Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РЕАЛІЗАЦІЯ ТЕЛЕГРАМ БОТУ ДЛЯ ВІДСТЕЖЕННЯ ІНФОРМАЦІЇ ПРО СЕАНС ФІЛЬМУ В ДЕКІЛЬКОХ КІНОТЕАТРАХ МІСТА ЛУЦЬКУ

IMPLEMENTATION OF A TELEGRAM BOT FOR TRACKING INFORMATION ABOUT A MOVIE SCREENING IN SEVERAL CINEMAS IN THE CITY OF LUTSK

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-31
Стасько Станіслав Іванович
_____
(підпис)

Керівник:
к.т.н., доцент
Суринович Олена Миколаївна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«4» 06 2023р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра інженерії програмного забезпечення
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«*02*» *02* 202*3* р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Стасько Станіслав Іванович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Реалізація телеграм боту для відстеження інформації про сеанс фільму в декількох кінотеатрах міста луцьку

Керівник роботи: д.т.н., доцент Суринович Олена Миколаївна
затверджені наказом закладу вищої освіти від «*23*» *грудня 2022* р. № *961-01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«8» червня 2023 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу, Мова програмування серверної частини – NodeJS.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, оцінка ергономічних та надійністних параметрів проектованої системи, функціонально-структурна схема роботи об'єкта проектування, розробка моделі бази даних об'єкта проектування.
5. Перелік графічного матеріалу: 23 рис; 1 схема; 1 діаграма.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Суринович О. М.*		
Специфікація вимог до розробленої системи	*Суринович О. М.*		
Розробка об'єкта проектування	*Суринович О. М.*		
Нормоконтроль	*Суринович О. М.*		
Гарант ОП	*Ліщина Н.М.*		
Показник запозичень тексту	10,1%		
Академічна доброчесність	*Яцук А.А.*		

7. Дата видачі завдання *«2» лютого 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*05.02.2023 р.*	вик
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2023 р.*	вик
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*10.03.2023 р.*	вик
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2023 р.*	вик
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2023 р.*	вик
6	*Тестування та налагодження об'єкта проектування*	*01.06.2023*	вик
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*08.06.2023 р.*	вик

Здобувач вищої освіти _____ Сисоєнко Є. І.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Суриновіч О.М.
(підпис) (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти: Стасько Станіслав Іванович
(ПІБ)

Тема кваліфікаційної роботи: *Реалізація телеграм боту для відстеження інформації про сеанс фільму в декількох кінотеатрах міста Луцьку*

(повна назва згідно наказу)

Коротка характеристика кваліфікаційної роботи та прийнятих рішень: *Робота складається з трьох розділів, протягом яких дипломантом здійснюється огляд представленої перед ним проблеми, пошук шляхів, її вирішення та опис їх реалізації. Для розробки було використано наступні програмні засоби: середовище виконання: Node js, а також бібліотеки puppeteer для парсингу даних із сайту та node-telegram-bot-api для взаємодії із API телеграму.*

Самостійні розробки і пропозиції автора: *Кваліфікаційна робота бакалавра присвячена розробці програмного продукту, який буде призначений для онлайн моніторингу сеансів в кінотеатрах м.Луцьку за допомогою телеграм боту.*

Практичне значення роботи: *Практичне значення даної роботи полягає в розробці та реалізації телеграм боту для зручного відстеження інформації про сеанси фільмів у кінотеатрах міста Луцька. Бот надає користувачам можливість швидко отримати актуальну інформацію про фільми, години сеансів та кінотеатри, що дозволяє зекономити час і забезпечити зручну організацію дозвілля.*

Недоліки: *Істотних недоліків в роботі не виявлено.*

Загальний висновок: *У кваліфікаційній роботі бакалавра було спроектовано та створено програму для забезпечення пошуку сеансу в кінотеатрах м.Луцьк за допомогю використання телеграм-чат боту. Робота є результатом самостійного дослідження. Істотних недоліків не виявлено. Кваліфікаційна робота здобувача освіти Стасько Станіслава рекомендується до захисту екзаменаційній комісії та заслуговує оцінки «відмінно».*

Рецензент _____ (к.т.н., доц., доц. каф. ЕІ Цукчинук Ю.А.)
(підпис) (науковий ступінь, вчене звання, посада, ПІБ)

«2» червня 2023 р.

АНОТАЦІЯ

Стасько С. І. Розробка та дослідження Телеграм боту-моніторингу інформації про сеанси фільмів в кінотеатрах м.Луцька. Рукопис.

Кваліфікаційна робота за спеціальністю 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2023. 42 с.

Кваліфікаційна робота присвячена розробці телеграм боту, який займатиметься моніторингом інформації про сеанси фільму в кінотеатрах м. Луцька на платформі NodeJS.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі було проаналізовано аналогічні чат-боти, сформовано конкретне завдання до розроблюваного програмного продукту.

В другому розділі були визначені необхідні вимоги до програмного забезпечення, також було обрано алгоритми, методи та засоби для розробки програмного забезпечення.

Третій розділ присвячено розробці чат-боту з використанням платформи NodeJS та необхідних бібліотек. Розглянуто реалізацію тестування та розробки, документації по використанню чат-бота, також було проведено розробку інсталяційного пакета та аналіз захисту інформації користувача.

У висновках роботи підводяться загальні підсумки досягнених результатів. Також додається список використаних джерел та додатки, що доповнюють зміст роботи.

Ключові слова: telegram, telegram api, telegram бот, моніторинг інформації, операційна система, Windows, JavaScript.

ANNOTATION

Stasko S. I. Development and research of a Telegram bot monitoring information about movie screenings in cinemas in Lutsk. Manuscript.

Qualification work on specialty 121 "Software engineering". Lutsk National Technical University. Lutsk, 2023. 42 p.

The qualification work is devoted to the development of a telegram bot that will monitor information about movie screenings in cinemas in Lutsk on the NodeJS platform.

The bachelor's qualification work consists of an introduction, three sections, conclusions, a list of used sources and appendices.

In the first section, similar chatbots were analyzed, and a specific task for the software product under development was formed.

In the second section, the necessary software requirements were defined, and algorithms, methods and tools for software development were also selected.

The third section is devoted to the development of a chatbot using the NodeJS platform and the necessary libraries. The implementation of testing and development, documentation on the use of the chatbot was considered, the development of the installation package and the analysis of the protection of user information were also carried out.

In the conclusions of the work, general summaries of the achieved results are summarized. A list of used sources and appendices supplementing the content of the work is also attached.

Keywords: telegram, telegram api, telegram bot, information monitoring, operating system, Windows, JavaScript.

ЗМІСТ

ВСТУП

Актуальність теми. В наш час є дуже багато телеграм ботів, які надають змогу завантажити фільм для перегляду по посиланню, завантажити фільму, тощо. Також дають змогу напряму у телеграмі переглянути завантажений в месенджері раніше фільм, серіал, або, переглянути інформацію про той чи інший фільм, серіал, тощо.

Розробка нового телеграм боту на Node.JS дасть змогу не просто переглянути інформацію про фільм, серіал, чи, наприклад, завантажити його, вона дасть змогу переглянути всю необхідну інформацію саме про сеанс обраного раніше фільму, мультфільму тощо.

Також, є необхідність забезпечити користувача лише необхідною його інформацією, і в цьому допоможуть дві бібліотеки: Puppeteer та Cheerio. Вони дають змогу отримувати необхідну інформацію з сайту для розробника, і передавати її користувачеві.

Також існує потреба для подальшого оновлення методу отримання інформації, адже прогрес не стоїть на місті і сайти доволі часто оновлюються, змінюється їхня структура, код, та інше. Саме тому слід слідкувати за оновленнями на сайті, це і є одним із мінусів отримання інформації через сайт, адже Puppeteer та Cheerio використовують так званий «парсинг», який збирає інформацію із сайту.

Cheerio може бути використаний для отримання даних з веб-сторінки, зокрема з отриманням тексту, HTML-коду, атрибутів, зображень та інших елементів. Цей модуль працює на основі DOM-структури документа, що дозволяє легко отримувати доступ до елементів та їх властивостей.

Основний принцип роботи Cheerio полягає в тому, що він приймає на вхід HTML-код сторінки, створює віртуальну DOM-структуру та дозволяє виконувати різноманітні дії з елементами на цій структурі. Cheerio надає доступ до методів, що дозволяють здійснювати пошук елементів за селекторами, отримувати їх властивості та виконувати з ними різноманітні дії.

Об’єктом дослідження є програмний продукт, який за допомогою месенджеру Телеграм дозволить отримувати необхідну інформацію про афіши в кінотеатрах м.Луцьку. Дослідження включає розробку програмного забезпечення, використання сучасних технологій та бібліотек для досягнення найкращої продуктивності та забезпечення користувачам максимального комфорту під час використання додатку.

Предметом дослідження є реалізація програмного телеграм чат-боту, використовуючи Node.JS, а також необхідні бібліотеки, зокрема: Node Telegram Bot Api, Puppeteer.

Ключовими аспектами дослідження є розробка, тестування програмного забезпечення.

Мета дослідження – розробка телеграм-чат боту для інформації щодо афіш в місті Луцьк. Основним завданням є розробка простого, сучасного, зрозумілого та швидкого в використанні боту, який буде надавати всю необхідну інформацію користувачеві, отримуючи її із сайту кінотеатрів.

Завдання дослідження:

- аналіз предметної області дослідження;
- розгляд та обґрунтування основних програмних засобів проектування;
- розробка телеграм-боту;
- тестування програмного продукту;
- аналіз інформаційної безпеки.

Новизною роботи є отримання інформації саме про афіши кінотеатрів, а не про сам фільм чи інше, на відміну від інших телеграм-ботів.

Розробка цього програмного забезпечення дає змогу вдосконалити знання в роботі із API, а також надає змогу глибше зрозуміти, як влаштовані веб сайти.

Практична цінність полягає у використанні продукту працівниками кінотеатрів для рекламування власного закладу, та їх відвідувачами для швидкого та зручного пошуку сенсів, що сприятиме отриманню позитивних емоцій.

РОЗДІЛ 1
АНАЛІЗ ТЕЛЕГРАМ БОТІВ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Чат-бот – це окремий обліковий запис у Telegram, який самостійно відповідає на повідомлення користувачів [1]. Чат-боти можуть бути запрограмовані на виконання різних завдань, наприклад, відповідати на запитання користувачів, надавати інформацію про товари та послуги, виконувати розрахунки або планувати зустрічі.

Використання ботів у Telegram наразі набуває все більшої популярності. У Telegram є низка ботів, які так чи інакше пов'язані з фільмами.

Було проведено аналіз декількох телеграм ботів, які містять в собі функцію для пошуку фільмів а також містять посилання для перегляду обраного фільму/серіалу. Серед них є наступні телеграм боти:

The Movie Bot (@MovieDatabaseBot);

Movies Bot(@Movies_teleBot).

Кінотеатр «Жовтень» (@ZhovtenBot).

1.1.1 The Movie Bot

Бот надає змогу шукати фільми та телешоу, знаходити популярні речі в кінотеатрах або на Netflix.

Також бот може надати інформацію про фільми, телешоу та людей, наприклад: афіша фільму, сюжет, актори тощо.

Для роботи з ботом необхідно надіслати назву для пошуку або скористатися меню, щоб знайти популярні речі, фільми в кінотеатрах тощо.

Також є можливість підключити свій обліковий запис TMDb, щоб синхронізувати список перегляду та вибране, а також оцінювати фільми та телешоу, не виходячи з Telegram.

Бот також містить зворотній зв’язок із підтримкою, користувач може напряму задати запитання тех.підтримці, та отримати відповідь на нього, також відповіді на деякі питання користувачів, що допоможуть в використанні бота.

Перевагами цього боту є:

– зручне меню для використання;
– необхідний функціонал;
– недоліки:
– мова змінюється не скрізь;
– переклад може бути не повний, не точний.

Також бот має навігаційне меню, що показано на рисунку 1.1

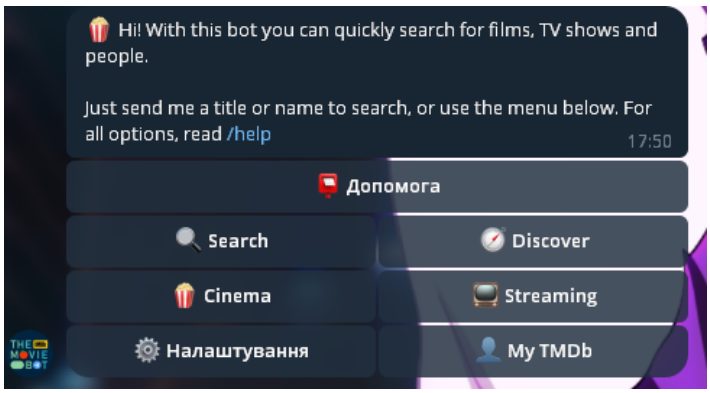

Рисунок 1.1 – Навігаційне меню бота «The Movie Bot»

1.1.2 Movies Bot

Movies Bot – також зарубіжний телеграм бот, який дає можливість отримати посилання на безкоштовний перегляд фільму/серіалу. Список фільмів та серіалів береться із бази даних бота, знайти фільм можливо використовуючи алфавітний порядок із контекстного меню. Можливості знайти серіал або фільм використовуючи ключові слова або повну назву не має змоги, що є одним із головних мінусів цього чат боту.

Із плюсів: зручне меню для використання.

Із мінусів:

– немає можливості змінити мову;

– немає можливості обрати фільм/серіал по назві використовуючи пошук;
– список фільмів та серіалів не є досить великим.

Контекстне меню бота показано на рисунку 1.2

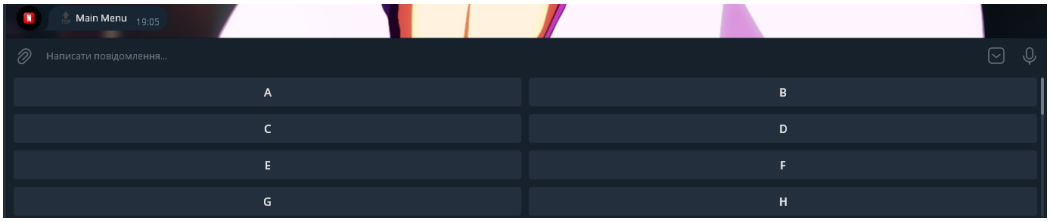

Рисунок 1.2 – Контекстне меню Movies Bot

1.1.3 Кінотеатр «Жовтень»

Кінотеатр «Жовтень» – кінотеатр у місті Києві, який надає змогу через телеграм забронювати онлайн квитки, дивитись інформацію про сеанси а також має взаємодію із особистим кабінетом.

Із плюсів:

– зручне меню для використання;

– купу необхідних можливостей для користувача (Бронювання, моніторинг сеансів, особистий кабінет).

Із мінусів: доступна лише одна мова.

Бот також містить в собі контекстне меню яке показано на рисунку 1.3.

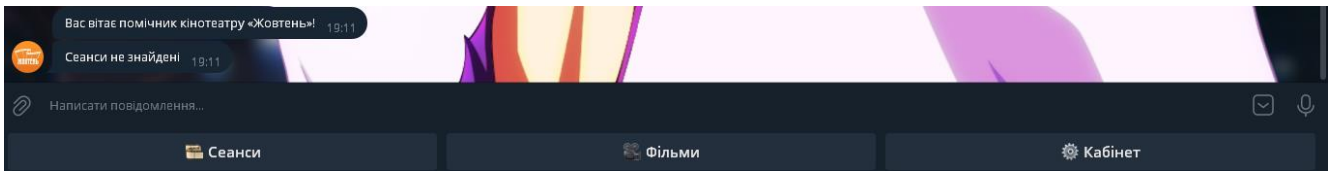

Рисунок 1.3 - Контекстне меню чат боту «Кінотеатр Жовтень»

Для зручності любителів переглянути фільм в кінотеатрі, було прийнято рішення створити телеграм бота, в наш час це актуально, адже досить багато людей використовує телеграм, також користувачу не доведеться моніторити декілька сайтів водночас, а просто написати назву фільму ботові, а вже сам

телеграм бот отримуватиме інформацію щодо сеансів в відомих кінотеатрах міста Луцьк, та відображати необхідну інформацію за вказаним користувачем фільмом:

– найменування кінотеатру;

– дата сеансів;

– час сеансів;

– вартість білетів.

Майже всі боти є однотипними, тому складно провести доволі глибокий аналіз телеграм ботів.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Ціль даного телеграм боту – забезпечити зручний онлайн-пошук доступних сеансів в кінотеатрах м. Луцька, використовуючи лише назву фільму/мультфільму, також надання всієї необхідної інформації про сеанс (Дата, назва кінотеатру, ціна за квиток, ціна білетів).

Функціональні вимоги:

– досягнення швидкої відповіді від бота користувачу;

– зручний інтерфейс використання (навігаційне меню);

– представлення всієї необхідної інформацію користувачу про обраний сеанс в обраному кінотеатрі;

– інформація про те, як використовувати бота.

Технічні вимоги:

– використовувати сучасні технології використання;

– забезпечити підтримку різних типів пристроїв;

– забезпечити можливість зберігання даних.

Процесні вимоги:

– забезпечити безперебійну роботу системи 24/7;

– забезпечити високу рівень безпеки та конфіденційності даних.

Також, слід звернути увагу на вимоги до самих чат-ботів:

– повинен мати зрозумілий і простий інтерфейс. Користувачі мають легко зрозуміти, як з ним спілкуватися і які можливості у нього та них є.

– пає бути швидким і відповідальним. Користувачі очікують швидкої реакції на свої запитання та запити, тому бот повинен бути готовий відповісти на будь-яке запитання в максимально короткий термін.

– має бути надійним. Бот повинен правильно обробляти інформацію, не викликати помилок і завжди надавати коректну інформацію користувачам.

– має мати розуміння мови. Хороший чат-бот повинен мати розуміння мови, в тому числі розуміти граматику та синтаксис запитів.

Після обраного сеансу також бот відправить посилання для бронювання квитків.

Висновки до розділу 1

Здійснено аналіз предметної області.

Розглянуто декілька аналогічних чат ботів, враховано їхні переваги та недоліки.

Обґрунтовано функціональні та технічні вимоги до розроблюваного продукту.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Для того, щоб проаналізувати та визначити вимоги до програмного забезпечення, яке буде розроблятися, необхідно зробити наступне:

– визначити цільові області та завдання програмного забезпечення;

– проаналізувати вимоги до програмного забезпечення;

– проаналізувати потреби та вимоги користувачів та інших зацікавлених сторін;

– визначити функціональні і нефункціональні вимоги;

– визначення вимог до програмного забезпечення для системи;

– визначення системних обмежень, таких як апаратне та програмне забезпечення;

– створення варіантів використання, діаграм послідовності та інших необхідних uml-моделей, які описують поведінку програмного забезпечення та його взаємодію з користувачами;

– створювати документи у стислому та лаконічному форматі.

2.1.1 Функціональні та нефункціональні вимоги

Функціональні вимоги визначають, які функції або операції має виконувати система, програмний продукт або процес. Вони є ключовим елементом при розробці будь-якого продукту або системи, оскільки визначають його основні можливості та функції, які повинні бути реалізовані.

Функціональні вимоги:

– зручний інтерфейс використання (навігаційне меню);

– досягнення швидкої відповіді від бота користувачу;

– представлення всієї необхідної інформацію користувачу про обраний сеанс в обраному кінотеатрі;

– інформація про те, як використовувати бота.

Нефункціональні вимоги є не менш важливим елементом вимог до розроблюваної програмної системи. Нефункціональні вимоги описують властивості, якості та характеристики, які має мати програмна система, щоб задовольнити потреби користувачів та відповідати стандартам та вимогам. Їх важливість полягає у тому, що вони визначають, як система повинна функціонувати з точки зору продуктивності, ефективності, безпеки, надійності, масштабованості та інших аспектів.

Нефункціональні вимоги:

– надійність: боти повинні витримувати велику кількість запитів і завжди бути доступними для користувачів;

– швидкість: боти повинні швидко обробляти запити користувачів і відповідати в режимі реального часу;

– відповідність інтерфейсу: боти повинні мати інтуїтивно зрозумілий і привабливий інтерфейс, який відповідає критеріям дизайну telegram;

– масштабованість: бот повинен бути масштабованим, щоб обробляти велику кількість запитів і користувачів;

– інтеграція з іншими системами: боти можуть інтегруватися з системами кінотеатрів, щоб отримувати актуальну інформацію про сеанси, квитки та наявність вільних місць. інтеграція має бути забезпечена безперервною та надійною передачею даних;

– Доступність: боти мають бути доступними для користувачів з різних пристроїв і платформ, таких як смартфони, планшети та комп'ютери. Вони також повинні підтримувати різні мови і бути доступними в різних регіонах.

2.1.2 Моделі елементів програмного додатку у нотації UML

Моделі дозволяють графічно спроектувати логіку системи. Для побудови моделей необхідно визначити основні компоненти системи та їх взаємозв'язки, а також функції та операції, що виконуються кожним компонентом. Також

необхідно визначити інтерфейси та структури даних, що використовуються для обміну інформацією між компонентами.

Сперш розглянуто діаграму прецедентів логіки взаємодії між клієнтом та чат-ботом (див. рис. 2.1).

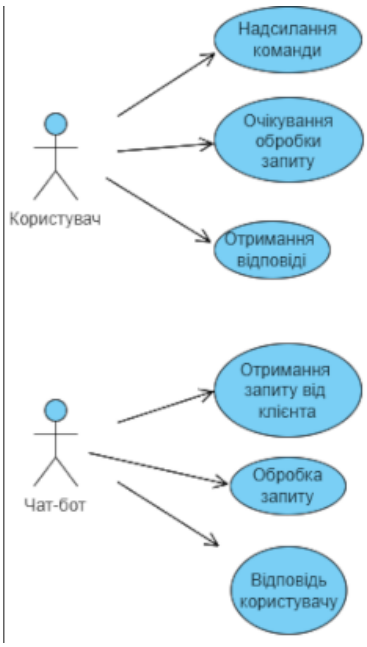

Рисунок 2.1 – Діаграма взаємодії користувача із чат ботом

Дана діаграма дає зрозуміти, як між собою відбувається взаємодія клієнта та чат боту, користувач надсилає запит та очікує на відповідь, в той час бот отримує запит користувача, обробляє та надсилає відповідь користувачу.

На наступній представленій діаграмі чітко представлено послідовність ітерацій, які виконує користувач в системі від початку чатування до отримання всієї необхідної інформації.

Схему послідовності користування чат ботом зображено на рисунку 2.2.

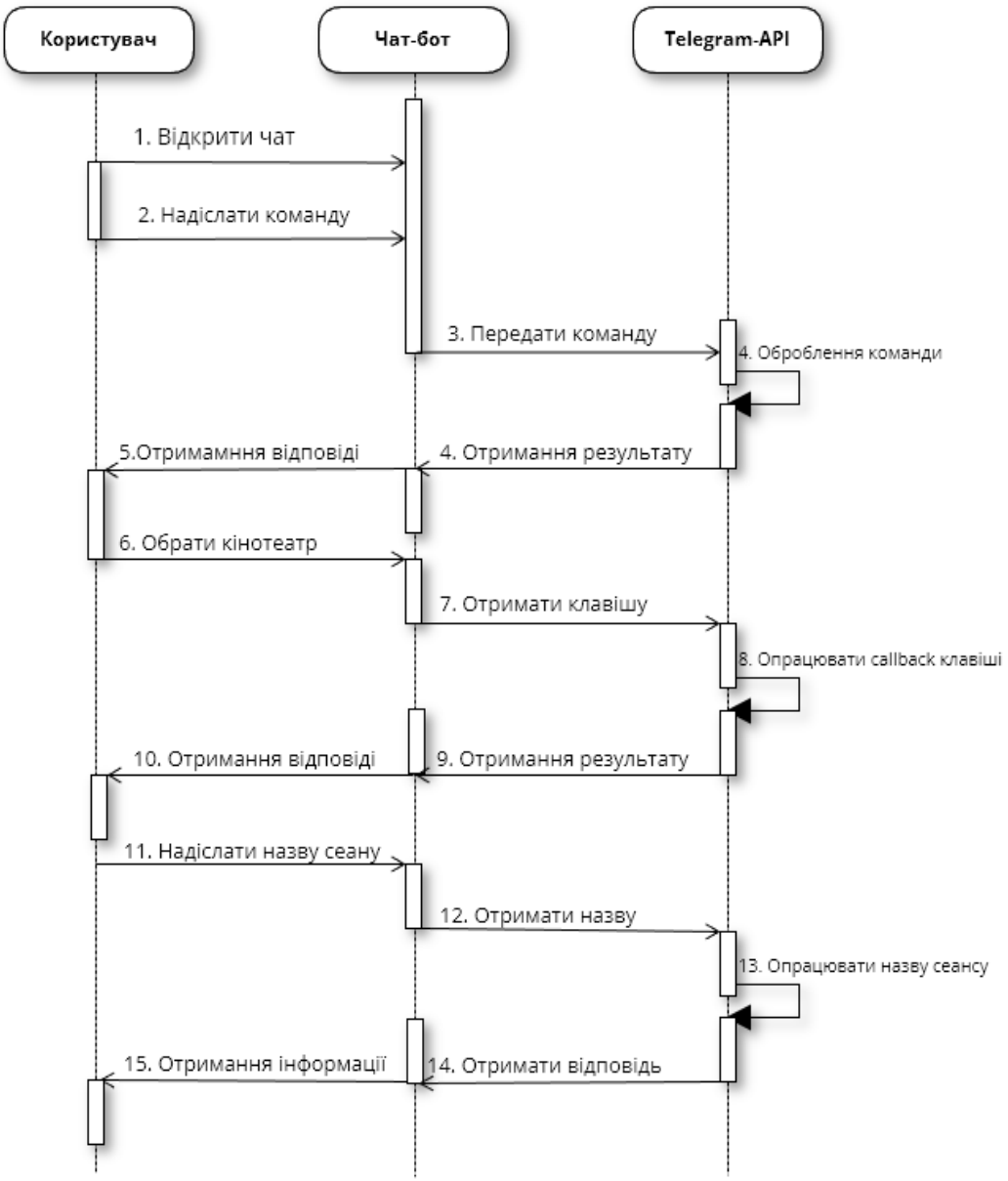

Рисунок 2.2 – Схема послідовності користування чат ботом

2.1.3 Створення UX/UI-дизайну

Для майбутнього дизайну чат-боту було проаналізовано інші чат-боти, та на основі аналізу було складено схему інтерфейсу.

В чат-боті має бути реалізований список команд, який буде відображати кожну команду чат боту та її функціонал. Також, в необхідний момент необхідно додати клавіші для більш зручного інтерфейсу користувача. Головна команда має включати в себе інструкцію по використанню чат-бота.

Клавіші будуть показуватись користувачеві в момент використання команди пошуку сеансів, представлено дві клавіши на вибір із двома кінотеатрами.

Після виведення інформації про сеанси буде взято у сторінки. На кожній сторінці буде не більше 5 сеансів, також повинні бути клавіші гортання сторінок (вперед/назад). На рисунку 2.3 показано схему інтерфейсу користувача.

Рисунок 2.3 - Інтерфейс користувача.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Під час проектування розробки програмного забезпечення було сформоване чітке представлення стеку технологій, які будуть використовуватись в розробці програмного забезпечення.

Також, під час обирання технологій було дотримано наступних критерій:

– актуальність технології;

– популярність технології;

– кількість функціональних можливостей;

– власні навички у обраній технології;

– наявність документацій до технології.

На основі критеріїв, описаних вище, було обрано технології для розробки.

Ця розробка є веб-орієнтованою, а тому, випливає висновок: необхідність у використанні веб-орієнтованої мови програмування.

Телеграм бот із афішами повинен містити в собі весь необхідний контент, щоб користувач дізнавався лише необхідну йому інформацію. Одним із варіантів для розробника є «парсинг» сайту [2]. Парсингом називається збір інформації з різних чужих сайтів. Парсити – означає збирати та аналізувати дані із відповідних сайтів за допомогою спеціальних програм. Суть даного процесу можна описати наступним чином: 1) бот заходить на сторінку ресурсу; 2) розбирає HTML код на окремі частини; 3) виокремлює необхідні дані; 4) зберігає їх у своїй базі. Роботів Google також називають свого роду парсерами. Саме тому захистити сайт від шпигунів є складно, адже одночасно можна обмежити доступ для пошуковиків.

2.2.1 Visual Studio Code

«Visual Studio Code (VS Code)– це безкоштовний та відкритий редактор коду, розроблений компанією Microsoft» [3].

На вибір даного редактору коду сплинули наступні переваги:

– розширюваність: vs code має власну систему із розширеннями, яка містить в собі безліч розширень, які допоможуть облегшити розробку програмного забезпечення;

– підсвічування синтаксису та автозавершення: vs code полегшує роботу з кодом, він має змогу підсвічування синтаксису для багатьох мов програмування, а також містить в собі функцію автозавершення коду, це допоможе розробнику швидше писати код;

– інтеграція з git'ом: vs code має вбудовану підтримку системи контролю версій git, що дозволяє зручно працювати з репозиторіями git безпосередньо з редактора;

– налагодження: vs code надає можливість налагоджувати код безпосередньо в редакторі. Можна встановлювати точки зупинки, відображати значення змінних і переходити по коду, щоб допомогти виправити помилки;

– простота використання: vs code має інтуїтивно зрозумілий і простий у використанні інтерфейс, придатний для початківців і повнофункціональний для досвідчених розробників;

– самодостатність: його можна використовувати як базовий інструмент розробки з інструментами для редагування та перегляду файлів, вбудованим терміналом і можливістю встановлювати розширення та налаштовувати їх відповідно до потреб.

2.2.2 NodeJS

Однією із популярних платформ для розробки телеграм-боту є Node.JS [4, 5, 6].

«Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, 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 їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.» [4]

Причиною вибору даного середовища стали наступні переваги:

– асинхроність: node.js побудований на подієво-керованих і неблокуючих операціях вводу/виводу, що дозволяє йому ефективно обробляти багато запитів одночасно;

– масштабованість: node.js структурований таким чином, що дозволяє легко горизонтально розширювати додаток, тобто додавати сервери для обробки більших навантажень;

– пакети для розробки: в node.js є низка пакетів та бібліотек, які полегшують розробку telegram-ботів. наприклад, є спеціальні бібліотеки telegraf та node-telegram-bot-api, які надають зручний інтерфейс для роботи з telegram api.

2.2.3 Node Telegram Bot API

Node Telegram Bot API– це бібліотека для розробки телеграм-ботів з використанням мови програмування Node.js [9, 10]. Вона надає зручний і простий спосіб взаємодії з Telegram API, ця бібліотека проста в використанні і має багато корисних функцій для розробки ботів.

Також node-telegram-bot-api має наступні переваги:

– повний доступ до функції телеграм: модуль надає повний доступ до всіх функцій telegram api;

– обробка команд і повідомлень node-telegram-bot-api надає можливість легко обробляти команди і повідомлення, отримані ботом;

– обробка подій: цей модуль може обробляти різні події, які відбуваються в telegram;

Так як бот має отримувати інформацію із сайтів, і після чого оброблювати її та передавати ботові, було прийнято рішення використати бібліотеку puppeteer.

2.2.4 Puppeteer

Puppeteer – одна з бібліотек NodeJS, яка дозволяє взаємодіяти з веб браузером та веб сторінками [11]. Також, Puppeteer дозволяє отримувати дані з веб-сторінок, шукаючи і витягуючи інформацію з HTML-елементів. Не мало важливим є те, що ця бібліотека також надає змогу відкривати нові сторінки, натискати клавіші на сайті що є важливим для розробки.

Переваги puppeteer:

– puppeteer browser control дає повний контроль над веб-браузером chrome або chromium;

– автоматизація тестування puppeteer - популярний інструмент для створення автоматизованих тестів для веб-додатків. він може імітувати різну поведінку користувачів, перевіряти, чи поводяться сторінки так, як очікується, і перевіряти, чи правильно працює додаток;

– витяг даних: puppeteer може витягувати дані з веб-сторінок шляхом вилучення та вилучення інформації з елементів html. дані можуть бути автоматично зібрані, проаналізовані, оброблені або збережені в базі даних;

– знімок екрана puppeteer дозволяє зробити знімок екрана веб-сторінки або частини веб-сторінки. Це корисно для створення скріншотів вашого веб-сайту, аналізу та налагодження розташування елементів;

– відображення на стороні сервера: puppeteer може відображати веб-сторінки на стороні сервера за допомогою chrome або chromium. Це особливо корисно, коли вам потрібно отримати html-код, запустити javascript або зробити скріншоти на сервері.

2.2.5 Telegram та Bot Father

Платформою, до якої було розроблено бота було обрано месенджер Telegram [12].

Telegram – один із найпопулярніших месенджерів із великою аудиторією, який був розроблений в 2013 році компанією Telegram Messenger LLP, засновник якої є Павло Дуров.

Телеграм має наступні переваги:

– легкий доступ: telegram надає зручне api для розробників, що спрощує розробку телеграм боту;

– багатий функціонал: telegram містить в собі розширені функції, які надають купу можливостей для реалізацій функцій чат-боту;

– зашифрована безпека telegram використовує протокол шифрування mtproto, який забезпечує високий рівень безпеки при надсиланні повідомлень;

це означає, що чат-боти можуть гарантувати конфіденційність і безпеку свого спілкування з користувачами;

– інтеграція з іншими сервісами: telegram пропонує інтеграцію з іншими сервісами та api;

Також телеграм має власного бота, який дозволяє створювати своїх ботів власноруч, щоб надалі розроблювати функціонал до нього, цей бот найменує себе як Bot Fhater.

Bot Father – це офіційний бот-керівник для створення та керування ботами в месенджері Telegram. З його допомогою можна створити нового бота, налаштувати його параметри, додати команди та відповіді, а також отримати токен для підключення бота до месенджера. BotFather є потужним інструментом для створення ботів в Telegram та дозволяє розширити можливості цього месенджера. Після створення боту ви отримаєте токен для авторизації, через який програмна частина буде взаємодіяти із ботом. Для розробки боту на мові JavaScript з використанням Node.JS застосовується бібліотека Node Telegram Bot Api [9, 10].

Висновки до розділу 2

Здійснено аналіз вимог до розроблюваного програмного забезпечення. Також означено функціональні та не функціональні вимоги до програмного продукту.

Виконано вибір засобів, методів та алгоритмів до поставленого завдання.

Обґрунтовано найоптимальніші програмні засоби для реалізації розроблюваного продукту.

Означено найпопулярніші месенджери та обрано найпопулярніший з них для подальшої роботи.

РОЗДІЛ 3
РОЗРОБКА ТЕЛЕГРАМ ЧАТ-БОТУ ДЛЯ ВІДСТЕЖЕННЯ ІНФОРМАЦІЇ ПРО СЕАНС ФІЛЬМУ В ДЕКІЛЬКОХ КІНОТЕАТРАХ МІСТА ЛУЦЬКУ З ВИКОРИСТАННЯМ NODEJS

3.1 Практична реалізація об’єкта проектування

В першу чергу, для створення проекту необхідно створити проект. Для цього, спочатку, необхідно встановити NodeJS, також, разом із ним буде здійсненна інсталяція Node Packet Manager (npm), для подальшого встановлення необхідних бібліотек, а саме puppeteer, node-telegram-bot-api.

Після чого, в теці з проектом необхідно клікнути пкм, та відкрити її через Visual Studio Code. Та за допомогою вже вбудованого терміналу ініціалізуємо проект, після чого встановимо необхідні бібліотеки за допомогою команд, які наведено у лістингу 3.1.

Лістинг 3.1 – Створення нового проекту на NodeJS.

```
npm init
npm i puppeteer
npm i node-telegram-bot-api
```

кінець лістингу 3.1

Під час виконання npm init буде задано декілька запитань:

- Package name – назва проекту;
- Version – номер версії проекту;
- Description – опис;
- Git repository – посилання на репозиторій у гітхабі;
- Keywords – ключові слова;
- Author – автор;
- License – ліцензія.

Після того, як вказано всі поля, буде поле з підтвердженням введеної вище інформації. Також, є можливість пропустити ці всі запитання додавши -y після команди npm init. Ця опція прийме значення по замовчуванню та без всіляких запитань ініціалізує проект.

Для початку, було розроблено масив, який, в подальшому, буде містити в собі дані з сайту, який буде парситись. Для самого парсингу було розроблено функції, які викликаються в основному коді через стороні файли. Наприклад, проведемо огляд асинхроної функції getSessionFromPremierCity, вона виконує «парсинг» сайту Premier City, та, після чого, записує дані у масив. Код наведено у лістингу 3.2.

Лістинг 3.2 – Функція getSessionFromPremierCity

```
const getSessionFromPremierCity = async () => {
   const browser = await puppeteer.launch();
   const page = await browser.newPage();

   await page.goto('https://premiercity.com.ua/rozklad/');

   const tableElement = await page.$('body > div > main > main > section > div > table');

   if (!tableElement) {
      console.log('Не можу знайти таблицю!');
      return;
   } else {
      console.log('Знайшов таблицю із фільмами!');
   }

   const data = await page.evaluate(table => {
      const rows = Array.from(table.querySelectorAll('tr'));

      let currentDate = null;

      const result = rows.reduce((data, tr) => {
         const sessionPremier = {};

         if (tr.classList.contains('schedule-table__row_head')) {
            currentDate = tr.querySelector('td:nth-child(1)').textContent.trim();

            return data;
         }

         sessionPremier.date = currentDate
```

```
            sessionPremier.time = tr.querySelector('td:nth-child(1)').textContent.trim();
            sessionPremier.movieName = tr.querySelector('td:nth-child(2)').textContent.trim();
            sessionPremier.hall = tr.querySelector('td:nth-child(3)').textContent.trim();
            sessionPremier.format = tr.querySelector('td:nth-child(4)').textContent.trim();

            data.push(sessionPremier);

            return data;
        }, []);


        return result;
    }, tableElement)

    await browser.close();

    return data;
}
```

кінець лістнгу 3.2

Тут можна побачити асинхрону функцію, яка отримує дані із веб-сторінки за допомогою бібліотеки puppeteer. Для початку було додано екземпляр браузера, після чого значення було задано змінній browser, після чого, відкривається нова сторінка, значення якої передається в змінну page. Після чого, йде перехід за вказаним посиланням, і, так як сайт містить дані саме в таблиці, було розроблено змінну tableElement, в яку було передано перший елемент table за допомогою вказаного css селектору.

Далі було зроблено перевірку, чи є tableElement – таблицею. В разі успіху код продовжується та виводить лог, що таблицю знайдено, в разі похибки – код перестане йти далі та виведе лог, що таблицю не знайдено.

Після чого було оголошено змінну data, вона містить в собі результат виконання функції, яка передається у метод page.evaluate.

Метод page.evaluate виконує функцію в сторінці веб-браузеру, і, після чого, отримує аргумент “table”.

Після чого, в функції було оголошено зміну rows, яка знаходить всі елементи <tr>, які знаходяться всередині таблиці, та, після чого, перетворює їх у масив.

Після чого, за допомогою reduce, ми заповнюємо масив відповідно до даних кожного рядку таблиці.

Далі йде перевірка на клас «schedule-table__row_head», і, якщо рядок містить в собі цей клас, то йде оновлення значення змінної currentDate, яку було оголошено раніше як null.

Якщо ж рядок не має класу, тоді йде заповнення властивостей sessionPremierm, дані з яких беруться із заданих селекторів (td комірок).

Потім, за допомогою push ми додаємо до масиву data ітерації об’єкту sessionPremier.

Ну а результат цієї функції – є сам масив data, який повертається через return.

За таким самим принципом було розроблено парсинг другого сайту, але з іншими селекторами, так як сайт має іншу структуру побудови.

Далі, в основному коді, використовуючи import було передано дві функції парсингу двох різних сайтів, та, після чого, розроблено масиви, які пізніше містили в собі інформацію згідно заданої функції, а також, було підключено бібліотеку node-telegram-bot-api, також було задано необхідні типи даних, використовуючи typedef, property. Код наведено у лістингу 3.3.

Лістинг 3.3 – Оголошення масивів та типів даних.

```
import TelegramApi from "node-telegram-bot-api";
import { getSessionFromPremierCity } from "./moviespremier.mjs";
import { getSessionFromAdrenaline } from "./moviesadrenaline.mjs";

/**
 * @typedef SessionPremiver
 * @property {string} date
 * @property {string} time
 * @property {string} movieName
 * @property {string} hall
 * @property {string} format
 */

/**
 * @type {SessionPremiver[]}
 */
```

```
let SESSIONS_PREMIER = [];

/**
 * @typedef Session
 * @property {string} movieName
 * @property {string} time
 * @property {string} cinemaPriceChair
 * @property {string} cinemaPriceSofa
 */

/**
 * @type {Session[]}
 */
let SESSIONS = [];
```

кінець лістингу 3.3

Після чого, для роботи із телеграм ботом було додано токен, та об’єкт, який представляє собою самого Telegram бота.

Також було розроблено кнопки для бота.

В коді використовується функція зворотного виклику - onText, яка виконується в момент надсилання повідомлення боту. Через цю функцію було реалізовано дві команди - /find – для пошуку сеансів, та /about – інформація щодо чат-боту. Код наведено у лістингу 3.4.

Лістинг 3.4 – Підключення бота, налаштування клавіш

```
const searchOption = {
  reply_markup: JSON.stringify({
    inline_keyboard: [
      [{ text: "Адреналін City", callback_data: "Adrenaline" }],
      [{ text: "PremierCity", callback_data: "Premier" }],
    ],
  }),
};

const TokenBot = "6254508815:AAEM9OWJEZyMdn1WuDLcT49KtZIJ3sGobp8";
const tBot = new TelegramApi(TokenBot, { polling: true });

tBot.onText(/\/find/, (msg) => {
  const timeResult = getTimerBlock(msg.chat.id);
  if(!timeResult.isBlocked) {
        tBot.sendMessage(msg.chat.id, "Оберіть кінотеатр для пошуку:", searchOption);
  }
  else {
```

```
    tBot.deleteMessage(msg.chat.id, msg.message_id);
    tBot.sendMessage(msg.chat.id, `Захист від спаму.\nОчікуйте: ${timeResult.timerBlock}
секунд.`);
  }
});

tBot.onText(/\/about/, async (msg) => {
  const timeResult = getTimerBlock(msg.chat.id);
  if(!timeResult.isBlocked) {
    tBot.sendMessage(
      msg.chat.id,
      "Привіт. Я є Lutsk Chat Bot.\
         \nЯ допоможу тобі знайти афішу в обраному кінотеатрі.\
         \nДля цього скористайся командою /find."
    );
  }
  else {
    tBot.deleteMessage(msg.chat.id, msg.message_id);
    tBot.sendMessage(msg.chat.id, `Захист від спаму.\nОчікуйте: ${timeResult.timerBlock}
секунд.`);
  }
});
```

кінець лістингу 3.4

Зміна searchOption містить об’єкт з опціями, він має властивість replay_markup, що вказує на розмітку відповіді бота. Властивість replay_markup є рядком, яка представляє з себе JSON об’єкт, цей об’єкт, в свою чергу, містить ключ inline_keyboard, який представляє собою масив із клавішами.

Кожна клавіша описується двома властивостями, це text – що має в собі назву клавіші, та callback_data, яка містить дані, які передаються ботові під час натискання цієї самої клавіші.

Змінна TokenBot має в собі токен боту, tBot – об’єкт, який представляє із себе телеграм бота, через нього і відбувається взаємодія із ботом. tBot містить в собі токен боту та об’єкт налаштувань polling.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування програмного забезпечення є одним із основних етапів в розробці.

Для тестування продуктивності системи в різних ситуаціях використовуються різні методи. До них відносяться тестування безпеки, продуктивності під навантаженням і перевірка полів введення на наявність невірних даних, а також перевірка відповідності отриманих результатів очікуванням.

Тестування проводилось в ручному режимі з позиції користувача. воно має перевагу у швидкому вирішенні будь-яких виявлених проблем і пошуку найбільш ймовірних помилок в інтерфейсі програмного забезпечення.

Для відладки пошуку всіх сеансів з сайту кінотеатрів використовується метод console.log, він виведе всю необхідну інформацію у термінал. На рисунку 3.1 та 3.2 показано результати тестувань двох кінотеатрів.

```
Знайшов таблицю із фільмами!
Дані із таблиці:    Дата: Вт,  30.05 Час: 16:55 Назва: Вартові Галактики 3 Зал: 1 Формат: 3D'
Дані із таблиці:    Дата: Вт,  30.05 Час: 18:00 Назва: Русалонька Зал: 4 Формат: 3D'
Дані із таблиці:    Дата: Вт,  30.05 Час: 18:15 Назва: Мегалодон. Черний демон Зал: 3 Формат: 2D'
Дані із таблиці:    Дата: Вт,  30.05 Час: 19:15 Назва: Форсаж X Зал: 2 Формат: 3D'
Дані із таблиці:    Дата: Вт,  30.05 Час: 19:40 Назва: Лицарі Зодіаку Зал: 1 Формат: 2D'
Дані із таблиці:    Дата: Вт,  30.05 Час: 20:15 Назва: Місія Кандагар Зал: 3 Формат: 2D'
Дані із таблиці:    Дата: Вт,  30.05 Час: 20:30 Назва: Мегалодон. Черний демон Зал: 4 Формат: 2D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 10:15 Назва: Русалонька Зал: 3 Формат: 3D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 10:30 Назва: Лицарі Зодіаку Зал: 1 Формат: 2D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 10:45 Назва: Вартові Галактики 3 Зал: 4 Формат: 3D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 11:00 Назва: Форсаж X Зал: 2 Формат: 3D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 12:40 Назва: Людина-павук: Крізь Всесвіт Зал: 1 Формат: 2D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 12:50 Назва: Місія Кандагар Зал: 3 Формат: 2D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 13:30 Назва: Русалонька Зал: 4 Формат: 3D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 13:45 Назва: Форсаж X Зал: 2 Формат: 3D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 15:10 Назва: Лицарі Зодіаку Зал: 1 Формат: 2D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 15:40 Назва: Русалонька Зал: 3 Формат: 3D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 16:00 Назва: Інспектор Сан. Таємниця чорної вдови Зал: 4 Формат: 2D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 16:30 Назва: Форсаж X Зал: 2 Формат: 3D'
Дані із таблиці:    Дата: Ср,  31.05 Час: 17:20 Назва: Людина-павук: Крізь Всесвіт Зал: 1 Формат: 2D'
```

Рисунок 3.1 – Вікно "TERMINAL" з інформацією із сайту PremierCity

```
Знайшов таблицю із фільмами!
Дані із таблиці:    Назва: Фільм Час: Сеанси Ціна за стілець: Вартість квитка Ціна за диван: undefined'
Дані із таблиці:    Назва: undefined Час: undefined Ціна за стілець: undefined Ціна за диван: undefined'
Дані із таблиці:    Назва: undefined Час: undefined Ціна за стілець: диван Ціна за диван: undefined'
Дані із таблиці:    Назва: БЛЕКБЕРРІ Час: 12:00 Ціна за стілець: 70 Ціна за диван: 80'
Дані із таблиці:    Назва: ПРИВИДИ БУДИНКУ-ВБИВЦІ Час: 12:30 Ціна за стілець: 70 Ціна за диван: 80'
Дані із таблиці:    Назва: Анімаційна феєрія Час: 13:00 Ціна за стілець: 70 Ціна за диван: '
Дані із таблиці:    Назва: ФОРСАЖ X Час: 14:00 Ціна за стілець: 70 Ціна за диван: 80'
Дані із таблиці:    Назва: РИЗИКИ Й ПОБІЧНІ ЕФЕКТИ Час: 14:30 Ціна за стілець: 70 Ціна за диван: 80'
Дані із таблиці:    Назва: ПРИВИДИ БУДИНКУ-ВБИВЦІ Час: 16:00 Ціна за стілець: 70 Ціна за диван: '
Дані із таблиці:    Назва: РУСАЛОНЬКА Час: 16:10 Ціна за стілець: 70 Ціна за диван: 80'
Дані із таблиці:    Назва: ЦНОТЛИВІ САМОГУБИЦІ Час: 16:30 Ціна за стілець: 70 Ціна за диван: 80'
Дані із таблиці:    Назва: ЦНОТЛИВІ САМОГУБИЦІ Час: 18:00 Ціна за стілець: 80 Ціна за диван: '
Дані із таблиці:    Назва: РИЗИКИ Й ПОБІЧНІ ЕФЕКТИ Час: 18:10 Ціна за стілець: 80 Ціна за диван: 90'
Дані із таблиці:    Назва: ФОРСАЖ X Час: 18:30 Ціна за стілець: 80 Ціна за диван: 90'
```

Рисунок 3.2 – Вікно "TERMINAL" з інформаціє із сайту Адреналін City

За таким же самим принципом відладкі інформації було реалізовано тестування пошуку за ключовим словом/частиною слова, результат зображено на рисунку 3.3 та рисунку 3.4.

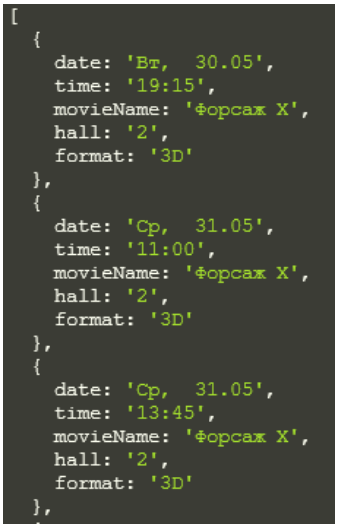

Рисунок 3.3 – Вікно "TERMINAL" з інформацією за ключовим словом PremierCity

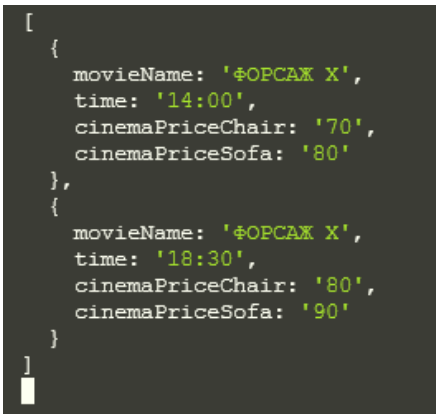

Рисунок 3.4 – Вікно "TERMINAL" з інформацією за ключовим словом Адреналін City

Також в консоль виводяться повідомлення про помилки, якщо вони виникають. Наприклад, якщо html селектор таблиці із фільмами не знайдено на сайті, тоді консоль про це повідомить.

Після тестування всіх основних функцій для отримання даних було розпочато тестування самого телеграм боту. Телеграм бот містить дві команди:

/about – коротка інформація про бота, та команду /find, яка викликає контекстне меню. Результати виконання цих команд показано на русинку 3.5 та на рисунку 3.6.

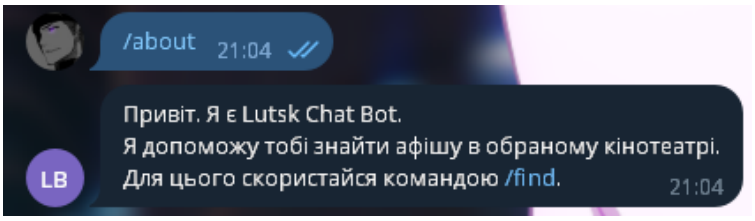

Рисунок 3.5 – Результат виконання команди /about

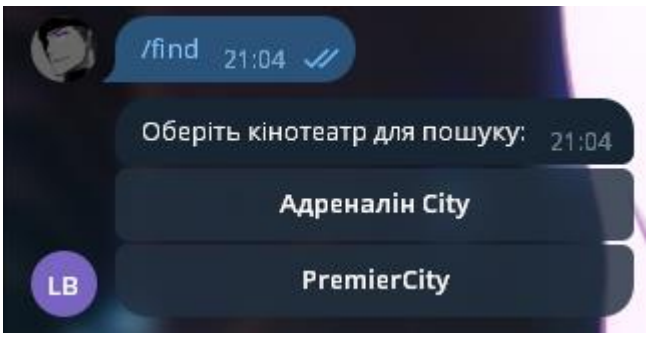

Рисунок 3.6 – Результат виконання команди /find

Команди виконуються, також можна побачити, що команда /find викликає меню, де користувач має обрати самостійно, якій кінотеатр його цікавить. Після обраного кінотеатру бот приховає клавіші, це зроблено для того, щоб клієнт навмисно не навантажував бота заставляючи виконувати запити безліч кількість раз, та попросить ввести назву фільму, або частину його назви, і, в разі успіху, виведе інформацію про сеанси в обраному раніше кінотеатрі. Більш детально це продемонстровано на рисунку 3.7 та рисунку 3.8.

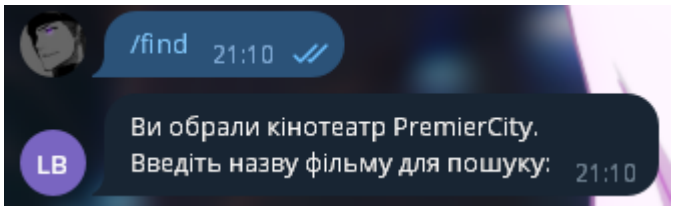

Рисунок 3.7 – Результат натискання клавіші після використання команди /find

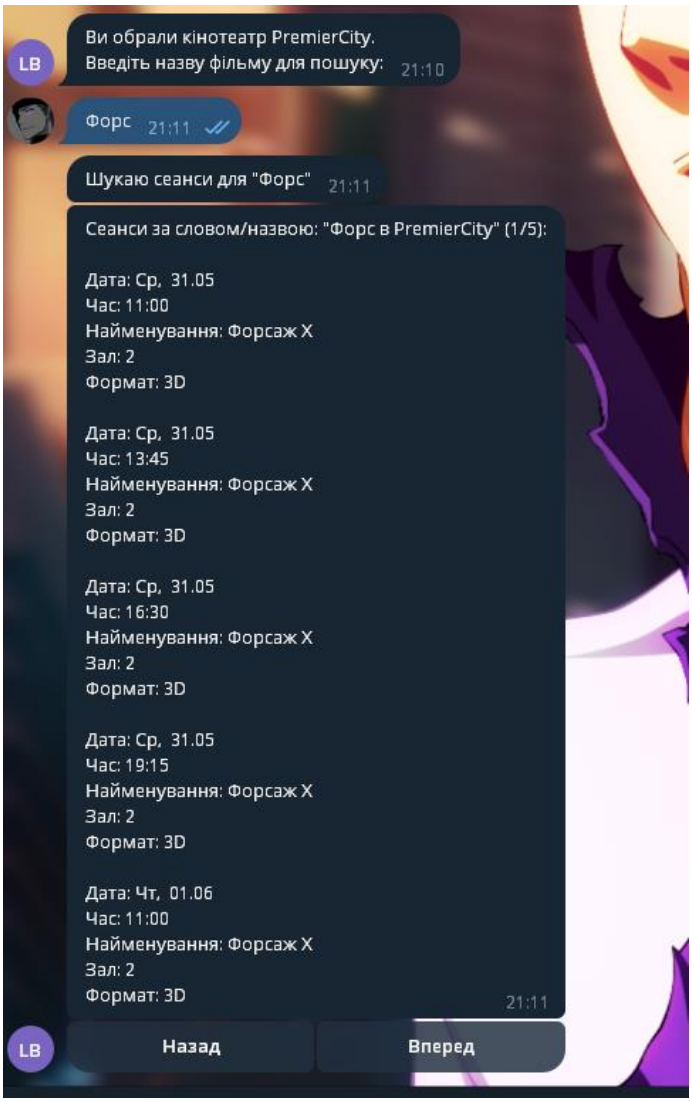

Рисунок 3.8 - Результат після написання назви фільму (PremierCity)

Якщо ж сеанс за назвою/частиною назви не було знайдено, бот також повідомить про це. Результат показано у рисунку 3.9.

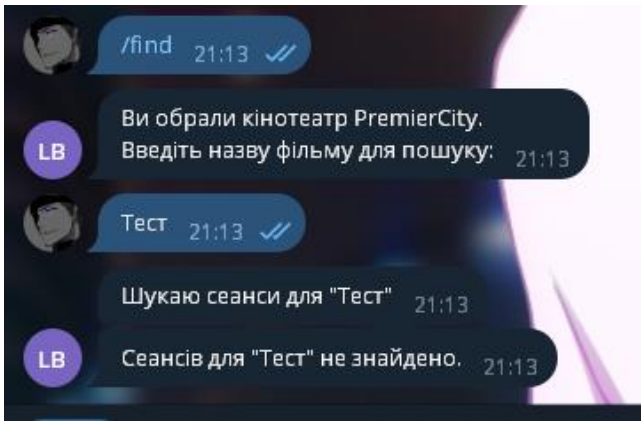

Рисунок 3.9 – Результат із неіснуючим сеансом (PremierCity)

Аналогічні тести також були проведені для іншого кінотеатру. Результат тестів показано на рисунку 3.10 та на рисунку 3.11.

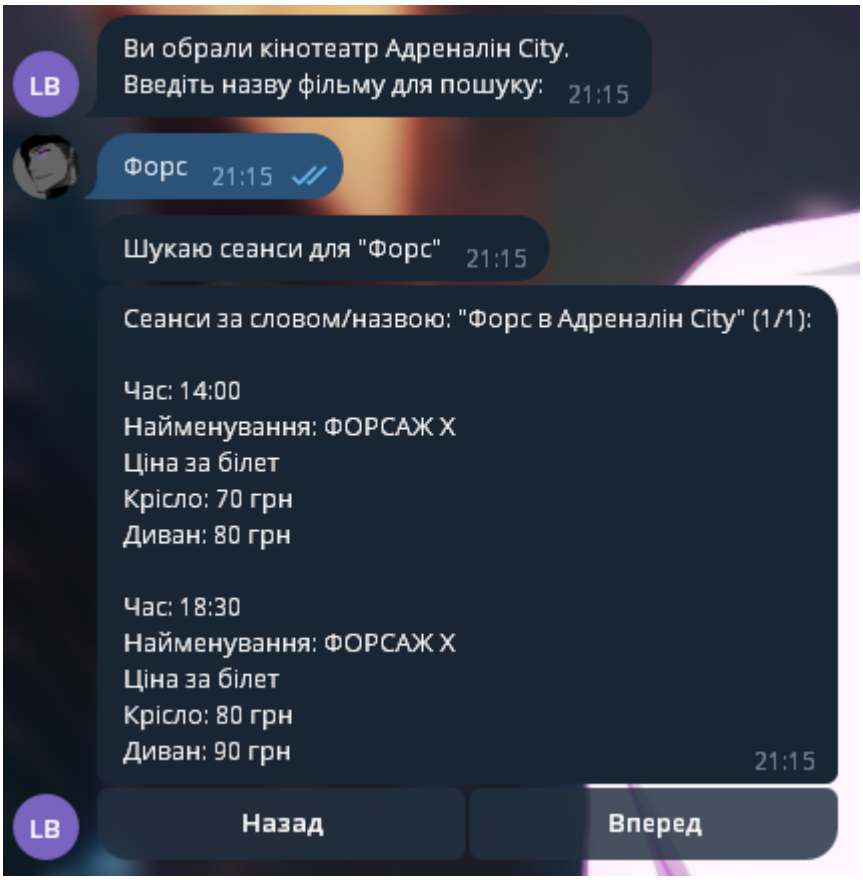

Рисунок 3.10 – Результат команди /find (Адреналін City)

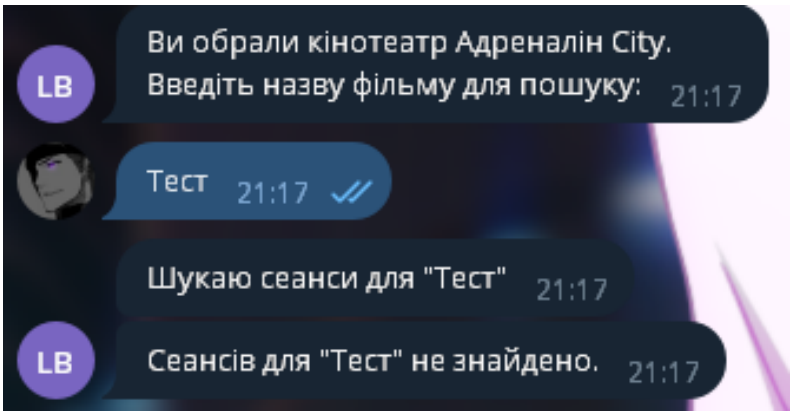

Рисунок 3.11 – Результат із неіснуючим сеансом (Адреналін City)

3.3 Розробка документації для програмного продукту

Розробники створюють документацію з інструкціями по роботі з програмним продуктом для забезпечення комфортного та надійного використання створеного програмного продукту та уникнення найбільш ймовірних проблем при взаємодії з ним.

Файли довідки та документації можуть відрізнятися за дизайном, рівнем деталізації та спеціалізованими навичками, необхідними для опанування матеріалу документації. Це можуть бути як загальні поради та описи незначних компонентів інтерфейсу, так і детальні інструкції з використання з ілюстраціями та прикладами.

Для розроблюваної системи було вирішено зробити пояснювальну записку у вигляді довідки, записка містить в собі інструкцію по використанню чат боту. Документація доступна усім авторизованим у телеграмі користувачам.

Структура довідки відповідатиме структурі розробленого чат боту, де для кожної команди буде описана своя функція призначення.

Візуально документація являє із собою просте pop-up меню, яке відображає користувачу всі доступні команди чат-боту, а також описує їхній функціонал.

Така структура та оформлення є оптимальними для даного програмного продукту та органічно вписуються в загальну будову сайту, зручні та доступні для кінцевого користувача.

3.4 Розробка інсталяційного пакета

Для інсталяції бота на веб-сервер, аби він працював самостійно, було обрано сервіс Heroku.

Heroku – це хмарна платформа розгортання та хостингу веб-додатків. Вона надає зручний спосіб розгортання, керування та масштабування вашого

веб-додатку без необхідності конфігурування та управління серверами. Головну сторінку сайту продемонстровано на рисунку 3.12.

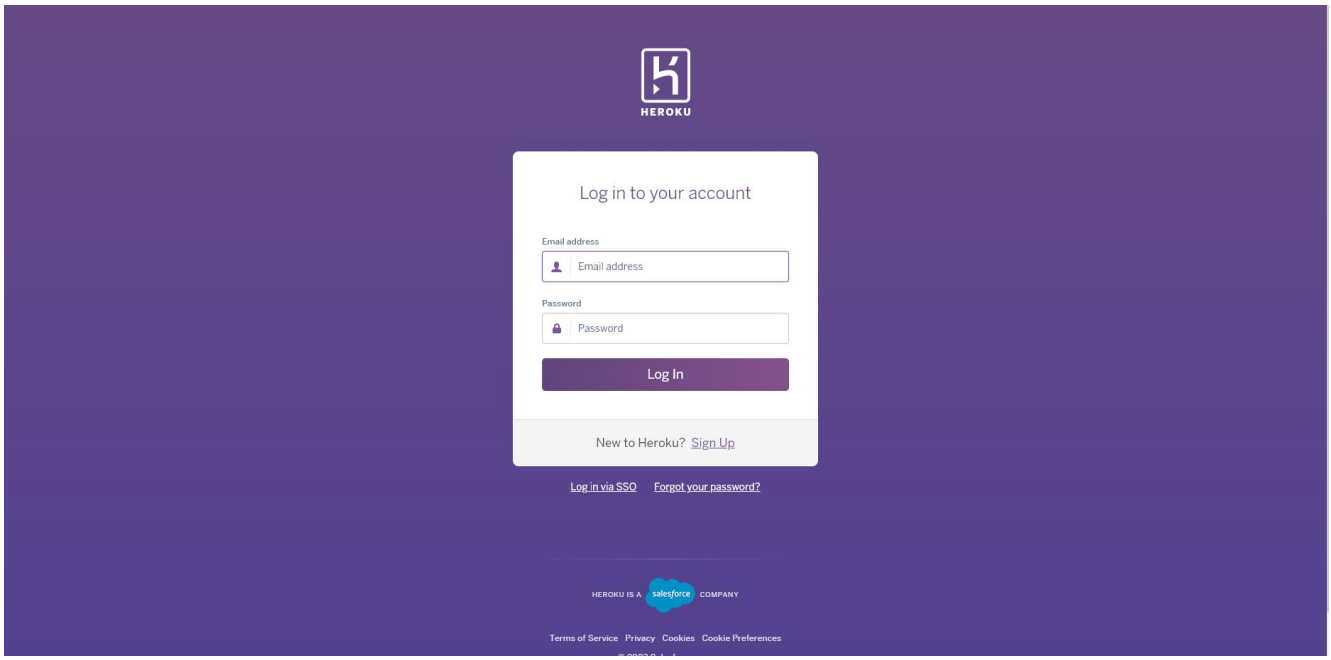

Рисунок 3.12 – Heroku

Для того, аби створити додаток, який буде тримати бота увімкненим завжди, необхідно створити аккаунт, а також мати аккаунт на GitHub.

Після того, як ми зареєстрували аккаунт, необхідно зайти в нього та на вікні що з’явилось натиснути клавішу Create new app. Після чого нам потрібно задати назву проекту (назва допустима лише з нижнім кейсом літер) та обрати регіон, на вибір доступно два регіони:

– Сполучені штати;

– Європа.

Обираємо Європу, після чого слід натиснути Create app. Детальніша інформація продемонстрована на рисунку 3.13.

Після чого з’явиться вікно із вкладкою Deployment. Сайт дає на вибір 2 типу деплоя боту:

– Heroku GIT;

– GitHub.

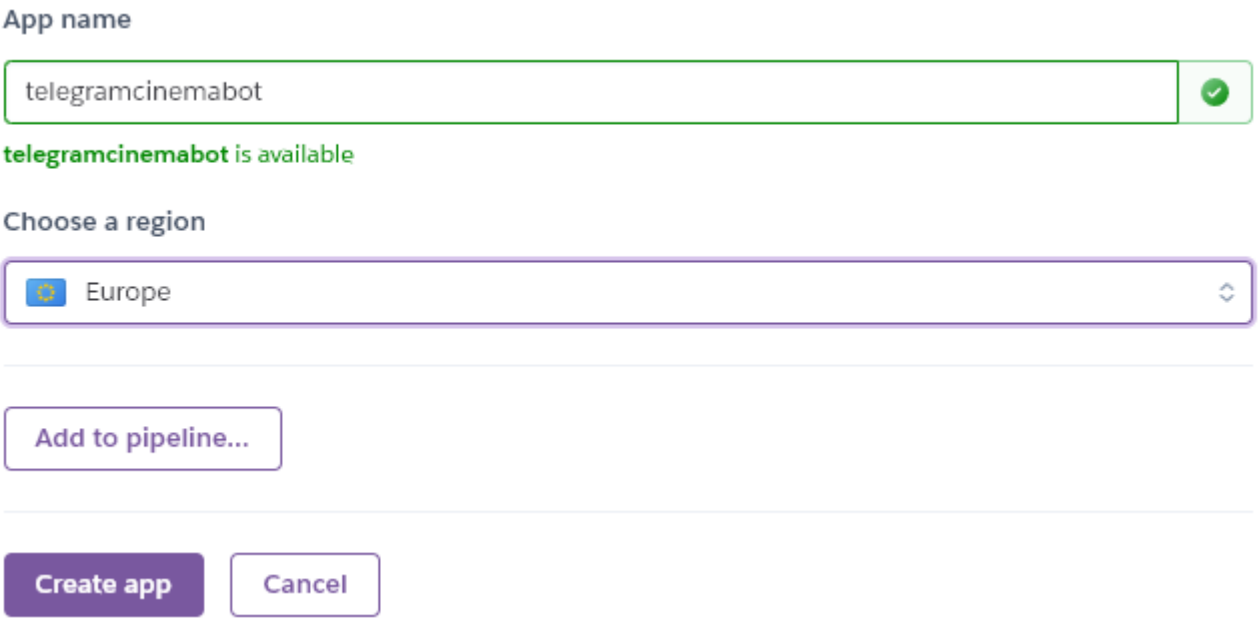

Рисунок 3.13 – Додавання нового додатку

Зручніше буде деплоїти саме через GitHub, також слід підготувати проект до самого деплою, у файлі package.json в рядку scripts необхідно додати два сценарії, а саме:

– команду start, яка буде запускати бота;

– команду для бібліотеки puppeteer, щоб він працював правильно (в 19 версії було змінено спосіб кешування інсталяції хрому).

Лістинг команди продемонстровано у лістингу 3.5.

Лістинг 3.5 – Скрипт для старту бота.

```
"scripts": {
  "start": "node index.mjs",
  "heroku-postbuild": "mkdir ./.cache && mv /app/.cache/puppeteer ./.cache"
},
```

кінець лістингу 3.5

Після чого слід додати файл .gitignore та додати туди наступний рядок (Лістинг 3.6).

Лістинг 3.6 – .gitignore

```
/node_modules
```

кінець лістингу 3.6

Це зроблено для того, аби папка node_modules (папка з модулями) не завантажувалась у репозиторій гітхаб.

Після цього слід створити репозиторій на гітхаб (публічний або приватний), і після чого запушити в гілку master наш проект (Heroku по замовчуванню працює саме із гілкою master).

Далі слід обрати метод деплою – GitHub. Після чого, Heroku запропонує зайти під власним GitHub акаунтом, і після чого, необхідно буде ввести назву репозиторію із GitHub і натиснути клавішу Search (рисунок 3.14).

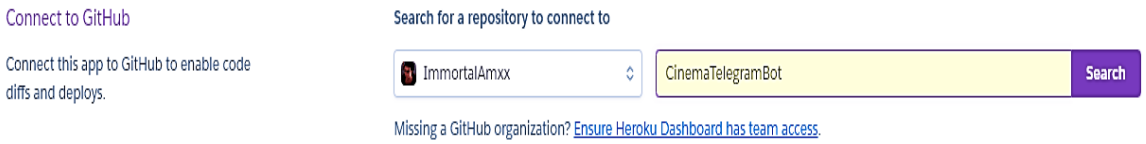

Рисунок 3.14 – Пошук GitHub репозиторію

І після того, коли Heroku знайде репозиторій, слід натиснути Connect.

Після конекту необхідно обрати гілку коміту (по замовчуванню – master). Та натиснути Deploy Branch, також, по можливості, можна ввімкнути Авто-деплой після змін в коді (рисунок 3.15).

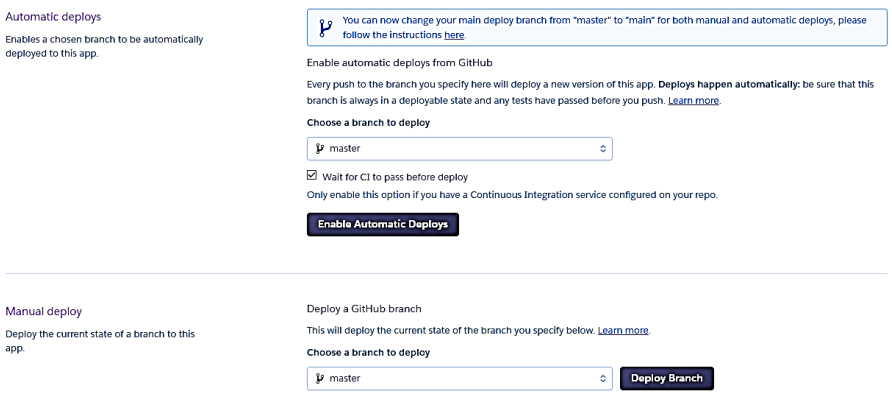

Рисунок 3.15 – Деплой репозиторію

Після закінчення деплою слід перейти в вікно Resources, та обрати в вікні Basic Dynos команду для запуску (її було додано раніше в сценарії) (рисунок 3.16).

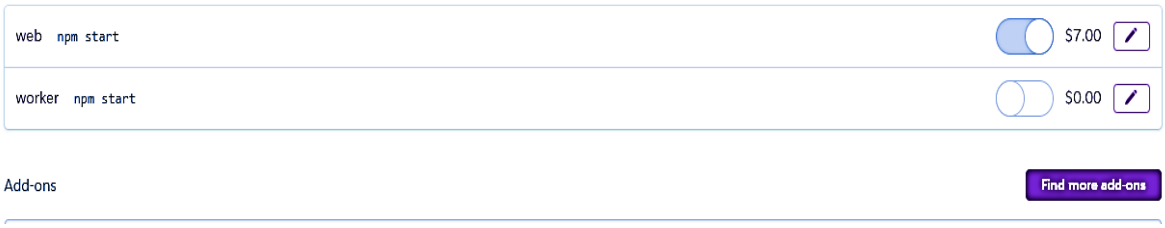

Рисунок 3.16 – Команда для старту боту

Після чого слід натиснути на клавішу More в верхньому правому куті та натиснути клавішу Restart all dynos. Після чого перейти в вікно View logs та подивитись, чи запустився наш бот (рисунок 3.17).

```
2023-06-06T17:11:22.513402+00:00 heroku[web.1]: Error R10 (Boot timeout) -> Web process failed to bind to $PORT within 60 seconds of launch
2023-06-06T17:11:22.530568+00:00 heroku[web.1]: Stopping process with SIGKILL
2023-06-06T17:11:22.693219+00:00 heroku[web.1]: Process exited with status 137
2023-06-06T17:11:22.726325+00:00 heroku[web.1]: State changed from starting to crashed
2023-06-06T17:12:38.682875+00:00 heroku[web.1]: State changed from crashed to starting
2023-06-06T17:12:51.279990+00:00 heroku[web.1]: Starting process with command `npm start`
2023-06-06T17:12:52.957729+00:00 app[web.1]:
2023-06-06T17:12:52.957751+00:00 app[web.1]: > parser@1.0.0 start
2023-06-06T17:12:52.957751+00:00 app[web.1]: > node index.mjs
2023-06-06T17:12:52.957751+00:00 app[web.1]:
```

Рисунок 3.17 – Логи запуску боту

Як можна бачити, бот успішно був запущений на хостингу, тепер немає потреби запускати його кожен раз локально.

Також бота можна «розкласти» на Лінуксі, або будь-якому веб хостингу, який має ос Лінукс і надає вільне місце під файли а також можливість виконувати команди в терміналі.

Із безкоштовних сервісів можна обрати хостинг Vercel, та також «розкласти» бота саме там.

3.5 Захист інформації користувача

Телеграм вживає ряд заходів для захисту своїх користувачів. Основні аспекти, пов’язані із захистом у телеграмі:

– двоетапна аутентифікація;

– керування під’єднаними пристроями;

– спосіб підтвердження номеру не за sms кодом, а за номером виклику.

Не мало важливим є захист інформації користувачів, наприклад, під час спілкування користувача із ботом. Телеграм надає такий захист для користувачів. Телеграм забезпечує шифер кінець-до-кінця для абсолютно всіх повідомлень, включаючи ті, які передаються саме через чат бота [13, 14]. Також телеграм містить в собі аутентифікацію чат-ботів, це допомагає користувачу запобігти чатування із ботами-шахраями.

Не менш важливим є правила використання чат-ботів, вони включають в себе обмеження на спам-розсилку або на розсилку шкідливого контенту через ботів, і, в разі необхідності, користувач має змогу поскаржитись на бота техпідтримці телеграму, що б підтримка, в свою чергу, заблокувала бота.

Висновки до розділу 3

Здійснено детальний опис проекту. Обґрунтовано етапи тестування програмного продукту. Означено помилки, що виявляються на відповідній стадії тестування.

Описано реалізацію документації з інструкціями по роботі з програмним продуктом для забезпечення комфортного та надійного використання.

Обґрунтовано кроки інсталяції бота на веб-сервер.

Виконано опис захисту інформації користувача.

ВИСНОВКИ

У кваліфікаційній роботі було проведено розробку телеграм чат-боту на NodeJS з використанням таких бібліотек як: puppeteer, node-telegram-bot-api.

На початку роботи було здійснено аналіз предметної області дослідження. Розглянуто та обґрунтовано основні програмні засоби проектування. Розроблений застосунок забезпечує весь необхідний функціонал.

Для розробки телеграм-боту було використано NodeJS, а також сторонні бібліотеки, що зробило процес розробки більш зручним та швидшим. Node Telegram Bot Api дав змогу легко та швидко співпрацювати із Telegram Api, puppeteer дав змогу «парсити» сторінки сайту ти витягувати з них необхідну з них інформацію використовуючи селектори.

Під час розробки також виникали проблеми при взаємодії із натиснутими клавішами а також при реалізації сторінок, але, завдяки відкритий інформації та вивченню API Телеграму ці проблеми було вирішено.

Наступним етапом було здійснено тестування програмного продукту. Результати, отримані під час розробки підтверджують реалізацію програмного продукту.

Одним з ефективних етапів роботи над проектом – був аналіз інформаційної безпеки користувача, що дозволить переконатися в надійності захисту інформації.

Головне досягнення роботи є створенням телеграм чат-боту, який матиме змогу надати необхідну користувачу інформацію про сеанси в обраному кінотеатрі міста Луцьк. Функціонал боту не є великим, але, при цьому, він виконує поставлені задачі.

В подальшому чат-бота можна покращити, додавши більше функціоналу: збільшити список кінотеатрів для пошуку сеансів, додати можливістю змінювати мову, а також, в разі потребі, додати змогу залишати відгуки, щоб покращити функціонал бота або виправити баги, які буде виявлено користувачем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке чат бот? URL: https://marchenko.marketing/scho-take-chat-bot-ta-yak-stvoriti-bota-v-telegram (дата звернення: 15.04.2023).

2. Що таке парсинг і для чого використовується? URL: https://dalistrategies.com/ua/shho-take-parsing-i-dlya-chogo-vikoristovuietsya/ (дата звернення 20.04.2023).

3. Що таке Visual Studio? URL: https://uk.education-wiki.com/3958588-what-is-visual-studio-code (дата звернення: 20.04.2023).

4. About Node.JS. URL: https://nodejs.org/en/about (дата звернення: 23.04.2023).

5. Уэкслер Д. Get Programming with Node.js, 2019. 480 с.

6. Сучасний підручник JavaScript. URL: https://uk.javascript.info (дата звернення: 04.05.2023)

7. X. Huang. Research and Application of Node.js Core Technology. 2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI). Sanya, China, 2020. Pp. 1-4, doi: 10.1109/ICHCI51889.2020.00008.

8. Zobair Ullah. Role of JavaScript function, method, event and object in website development. International Journal of Recent Scientific Research, Vol. 11, Issue, 01(F), pp. 37137-37146.

9. Node Telegram Bot Api Usage. URL: https://github.com/yagop/node-telegram-bot-api/blob/master/doc/usage.md (дата звернення: 30.04.2023).

10. Node Telegram Bot Api. URL: https://npm.io/package/node-telegram-bot-api (дата звернення: 29.04.2023).

11. FAQ | Puppeteer. URL: https://pptr.dev/faq (дата звернення: 28.04.2023).

12. Dargahi Nobari, Arash Reshadatmand, Negar Neshati Mahmood. Analysis of Telegram, An Instant Messaging Service. CIKM'18, November 6–10, 2018, Singapore. Singapore, 2018. Pp. 2035-2038.

13. Anonymity, emotions, and questionable information: Seven takeaways on how Telegram channels operate in Ukraine. URL: https://akademie.dw.com/en/anonymity-emotions-and-questionable-information-seven-takeaways-on-how-telegram-channels-operate-in-ukraine/a-65646442 (дата звернення: 20.05.2023 р.)

14. Telegram Privacy Policy. URL: https://telegram.org/privacy?setln=fa (дата звернення: 20.05.2023 р.).

ДОДАТКИ

Додаток А

Лістинг 1 – Функціонал взаємодії із Telegram API

```
tBot.onText(/\/find/, (msg) => {
  const userChatId = msg.chat.id;
  const timeResult = getTimerBlock(userChatId);

  if(!timeResult.isBlocked) {
    tBot.sendMessage(userChatId, "Оберіть кінотеатр для пошуку:",
searchOption);
  }
  else {
    tBot.deleteMessage(userChatId, msg.message_id);
    tBot.sendMessage(userChatId, `Захист від спаму.\nОчікуйте:
${timeResult.timerBlock} секунд.`);
  }
});

tBot.onText(/\/about/, async (msg) => {
  const userChatId = msg.chat.id;
  const timeResult = getTimerBlock(userChatId);

  if(!timeResult.isBlocked) {
    tBot.sendMessage(
      userChatId,
      "Привіт. Я є Lutsk Chat Bot.\
          \nЯ допоможу тобі знайти афішу в обраному кінотеатрі.\
          \nДля цього скористайся командою /find."
    );
  }
  else {
    tBot.deleteMessage(userChatId, msg.message_id);
    tBot.sendMessage(userChatId, `Захист від спаму.\nОчікуйте:
${timeResult.timerBlock} секунд.`);
  }
});

tBot.on("callback_query", async (msg) => {
  const data = msg.data;
  const userChatId = msg.message.chat.id;
  const messageId = msg.message.message_id;

  const handleAdrenaline = async () => {
    tBot.editMessageText(`Ви обрали кінотеатр Адреналін
City.\nВведіть назву фільму для пошуку:`, {
      chat_id: userChatId,
      message_id: messageId,
      reply_markup: JSON.stringify({ inline_keyboard: [] }),
    });
```

```
    tBot.once("message", async (subMsg) => {
      const movieName = subMsg.text;

      if(movieName[0] === '/')
      {
        tBot.sendMessage(userChatId, "Недопустимий символ для
пошуку!");
        tBot.sendMessage(userChatId, "Оберіть кінотеатр для
пошуку:", searchOption);

        return;
      }

      if(movieName.length < lengthSize)
      {
        tBot.sendMessage(userChatId, 'Замало символів для
пошуку.');
        tBot.sendMessage(userChatId, "Оберіть кінотеатр для
пошуку:", searchOption);

        return;
      }

      tBot.sendMessage(userChatId, `Шукаю сеанси для
"${movieName}"`);

      SESSIONS = await getSessionFromAdrenaline();
      const sessions = findSessionsByString(movieName, data);

      console.log(sessions);

      const totalPages = Math.ceil(sessions.length / pageSize);

      const getSessionsForCurrentPage = () => {
        const startIndex = (currentPage - 1) * pageSize;
        const endIndex = startIndex + pageSize;

        return sessions.slice(startIndex, endIndex);
      };

      const sendMessageWithPagination = () => {
        const sessions = getSessionsForCurrentPage();

        if (sessions.length > 0) {
          const message = sessions
            .map(
              (session) =>
                `Час: ${session.time}\nНайменування:
${session.movieName}\nЦіна за білет\nКрісло:
```

```
${session.cinemaPriceChair} грн\nДиван: ${session.cinemaPriceSofa}
грн`
            )
            .join("\n\n");

          const paginationMessage = `Сеанси за словом/назвою:
"${movieName} в Адреналін City"
(${currentPage}/${totalPages}):\n\n${message}`;

          tBot.sendMessage(userChatId, paginationMessage, {
            reply_markup: JSON.stringify({
              inline_keyboard: [
                [
                  { text: "Назад", callback_data: "PREV_PAGE_ARD"
},
                  { text: "Вперед", callback_data: "NEXT_PAGE_ADR"
},
                ],
              ],
            }),
          });
        } else {
          tBot.sendMessage(userChatId, `Сеансів для "${movieName}"
не знайдено.`);
        }
      };

      sendMessageWithPagination();

      tBot.on("callback_query", async (msg) => {
        const queryData = msg.data;

        if (queryData === "PREV_PAGE_ARD") {
          if (currentPage > 1) {
            currentPage--;
          }
          tBot.deleteMessage(userChatId, messageId);
          sendMessageWithPagination();
        } else if (queryData === "NEXT_PAGE_ADR") {
          if (currentPage < totalPages) {
            currentPage++;
          }
          tBot.deleteMessage(userChatId, messageId);
          sendMessageWithPagination();
        }
      });
    });
  };

  const handlePremier = async () => {
```

```
    tBot.editMessageText(`Ви обрали кінотеатр
PremierCity.\nВведіть назву фільму для пошуку:`, {
      chat_id: userChatId,
      message_id: messageId,
      reply_markup: JSON.stringify({ inline_keyboard: [] }),
    });

    tBot.once("message", async (subMsg) => {
      const movieName = subMsg.text;

      if(movieName[0] === '/')
      {
        tBot.sendMessage(userChatId, "Недопустимий символ для
пошуку!");
        tBot.sendMessage(userChatId, "Оберіть кінотеатр для
пошуку:", searchOption);

        return;
      }

      if(movieName.length < lengthSize)
      {
        tBot.sendMessage(userChatId, 'Замало символів для
пошуку.');
        tBot.sendMessage(userChatId, "Оберіть кінотеатр для
пошуку:", searchOption);

        return;
      }

      tBot.sendMessage(userChatId, `Шукаю сеанси для
"${movieName}"`);

      SESSIONS_PREMIER = await getSessionFromPremierCity();
      const sessionPremier = findSessionsByString(movieName,
data);

      console.log(sessionPremier);

      const totalPages = Math.ceil(sessionPremier.length /
pageSize);

      const getSessionsForCurrentPage = () => {
        const startIndex = (currentPage - 1) * pageSize;
        const endIndex = startIndex + pageSize;
        return sessionPremier.slice(startIndex, endIndex);
      };

      const sendMessageWithPagination = () => {
        const sessions = getSessionsForCurrentPage();
```

```
        if (sessions.length > 0) {
          const message = sessions
            .map(
              (session) =>
                `Дата: ${session.date}\nЧас:
${session.time}\nНайменування: ${session.movieName}\nЗал:
${session.hall}\nФормат: ${session.format}`
            )
            .join("\n\n");

          const paginationMessage = `Сеанси за словом/назвою:
"${movieName} в PremierCity"
(${currentPage}/${totalPages}):\n\n${message}`;

          tBot.sendMessage(userChatId, paginationMessage, {
            reply_markup: JSON.stringify({
              inline_keyboard: [
                [
                  { text: "Назад", callback_data: "PREV_PAGE" },
                  { text: "Вперед", callback_data: "NEXT_PAGE" },
                ],
              ],
            }),
          });
        } else {
          tBot.sendMessage(userChatId, `Сеансів для "${movieName}"
не знайдено.`);
        }
      };

      sendMessageWithPagination();

      tBot.on("callback_query", async (msg) => {
        const queryData = msg.data;

        if (queryData === "PREV_PAGE") {
          if (currentPage > 1) {
            currentPage--;
          }
          tBot.deleteMessage(userChatId, messageId);
          sendMessageWithPagination();
        } else if (queryData === "NEXT_PAGE") {
          if (currentPage < totalPages) {
            currentPage++;
          }
          tBot.deleteMessage(userChatId, messageId);
          sendMessageWithPagination();
        }
      });
```

```
    });
  };

  switch(data) {
    case "Adrenaline": {
        handleAdrenaline();
        break;
    }
    case "Premier": {
        handlePremier();
        break;
    }
  }
});
```

кінець лістингу 1.

Лістинг 2 – Затримка між відправленнями повідомлення.

```
const lastMessageTimes = {};
const messageDelay = 30000;

function getTimerBlock (chatId) {
    if (lastMessageTimes.hasOwnProperty(chatId))
    {
        const currentTime = Date.now();
        const timerBlock = currentTime - lastMessageTimes[chatId];

        if (timerBlock < messageDelay)
        {
            return {
                isBlocked: true,
                timerBlock: Math.floor((messageDelay - timerBlock)
/ 1000)
            };
        }
    }

    lastMessageTimes[chatId] = Date.now();
    return {
        isBlocked: false,
        timerBlock: 0
```

```
    };
}

export { getTimerBlock };
```

кінець лістингу 2.