

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

Кафедра дискретного аналізу та інтелектуальних систем

(повна назва кафедри)

КУРСОВА РОБОТА

**КОМПЛЕКСНИЙ ЗАХИСТ ВЕБ-ДОДАТКІВ, НАПИСАНИХ МОВОЮ
ПРОГРАМУВАННЯ JAVASCRIPT**

Виконав: студент групи ПМІ-33

спеціальності 122 – комп'ютерні науки

(шифр і назва спеціальності)

Ваврикович М. І.

(підпис)

(прізвище та ініціали)

Керівник

Позднякова І.В.

(підпис)

(прізвище та ініціали)

Зміст

ВСТУП.....	1
РОЗДІЛ 1. МЕТОДИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ.....	3
2.1 JavaScript.....	3
2.2 TypeScript.....	3
2.3 Node.js.....	4
2.4 Express.....	4
2.5 Sequelize.....	4
2.6 PostgreSQL.....	5
2.7 Gmail.....	5
2.8 Next.js.....	5
2.9 React.....	6
2.10 JSON Web Token (JWT).....	6
2.11 REST (Representational State Transfer).....	6
2.12 RESTful API.....	7
2.13 Клієнт-серверна архітектура.....	7
РОЗДІЛ 2. ЗБЕРІГАННЯ ДАНИХ ЗАСТОСУНКУ.....	8
2.1 Таблиця "users".....	8
2.2 Таблиця "messages".....	8
2.3 Таблиця "verify_code".....	9
РОЗДІЛ 3. ДЕТАЛЬНИЙ ОПИС ІНТЕРФЕЙСУ ТА КОРИСТУВАЦЬКИХ МОЖЛИВОСТЕЙ ЗАСТОСУНКУ.....	10
3.1 Головна сторінка.....	10
3.2 Реєстрація користувачів.....	11
3.3 Автентифікація та вхід в систему.....	11
3.4 Персональний профіль.....	12

3.5 Список користувачів.....	13
3.6 Приватний чат.....	15
РОЗДІЛ 4. КОМПЛЕКСНИЙ ЗАХИСТ ЗАСТОСУНКУ.....	17
3.1 Двофакторна автентифікація.....	17
3.2 Аксес та рефреш токен.....	17
3.3 Хешування паролів користувачів.....	18
3.4 Шифрування повідомлень користувачів.....	18
3.5 Валідація тіла запиту.....	19
РОЗДІЛ 5. ЕФЕКТИВНІСТЬ ТА ПЕРЕВАГИ ЗАСТОСУНКУ.....	21
5.1. Переваги застосунку "Chat App" в порівнянні з існуючими застосунками. 21	
5.2. Визначення потенційних можливостей для подальшого вдосконалення	21
ВИСНОВОК.....	23
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	24

ВСТУП

У сучасному світі Інтернет-технології стали неодмінною складовою нашого повсякденного життя. Веб-додатки, розроблені з використанням мови програмування JavaScript, стали надзвичайно поширеними та мають велику значущість для бізнесу, розваг та комунікації. Ці додатки забезпечують широкий спектр функціональних можливостей, від інтерактивних інтерфейсів до обробки складних бізнес-логік, що робить їх незамінними компонентами сучасного веб-середовища.

Проте, разом зі зростанням популярності веб-додатків, з'явилися й нові виклики у сфері безпеки. Веб-додатки, зокрема ті, що використовують JavaScript, стали об'єктом специфічних загроз, які можуть вплинути на їх конфіденційність, цілісність та доступність. Зловмисники використовують різноманітні техніки, такі як міжсайтовий скриптинг (XSS), ін'єкція SQL-запитів (SQL injection), викрадення ідентифікаційних даних та багато інших, щоб отримати незаконний доступ до даних користувачів або спричинити шкоду системі.

В цьому контексті, комплексний захист веб-додатків, написаних мовою програмування JavaScript, стає критично важливим завданням. Такий захист передбачає впровадження широкого спектру технологій, методів та практик, які допомагають запобігти та виявити можливі вразливості, а також ефективно реагувати на атаки. Він забезпечує високий рівень безпеки та захисту для веб-додатків, що дозволяє користувачам впевнено використовувати їх та довіряти їхній безпеці.

Актуальність. У сучасному світі, де онлайн-платформи та електронні сервіси займають важливе місце, безпека веб-додатків стає критично важливою. Використання JavaScript у веб-розробці дозволяє створювати інтерактивні та ефективні додатки, але разом з цим посилюється ймовірність вразливостей та атак на ці додатки. Забезпечення комплексного захисту таких веб-додатків є нагальним завданням для забезпечення конфіденційності, цілісності та доступності інформації.

Мета дослідження: розробити комплексний захист веб-додатків, написаних мовою програмування JavaScript. Визначення найбільш ефективних заходів та методів, які допоможуть у запобіганні вразливостям та атакам на ці додатки. Аналізу існуючих методів захисту та їх ефективність, а також визначення можливих напрямів подальшого розвитку комплексного захисту веб-додатків на базі JavaScript.

Предмет дослідження: комплексний захист веб-додатків, що розроблені з використанням мови програмування JavaScript. Вразливості, що можуть виникнути в процесі розробки таких додатків, пошук ефективних заходів для запобігання атакам, таким як міжсайтовий скриптинг, SQL-ін'єкція та інші типи атак, що загрожують безпеці веб-додатків на JavaScript.

Методи дослідження: аналіз наукової літератури, статей, публікацій та стандартів, що стосуються захисту веб-додатків. Тестування для оцінки ефективності запропонованих методів та заходів безпеки. Вивчення практик та методик, що використовуються в реальних проектах для захисту веб-додатків на JavaScript.

Структура роботи: курсова робота складається зі вступу, п'ятих розділів, висновку, списку використаних джерел.

РОЗДІЛ 1. МЕТОДИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ

2.1 JavaScript

JavaScript - це програмна мова, яка використовується для розробки веб-додатків і веб-сторінок. Вона є однією з основних мов, які використовуються у фронтенд-розробці, тобто для створення взаємодії із користувачем на веб-сторінках. JS може бути використана для додавання динамічної поведінки до веб-сторінок, маніпуляції елементами DOM (об'єктна модель документа), валідації форм, роботи з HTTP-запитами та багатьма іншими задачами.

2.2 TypeScript

Це розширення мови програмування JavaScript, яке надає статичну типізацію та додаткові функції для покращення розробки великих та складних проектів. Він компілюється в чистий JavaScript, що дозволяє використовувати його в будь-якому середовищі, де підтримується JavaScript.

Однією з головних переваг TypeScript є його підтримка статичної типізації. За допомогою TypeScript можна вказати типи для змінних, параметрів функцій, повертаємого значення та інших елементів коду. Це дозволяє виявити та уникнути помилок типізації на етапі компіляції, забезпечуючи більшу стабільність та надійність програмного коду. Також TypeScript надає інтелектуальний автодоповнювач коду, перевірку на використання невизначених значень та інші корисні функції, що полегшують розробку.

Крім статичної типізації, TypeScript підтримує всі можливості, що є в JavaScript. Це означає, що всі існуючі бібліотеки та фреймворки JavaScript можна використовувати безпосередньо в TypeScript. Більше того, TypeScript активно розвивається та має велику спільноту розробників, що забезпечує наявність багато інструментів, бібліотек та документації для підтримки та розширення можливостей мови.

2.3 Node.js

Node.js є потужним середовищем виконання на базі JavaScript, яке відкриває безліч можливостей для розробки веб-додатків. Воно побудоване на швидкому та ефективному двигуні V8, розробленому Google, що забезпечує високу швидкодію виконання JavaScript-коду. Однак, Node.js не просто дозволяє виконувати JavaScript на стороні сервера, воно також надає розробникам доступ до багатьох потужних інструментів і бібліотек для розробки серверної логіки.

Node.js є ідеальним вибором для створення масштабованих та ефективних веб-додатків, оскільки воно дозволяє обробляти багато запитів одночасно завдяки необхідній асинхронності. Ви можете легко створювати веб-сервери, реалізовувати API, керувати файлами та багато іншого за допомогою Node.js. Його широкі можливості, активна спільнота розробників і регулярні оновлення роблять його важливим інструментом для будь-якого розробника веб-додатків.

2.4 Express

Express - це легковаговий та гнучкий фреймворк для розробки веб-додатків на платформі Node.js. Він надає простий та інтуїтивно зрозумілий спосіб створення серверної логіки, маршрутизації та обробки HTTP-запитів. Express дозволяє розробникам швидко відтворювати веб-сервери та створювати ефективні API. Завдяки своїй гнучкості, Express дозволяє налаштувати розробку веб-додатків під конкретні потреби проекту. Ви можете легко додавати маршрути, обробники запитів та middleware для обробки різних типів запитів. Express також надає вам доступ до широкого спектру плагінів та додаткових модулів, що спрощують розробку різноманітних функціональностей, таких як аутентифікація, обробка форм, робота з базами даних та багато іншого.

2.5 Sequelize

Sequelize - це об'єктно-реляційний мапер (ORM) для Node.js, який спрощує взаємодію з реляційними базами даних. Він надає розробникам зручний спосіб моделювати та взаємодіяти з базою даних за допомогою об'єктно-орієнтованого підходу. Sequelize підтримує багато реляційних СУБД,

таких як PostgreSQL, MySQL, SQLite і MSSQL, що дає вам можливість використовувати будь-яку з цих баз даних у вашому проекті.

За допомогою Sequelize ви можете визначати моделі, які відображають таблиці бази даних, і здійснювати з ними операції CRUD (створення, читання, оновлення, видалення). Він також надає можливості збірки складних запитів, включаючи з'єднання таблиць, умови, сортування та групування даних. Sequelize дозволяє зручно працювати з базою даних, забезпечуючи вам потужні інструменти для ефективної роботи з даними.

2.6 PostgreSQL

PostgreSQL є потужною і відкритою системою управління реляційними базами даних (RDBMS). Вона володіє широким спектром функцій і можливостей, що дозволяють ефективно зберігати, організовувати та обробляти дані. PostgreSQL підтримує розширену мову запитів SQL і забезпечує надійність, стабільність та безпеку даних.

PSQL є вибором багатьох розробників через свою надійність і високу продуктивність. Вона має велику кількість вбудованих функцій, таких як транзакції, індексація, забезпечення цілісності даних та багато іншого. PostgreSQL також підтримує розширення та може бути налаштована під ваші потреби. Вона є досить гнучкою і може працювати з різними типами даних, включаючи числа, рядки, дати, географічні дані та багато іншого.

2.7 Gmail

Для відправки електронних листів з Node.js ви можете використовувати Gmail API. Це дозволяє вам підключити свій Node.js додаток до облікового запису Gmail і надсилати електронні листи від імені користувача.

Gmail API забезпечує розширений набір функцій, таких як створення та надсилання електронних листів, робота з вкладеннями, розсилки, керування мітками та багато іншого. Він пропонує простий та зрозумілий API, який дозволяє легко інтегрувати функціональність електронної пошти в ваш додаток.

2.8 Next.js

Next.js є популярним фреймворком для розробки веб-додатків з використанням React.js. Він надає потужні інструменти для побудови швидких, масштабованих та SEO-придатних додатків. Next.js підтримує серверний рендеринг та статичне генерування, що дозволяє досягти оптимальної продуктивності та швидкого відгуку.

Завдяки Next.js ви можете ефективно створювати односторінкові та багатосторінкові додатки, керувати маршрутизацією, працювати зі статичними ресурсами, управляти станом додатка та багато іншого. Він має широкий спектр функцій, включаючи підтримку серверного API, автоматичне кодогенерування, оптимізацію зображень, підтримку CSS-модулів та багато іншого.

2.9 React

React є однією з найпопулярніших бібліотек JavaScript для розробки користувацького інтерфейсу. Він дозволяє створювати динамічні та інтерактивні веб-додатки з використанням компонентного підходу.

React забезпечує зручний спосіб організації коду та керування станом додатка. Він використовує віртуальний DOM, що дозволяє ефективно оновлювати тільки змінені частини інтерфейсу. React також підтримує розширення за допомогою багатьох сторонніх бібліотек та надає багато можливостей для розробки високоякісних і швидких додатків.

2.10 JSON Web Token (JWT)

Це відкритий стандарт для створення токенів, які можна використовувати для безпечної автентифікації та авторизації в розподілених системах. Він забезпечує надійний спосіб передавати інформацію між двома сторонами в JSON-форматі. JWT має просту структуру, що складається з трьох частин: заголовку, тіла та підпису.

2.11 REST (Representational State Transfer)

Є набором принципів для взаємодії між клієнтом та сервером у веб-розробці. Цей підхід базується на використанні стандартних HTTP методів, таких як GET, POST, PUT і DELETE, для виконання операцій з ресурсами. REST передбачає, що сервер не зберігає стан користувача між запитами, а кожен запит містить всю необхідну інформацію для обробки. Це забезпечує безстанну комунікацію між клієнтом і сервером.

2.12 RESTful API

Використовує принципи REST для побудови веб-сервісів. Це означає, що він надає доступ до ресурсів через інтернет, використовуючи HTTP методи для взаємодії з цими ресурсами. RESTful API дозволяє розділити клієнтську та серверну частини додатку, що сприяє модульності та масштабованості. Він забезпечує передачу даних у форматі, такому як JSON або XML, для простоти обробки та інтеграції з різними системами.

2.13 Клієнт-серверна архітектура

Є шаблоном програмного забезпечення, в якому функції додатку розділені між клієнтом і сервером. У цій архітектурі сервери надають інформацію та послуги, а клієнти використовують ці послуги, взаємодіючи з сервером за допомогою мережі. Ця архітектура забезпечує ефективну комунікацію між компонентами, зменшуючи залежність та покращуючи надійність та продуктивність додатків.

Ці технології та методи використовуються разом для створення сучасних веб-додатків з високою продуктивністю, надійністю та швидкістю відгуку. Така комбінація дозволяє розробникам швидко створювати потужні та функціональні додатки з використанням сучасних практик розробки.

РОЗДІЛ 2. ЗБЕРІГАННЯ ДАНИХ ЗАСТОСУНКУ

У застосунку Chat App дані зберігаються у базі даних PostgreSQL (psql). Для організації ефективного та структурованого зберігання інформації, використовуються три таблиці: "users" (користувачі), "verify_code" (коди підтвердження) та "messages" (повідомлення).

2.1 Таблиця "users"

Таблиця "users" містить дані про зареєстрованих користувачів застосунку. Вона включає такі поля:

- id: унікальний ідентифікатор користувача;
- firstName: ім'я користувача;
- lastName: прізвище користувача;
- email: електронна пошта користувача;
- password: хеш пароля користувача;
- phone: номер телефону користувача.
- createdAt: час створення запису

Ці дані дозволяють ідентифікувати користувачів та забезпечувати їх аутентифікацію в системі.

2.2 Таблиця "messages"

Таблиця "messages" використовується для зберігання повідомлень, які надсилаються між користувачами. В таблиці містяться наступні поля:

- id: унікальний ідентифікатор повідомлення;
- message: текст повідомлення;
- senderId: ідентифікатор користувача, що відправляє повідомлення;
- recipientId: ідентифікатор користувача, що отримує повідомлення.
- createdAt: час створення запису

Ці дані дозволяють відстежувати комунікацію між користувачами та забезпечувати обмін повідомленнями.

2.3 Таблиця "verify_code"

Таблиця "verify_code" використовується для зберігання тимчасових кодів підтвердження, які використовуються під час процесу автентифікації. Вона містить наступні поля:

- code: код підтвердження;
- id: унікальний ідентифікатор коду;
- email: електронна пошта, пов'язана з кодом підтвердження.
- createdAt: час створення запису

Ця таблиця дозволяє зберігати та перевіряти коди підтвердження для підтвердження ідентичності користувачів.

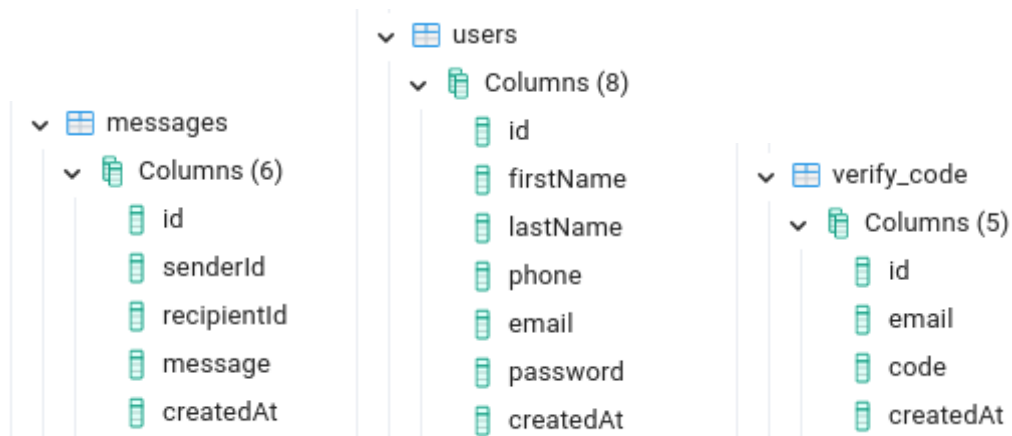


Рисунок 2.1 – Таблиці 'users', 'messages', 'verify_code'

Використання бази даних PostgreSQL (psql) забезпечує надійне та масштабоване зберігання даних для застосунку Chat App. Таблиці "users", "messages" та "verify_code" дозволяють ефективно управляти інформацією про користувачів, повідомлення та коди підтвердження, що забезпечує зручну та безпечну роботу з даними у застосунку.

РОЗДІЛ 3. ДЕТАЛЬНИЙ ОПИС ІНТЕРФЕЙСУ ТА КОРИСТУВАЦЬКИХ МОЖЛИВОСТЕЙ ЗАСТОСУНКУ

1. Додаток "Chat App" є мобільним застосунком, створеним для спілкування та реєстрації користувачів. Цей додаток надає зручну та ефективну платформу для обміну повідомленнями між користувачами, де вони можуть зареєструватися, створювати профілі та спілкуватися один з одним у режимі реального часу.

Основні функції та можливості “Chat App” включають:

3.1 Головна сторінка

При запуску застосунку, буде відображена головна сторінка

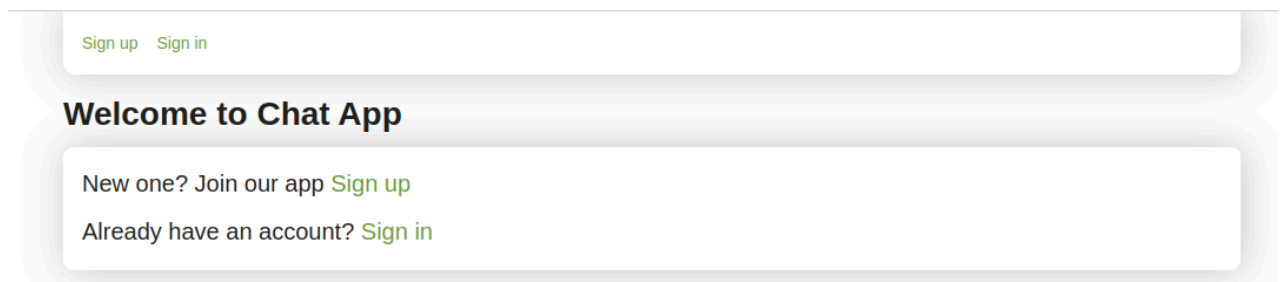
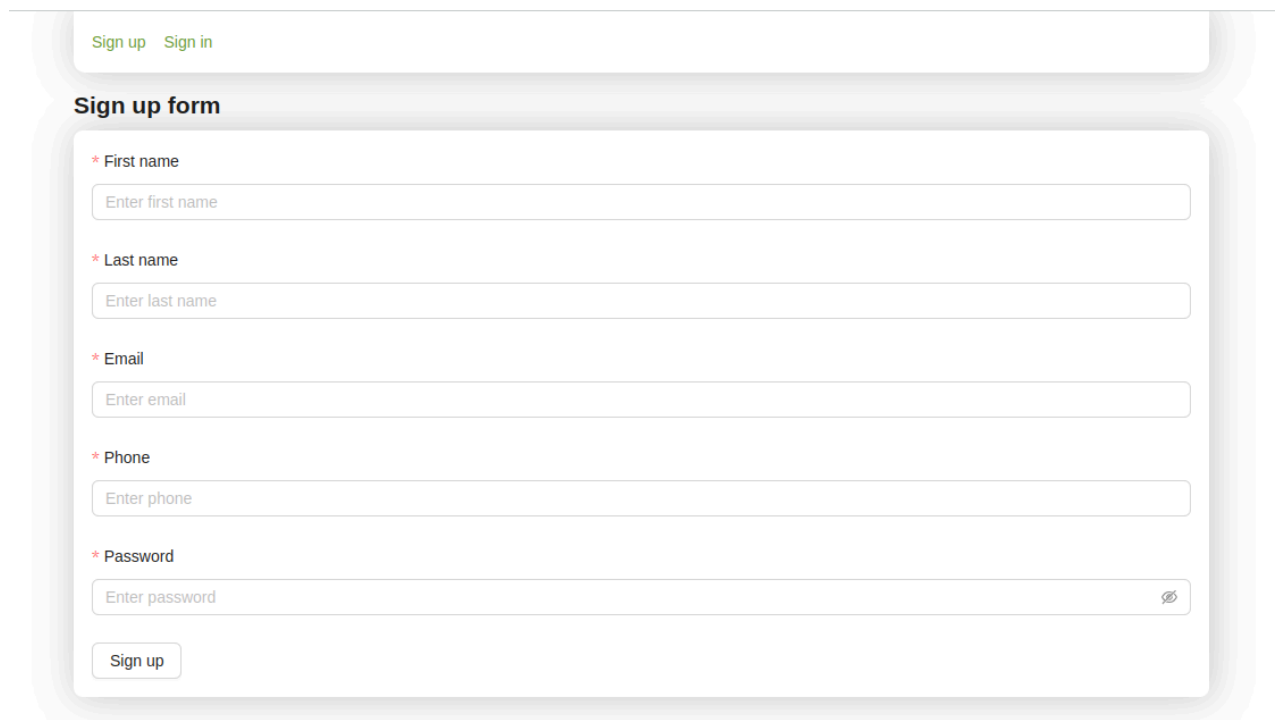


Рисунок 3.1 – Головна сторінка

Звідси можна перейти або на сторінку реєстрації для створення нового акаунта користувача, або сторінку авторизації для входу у свій профіль.

3.2 Реєстрація користувачів



The image shows a registration form titled "Sign up form". At the top, there are two links: "Sign up" and "Sign in". The form contains five input fields, each with a red asterisk indicating a required field: "First name" (placeholder: "Enter first name"), "Last name" (placeholder: "Enter last name"), "Email" (placeholder: "Enter email"), "Phone" (placeholder: "Enter phone"), and "Password" (placeholder: "Enter password" with a toggle icon). A "Sign up" button is located at the bottom of the form.

Рисунок 3.2 – Сторінка реєстрації

Додаток дозволяє новим користувачам створювати облікові записи, заповнюючи необхідну інформацію, таку як ім'я користувача, прізвище, електронна пошта, телефон та пароль. Після цього потрібно натиснути кнопку 'Sign up' для відправлення запиту на сервер.

3.3 Автентифікація та вхід в систему

Користувачі можуть увійти в свої облікові записи, використовуючи введені при реєстрації дані. Автентифікація забезпечує безпеку та обмежує доступ до особистої інформації.

Sign up Sign in

Sign in form

* Email

skystar.peach+1@gmail.com

* Password

.....

Sign in

Have not an account? [Sign up](#)

Рисунок 3.3 – Сторінка першого кроку авторизації

На першому етапі автентифікації користувач повинен ввести емейл та пароль введений при реєстрації. Після цього потрібно натиснути кнопку ‘Sign in’ для відправлення запиту на сервер.

Sign up Sign in

Verify code form

* Code

Enter code

Verify code

Рисунок 3.4 – Сторінка другого кроку авторизації

Після успішного проходження першого етапу, користувач повинен ввести код, який прийшов йому на емейл вказаний при реєстрації. Після цього потрібно натиснути кнопку ‘Verify code’ для відправлення запиту на сервер.

Після успішного проходження цих кроків, користувач входить у додаток.

3.4 Персональний профіль

Після успішної автентифікації користувач перенаправляється на сторінку його профілю

Profile info		
First name :	Last name :	Email :
Tom	Holland	skystar.peach+1@gmail.com
Phone :	Created at :	
123123	2023-05-30	

Рисунок 3.5 – Сторінка першого профілю

Тут він може переглянути свої дані, вказані при реєстрації

3.5 Список користувачів

На цій сторінці відображається список усіх користувачів зареєстрованих у застосунку з їхніми основними даними, такими як ім'я та прізвище. Список представлений у виді таблиці. Під ім'ям користувача присутня кнопка 'Send a message', яка перенаправляє на сторінку чату з даним користувачем.

ProfileUsers

Sign out

Filters

First nameLast name

Users list

Gojo Satoru

Send a message

Capu Brown

Send a message

Jim Hellbert

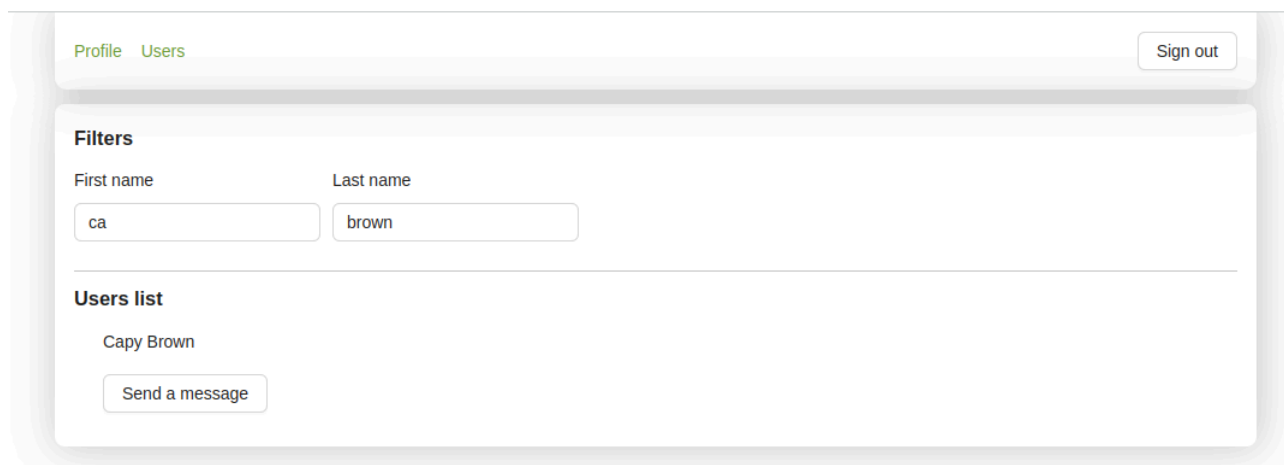
Send a message

John Wick

Send a message

Рисунок 3.6 – Сторінка першого кроку авторизації

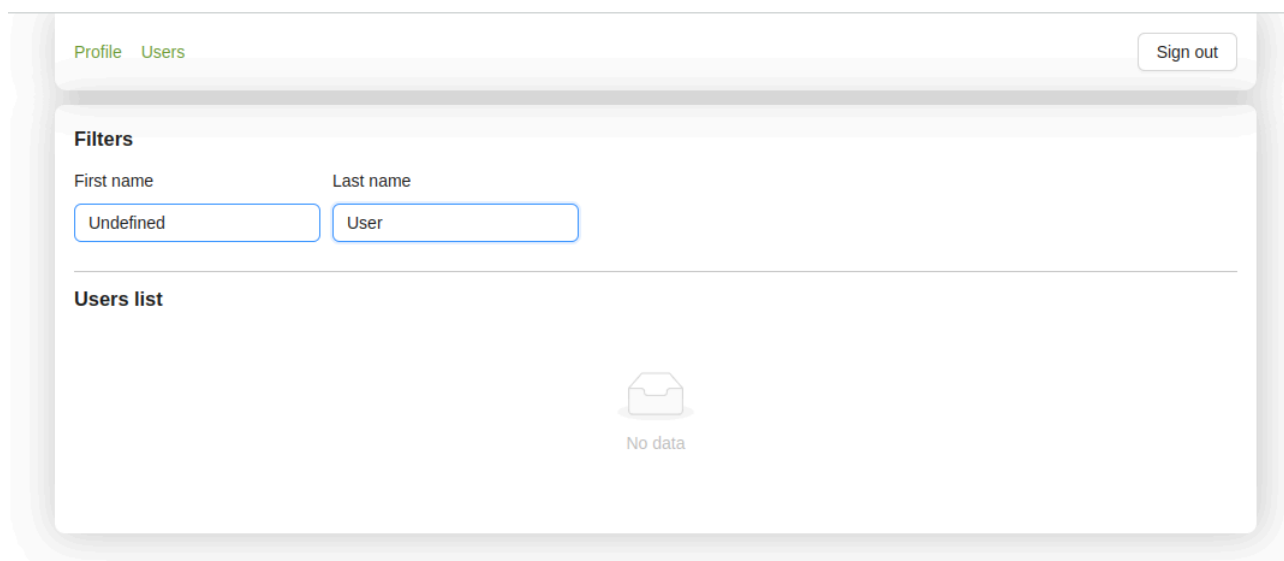
Над списком присутні фільтри для зручнішого пошуку користувачів за їх ім'ям та прізвищем.



The screenshot shows a web interface with a top navigation bar containing 'Profile' and 'Users' (the latter is highlighted in green), and a 'Sign out' button. Below the navigation bar is a 'Filters' section with two input fields: 'First name' containing 'ca' and 'Last name' containing 'brown'. Underneath the filters is a 'Users list' section. It displays a single user entry, 'Capy Brown', with a 'Send a message' button below it.

Рисунок 3.7 – Фільтр пошуку

При відсутності користувачів з таким прізвищем та ім'ям відображається пустий список.



This screenshot shows the same web interface as Figure 3.7, but with different search criteria. The 'First name' filter is set to 'Undefined' and the 'Last name' filter is set to 'User'. The 'Users list' section is now empty, displaying a folder icon and the text 'No data'.

Рисунок 3.8 – Пустий результат пошуку

3.6 Приватний чат

При натисканні кнопки ‘Send a message’ біля імені користувача в списку користувачів, нам відкривається сторінка чату з цим користувачем

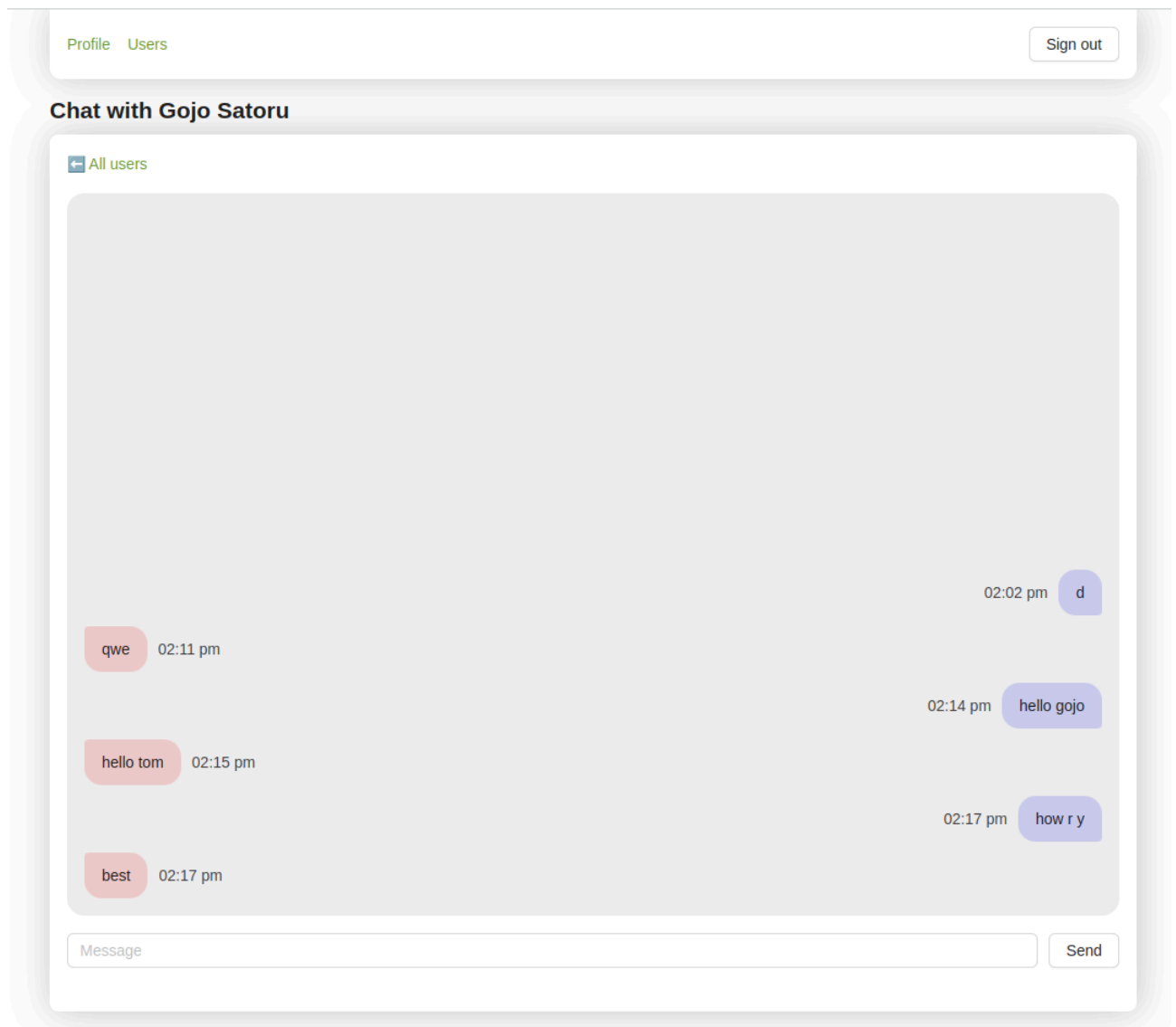


Рисунок 3.9 – Сторінка чату

На цій сторінці ми можемо обмінюватися повідомленнями з користувачем. Для цього потрібно вписати своє повідомлення у полі під чатом та натиснути кнопку ‘Send’.

Якщо ми раніше не обмінювалися повідомленнями з користувачем, тоді чат буде пустим.

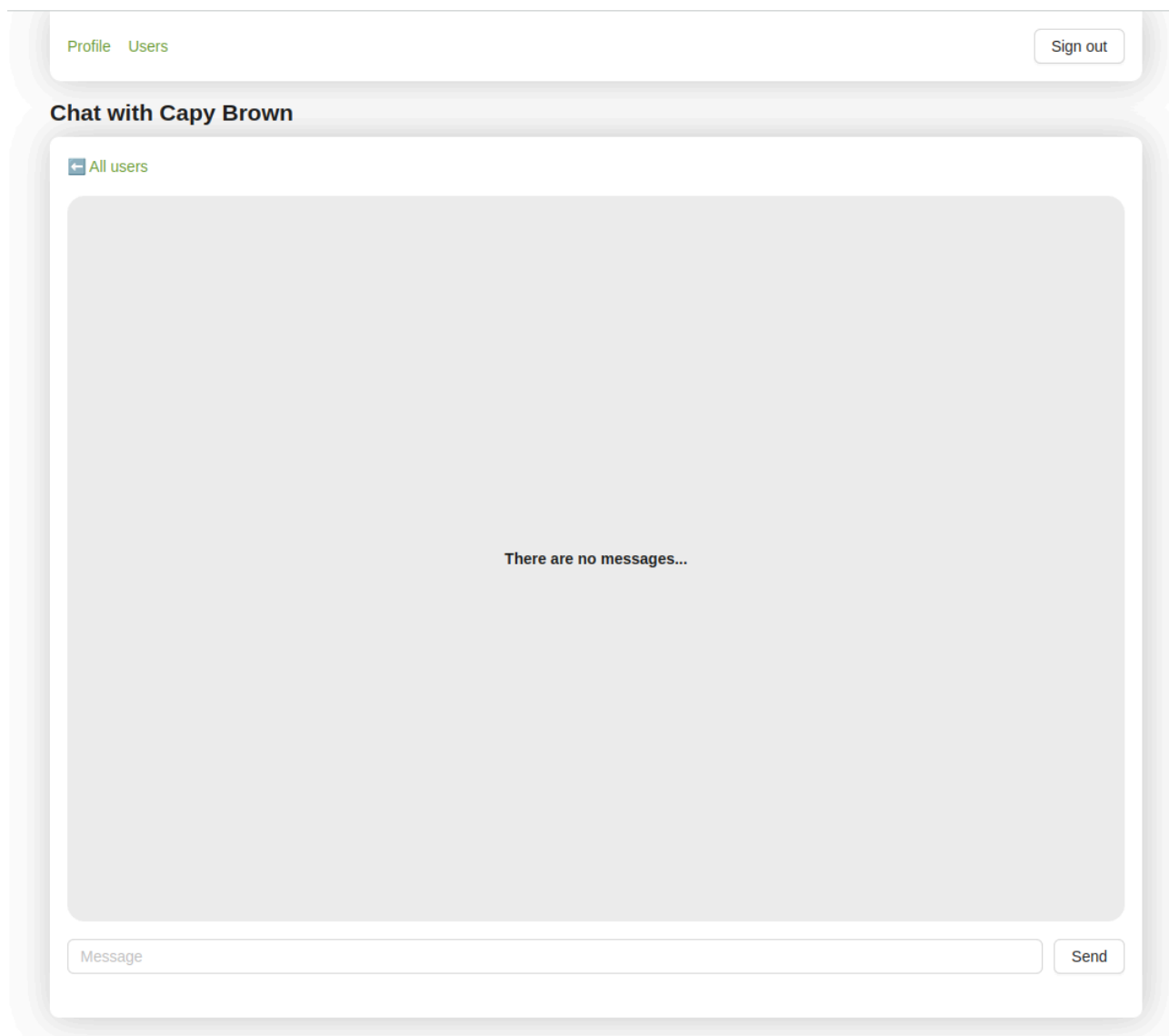


Рисунок 3.10 – Сторінка нового чату

Над чатом є кнопка 'All users', яке поверне назад до списку користувачів.

РОЗДІЛ 4. КОМПЛЕКСНИЙ ЗАХИСТ ЗАСТОСУНКУ

Застосунок Chat App був розроблений з урахуванням високого рівня безпеки для забезпечення конфіденційності та цілісності користувачів. У цьому розділі описуються різні рівні захисту, що використовуються в застосунку.

3.1 Двофакторна автентифікація

Двофакторна автентифікація є одним з найефективніших способів захисту від несанкціонованого доступу до облікових записів користувачів. У застосунку Chat App використовується двофакторна автентифікація для підтвердження ідентичності користувача перед наданням доступу до облікового запису. Крім звичайного пароля, користувачам потрібно підтвердити свою особу за допомогою додаткового фактора, такого як одноразовий код, відправлений на зареєстровану електронну пошту. Це значно підвищує рівень безпеки та унеможливорює несанкціонований доступ навіть у випадку, якщо пароль користувача був викрадений або скомпрометований.

3.2 Аксес та рефреш токен

У застосунку Chat App використовуються аксес токени та рефреш токени для забезпечення безпеки і захисту сесій користувачів. Після успішної автентифікації користувача, видається аксес токен, який використовується для авторизації. Рефреш токен, який також надається при автентифікації, використовується для отримання нового аксес токена після закінчення строку його дії. Цей механізм дозволяє уникнути передачі пароля через мережу при кожному запиті та забезпечує більшу безпеку для користувачів.

Також застосунок Chat App забезпечує безпечний доступ до свого сервера за допомогою авторизаційного токена. Кожен запит до бекенду повинен містити авторизаційний токен для підтвердження ідентифікації та авторизації користувача.

Після успішної автентифікації, при вході в систему або під час отримання аксес токена та рефреш токена (згадані у попередньому розділі), користувач отримує авторизаційний токен. Цей токен зберігається на стороні клієнта і використовується для кожного запиту до бекенду. При виконанні запиту до бекенду, клієнтська програма включає авторизаційний токен у заголовок запиту. Заголовок містить інформацію про тип токена та сам токен, який дозволяє бекенду перевірити ідентифікацію та авторизацію користувача перед обробкою запиту.

Бекендова частина застосунку перевіряє валідність авторизаційного токена, перевіряє права доступу користувача та забезпечує виконання запиту тільки для дійсних та авторизованих користувачів. Це дозволяє попередити несанкціонований доступ до ресурсів та функціональності бекенду.

Використання авторизаційного токена забезпечує додатковий рівень безпеки та контролю доступу до бекенду застосунку Chat App. Це дозволяє підтримувати високу безпеку даних та захистити конфіденційну інформацію користувачів.

3.3 Хешування паролів користувачів

З метою запобігання несанкціонованому доступу до облікових записів користувачів, паролі в застосунку Chat App зберігаються у захешованому вигляді. Хешування паролів - це процес перетворення пароля в нерозшифровувальний рядок (хеш), який неможливо відновити до початкового пароля. При авторизації користувача введений пароль хешується та порівнюється зі збереженим хешем пароля в базі даних. Це забезпечує високий рівень безпеки, оскільки навіть при можливості злому бази даних з хешами паролів, зловмисник не зможе отримати початковий пароль.

3.4 Шифрування повідомлень користувачів

Застосунок Chat App забезпечує захист повідомлень, що зберігаються в базі даних, шляхом їх зашифрування за допомогою алгоритму RSA.

RSA (Rivest-Shamir-Adleman) є криптографічним алгоритмом, який використовує пару ключів - публічний та приватний ключі. Публічний ключ використовується для шифрування повідомлень, тоді як приватний ключ використовується для розшифрування.

У застосунку Chat App, перед збереженням повідомлень у базі даних, вони шифруються за допомогою публічного ключа RSA. Це означає, що навіть якщо сторонні особи отримають доступ до бази даних, вони не зможуть прочитати зміст повідомлень без приватного ключа. Приватний ключ зберігається на сервері, що забезпечує тільки авторизованим користувачам можливість розшифрувати повідомлення після отримання їх з бази даних.

Цей підхід забезпечує високий рівень конфіденційності та безпеки для повідомлень у застосунку Chat App. Зашифровані повідомлення зберігаються в безпечному стані, знижуючи ризик несанкціонованого доступу до конфіденційної інформації користувачів.

3.5 Валідація тіла запиту

У застосунку Chat App на сервері реалізована валідація тіла запиту для забезпечення цілісності та безпеки даних.

Валідація тіла запиту - це процес перевірки вхідних даних, що надходять з клієнтської сторони, на відповідність певним правилам, формату та обмеженням. Вона дозволяє впевнитися, що дані, які передаються на бекенд, відповідають очікуваному формату та не містять шкідливих або некоректних значень. Валідація тіла реквесту реалізована шляхом перевірки його структури, типів даних, обов'язкових полів та діапазонів значень. Якщо дані не задовольняють встановлені вимоги, сервер повертає відповідну помилку та повідомлення про некоректність даних.

Валідація тіла реквесту на сервері є важливим етапом, який допомагає уникнути вразливостей, таких як некоректне введення даних, переповнення буфера або виконання зловмисного коду. Вона сприяє забезпеченню безпеки, надійності та стабільності застосунку, а також забезпечує коректну обробку запитів з клієнтської сторони.

Усі ці заходи безпеки разом створюють комплексний захист застосунку Chat App. Вони забезпечують високий рівень конфіденційності, цілісності та доступності для користувачів, запобігають несанкціонованому доступу до облікових записів і зберігають комунікацію між користувачами в безпечному стані.

РОЗДІЛ 5. ЕФЕКТИВНІСТЬ ТА ПЕРЕВАГИ ЗАСТОСУНКУ

5.1. Переваги застосунку "Chat App" в порівнянні з існуючими застосунками

Застосунок "Chat App" має декілька переваг порівняно з існуючими аналогами, що робить його привабливим для користувачів. Нижче наведені основні переваги цього застосунку:

1. Зручність використання: "Chat App" надає інтуїтивно зрозумілий та легкий у використанні інтерфейс, що дозволяє користувачам швидко освоїти його. Простота та зручність використання сприяють покращенню користувацького досвіду.
2. Безпека: Застосунок "Chat App" враховує важливі аспекти безпеки, такі як 2-факторна автентифікація та хешування паролів, що дозволяє забезпечити конфіденційність та цілісність даних користувачів. Ці заходи безпеки допомагають уникнути незаконного доступу до інформації та зберегти приватність користувачів.
3. Ефективна комунікація: "Chat App" забезпечує швидку та безперервну комунікацію між користувачами. Використання кодування повідомлень дозволяє зберігати конфіденційність.

5.2. Визначення потенційних можливостей для подальшого вдосконалення

Незважаючи на наявні переваги, застосунок "Chat App" також може бути покращений та розширений з метою забезпечення ще більшого задоволення користувачів. Нижче наведені декілька потенційних напрямків для подальшого вдосконалення:

1. Розширення функціональності. Можливість додати додаткові функції, такі як відео- або голосовий дзвінок, створення списків завдань або календарних подій, може покращити використання застосунку та розширити його потенціал.
2. Покращення безпеки. Вдосконалення заходів безпеки, таких як виявлення та запобігання атакам, автоматична блокування спаму або фішингових

повідомлень, може забезпечити ще вищий рівень безпеки для користувачів.

3. Виявлення та блокування шкідливих вмісту. Розробка алгоритмів виявлення шкідливого вмісту, такого як спам, образливі повідомлення або небажана реклама, допоможе забезпечити безпеку та комфортну взаємодію користувачів. Автоматичне блокування такого вмісту перед його досягненням до отримувача допоможе зменшити ризик негативного впливу на користувачів.
4. Система сповіщень про підозрілу активність. Розробка системи сповіщень, яка інформує користувачів про підозрілу активність на їх облікових записах, таку як невдалі спроби входу або зміна пароллю, допоможе покращити виявлення можливих кіб

Реалізація цих покращень може підвищити ефективність та користувацьке задоволення від застосунку "Chat App" та забезпечити його конкурентоспроможність на ринку веб-додатків для спілкування.

ВИСНОВОК

Було розглянуто проблеми безпеки веб-додатків, розроблених з використанням мови програмування JavaScript, а також був розроблений веб-додаток "Chat App" з урахуванням комплексного захисту та забезпечення безпеки користувачів.

Під час розробки "Chat App" було враховано важливі аспекти безпеки, включаючи двофакторну автентифікацію, хешування паролів, кодування повідомлень та використання токенів доступу. Ці заходи сприяють забезпеченню конфіденційності, цілісності та доступності інформації для користувачів.

Враховуючи постійний розвиток загроз безпеці, важливо пам'ятати, що захист веб-додатків є процесом, який вимагає постійного оновлення та вдосконалення. Тільки за допомогою комплексного підходу, поєднання різних заходів та постійного моніторингу можна забезпечити надійний рівень безпеки веб-додатків на базі JavaScript, що в свою чергу сприятиме захисту та впевненості користувачів у використанні цих додатків.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. JavaScript [Електронний ресурс]. – Режим доступу:
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>
2. TypeScript [Електронний ресурс]. – Режим доступу:
<https://www.typescriptlang.org/>
3. Node.js [Електронний ресурс]. – Режим доступу:
<https://nodejs.org/>
4. Express [Електронний ресурс]. – Режим доступу:
<https://expressjs.com/>
5. Sequelize [Електронний ресурс]. – Режим доступу:
<https://sequelize.org/>
6. PSQL [Електронний ресурс]. – Режим доступу:
<https://www.postgresql.org/>
7. Gmail [Електронний ресурс]. – Режим доступу:
<https://www.google.com/intl/uk/gmail/about/>
8. Next.js [Електронний ресурс]. – Режим доступу:
<https://nextjs.org/>
9. React [Електронний ресурс]. – Режим доступу:
<https://react.dev/>
10. JWT [Електронний ресурс]. – Режим доступу:
<https://jwt.io/>
11. Password hashing [Електронний ресурс]. – Режим доступу:
<https://auth0.com/blog/hashing-passwords-one-way-road-to-security/>
12. REST [Електронний ресурс]. – Режим доступу:
<https://uk.wikipedia.org/wiki/REST>
13. RESTful API [Електронний ресурс]. – Режим доступу:
<https://www.redhat.com/en/topics/api/what-is-a-rest-api>
14. RESTful API [Електронний ресурс]. – Режим доступу: <http://surl.li/gszpn>