися посиланнями на точну «підкладну» сторінку в межах певної веб-сторінки. Оскільки SPA обслуговує своїх користувачів лише одну відповідь сервера на основі URL-адреси (зазвичай він обслуговує index.html або index.vue), створення закладок для певних екранів або обмін посиланнями на певні розділи зазвичай складні, якщо взагалі неможливі. Щоб вирішити цю проблему, багато маршрутизаторів на стороні клієнта розмежовують свої динамічні URL-адреси символом «hashbang» (#!), напр. page.com/#!/. Однак із HTML5 більшість сучасних браузерів підтримують маршрутизацію без

хешбінгів.

Vue надає інтерфейс для зміни того, що відображається на сторінці на основі поточного шляху URL-адреси – незалежно від того, як він був змінений (через посилання електронною поштою, оновлення чи посилання на сторінці). Крім того, використання інтерфейсного маршрутизатора дозволяє навмисно переносити шлях браузера, коли певні події браузера (тобто кліки) відбуваються на кнопках або посиланнях. Сам Vue не має фронтальної хешованої маршрутизації. Але пакет «vue-router» з відкритим кодом надає API для оновлення URL-адреси програми, підтримує кнопку «Назад» (історія навігації), а також скидання пароля електронної пошти або посилання для перевірки електронної пошти з параметрами URL-адреси автентифікації. Він підтримує відображення вкладених маршрутів до вкладених компонентів і пропонує дрібнозернистий контроль переходів. За допомогою Vue розробники вже створюють програми з невеликих будівельних блоків, створюючи більші компоненти. Коли до суміші додається vue-router, компоненти повинні бути просто зіставлені з маршрутами, яким вони належать, а батьківські/кореневі маршрути мають вказувати, де мають відображатися дочірні елементи.

Node.js розроблено як асинхронне кероване подіями середовище виконання JavaScript, для створення масштабованих мережевих додатків. У цьому середовищі багато з'єднань можна обробляти одночасно. Після кожного підключення запускається зворотний виклик, але якщо немає ніякої роботи, Node.js перейде в режим сну. Це на відміну від сьогоднішньої більш поширеної моделі паралельності, в якій використовуються потоки ОС. Мережа на основі потоків є відносно неефективною та дуже важкою у використанні. Крім того, користувачі Node.js не турбуються про блокування процесу, оскільки блокування немає. Майже жодна функція в Node.js безпосередньо не виконує введення-виведення, тому процес ніколи не блокується, за винятком випадків, коли ввід-вивід виконується за допомогою синхронних методів стандартної бібліотеки Node.js. Оскільки ніщо не блокує, масштабовані системи дуже розумно розробляти в Node.js.

Node.js схожий за дизайном на такі системи, як Ruby's Event Machine і

Python's Twisted, і знаходиться під впливом таких систем. Node.js розширює модель подій. Він представляє цикл подій як конструкцію під час виконання, а не як бібліотеку. В інших системах завжди існує блокуючий виклик для запуску циклу подій. Зазвичай поведінка визначається за допомогою зворотних викликів на початку сценарію, а в кінці сервер запускається через блокуючий виклик. У Node.js такого виклику циклу немає. Node.js просто входить у цикл подій після виконання сценарію введення. Node.js виходить з циклу подій, коли більше немає зворотних викликів для виконання. Ця поведінка схожа на JavaScript у браузері — цикл подій прихований від користувача.

HTTP — це перший класс Node.js, розроблений з урахуванням потокової передачі та низької затримки. Завдяки цьому Node.js добре підходить для створення веб-бібліотеки або фреймворку. Розробка Node.js без потоків не означає, що не можна скористатися перевагами кількох ядер у своєму середовищі. Дочірні процеси можуть бути створені за допомогою API і розроблені так, щоб з ними було легко спілкуватися. На цьому ж інтерфейсі побудований модуль кластера, який дозволяє вам спільно використовувати сокети між процесами, щоб забезпечити балансування навантаження на ваші ядра.

3.4 Інструменти для розробки програмного забезпечення з використанням набору технологій FullStack JS Bundle

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих додатків, написаних мовою JavaScript. Якщо раніше Javascript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників [8].

Node.js має наступні властивості:

- асинхронна одно-нитева модель виконання запитів;
- неблокуючий ввід/вивід;

– система модулів CommonJS;
– рушій JavaScript Google V8.

Для керування модулями використовується npm (node package manager) – це менеджер пакетів для мови програмування JavaScript. Для середовища виконання Node.js це менеджер пакетів за замовчуванням. Включає в себе клієнт командного рядка, який також називається npm, а також онлайн-базу даних публічних та приватних пакетів, яка називається реєстром npm. Реєстр доступний через клієнт, а доступні пакети можна переглядати та шукати через веб-сайт npm. Менеджер пакетів та реєстр керуються npm, Inc.

Express – програмний каркас розробки серверної частини веб-додатків для Node.js, реалізований як вільне і відкрите програмне забезпечення під ліцензією MIT. Він спроектований для створення веб-додатків і API. Де-факто є стандартним каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описує його як створений на основі написаного на мові Ruby каркаса Sinatra, маючи на увазі, що він мінімалістичний, але має велику кількість плагінів, шо підключаються [9].

Express є бекендом для програмного стека MEAN, разом з базою даних MongoDB і каркасом AngularJS для фронтенду.

Vue.js – це фреймворк JavaScript із відкритим вихідним кодом model–view–viewmodel для створення інтерфейсів користувача та односторінкових програм [10].

Vue.js має поступово адаптивну архітектуру, яка зосереджена на декларативній візуалізації та композиції компонентів. Основна бібліотека зосереджена лише на шарі перегляду. Розширені функції, необхідні для складних додатків, таких як маршрутизація, керування станом і інструменти для збирання, пропонуються через офіційно підтримувані допоміжні бібліотеки та пакети. Vue.js дозволяє розширювати HTML атрибутами HTML, які називаються директивами. Директиви пропонують функціональні можливості для HTML-додатків і є вбудованими або визначеними користувачами директивами.

Vue використовує синтаксис шаблону на основі HTML, який дозволяє прив’язувати відтворений DOM до даних базового екземпляра Vue. Усі шаблони Vue є дійсним HTML, який можна проаналізувати браузерами, що відповідають

специфікації, і синтаксичними аналізаторами HTML. Vue компілює шаблони у віртуальні функції візуалізації DOM. Віртуальна модель об’єкта документа (або «DOM») дозволяє Vue відображати компоненти у своїй пам’яті перед оновленням браузера. У поєднанні з системою реактивності Vue може обчислити мінімальну кількість компонентів для повторного візуалізації та застосувати мінімальну кількість маніпуляцій DOM, коли стан програми змінюється. Користувачі Vue можуть використовувати синтаксис шаблону або безпосередньо писати функції візуалізації за допомогою гіперскрипту або за допомогою викликів функцій або JSX. Функції візуалізації дозволяють створювати програми з програмних компонентів.

4 ПРОЕКТУВАННЯ ВЕБ- ДОДАТКУ

Після проведення аналізу предметної області, програмних продуктів-аналогів та вибору засобів розробки, наступним етапом є проектування веб-додатку. На цьому етапі необхідно створити UML-діаграми варіантів використання (прецедентів), активності і послідовності та розробити макети інтерфейсу застосунку.

4.1 Створення UML-діаграми варіантів використання системи

Діаграма варіантів використання (прецедентів) відображає функціональне призначення проектованої програмної системи. Суть діаграми прецедентів полягає в тому, що систему представляють як групу акторів, які за допомогою варіантів використання взаємодіють із нею.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених межею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть діаграми прецедентів полягає в тому, що проєктована система подається у вигляді множини сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система під час діалогу з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

- асоціації;
- включення;
- розширення;

– узагальнення.

При цьому загальні властивості варіантів використання можна подати трьома різними способами, а саме — за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації — одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується під час побудови всіх графічних моделей систем у формі канонічних діаграм.

Включення у мові UML — це різновид відношення залежності між базовим варіантом використання і його окремим випадком. При цьому відношенням залежності є таке відношення між двома елементами моделі, за якого зміна одного елемента (незалежного) спричиняє зміну іншого елемента (залежного).

Відношення розширення визначає взаємозв'язок базового варіанту використання з іншим варіантом використання, функціональна поведінка якого залучається базовим не завжди, а тільки за виконання додаткових умов.

Актор – це сутність, що взаємодіє з системою для вирішення деяких завдань. Актором може бути людина, інша система, пристрій або програмний засіб. В даному випадку акторами визначено користувача застосунку, який отримує скорочений текст та скрипт, який створює скорочений текст.

Після визначення акторів системи, необхідно сформувати перелік усіх варіантів використання, з якими будуть взаємодіяти визначені актори. Серед основних варіантів використання системи: з боку користувача ввід тексту у спеціальне поле вводу; посилання тексту у вигляді тіла запиту на сервер; маніпулювання налаштуваннями додатку такими як мова тексту, кількість символів для збереження; перегляд скороченого тексту; можливість скопіювати скорочений текст; можливість змінювати розмір вікна додатку та елементів інтерфейсу користувача. З боку скрипта перетворення вхідного тексту у вихідний скорочений текст.

За результатами сформованих варіантів використання та акторів системи було розроблено діаграму варіантів використання, яку наведено на рис. 4.1.

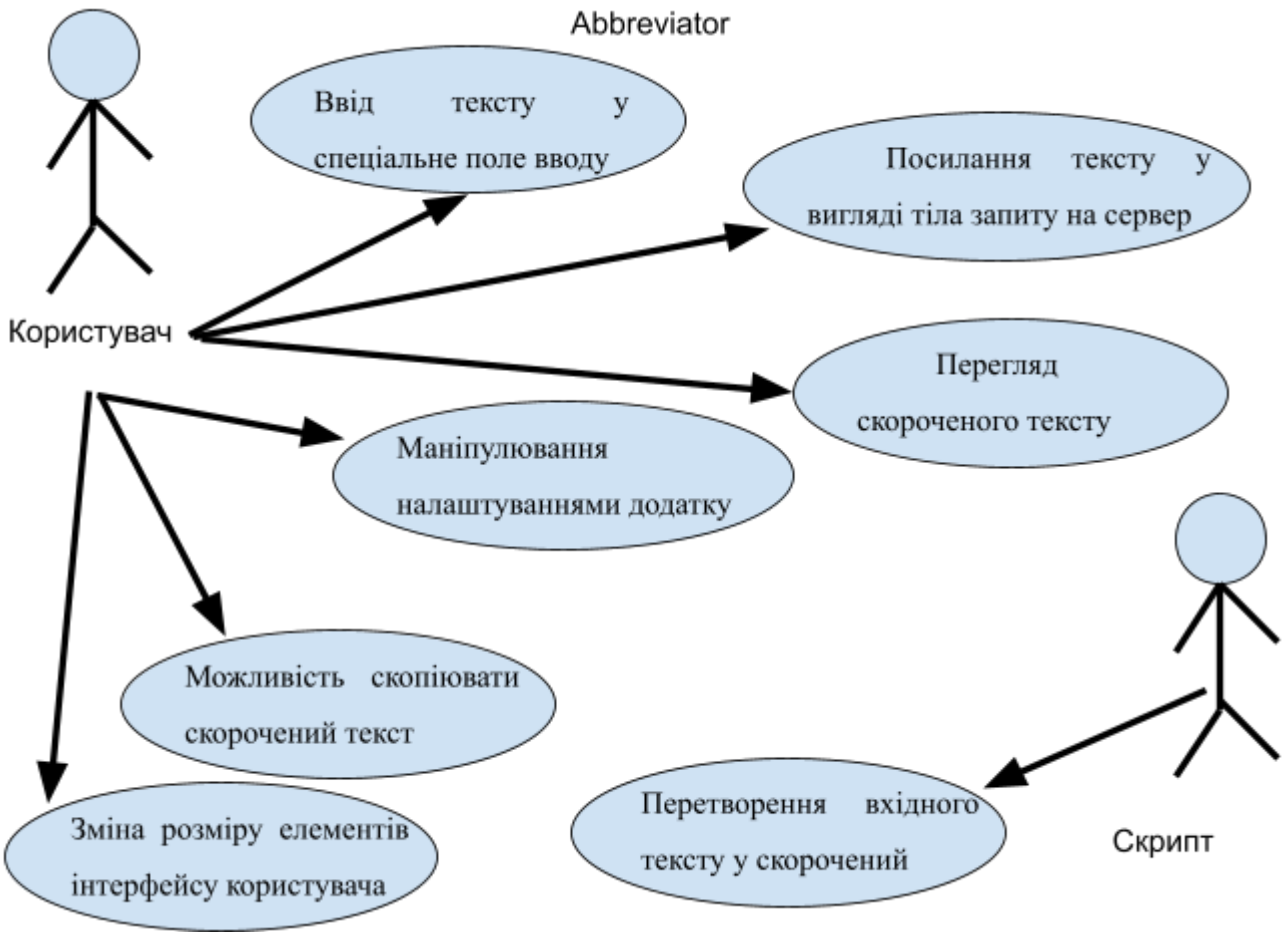

Рисунок 4.1 – Діаграма варіантів використання (прецедентів) системи

4.2 Створення діаграми діяльності (активностей)

Для моделювання процесу виконання операцій в мові UML використовуються діаграми діяльності. На діаграмі діяльності відображається логіка або послідовність переходу від однієї діяльності до іншої, при цьому увагу фіксується на результаті діяльності. Сам же результат може призвести до зміни стану системи або повернення деякого значення.

Діаграма діяльності в UML та SysML – це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас,

можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграми активностей будуються з обмеженої кількості фігур, з'єднаних стрілочками. Найважливіші типи фігур:

– скруглені прямокутники позначають дії;

– ромби позначають рішення;

– риски позначають початок (розподіл) чи кінець (об'єднання) паралельних активностей;

– чорний кружок позначає старт (початковий стан) процесу;

– чорний кружок в колі позначає кінець (кінцевий стан).

Для системи була створена діаграма діяльності, яка відображає основний процес проектованої системи – сканування зображення і його розпізнавання (рис. 4.2).

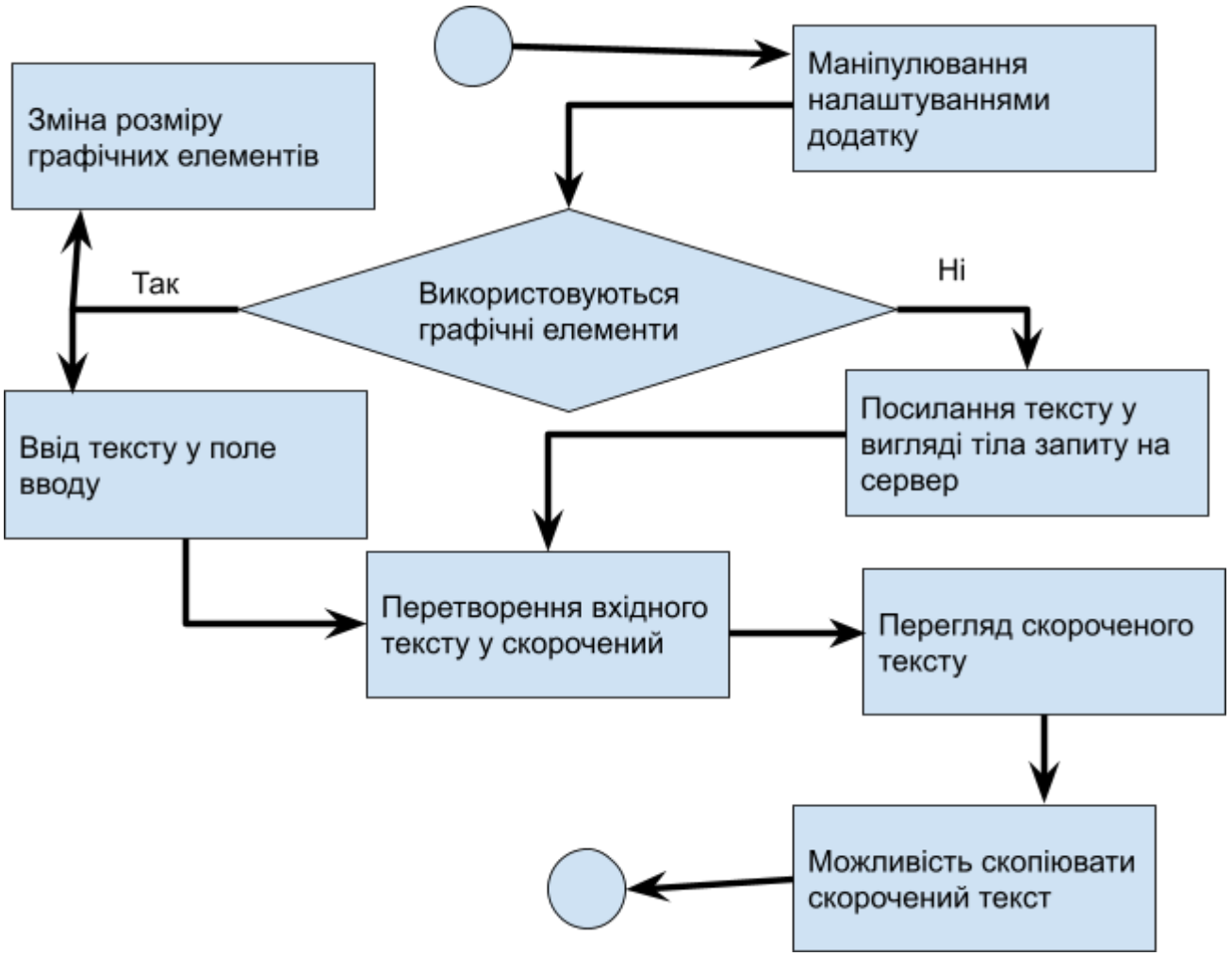

Рисунок 4.2 – Діаграма діяльності (активностей)

4.3 Створення діаграми послідовності

Також була створена діаграма послідовностей. Такі діаграми використовуються для уточнення діаграм прецедентів і більш детального опису логіки сценаріїв використання.

Діаграма послідовності — різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність надісланих повідомлень.

На діаграмі послідовностей показано у вигляді вертикальних ліній різні процеси або об'єкти, що існують водночас. Надіслані повідомлення зображуються у вигляді горизонтальних ліній, в порядку відправлення.

Визначені стандартом UML 2.0 діаграми послідовностей мають ті ж можливості, що і визначені стандартом UML 1.x, і підтримують додаткові можливості зміни стандартного порядку повідомлень.

Головна послідовність сценарію розпізнавання зображення складається з взаємодій, які відображені на діаграмі на рис. 4.3: користувач вводить текст для обробки в графічний елемент. Цей текст посилається на сервер Express.js, де Node.js скорочує текст. Текст обробляється і повертається у графічний елемент для вихідного тексту, який відображається користувачу.

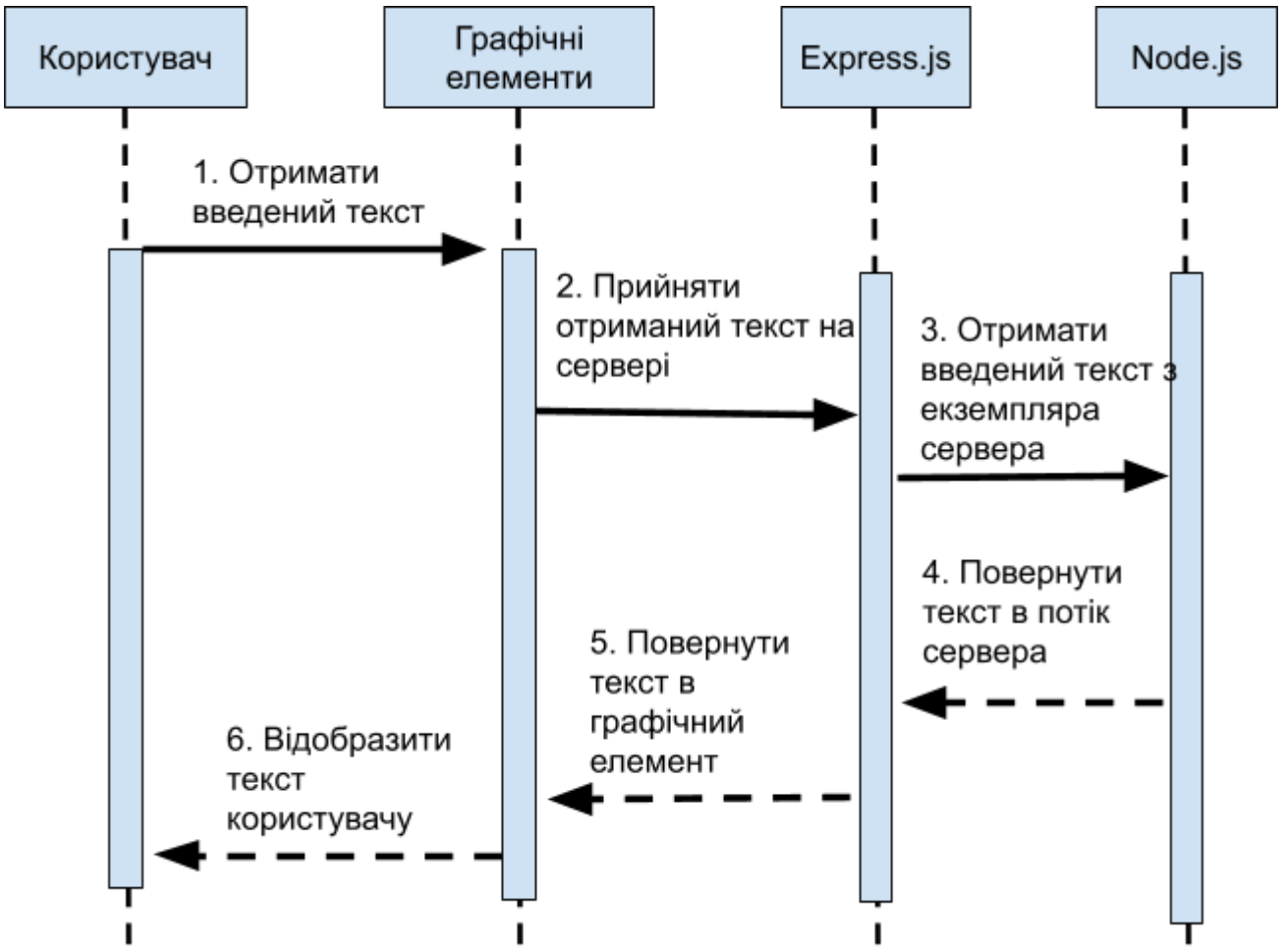

Рисунок 4.3 – Діаграма послідовностей

4.4 Проектування макету інтерфейсу застосунку

Після цього були створені макет інтерфейсу. Було вирішено зробити дизайн легким та простим. Усі необхідні елементи розташовані на одній веб-сторінці. Прототип інтерфейсу наведений на рис. 4.4.

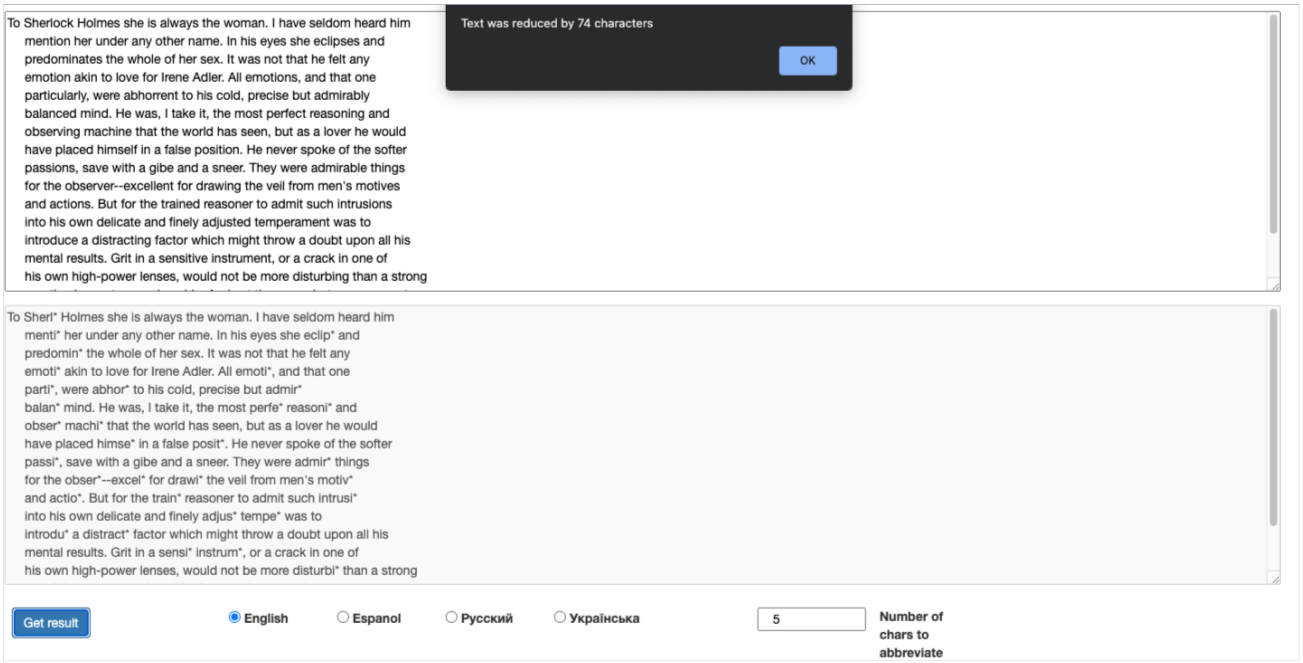

Рисунок 4.4 – Макет інтерфейсу застосунку

На сторінці відображаються поле для вводу та виводу тексту із можливістю прокрутки, якщо вміст полів перевищує розмір графічних елементів. Поле для виводу є недоступним для редагування користувачем. Ці поля займають відповідно верхню та нижню частину вікна браузера. Користувач має змогу змінювати розмір текстових полів. В нижній панелі розташовані кнопка для генерації скороченого тексту у полі виводу, кнопки-перемикачі для вибору мови та текстове поле для вводу мінімальної кількості символів, яку потрібно зберегти у слові.

Таким чином, на етапі проектування системи були створені UML-діаграми варіантів використання (прецедентів), активності і послідовності та розроблений макет інтерфейсу застосунку. Це допомогло зрозуміти основні варіанти використання системи та логіку роботи застосунку.

5 РЕАЛІЗАЦІЯ ВЕБ- ДОДАТКУ «ABBREVIATOR»

5.1 Створення інтерфейсу веб-сторінки за допомогою Vue.js

Для використання Vue.js необхідно підключити скрипт в HTML файлі.

```
<script
src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
```

Повний код додатку наведений у Додатку А.

Далі приведений фрагмент коду HTML коду із використанням Vue.js:

```
div id="radioButtonListCmp" >
                <div v-for="radioButton in radioButtonList">
                     <div :class="radioButton.cssClass">
                          <input class="form-check-input"
name="languageRadioButton" type="radio" :value="radioButton.value"
:id="radioButton.id" :checked="radioButton.checked"
onchange="radioButtonOnChange(this)">
                          <label class="form-check-label"
:for="radioButton.id">
                               {{radioButton.label}}
                          </label>
                     </div>
                </div>
           </div>
```

Javascript метод, який повертає Vue.js компонент із кнопками-перемикачами:

```
var radioButtonListCmp = new Vue({
    el: '#radioButtonListCmp',
    data: {
      radioButtonList: getRadioButtonList()
    }
})
```

Далі наведений скрипт для встановлення мови:

```
function setLanguage(language) {
    console.log('language', language);
    selectedLanguage = language;
```

```
        let req = JSON.stringify({lang: selectedLanguage});
        let request = new XMLHttpRequest();
        request.open("POST", "/lang", true);
        request.setRequestHeader("Content-Type", "application/json");
        request.addEventListener("load", function () {
            let data = JSON.parse(request.response).lang;
            document.getElementById('input').placeholder = data[0];
            document.getElementById('output').placeholder = data[1];
            document.getElementById('getResult').innerHTML =
data[2];
            message = [data[3], data[4]];

//document.getElementById('charsToAbbreviateInput').innerHTML =
data[5];
        });
        request.send(req);
    }

    function radioButtonOnChange(radioButton){
        console.log('radioButtonOnChange value', radioButton);
        setLanguage(radioButton.value);
    }

    let message = [];

    const languageValueList = abbreviator.getLanguageList();
    let selectedLanguage = languageValueList[0];
    setLanguage(selectedLanguage);
```

На веб-сторінці була використана анімація. Фрагмент коду для руху текстових полів під час завантаження сторінки:

```
    <link
        rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate
.min.css"/>
        ..........................
```

```
        <textarea  class = "textarea animate__animated
animate__bounce" id = "input" placeholder="Input data"></textarea>
```

Для розбиття сторінки на вертикальні стовпці однакової ширини була використана CSS бібліотека Bootstrap:

```
    <link rel="stylesheet"
href="http://netdna.bootstrapcdn.com/bootstrap/3.0.2/css/bootstrap.min
.css">
    ……….
    <div class="col-sm-2">
            <button type="button" class="btn btn-primary" id =
"getResult" value = ""  onclick="getResult()">Get Result</button>
        </div>
```

Для розбиття сторінки на верхню та нижню частини із полями для вводу та виводу були використані CSS стилі:

```
.textarea {
width: 98vw;
height: 42vh;
margin: 1vh;
}

#output{
margin-bottom: 2vh;
}
```

Вся клієнтська частина знаходиться в одній окремій директорії. Її вміст пересилається сервером, який може знаходитися віддалено. Там знаходяться файли формату .html, .css, .js.

5.2 Використання node.js, express.js

Завантажити node.js можна з офіційного сайту https://nodejs.org/en/. Інтерфейс сайту node.js із кнопками для завантаження зображений на рисунку 5.1:

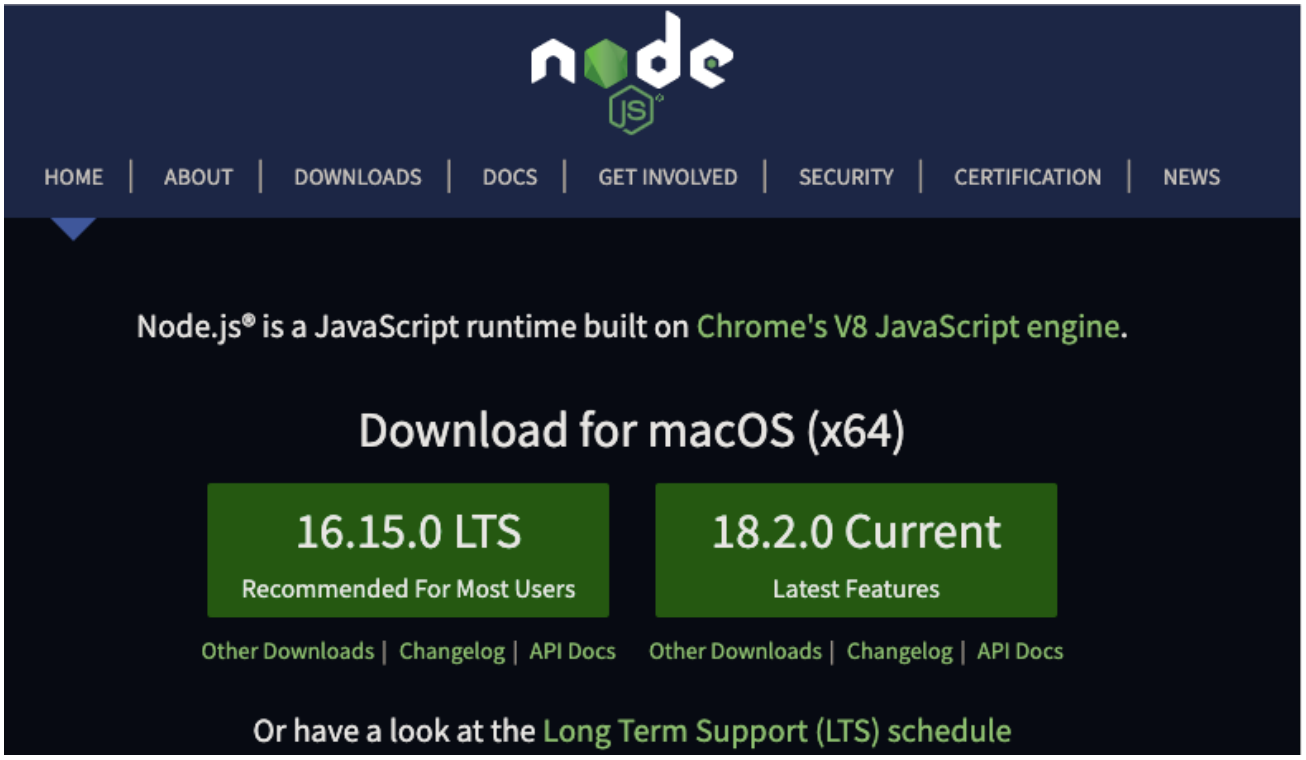

Рисунок 5.1 – Інтерфейс сайту node.js із кнопками для завантаження

Завантажити express.js та інші модулі можна створивши файл package.json:

```
{
  "name": "abbreviator",
  "version": "1.0.0",
  "scripts": {
    "start": "node server/server.js"
  },
  "dependencies": {
    "file-system": "^1.0.0",
    "express": "^4.16.4"
  }
}
```

Потім для встановлення express та file-system достатньо ввести команду npm install у терміналі. Для того, щоб запустити сервер достатньо ввести команду npm start, яка еквівалентна команді node server/server.js [11].

В залежності від способу використання веб-додатку, Javascript код може виконуватися як на клієнтській машині, так і на серверній. Для того, щоб уникнути дублювання коду, алгоритм для скорочення слів був винесений в

окремий файл і експортується для інших компонентів додатку [12]:

```
(function(exports){
   exports.getAbbreviatedText = function(input){
       prefixes = input.languageDetails.prefixes;
       utf = input.languageDetails.utf;
       textToAbbreviate = input.textToAbbreviate;
       lettersToAbbr = input.lettersToAbbr;
        return inspectText();
    };
})(typeof exports === 'undefined'? this['abbreviator']={}:
exports);
```

Використання експортованих функцій на клієнтській машині:

```
const output = abbreviator.getAbbreviatedText(input);
```

Використання експортованих функцій на серверній машині:

```
const abbreviator =
require('./../public/customUtil/abbreviator.js');
.....
const output = abbreviator.getAbbreviatedText(input);
```

Алгоритм скорочення слів використовує найуживаніші префікси тієї чи іншої мови. Файл префіксів англійської мови:

```
anti auto de dis down extra hyper il
im inter in ir mega mid mis non
over out post pre pro re semi sub
super tele trans ultra under un up
```

Веб-сторінка доступна на 4-ох мовах: англійській, іспанській, українській та російській. Текст повідомлень веб-сторінки для кожної мови зберігається у відповідному файлі gui.txt. Далі наведений приклад файлу для української мови:

```
Вхідні дані
Вихідні дані
Отримати результат
Текст скоротився на
 символів
```

Структура проекту зображена на малюнку 5.1:

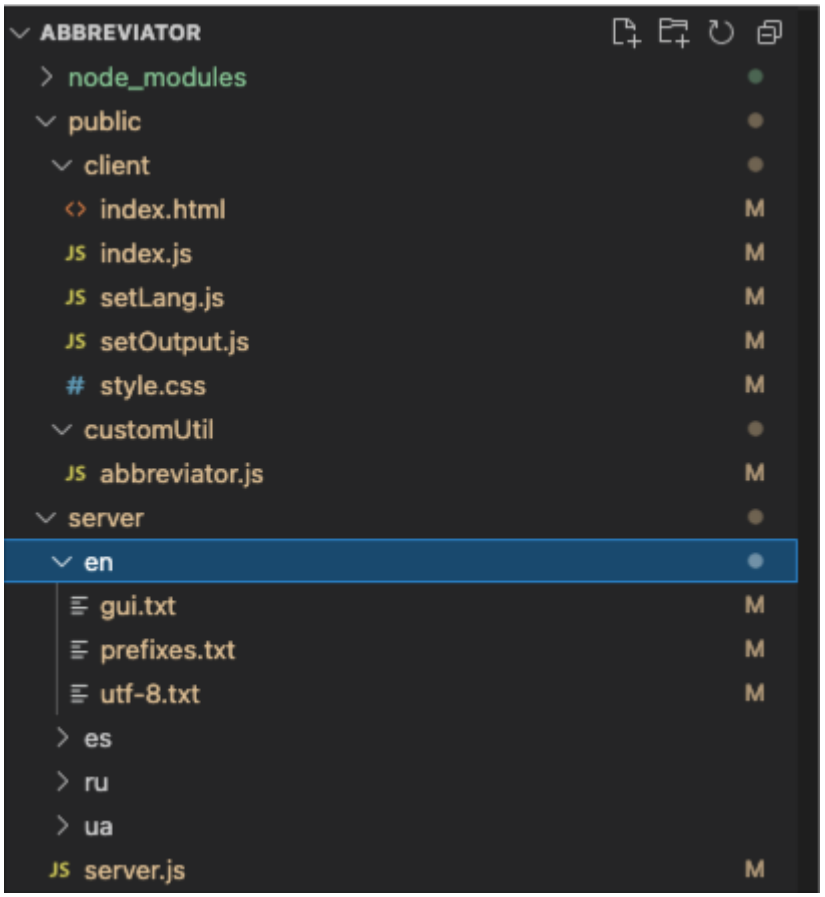

Рисунок 5.1 – Структура проекту

5.3 Опис та тестування застосунку

Для роботи з додатком знадобиться програмна бібліотека node.js. Слід виконати встановлення програми за замовчуванням. Запустимо сервер. Потрібно встановити необхідні модулі. Для цього у командному рядку у папці проекту виконати команду npm install. Коли модулі встановлені – npm start. Тепер можна підключити клієнта. У браузері перейти за посиланням ip:3000, де ip – ip адрес сервера. У файлі server.js можна змінити порт. При цьому пристрій клієнта має бути під'єднаний по одній локальній мережі із сервером.

Успішний запуск сервера представлений на рисунку 5.2:

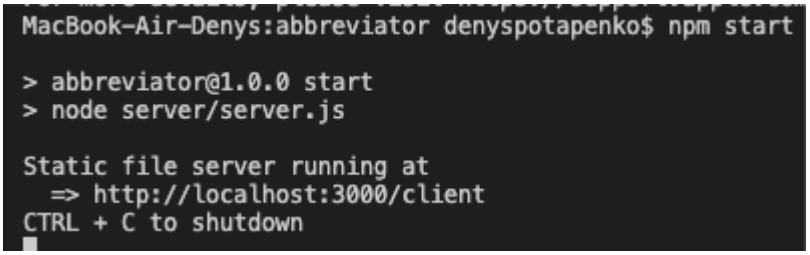

Рисунок 5.2 – Успішний запуск сервера

Протестуємо скорочення речення «Інтернет не має централізованого управління, правил використання чи доступу. Кожна складова мережа встановлює свої власні стандарти. Централізовано визначаються правила використання адресного простору Інтернет-протоколу та системи доменних імен.». Очікуваний результат: «Інтернет не має центр* управл*, правил використ* чи доступу. Кожна складо* мережа встано* свої власні станда*. Центр* визнача* прави* використ* адресн* простору Інтернет-протоколу та системи доменних імен.». Актуальний результат представлений на рисунку 5.3:

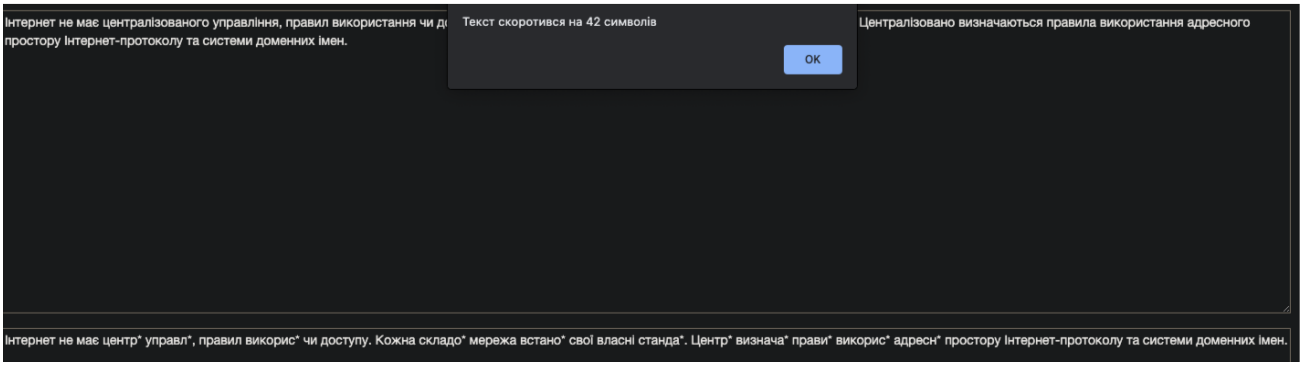

Рисунок 5.3 – Актуальний результат

Додаток був протестований на різних розширеннях екрану. Вигляд

додатку на екрані із розширенням 320x700px представлений на рисунку 5.4:

Рисунок 5.4 – Вигляд додатку на екрані із розширенням 320x700px

Вигляд додатку на екрані із розширенням 768x1000px представлений на рисунку 5.5:

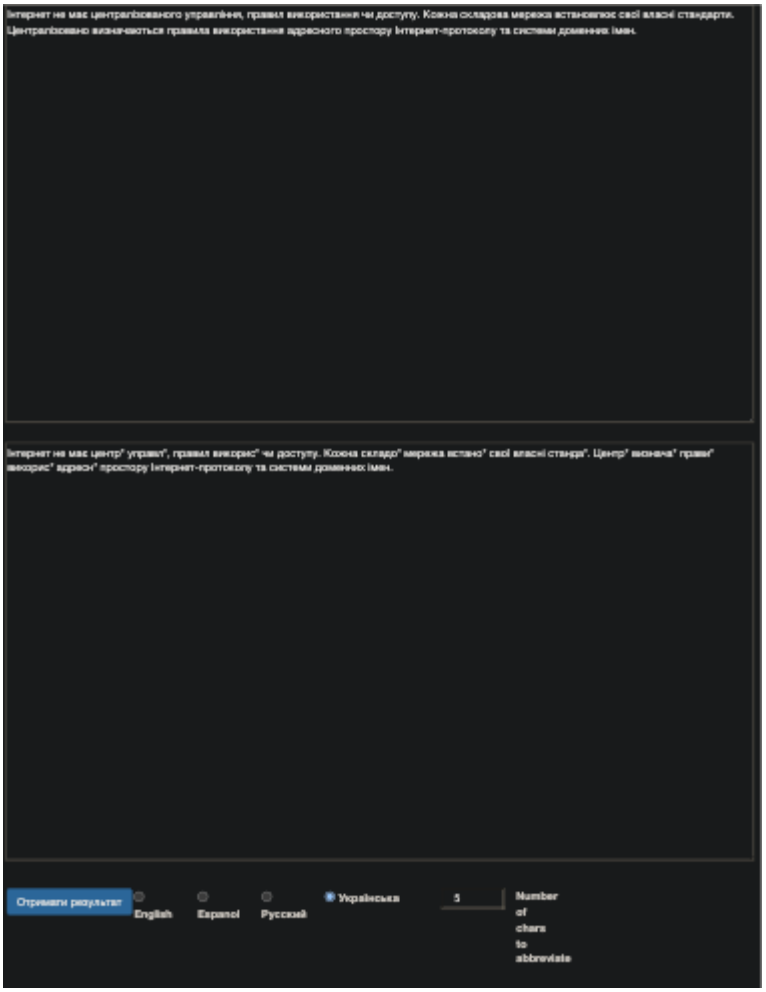

Рисунок 5.5 – Вигляд додатку на екрані із розширенням 768x1000px

Вигляд додатку на екрані із розширенням 1024x1000px представлений на рисунку 5.6:

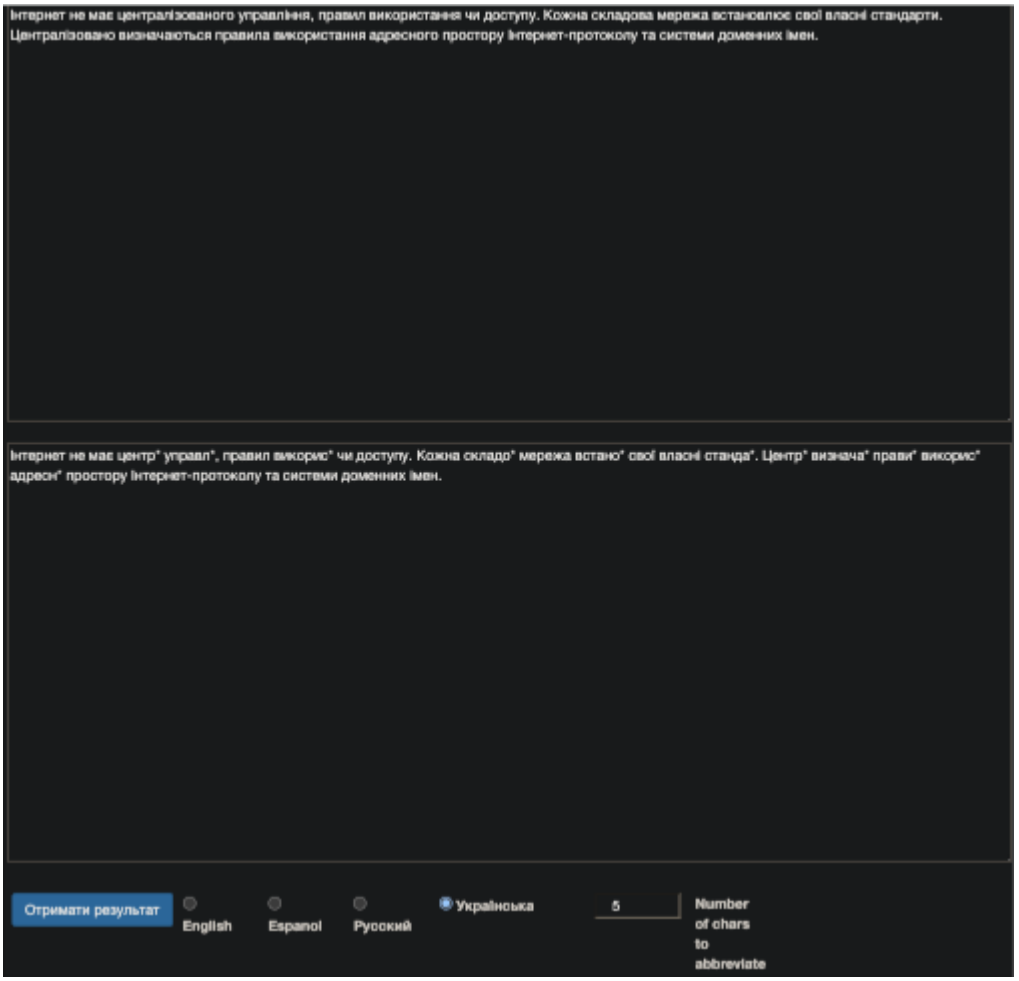

Рисунок 5.6 – Вигляд додатку на екрані із розширенням 1024x1000px

Вигляд додатку на екрані із розширенням 1400x1000px представлений на рисунку 5.7:

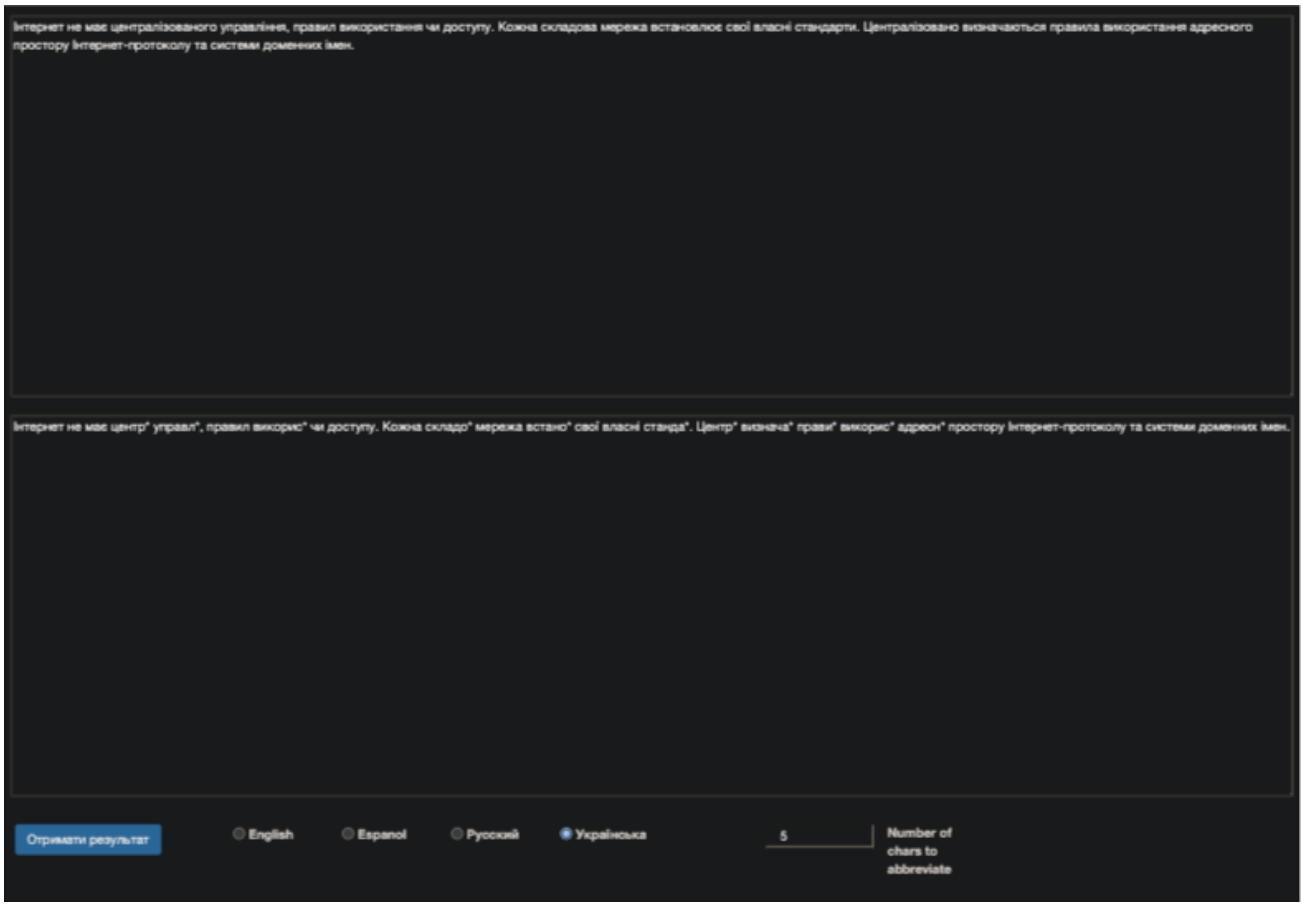

Рисунок 5.7 – Вигляд додатку на екрані із розширенням 1400x1000px

Додаток обробив текст, який складається з 575 000 символів за 0,5 секунди. Це зображено на рисунку 5.8.

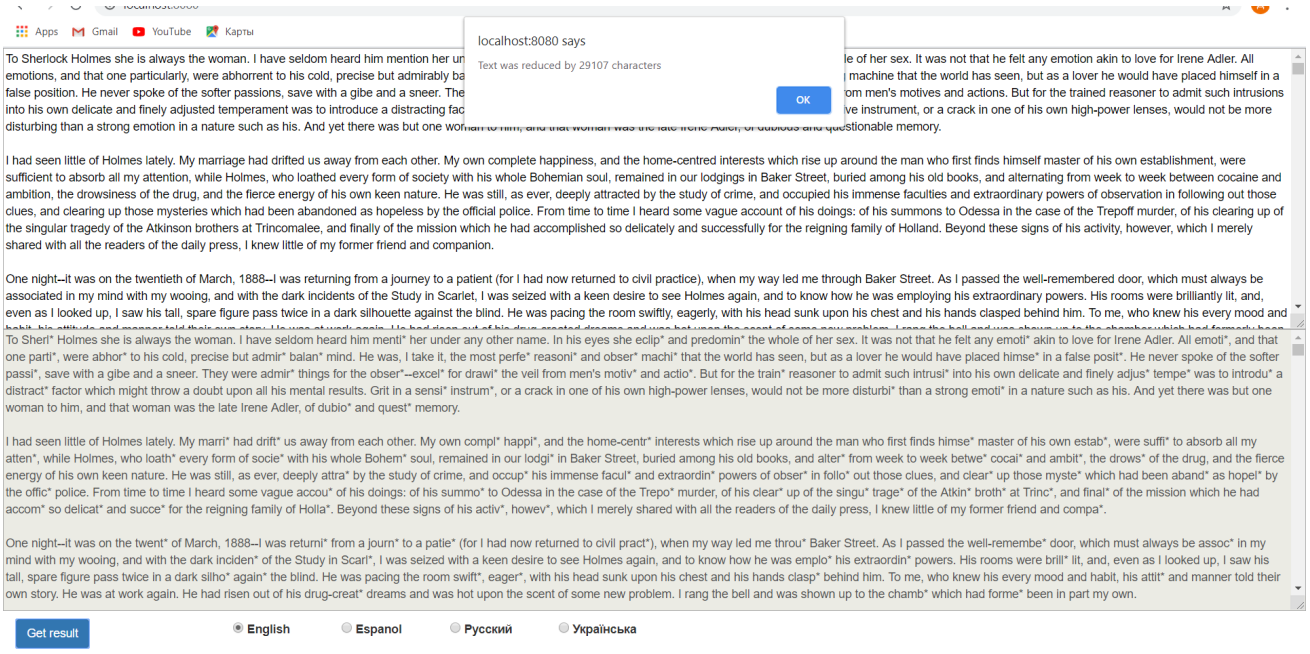

Рисунок 5.8 – Обробка великої кількості символів

Результат після посилання POST запиту зображений на рисунку 5.9.

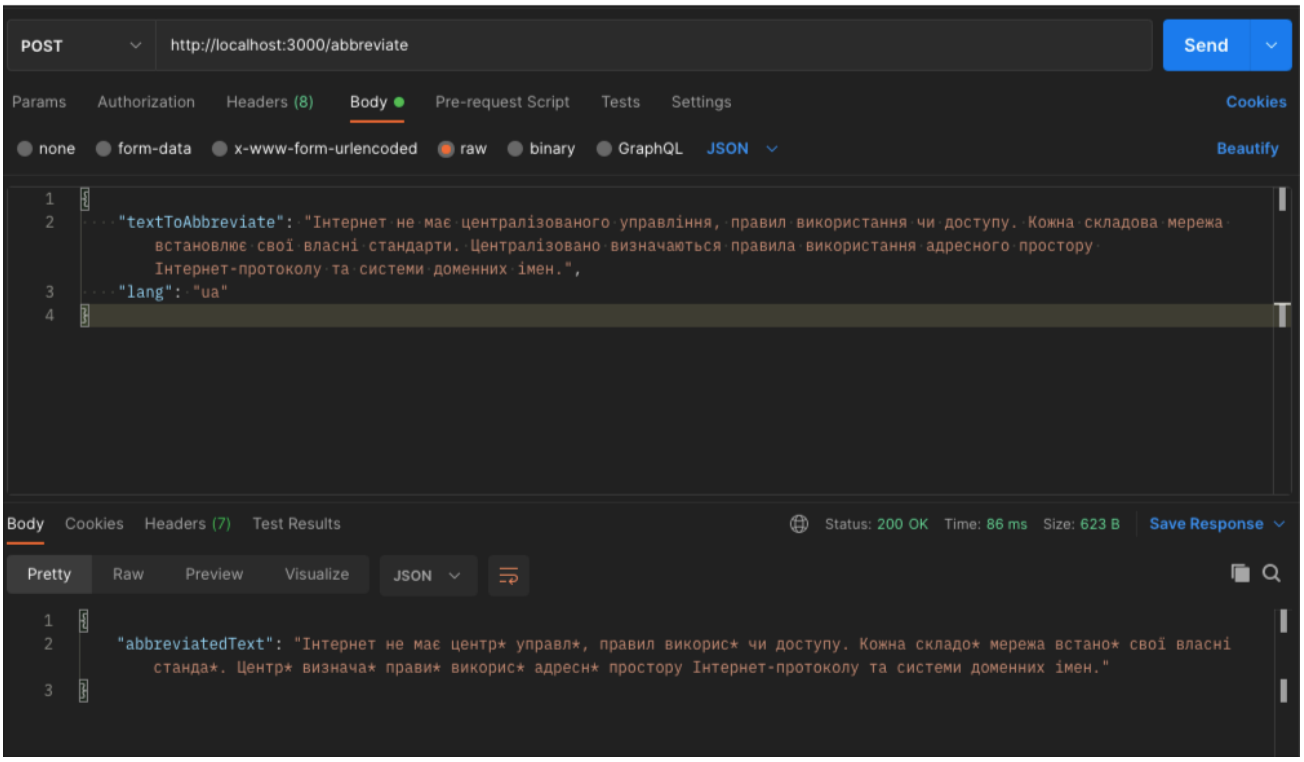

Рисунок 5.9 – Результат після посилання POST запиту

Ефективність скорочення слів в залежності від мови зображена в таблиці 5.1.

Таблиця 5.1– Ефективність скорочення слів в залежності від мови

Мова	Скорочує на
Англійська	7%
Іспанська	9%
Українська	10%
Російська	10%

ВИСНОВКИ

Веб-додатки мають явні переваги перед своїми програмними аналогами це їх мобільність (можливість роботи з ними з будь-якого місця), простота в створенні та використанні (для роботи потрібен лише браузер, незалежно від встановленої операційної системи), тому ця технологія стрімко набуває популярності, як серед користувачів, так і серед розробників.

За час написання дипломного проекту було поглиблено загальні базові знання в галузі веб-програмування. Було здобуто навички розробки веб-додатків з використанням сучасних веб-технологій. Поглиблено знання з аналізу та застосування інформаційних моделей і процесів з використанням сучасного програмного забезпечення. Отримано теоретичні та практичні знання з питань проектування, розробки, тестування, просування та подальшої підтримки програмного забезпечення.

У результаті виконання кваліфікаційної роботи бакалавра був створений веб-застосунок «Abbreviator» для скорочення слів. Для цього був проведений аналіз предметної області та порівняльний аналіз існуючих програм-аналогів. В якості програмного засобу розробки було обрано середовище Visual Studio Code з огляду на те, що цей редактор є найпопулярнішим серед веб розробників.

На етапі проектування були створені UML-діаграми варіантів використання (прецедентів), активності і послідовності та розроблені макети інтерфейсу застосунку. Були досліджені ролі акторів додатку та їх значимість. В роботі представлені можливості застосунку та поетапний приклад його використання. Було досліджено широкі можливості мови JavaScript для написання додатків як для серверних, так і для клієнтських машин при використанні технологій node.js. express.js, vue.js.

Було виконано тестування роботи. Було протестовано використання інтерфейсу користувача із полями для вводу на різних розширеннях екрану, виводу тексту; посилання POST запитів на сервер. Додаток успішно та швидко пройшов випробування на велику кількість вхідних символів.

Проблем при використанні виявлено не було.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Вебсайт – Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1%D1%81%D0%B0%D0%B9%D1%82 (дата звернення 14.04.2022)
2. Вебзастосунок – Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA (дата звернення 14.04.2022)
3. Багатоплатформенність – Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B3%D0%B0%D1%82%D0%BE%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%BD%D1%96%D1%81%D1%82%D1%8C (дата звернення 14.04.2022)
4. zip – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Zip (дата звернення 14.04.2022)
5. Аналіз вимог – Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D1%96%D0%B7_%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3 (дата звернення 14.04.2022)
6. Клієнт-серверна архітектура – Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0 (дата звернення 14.04.2022)
7. JavaScript – Вікіпедія. URL: https://uk.wikipedia.org/wiki/JavaScript (дата звернення 14.04.2022)
8. Node.js URL: https://nodejs.org/en/ (дата звернення 14.04.2022)
9. Express.js – Node.js web application framework. URL: https://expressjs.com (дата звернення 14.04.2022)

10. Vue.js - The Progressive JavaScript Framework | Vue.js. URL: https://vuejs.org (дата звернення 14.04.2022)
11. Кантелон М. Node.js в действии / Кантелон М. // Видавничий дім «Пітер». – 2014. – С. 13-19. (дата звернення 14.04.2022)
12. Metanit NodeJS | JSON и AJAX. URL: https://metanit.com/web/nodejs/4.6.php (дата звернення 14.04.2022)

ДОДАТОК А

Вихідний код програми

server.js

```
const fs = require("fs");
const express = require("express");
const app = express();
app.use(express.static("public"));
const jsonParser = express.json();
const abbreviator =
require('./../public/customUtil/abbreviator.js');

function readFromFile(path, regex) {
    return fs.readFileSync(path).toString('utf-8').split(regex);
}

function getPathByLanguage(language){
    return './server/' + language;
}

function getLanguageDetails(path){
    return {
         prefixes: readFromFile(path + '/prefixes.txt', /\s+/),
         utf: readFromFile(path + '/utf-8.txt', /\s+/)
    };
}

app.post("/prefixes", jsonParser, function (request, response) {
    let language = request.body.lang;
    const path = getPathByLanguage(language);
    console.log('loading ' + language + ' prefixes');
    const languageDetails = getLanguageDetails(path);
    response.json({
         "prefixes": languageDetails.prefixes,
         "utf": languageDetails.utf
    });
});

app.post("/lang", jsonParser, function (request, response) {
    let language = request.body.lang;
    const path = getPathByLanguage(language);
    console.log('loading ' + language + ' gui');
    let gui = readFromFile(path + '/gui.txt', /[\r\n]+/gm);
    response.json({"lang": gui});
});
```

```
app.post("/abbreviate", jsonParser, function (request, response)
{
    const textToAbbreviate = request.body.textToAbbreviate;
    const lettersToAbbr = request.body.lettersToAbbr
    ? request.body.lettersToAbbr
    : abbreviator.getDefaultLettersToAbbr();
    const language = request.body.lang
    ? request.body.lang
    : abbreviator.getLanguageList()[0];
    const path = getPathByLanguage(language);
    const languageDetails = getLanguageDetails(path);
    const input = {
         textToAbbreviate: textToAbbreviate,
         languageDetails: languageDetails,
         lettersToAbbr: lettersToAbbr
    };
    const abbreviatedText = abbreviator.getAbbreviatedText(input);
    response.json({"abbreviatedText": abbreviatedText});
});

let port = 3000;
app.listen(port);

console.log("Static file server running at\n  =>
http://localhost:" + port + "/client\nCTRL + C to shutdown");
```

index.html

```
<!DOCTYPE html>
<html>
    <head>
    <meta charset="utf-8">
    <link rel="stylesheet"
href="http://netdna.bootstrapcdn.com/bootstrap/3.0.2/css/bootstra
p.min.css">
    <link rel="stylesheet" href="style.css">
    <link
     rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/an
imate.min.css"
  />
    <script
src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
         <title>Abbreviator</title>
```

```
    </head>
    <body>
    <textarea  class = "textarea animate__animated
animate__bounce" id = "input" placeholder="Input
data"></textarea>
    <textarea  class = "textarea animate__animated
animate__bounce" id = "output" placeholder="Output data"
disabled></textarea>
    <p></p>
    <div class="col-sm-2">
         <button type="button" class="btn btn-primary" id =
"getResult" value = ""  onclick="getResult()">Get Result</button>
    </div>
    <form name="languages">
         <div id="radioButtonListCmp" >
              <div v-for="radioButton in radioButtonList">
                   <div :class="radioButton.cssClass">
                        <input class="form-check-input"
name="languageRadioButton" type="radio"
:value="radioButton.value" :id="radioButton.id"
:checked="radioButton.checked"
onchange="radioButtonOnChange(this)">
                        <label class="form-check-label"
:for="radioButton.id">
                             {{radioButton.label}}
                        </label>
                   </div>
              </div>
         </div>
    </form>
    <input id="charsToAbbreviateInput" class="col-sm-1" value="5">
                        <label for="charsToAbbreviateInput"
class="col-sm-1">
                             Number of chars to abbreviate
                        </label>

    <script src="./../customUtil/abbreviator.js"></script>
    <script src="setLang.js"></script>
    <script src="setOutput.js"></script>
```

```
    <script src="index.js"></script>
    </body>
</html>
```

index.js

```
var radioButtonListCmp = new Vue({
    el: '#radioButtonListCmp',
    data: {
      radioButtonList: getRadioButtonList()
    }
})

function getRadioButtonList(){
  let radioButtonList = [];
  const languageLabelList = ['English', 'Espanol', 'Русский',
'Українська'];

  let idCounter = 0;
  let cssClass = 'form-check ';
  for(const language of languageLabelList){
    radioButtonList.push({
     value: languageValueList[idCounter],
      id: 'radioButton' + idCounter++,
      label: language,
     cssClass: cssClass + 'col-sm-1'
    });
  }
  radioButtonList[languageLabelList.length-1].cssClass = cssClass
+ 'col-sm-2';
  radioButtonList[0].checked = true;

  return radioButtonList;
}
```

setLang.js

```
            function setLanguage(language) {
              console.log('language', language);
                 selectedLanguage = language;
    let req = JSON.stringify({lang: selectedLanguage});
             let request = new XMLHttpRequest();
            request.open("POST", "/lang", true);
          request.setRequestHeader("Content-Type",
                 "application/json");
       request.addEventListener("load", function () {
          let data = JSON.parse(request.response).lang;
         document.getElementById('input').placeholder =
                       data[0];
```

```
            document.getElementById('output').placeholder =
                            data[1];
            document.getElementById('getResult').innerHTML =
                            data[2];
                        message = [data[3], data[4]];

//document.getElementById('charsToAbbreviateInput').innerH
                        TML = data[5];
                                });
                        request.send(req);
                              }

          function radioButtonOnChange(radioButton){
              console.log('radioButtonOnChange value',
                        radioButton);
                    setLanguage(radioButton.value);
                              }

                        let message = [];

 const languageValueList = abbreviator.getLanguageList();
       let selectedLanguage = languageValueList[0];
              setLanguage(selectedLanguage);
```

setOutput.js

```
function getResult() {
    let req = JSON.stringify({lang: selectedLanguage});
    let request = new XMLHttpRequest();
    request.open("POST", "/prefixes", true);
    request.setRequestHeader("Content-Type", "application/json");
    request.addEventListener("load", function () {
         const textToAbbreviate =
document.getElementById('input').value;
         const lettersToAbbr =
parseInt(document.getElementById('charsToAbbreviateInput').value)
;
         const input = {
              textToAbbreviate: textToAbbreviate,
              languageDetails: JSON.parse(request.response),
              lettersToAbbr: lettersToAbbr
         };
         const output = abbreviator.getAbbreviatedText(input);

        document.getElementById('output').innerHTML = output;
        let difference = textToAbbreviate.length - output.length;
        setTimeout(() => alert(message[0] + difference +
message[1]), 50);
        console.log(textToAbbreviate.length);
```

```
    });
    request.send(req);
}
```

abbreviator.js

```
function inspectText() {
    output = '', lettersCounter = 0;
    let isCurrentLetter = false;
    let coherentChars = parseInt(utf[0]), charsToUpper =
parseInt(utf[1]);

    for(let i = 0; i < textToAbbreviate.length; i++) {
         isCurrentLetter = false;
         for(let j = 2; j < coherentChars + 2; j+=2) {
              if(textToAbbreviate[i] >= utf[j].charAt(0) &&
textToAbbreviate[i] <= utf[j+1].charAt(0)) {
                   isCurrentLetter = true;
                   break;
              }
         }
         if(!isCurrentLetter) {
              for(let j = coherentChars + 2; j < utf.length; j++)
{
                   if(textToAbbreviate[i] == utf[j].charAt(0)) {
                        isCurrentLetter = true;
                        break;
                   }
              }
         }

         if(!isCurrentLetter) {
              for(let j = 2; j < coherentChars + 2; j+=2) {
                   if(textToAbbreviate[i] >=
utf[j].toUpperCase().charAt(0) && textToAbbreviate[i] <=
utf[j+1].toUpperCase().charAt(0)) {
                        isCurrentLetter = true;
                        break;
                   }
              }
         }

         if(!isCurrentLetter) {
              for(let j = coherentChars + 2; j < charsToUpper;
j++) {
                   if(textToAbbreviate[i] ==
utf[j].toUpperCase().charAt(0)) {
                        isCurrentLetter = true;
                        break;
```

```
                    }
               }
          }

          if(isCurrentLetter) {
               ++lettersCounter;
          } else {
               if(lettersCounter > 0) {
                    processWord(i);
               }
               output += textToAbbreviate[i];
               lettersCounter = 0;
          }
     }

     if(isCurrentLetter) {
          processWord(textToAbbreviate.length);
     }

     return output;
}

function processWord(lastIndex) {
     let startIndex = lastIndex - lettersCounter;
     if(lettersCounter > lettersToAbbr + 1) {
          let firstLetter = textToAbbreviate[startIndex];
          let word = firstLetter.toLowerCase();
          for(let i = startIndex + 1; i < lastIndex; i++) {
               word += textToAbbreviate[i];
          }
          let letSymbols, hasPrefix = false, abbreviate = true;
          for(let i = 0; i < prefixes.length; i++) {
               if(word.startsWith(prefixes[i])) {
                    letSymbols = prefixes[i].length +
lettersToAbbr;
                    if(letSymbols > word.length - 2)
                         abbreviate = false;
                    hasPrefix = true;
                    break;
               }
          }
          if(!hasPrefix){
               letSymbols = lettersToAbbr;
          }
          let newWord = firstLetter + word.substr(1);
          if(abbreviate) {
               output += newWord.substr(0, letSymbols);
               output += '*';
```

```
        } else {
            output += newWord;
        }
    } else {
        for(let i = startIndex; i < lastIndex; i++) {
            output += textToAbbreviate[i];
        }
    }
}

let lettersToAbbr;
let prefixes;
let utf;
let output = '', lettersCounter = 0;
let textToAbbreviate;
const languageList = ['en', 'es', 'ru', 'ua'];

(function(exports){
   exports.getAbbreviatedText = function(input){
      prefixes = input.languageDetails.prefixes;
      utf = input.languageDetails.utf;
      textToAbbreviate = input.textToAbbreviate;
      lettersToAbbr = input.lettersToAbbr;
        return inspectText();
    };
    exports.getLanguageList = function(){
          return languageList;
     };
     exports.getDefaultLettersToAbbr = function(){
          return 5;
     };
})(typeof exports === 'undefined'? this['abbreviator']={}:
exports);
```

style.css

```
.textarea {
    width: 98vw;
    height: 42vh;
    margin: 1vh;
}

#output{
    margin-bottom: 2vh;
}
```