Вищий навчальний заклад
«Університет економіки та права «КРОК»
Фаховий коледж

Циклова комісія з інформаційних технологій

Кваліфікаційна робота фахового молодшого бакалавра

На тему Створення месенджер-бота для кіноманів з системою рекомендації фільмів.

Виконала _____
(Підпис)

_____Войтенко Дарина Дмитрівна_____
(прізвище, ім'я, по батькові)

_____

Науковий керівник
__Головань Володимир Володимирович__
(прізвище, ім'я, по батькові)

_____

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Голова циклової комісії _____ _____ _____
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2025 року

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Спеціальність 121 інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Голова циклової комісії _____ Леонід УВАРОВ
(підпис)

«____» _____ 2025 року

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач освіти Войтенко Дарина Дмитрівна

1. Тема роботи Створення месенджер-бота для кіноманів з системою рекомендації фільмів затверджена наказом по університету
   від «___» _____ 202__ р. № _____
2. Термін здачі закінченої роботи «30» травня 2025 року
3. Вихідні дані до роботи:
   1) цільова аудиторія – кіномани, глядачі з різними вподобаннями;
   2) функціональність – рекомендації фільмів, перегляд добірок фільмів за жанром;
   3) технічні вимоги - платформа для розробки бота (Telegram), мова програмування, база даних та інші технології;
   4) можливість інтеграції (взаємодії) з платформа для розробки бота (Telegram), мова програмування, база даних та інші технології;
   5) існуючі рішення та кращі практики у сфері розробки ботів для рекомендації фільмів.
4. Зміст пояснювальної записки
   1) Розділ 1. Аналіз існуючих рішень та кращих практик у сфері створення Telegram-ботів для рекомендації фільмів. Огляд прикладів подібних ботів та функціональних можливостей стрімінгових сервісів. Обґрунтування вибору

Telegram як платформи, як мови програмування та як бази даних. Вибір сторонніх API для отримання фільмів.

2) Розділ 2. Створення зручного та інтуїтивно зрозумілого інтерфейсу бота за допомогою Telegram Web Apps, кнопок та меню. Розробка алгоритму, за яким бот обробляє запити користувачів і надає персоналізовані рекомендації. Забезпечення можливості збереження вподобань користувача та історії перегляду. Побудова архітектури, налаштування бази даних та інтеграція з зовнішніми джерелами інформації про фільми.
3) Розділ 3. Перевірка роботи бота на різних пристроях (смартфонах, планшетах, ПК), у різних середовищах Telegram. Виявлення і виправлення помилок у логіці, інтерфейсі та візуалізації даних. Аналіз відповідності функціональності поставленим цілям і очікуванням користувачів.

5. Перелік графічного матеріалу:

Скріншоти існуючих рішень

Скріншоти інтерфейсу продукту

Схеми і таблиці щодо візуалізації аналізу

Блок-схеми алгоритмів

Діаграми щодо проєктування продукту (наприклад, потоків даних, переходів станів, сутність-зв'язок, UML, бази даних тощо)

Дата видачі завдання «12» лютого 2025 року

Науковий керівник _____ __Головань В. В.__
(підпис) (прізвище, ім'я, по батькові)

Завдання прийняла до виконання _____ ___Войтенко Д.Д.__
(підпис) (прізвище, ім'я, по батькові)

РЕФЕРАТ

Пояснювальна записка: 51 сторінок, 5 рисунків, 2 таблиць, 4 додатків, 15 джерел.
Об'єкт дослідження — процес автоматизації рекомендації фільмів та взаємодії з користувачами за допомогою месенджер-ботів.
Мета роботи — розробка Telegram-бота “Bot_film” для організації персоналізованих рекомендацій фільмів, збереження вподобань користувачів, переглядів та взаємодії з кіно-контентом через зручний інтерфейс.
У кваліфікаційній роботі проведено аналіз сучасних підходів до створення чат-ботів у сфері розваг, досліджено існуючі сервіси з рекомендацій фільмів, обґрунтовано доцільність використання платформи Telegram для реалізації інтерактивного бота. Було створено Telegram-бота “Bot_film” із функціоналом:

- інтерактивного вибору фільмів через кнопки та меню,
- отримання опису, рейтингу фільмів через зовнішні API.

Бот реалізовано з використанням мови JavaScript (Node.js), бібліотеки Telegraf для роботи з Telegram API та бази даних MongoDB, яка використовується для зберігання користувацьких профілів, історій переглядів та інформації про фільми. У роботі детально описано логіку функціонування бота, структуру бази даних, алгоритм рекомендацій, а також виконано тестування взаємодії на різних пристроях.
Надано рекомендації щодо масштабування системи, підвищення продуктивності та безпеки персональних даних користувачів.
Результати розробки можуть бути застосовані в освітніх або розважальних проєктах, стримінгових сервісах, бот-платформах або стартапах у сфері цифрового дозвілля.
Ключові слова: Telegram-бот, рекомендації фільмів, JavaScript, Node.js, MongoDB, персоналізація, чат-бот, медіа-контент**.**

ABSTRACT

Explanatory note: 51 pages, 5 figures, 2 tables, 4 appendices, 15 sources.
The object of the study is the process of automating movie recommendations and interaction with users using messenger bots.
The purpose of the work is to develop a Telegram bot “Bot_film” for organizing personalized movie recommendations, saving user preferences, viewings and interacting with movie content through a convenient interface.
The qualification work analyzed modern approaches to creating chat bots in the entertainment industry, investigated existing movie recommendation services, and justified the feasibility of using the Telegram platform to implement an interactive bot. A Telegram bot “Bot_film” was created with the following functionality:
• interactive selection of movies through buttons and menus,
• obtaining a description, rating of movies through external APIs.
The bot is implemented using JavaScript (Node.js), the Telegraf library for working with the Telegram API, and the MongoDB database, which is used to store user profiles, viewing histories, and information about movies. The work describes in detail the logic of the bot's operation, the database structure, the recommendation algorithm, and also tested interaction on various devices.
Recommendations are provided for scaling the system, increasing productivity, and security of users' personal data.
The results can be applied in educational or entertainment projects, streaming services, bot platforms, or startups in the field of digital leisure.
Keywords: Telegram bot, movie recommendations, JavaScript, Node.js, MongoDB, personalization, chat bot, media content.

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА КРАЩИХ ПРАКТИК 10
1.1 Поняття чат-боту 10
1.2. Аналіз предметної області 11
1.3. Створення та основні задачі чат-боту 16
1.4. Огляд Telegram-ботів для рекомендації фільмів 18
1.5. Функціональні можливості стрімінгових сервісів 20
1.6. Обґрунтування вибору платформи Telegram, мови програмування та бази даних 22
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА 26
2.1. Створення зручного інтерфейсу бота з використанням Telegram Web Apps, кнопок і меню 26
2.2. Розробка алгоритму персоналізованих рекомендацій 27
2.3. Забезпечення збереження вподобань користувачів і історії переглядів 29
2.4. Архітектура системи, налаштування бази даних і інтеграція з API 30
РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА 33
3.1. Тестування роботи бота на різних пристроях і платформах 33
3.2. Виявлення та усунення помилок у логіці, інтерфейсі та візуалізації 34
3.3. Аналіз відповідності функціоналу цілям і очікуванням користувачів 36
ВИСНОВКИ 38
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 42
ДОДАТКИ 44
Додаток А. Скріншоти інтерфейсу розробленого бота 44
Додаток В. Фрагменти коду та конфігураційні файли 47
Додаток Г. Схеми і таблиці щодо візуалізації аналізу 50
Додаток Д. Узагальнена схема алгоритму функціонування чат-боту 51

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – інтерфейс програмування додатків

ШІ – штучний інтелект

GUI – графічний інтерфейс користувача

ID – унікальний ідентифікатор

Node.js – середовище виконання JavaScript на серверній стороні

JavaScript – мова програмування для створення інтерактивних веб-додатків

MongoDB – нереляційна документ орієнтована база даних

Bot_film – назва Telegram-бота для рекомендації фільмів

ВСТУП

Актуальність теми. Сучасна цифрова епоха характеризується стрімким розвитком месенджерів і чат-ботів, які значно спрощують взаємодію користувачів з інформаційними сервісами. Одним із перспективних напрямів є створення ботів для рекомендації фільмів, що дозволяє кіноманам оперативно отримувати персоналізовані пропозиції відповідно до власних вподобань. Такий підхід сприяє економії часу, підвищує якість користувацького досвіду та популяризує якісний контент. У зв'язку з цим розробка зручного та інтуїтивно зрозумілого месенджер-бота з функцією рекомендацій є актуальним завданням сучасної інформаційної технології.

Мета роботи. Метою даної роботи є створення Telegram-бота для кіноманів, який забезпечує:

- зручний та інтуїтивно зрозумілий інтерфейс;
- персоналізований підбір фільмів;
- збереження історії взаємодій користувача;
- інтеграцію з популярними зовнішніми сервісами (через API).

Завдання роботи. Для досягнення поставленої мети необхідно виконати наступні завдання:

- Провести аналіз існуючих рішень та найкращих практик у сфері чат-ботів і систем рекомендацій фільмів.
- Розробити зручний інтерфейс бота з використанням функціоналу Telegram.
- Реалізувати алгоритми формування персоналізованих рекомендацій на основі вподобань користувачів.
- Налагодити збереження історії переглядів та вибору фільмів.
- Інтегрувати зовнішні API для отримання актуальної інформації про фільми.
- Провести тестування бота на різних пристроях та в різних середовищах.

Об'єкт дослідження. Методи та технології створення Telegram-бота для персоналізованих рекомендацій фільмів.

Предмет дослідження. Функціональні можливості чат-бота для реалізації персоналізованих рекомендацій фільмів у середовищі Telegram.

Методи дослідження. У процесі роботи використано такі методи:

- **Аналіз** існуючих рішень у сфері чат-ботів та рекомендаційних систем, що дозволило виділити найефективніші підходи до реалізації.
- **Проєктування програмного забезпечення** із застосуванням блок-схем, UML-діаграм і архітектурних моделей.
- **Розробка** інтерфейсу із використанням Telegram Web Apps для забезпечення кросплатформенності та зручності.
- **Програмування** на JavaScript із використанням Node.js та бази даних MongoDB для ефективного збереження інформації.
- **Інтеграція API** за допомогою REST-запитів для отримання актуальних даних про фільми.
- **Тестування** продукту на різних пристроях з метою виявлення помилок та оцінки стабільності.
- **Опитування користувачів** для аналізу рівня задоволеності функціональністю бота.

Практичне значення одержаних результатів. Розроблений месенджер-бот має високу практичну цінність для широкої аудиторії кіноманів. Він дозволяє швидко знаходити персоналізовані рекомендації, зберігати історію переглядів і формувати унікальний користувацький досвід. Бот може використовуватись як окремий сервіс або як модуль для інтеграції у стрімінгові платформи, сайти кінотеатрів чи онлайн-спільноти. Реалізоване рішення демонструє ефективність застосування сучасних вебтехнологій, API та підходів до створення масштабованих систем рекомендацій.

Структура роботи. Пояснювальна записка складається з трьох розділів:

Розділ 1 — теоретичний, присвячений аналізу аналогів, опису сучасних підходів і вибору технологій.

Розділ 2 — практичний, містить опис архітектури системи, інтерфейсу користувача та програмної реалізації.

Розділ 3 — експериментальний, містить результати тестування, аналіз ефективності роботи бота, виявлені недоліки та шляхи їх усунення.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА КРАЩИХ ПРАКТИК

1.1 Поняття чат-боту

Чат-боти – це такі діалогові програми, які здатні імітувати людське спілкування за допомогою тексту або голосових повідомлень. Вони допомагають нам автоматизувати завдання, просто працюючи за деяким заданим алгоритмом, а також самостійно вести простий діалог з людиною, відповідаючи на її запити. У 1966 році світ побачив перші програми, що моделюють людське спілкування. Віртуальний співрозмовник Еліза імітував діалог з психотерапевтом. Популярність чат-ботів почала зростати у 2010 році, коли люди почали активно користуватись месенджерами. Більшість з них працює на популярних платформах обміну повідомленнями: Facebook Messenger, Телеграм, Viber, Skype, Slack. Всі боти мають можливість працювати як відокремлені програми або бути об’єднаними з функціональністю будь-якої пошукової системи.

Чат-боти призначені і часто використовуються в таких сферах, як інтернетбанкінг, освіта, готельно-ресторанний бізнес, ігрова індустрія тощо. Зазвичай чатботів використовують для задач з вузькою спеціалізацією та обмеженим призначенням і вони не мають можливості знайти застосування для широкого кола потреб. Всі компанії мають свою думку, щодо класифікації чат-ботів. Однак можна 9 виділити лише 2 типи класифікації:

– Бізнес-класифікація чат-ботів

– Технічна класифікація чат-ботів.

Бізнес-класифікація чат-ботів:

– Чат-боти для розмов. Вони були розроблені для простого спілкування з користувачем. Такі боти не мають ніякої конкретної мети.

– Чат-помічники. У таких ботів є визначена ціль. На основі відповідей людини вони збирають дані, що їм потрібні для вирішення проблеми. Часто знаходять своє призначення для сфери онлайн-банкінгу.

– Чат-боти Q&A. Q&A (запитання та відповіді). Це доволі прості боти, розроблені для того, щоб отримати 1 відповідь на 1 запитання. Іноді створюють для того, щоб замінити розділи FAQ деяких інтернет-ресурсів.

1.2. Аналіз предметної області

Що таке Телеграм

Завдяки активному онлайн-розвитку після 2015 року месенджер Телеграм став однією з найвідоміших комунікаційних платформ для аудиторії та своєрідним порталом новин. Створення та розвиток мобільної платформи Телеграм, побудованої за принципом платформи, прив'язаної до номера телефону. У серпні 2013 року Павло Дуров презентував свою нову роботу, тоді і почалась розробка месенджера. Спершу Телеграм виконував функції старого месенджера зі стандартними функціями для цієї платформи. Основною та унікальною особливістю Телеграму було шифрування всього процесу листування. Безпека передачі особистих відомостей і можливість ведення групового облікового запису привернули увагу месенджера до анонімної комп'ютерної мережі «Невидимий інтернет», комунікаційна технологія конфіденційності якого була значно спрощена. Цим скористались кіберзлочинці, що почали активно використовувати Телеграм і звісно ж звичайні користувачі, які також відкрили для себе неповторну систему захисту даних. А також найпростішу форму листування, яка була дуже корисною для тих, хто хотів спілкуватись в простій розмові в Telegram, не заповнюючи її.

Телеграм-канали призначені для заохочення нової аудиторії. Перші групи були створені користувачами для обговорення групових дискусій, і врешті-решт відкрилися і ЗМІ, щоб залучити активну дорослу аудиторію, що все ще залишається складним завданням. Єдина проблема – це наповнення платформи контентом. Якщо окремі користувачі публікують свої новини та дописи без конкретики та своєчасності, ЗМІ зосереджуватимуть увагу на їх змісті (особливо на онлайн-рейтингу їхніх публікацій), їм доведеться шукати інструменти, щоб автоматично наповнювати сторінку. Такими помічниками стали боти - написане програмне забезпечення, яке знав, де взяти інформацію і що завантажувати в Телеграм. Боту достатньо було відразу наказати, який документ подати в месенджер, усно показати результати пошуку чи мультимедіа, а журналістам залишалося лише виконувати свої прямі обов'язки, чим і займається вся решта бота. Боти завоювали багато онлайн-ЗМІ, наприклад ті, що працювали з каналами в телеграмі, що значно спростило їм роботу. Інтерес до ботів настільки зріс, коли месенджер Телеграм створив окреме середовище управління ботами - Botfather. Кожен, хто володіє базовими знаннями використання інтерфейсів додатків, може написати програму-помічника та використовувати її для побудови власної мережі. Незважаючи на свій молодий вік, Телеграм зараз користується понад 100 мільйонами користувачів щомісяця, особливо популярний в Західній Європі. Розробники месенджера наполягають на тому, що цей месенджер має найкращий захист серед подібних продуктів на ринку, але в цілому довіра клієнтів ґрунтується тільки на історії цього додатка та таланті творців. Телеграм — неповторне явище між технологічних стартапів. І в Телеграм немає ні реклами, ні платного клієнта. У Телеграм також є функція під назвою «Секретний чат», яка спочатку не була включена. Секретні чати доступні у версії Телеграм, наскрізно зашифровані. Повідомлення видаляються через зазначений користувачем час і не можуть бути відновлені. Розробники Телеграм вирішили не вводити наскрізне шифрування для зручності: секретні чати залежать від пристрою, і ви не можете продовжувати розмову там, де розмова не почалася.

Користувачі можуть створювати облікові записи та входити в систему за допомогою SMS-коду доступу. Після первинного входу користувачі можуть входити в налаштування та шукати один одного. Телеграм також має функцію двоетапної перевірки, якщо ви хочете вводити свій пароль щоразу, коли ви входите у свій обліковий запис [4]. Тобто можемо виділити цілий ряд переваг, які виникли за час існування месенджера, а саме: – Конфіденційність даних. Усі чати зашифровані, і повідомлення можна видалити через певний період часу; – Швидкість та зручність у використанні. Завдяки оптимізації месенджер має високу швидкість надсилання та отримання повідомлень; – Відкритість . Має прозорий протокол; – Масовість та загальнодоступність. Сервіс не потребує коштів і не має інтегрованої реклами; – Без обмежень. У Telegram немає чітких вузьких рамок на розмір повідомлень і файлів, що надсилаються. – Систематизація . Сервери Телеграм розгорнуті практично по всьому світу, що підвищує надійність і стабільність;

JavaScript

JavaScript – динамічна, об'єктно орієнтована мова програмування широкого призначення. Реалізація стандарту ECMAScript. Її застосовують для: сценаріїв вебпорталів для надання їм інтерактивності, розробки односторінкових веб-додатків (React, Angular, Vue.js), створення програмного забезпечення на боці сервера 24 (Node.js), мобільних додатків (React Native, Cordova) та інших.

Node.js

Node або Node.js — це програмна платформа, заснована на движку V8 (який перекладає JavaScript у машинний код), який перетворює JavaScript із спеціалізованої мови в мову широкого застосування. Node.js додає змогу для JavaScript не тільки працювати самостійно, але й з пристроями вводу/виводу через свій AP+, написаний на C++, підключати інші зовнішні бібліотеки, реалізовані багатьма мовами, надаючи їм виклики з коду JavaScript. Node.js використовують в основному на сервері, виступаючи як веб-сервер, але його можна створювати на Node.js і віконних додатках робочого столу (за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмні мікроконтролери (наприклад, tessel, low.js і espruino). Node.js заснований на керованому подіями та асинхронному (або реактивному) програмуванні з неблокуючим введеннямвиводом. Node.js містить в собі кілька атрибутів, які роблять його особливо цікавим для мережевого або Інтернет-програмування. Перший пов’язаний з усіма накладними витратами та упаковкою, що використовуються існуючими технологіями для спілкування в Інтернеті.

Давайте уявимо, що ви доставляєте невелику посилку від FedEx і помічаєте всі «контейнери», які ваше відправлення доставить до місця призначення. Там буде вантажівка, яка везе всі пакунки до центру перевірки та розподілення. У цьому центрі будуть великі вантажні трюми, які будуть переміщені в авіаконтейнери, що використовуються для транспортування до центру призначення. І як тільки посилка приходить, відбувається зворотний рух, коли інші посилки відходять у зворотному напрямку. Все це пакування та переупакування є трудомістким і дорогим процесом, і 25 саме це роблять сучасні технології програмування, такі як JSON і REST, для переміщення даних через Інтернет. Node.js значно зменшує цю умовність і надає прості інструменти для того ж завдання. Ще один привабливий атрибут Node.js відноситься до моделі подій вебпрограмування

Більшість існуючих технологій створені для досягнення «великих прогалин» у даних для швидкого запиту та відповіді. Іншими словами, цілу сторінку даних можна відправити на сервер, навіть якщо є лише незначні зміни. Ці технології оптимізовані для використання великих обсягів даних з меншою кількістю подій. Node.js робить навпаки; він розроблений для використання з більшою інтерактивністю – меншими даними, які реагують на багато інших подій.

Бібліотека Mongoose

Mongoose представляє спеціальну бібліотеку моделювання даних об’єктів (ODM) для роботи з MongoDB, яка дозволяє порівнювати об’єкти класів і документи колекції з бази даних. Mongoose — це, можна сказати, фреймворк JavaScript, який зазвичай використовується в Node.js з базою даних MongoDB. У цій статті я познайомлю вас з Mongoose і MongoDB, і, що ще важливіше, де ці технології підходять для вашої програми. Почнемо з MongoDB. MongoDB — це база даних, де ваші дані зберігаються як документи. Найчастіше ці документи виглядають як JSON-подібна структура:

```
{
ім'я: "Джеймі",
Ім'я: "Мунро"
}
```

Потім документ поміщається в колекцію. Як приклад, зразок документа вище визначає об’єкт користувача. Одним із ключових факторів MongoDB є його структурна гнучкість. Хоча в першому прикладі об’єкт користувача містить властивості firstName та lastName, ці властивості не є обов’язковими в кожному документі користувача, який є частиною колекції користувачів. Це та різниця між MongoDB і бази даних SQL, такої як MySQL або Microsoft SQL Server, яка вимагає строго визначеної схеми бази даних для всіх об’єктів, які він зберігає. Можливість створювати динамічні об’єкти, що зберігаються як документи в БД – ось тут на допомогу приходить Mongoose. Mongoose — програма для створення об’єктних документів (ODM). Це означає, що Mongoose дозволяє визначати об’єкти за допомогою строго типізованої схеми, яка відображається в документі MongoDB. Mongoose надає неймовірну кількість функцій для створення та роботи зі схемами. Наразі Mongoose містить вісім типів схем, які реєструються для властивості, коли вона збережена в MongoDB.

Кожен тип дає змогу вказати: – стандартні значення; – користувацька функція перевірки; – вкажіть необхідне поле; – функція get, яка надає змогу робити маніпуляції з даними до того, як вони повернуться як об’єкт; – функція набору, яка дає можливість робити маніпуляції з даними перед їх збереженням у БД; – створювати індекси, які надають змогу миттєво отримувати дані.

1.3. Створення та основні задачі чат-боту.

Створення чат-боту

Щоб розпочати розробку нашого боту потрібно написати @BotFather . BotFather - це такий асистент в месенджері, що допомагає створювати та налаштовувати власних помічників, а також керувати ними. Можна навіть сказати, що це такий тато для всіх ботів Телеграму. Тепер будемо дотримуватись деяких правил:

1. Спершу введемо команда /newbot. Вона допоможе конкретно створити помічника.

2. Далі BotFather відправить вам повідомлення з проханням ввести ім’я боту. Воно не повинно повторювати жодного вже існуючого імені.

3. Пам’ятаємо про правило для створення ботів. Ім’я обов’язково закінчується на «…_bot» або «..Bot». Про це нагадує нам і сам BotFather.

4. Також ми маємо можливість одразу коротко описати нашу програму. Цю візитівку зможе побачити кожен новий співрозмовник і одразу зрозуміти, чи допоможе йому наш бот у вирішенні своїх питань. Також можна додати фото і налаштувати деякі інші можливості.

5. Тепер «тато-бот» відправить нам унікальний ідентифікатор нашого боту - токен. Він складається з різних символів, у тому числі цифр та букв латинського алфавіту. Без нього неможливо продовжувати розробку, тому що програма просто не знатиме, як зв’язатись з сервером. Далі цей ключ треба буде указати в коді.

6. Тепер нам треба зберегти токен в файлі, який не можна ні в якому разі загубити або ділитись з іншими. Якщо це все ж станеться, то доступ до вашого помічника зможе отримати будь-хто інший.

7. Тож, на основі минулого пункту, сформуємо правило : «Не передавайте токен в чужі руки!».

BotFather містить дуже багато команд для налаштування , проте ми розглянемо тільки деякі з них. Конкретно ті, які знадобились нам для розробки: – /setname - нове ім'я боту; – /setdesctiption - опис боту; – /setabouttext - корегувати вже існуючі дані; – /setuserpic - корегувати профіль боту; – /setcommands - список команд; – /token - створює новий унікальний ключ; – /setinline - увімкнути або вимкнути можливість виклику бота з інших чатів; – /setinlinegeo - дозвіл або заборона боту робити запит отримати місцезнаходження; – /setjoingroups - можливість запрошувати бота в інші розмови;

Ці кроки допомогли створити та налаштувати бота. Коли всі ці дії будуть виконані, BotFather сповістить про успішне створення бота. Для того, щоб навчити його виконувати будь-яку дію, треба написати відповідний програмний код.

Основні задачі чат-боту

Програмний продукт багатофункціональний та має багато можливостей для розширення. Головне питання, яке ми поставили перед собою у реалізації програмного продукту - вирішення проблеми пошуку актуальних фільмів та відповідно кінотеатрів, де вони транслюються. Основні функції і задачі, які виконує чат-бот для пошуку фільмів:

– Надсилання повідомлення-привітання користувачу.

– За запитом здійснюється пошук фільмів за жанрами.

– Надсилання результатів пошуку користувачу з відповідним фільмом;

– Надсилання всіх доступних даних про фільм після того, як користувач зробить свій вибір;

Отже, ми визначили основні задачі боту, завдяки виконанню яких, ми отримаємо повноцінну програму, яка вирішує велику проблему кіноманів.

1.4. Огляд Telegram-ботів для рекомендації фільмів

На сучасному етапі розвитку цифрових технологій Telegram-боти займають важливе місце у сфері надання інформаційних послуг. Одним із напрямів їхнього застосування є формування персоналізованих рекомендацій фільмів. Такий функціонал дозволяє користувачам швидко знаходити контент, що відповідає їхнім інтересам, без потреби в тривалому пошуку на сторонніх ресурсах [1], [2].

На ринку Telegram-ботів уже представлено декілька успішних рішень, серед яких варто виділити **IMDb Bot** [15] та **MovieMate Bot** [14].

IMDb Bot надає можливість здійснювати пошук фільмів за назвою, переглядати рейтинги, рецензії, трейлери та базову інформацію про акторів і знімальні групи, використовуючи ресурси IMDb API [9]. Основною перевагою бота є швидкий доступ до великої та авторитетної бази даних, що забезпечує високу інформативність результатів.

MovieMate Bot, у свою чергу, орієнтований саме на рекомендації. Він аналізує смаки користувача (шляхом вибору жанрів або оцінювання фільмів) і надає персоналізовані підбірки. Крім того, бот враховує популярність фільмів серед інших користувачів, формуючи рекомендації на основі загальних тенденцій [14].

Незважаючи на свої переваги, такі боти здебільшого мають обмежений функціонал у плані візуального оформлення. Вони зазвичай використовують класичний текстовий інтерфейс і стандартні кнопки Telegram, що робить взаємодію менш привабливою для сучасного користувача [3], [6]. Відсутність інтерактивних візуальних елементів може знижувати рівень залученості, особливо серед молодіжної аудиторії.

Окремо слід розглянути досвід стрімінгових сервісів, таких як **Netflix** або **Taste.io**, які успішно впроваджують алгоритми машинного навчання для формування рекомендацій [10], [13]. Ці системи враховують історію переглядів, рейтинг, поведінкові патерни, а також кластеризацію користувачів за інтересами. Проте застосування подібних алгоритмів у рамках Telegram-ботів має низку складностей — як технічного (складність реалізації ML-моделей), так і юридичного характеру (ліцензійні обмеження API стрімінгів) [7], [10].

Додатково варто згадати потенціал використання тривимірної візуалізації (3D-інтерфейсу) в середовищі Telegram Web Apps, що може значно покращити користувацький досвід. Сучасні технології на базі бібліотеки **Three.js** [12] дозволяють створити інтерактивні елементи навіть у межах чат-бота. Попередні дослідження демонструють ефективність таких рішень для візуалізації великих масивів даних [11].

Таким чином, аналіз існуючих рішень показує, що:

- Вже реалізовані Telegram-боти можуть ефективно шукати фільми та надавати загальні рекомендації;
- Існує потенціал удосконалення персоналізації та візуального оформлення;

- Застосування сучасних вебтехнологій (наприклад, інтеграція з TMDb API [7], YouTube API [8], MongoDB [5], Node.js [4]) дозволяє розширити функціональні можливості бота і вийти за межі стандартного досвіду взаємодії.

Отже, є потреба у створенні нового Telegram-бота, який поєднував би:

- простоту використання і швидкодію месенджера;
- глибоку персоналізацію рекомендацій;
- сучасний, візуально привабливий інтерфейс, включаючи 3D-елементи.

Це дозволить не лише задовольнити поточні потреби кіноманів, а й створити інноваційний продукт, що вирізнятиметься серед конкурентів.

1.5. Функціональні можливості стрімінгових сервісів

Сучасні стрімінгові сервіси, зокрема **Netflix**, **Amazon Prime Video**, **Hulu**, а також менш відомі інноваційні платформи, як-от **Taste.io**, відіграють важливу роль у формуванні користувацького досвіду в галузі перегляду фільмів та серіалів. Їхня функціональність значно виходить за межі простого перегляду відеоконтенту, охоплюючи гнучкі рекомендаційні системи, персоналізацію, інтерактивні елементи та мультіплатформний доступ [10], [13].

До ключових функціональних можливостей сучасних стрімінгових платформ належать:

- **Персоналізовані рекомендації.** Сервіси використовують алгоритми машинного навчання для аналізу історії переглядів, оцінок, жанрових уподобань і поведінкових моделей користувача. Крім того, враховується активність схожих за інтересами осіб, що дозволяє формувати максимально релевантні підбірки контенту [2], [10], [13]. Наприклад, Netflix застосовує гібридну модель, яка поєднує колаборативну фільтрацію з контентним аналізом, щоб удосконалити точність рекомендацій [10].
- **Системи фільтрації контенту.** Користувачі мають змогу сортувати фільми за жанрами, настроєм, тривалістю, рейтинговими оцінками (наприклад, за шкалою IMDb) або новизною. Це суттєво спрощує навігацію у великому обсязі доступного контенту [9], [14].

- **Інтерактивні елементи.** Більшість сервісів надають змогу додавати фільми до списку "Подивитись пізніше", залишати власні рецензії, ставити оцінки, а також отримувати сповіщення про нові релізи, ґрунтуючись на персональних інтересах [13].
- **Мультіплатформність.** Стрімінгові сервіси оптимізовані для різних пристроїв — смартфонів, планшетів, комп’ютерів, Smart TV тощо. Це забезпечує гнучкість і доступ до улюбленого контенту з будь-якого місця та в будь-який час [10].
- **Мовна підтримка та субтитри.** Платформи, орієнтовані на міжнародну аудиторію, надають можливість вибору мови інтерфейсу, озвучення та субтитрів, що суттєво підвищує інклюзивність [10].
- **Висока якість відео.** Підтримка форматів HD, Full HD, 4K, а також адаптивного стрімінгу, який автоматично підлаштовується під швидкість інтернет-з’єднання, гарантує якісне зображення за різних умов перегляду [10].

Попри численні переваги, ці сервіси мають обмеження у контексті інтеграції з месенджерами та Telegram-ботами. Насамперед це пов’язано з ліцензійною закритістю API стрімінгових гігантів, а також високими вимогами до обробки даних у режимі реального часу [7], [10]. Наприклад, Netflix API більше не є відкритим для сторонніх розробників, що унеможливлює безпосередній доступ до рекомендаційної логіки платформи.

Однак стрімінгові сервіси залишаються важливими орієнтирами для розробників, які працюють над створенням Telegram-ботів. Досвід цих платформ може бути використаний для формування ефективних, гнучких і привабливих систем рекомендацій. Розробники можуть впроваджувати спрощені моделі персоналізації, побудовані на відкритих базах даних, таких як **TMDb API** [7] або **IMDb API** [9], адаптуючи функціональність під обмеження Telegram-екосистеми.

Таким чином, вивчення функціоналу провідних стрімінгових платформ дозволяє зрозуміти сучасні очікування користувачів і сформувати технічне бачення майбутнього продукту, що об'єднає інтуїтивність Telegram-бота з можливостями рекомендованих систем.

1.6. Обґрунтування вибору платформи Telegram, мови програмування та бази даних

Вибір технологічного стека є критично важливим на етапі розробки будь-якого програмного рішення, оскільки саме від нього залежить подальша ефективність роботи, можливості масштабування та зручність користувацької взаємодії. У межах реалізації Telegram-бота для рекомендації фільмів було обрано платформу **Telegram**, мову програмування **JavaScript (Node.js)**, а також базу даних **MongoDB**.

Вибір платформи Telegram

Telegram є одним із найбільш відкритих та функціональних месенджерів для створення ботів. Основними перевагами Telegram як платформи є:

- **Простий у використанні API**, який дозволяє швидко інтегрувати ботів із зовнішніми сервісами [6].
- **Можливість використання Web Apps**, що дозволяє реалізувати розширений інтерфейс прямо в Telegram, зокрема, додавати інтерактивні елементи, такі як 3D-візуалізація [11].
- **Велика аудиторія користувачів**, у тому числі активна спільнота кіноманів, які шукають рекомендації безпосередньо в месенджері [14], [15].

Завдяки цим перевагам Telegram є ідеальною платформою для реалізації чат-бота, який поєднує доступність, швидкодію та сучасний інтерфейс.

Вибір мови програмування — JavaScript та середовища Node.js

Для розробки бота було обрано мову JavaScript у середовищі Node.js. Такий вибір зумовлений кількома факторами:

- **Асинхронна природа Node.js** забезпечує високу продуктивність при обробці запитів Telegram API та зовнішніх джерел даних, таких як TMDb або YouTube [4], [8].
- **Широка екосистема бібліотек**, наприклад, node-telegram-bot-api, axios, mongoose, спрощує розробку логіки бота та роботу з базою даних [3].

- **Простота розгортання та масштабування**, що дозволяє адаптувати проєкт як для локального використання, так і для хмарних платформ.

Крім того, JavaScript ідеально підходить для розробки інтерфейсів у Web Apps з використанням бібліотек для 3D-графіки, зокрема **Three.js** [12].

Вибір бази даних — MongoDB

MongoDB є популярною **документно-орієнтованою базою даних**, яка забезпечує гнучкість структури зберігання даних. Її переваги у контексті цього проєкту включають:

- **Можливість зберігання динамічних профілів користувачів**, включно з вподобаннями, історією перегляду, збереженими фільмами та рейтингами.
- **Гнучка схема даних**, яка дозволяє легко змінювати структуру бази без необхідності складного рефакторингу [5].
- **Масштабованість і висока продуктивність**, що важливо для обробки великої кількості запитів у реальному часі [5].

MongoDB чудово інтегрується з Node.js через драйвер mongoose, що спрощує моделювання структури даних та виконання CRUD-операцій.

1.4. Вибір сторонніх API для отримання інформації про фільми

Для забезпечення широкого функціоналу та актуального контенту в Telegram-боті *Bot_film*, було прийнято рішення інтегрувати **сторонні програмні інтерфейси (API)**, які надають доступ до професійних баз даних фільмів, відео та метаданих. Використання API дозволяє не лише отримувати структуровану інформацію в режимі реального часу, але й динамічно адаптувати бот під нові запити користувачів.

Критерії вибору API

Основними критеріями, які визначали доцільність використання певного API, стали:

- **Обсяг та повнота інформації**, включно з даними про акторів, описами, жанрами, рейтингами, трейлерами тощо.
- **Надійність джерела**, авторитетність у кіногалузі.

- **Зручність інтеграції**, наявність документації, формат відповіді JSON, підтримка HTTP/HTTPS-запитів.
- **Додаткові можливості**, такі як пошук за ключовими словами, категоріями, мовами, а також вивантаження зображень і відео.
- **Безкоштовний доступ із відкритим API-ключем або доступними тарифними планами**.

Обрані API

TMDb (The Movie Database) API

TMDb є однією з найбільш повних та відкритих баз даних фільмів і телесеріалів. Вона містить:

- детальну інформацію про фільми, серіали, акторів;
- рейтинг користувачів;
- офіційні постери, зображення та трейлери;
- можливість здійснювати пошук і фільтрацію за мовою, жанром, роком випуску тощо [7].

Перевагою TMDb є також активне оновлення контенту спільнотою, що забезпечує актуальність даних. Це API ідеально підходить для створення персоналізованих рекомендацій, візуального контенту та побудови гнучкого пошуку.

IMDb API

IMDb API використовується для доповнення інформації із TMDb. IMDb — це одна з найбільш авторитетних баз даних у світі кіноіндустрії, яка містить:

- перевірені рецензії;
- рейтинги, що формуються на основі великої кількості голосів;
- детальну фільмографію акторів і режисерів [9].

Завдяки авторитетності платформи, інтеграція IMDb API підвищує довіру користувачів до контенту, що рекомендується ботом.

Висновок

Комбінування **двох API** — TMDb, IMDb — забезпечує багатовимірний підхід до подачі інформації. Така інтеграція дозволяє:

- покривати як **структуровану метаінформацію** (TMDb, IMDb), так і **мультимедійний контент** (YouTube);
- підвищити точність та повноту рекомендацій;
- створити **більш інформативний і захоплюючий досвід** взаємодії користувача з ботом.

Таким чином, використання сторонніх API є **ключовим технологічним рішенням**, що дозволяє підняти функціональність Telegram-бота на рівень сучасних стрімінгових платформ [2], [7], [8], [9], [10].

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА

2.1. Створення зручного інтерфейсу бота з використанням Telegram Web Apps, кнопок і меню

Одним із ключових факторів ефективного функціонування Telegram-бота *Bot_film* є наявність **інтуїтивно зрозумілого, зручного та естетично привабливого інтерфейсу**, що забезпечує комфортну взаємодію користувачів із системою. З цією метою в реалізації було використано **сучасні можливості Telegram**, зокрема:

- **Telegram Web Apps** — для побудови розширених інтерфейсів із підтримкою HTML/CSS/JavaScript;
- **Inline-кнопки, reply-кнопки та меню** — для структурування діалогу та швидкої навігації по функціоналу бота.

Telegram Web Apps як засіб розширення інтерфейсу

Telegram Web Apps — це вбудовані вебзастосунки, які відкриваються всередині Telegram-клієнта та забезпечують повноцінну інтерактивну взаємодію з користувачем. За допомогою цієї технології у *Bot_film* реалізовано:

- інтерактивний каталог фільмів;
- візуалізацію постерів, оцінок та описів у єдиному вікні;
- можливість запуску трейлерів без переходу на зовнішні сайти;
- адаптивний дизайн для мобільних пристроїв.

Крім того, Web Apps надає можливість у майбутньому реалізувати **3D-інтерфейси чи віртуальні простори**, де користувач зможе «прогулятися» віртуальним кінотеатром, переглядаючи афіші фільмів (реалізація на базі WebGL/Three.js).

Кнопки та меню для швидкої навігації

Використання стандартних елементів Telegram API, таких як **Inline-кнопки** (інтерактивні кнопки під повідомленням), **Reply-кнопки** (фіксовані елементи клавіатури внизу чату) та **вбудоване меню бота**, дозволяє:

- швидко переходити до основних функцій: «🎬 Отримати рекомендацію», «⚙ Обрати жанр»;
- скорочувати кількість необхідного текстового введення;
- уникати помилок у синтаксисі запитів;
- спрямовувати користувача по заздалегідь визначених сценаріях (діалогових шаблонах).

Кожна кнопка викликає певну функцію бота, зокрема, надсилає дані на сервер або відкриває Web App із вибраною тематикою чи добіркою.

Висновок

Застосування Telegram Web Apps у поєднанні з класичними елементами управління дозволяє створити **збалансований інтерфейс**, що відповідає вимогам сучасних користувачів до простоти, швидкості та візуального комфорту. Такий підхід забезпечує **високу ефективність навігації**, знижує навантаження на користувача та підвищує загальний рівень взаємодії з ботом.

2.2. Розробка алгоритму персоналізованих рекомендацій

Ефективна система рекомендацій є ключовим елементом сучасного стрімінгового сервісу або бота, орієнтованого на індивідуальні вподобання користувачів. У Telegram-боті *Bot_film* реалізовано персоналізований алгоритм, який формує релевантні добірки фільмів на основі аналізу користувацьких дій і контенту.

Основні джерела даних для аналізу:

- історія переглядів фільмів через бот;
- виставлені користувачем оцінки (like/dislike);
- обрані жанри й ключові тематики;
- час взаємодії з контентом (утримання уваги);
- реакції на запропоновані добірки (перехід, перегляд трейлерів, додавання до «обраного»).

Підхід до побудови системи рекомендацій

Алгоритм поєднує **два основні методи фільтрації**:

1. **Колаборативна фільтрація (Collaborative Filtering)**

2. Використовується для виявлення схожих користувачів та їхніх вподобань. Якщо кілька користувачів мають подібні оцінки для одних і тих самих фільмів, система припускає, що інші вподобання цих користувачів також можуть бути релевантними. Таким чином, нові фільми пропонуються на основі «схожої поведінки інших».

3. **Контентна фільтрація (Content-Based Filtering)**

4. Застосовується для аналізу безпосередніх характеристик фільмів — жанр, режисер, акторський склад, рік випуску, країна, ключові слова тощо. На основі цього аналізу формується профіль користувача (вектор жанрових уподобань), і кожен новий фільм оцінюється на предмет відповідності цьому профілю.

Додаткові параметри:

- **Актуальні тренди** (на основі TMDb/IMDb API): фільми, що наразі у тренді, мають вищу вагу в рейтингу рекомендацій.
- **Часовий фільтр**: користувачам можуть пропонуватися новинки за останні 1–2 роки.
- **Сезонні добірки**: наприклад, тематичні фільми до свят, новорічні комедії, літні блокбастери тощо.

Реалізація та технічна оптимізація

Алгоритм реалізовано у середовищі **Node.js**, з використанням бази даних **MongoDB** для зберігання історії користувачів і результатів обробки. MongoDB дозволяє динамічно зберігати профілі вподобань і швидко здійснювати запити для пошуку відповідних фільмів.

Додатково, інформація про фільми отримується з **зовнішніх API (TMDb, IMDb)**, що забезпечує доступ до актуальних метаданих, включно з жанрами, постерами, рейтингами, описами та трейлерами.

Висновок

Розроблений алгоритм персоналізованих рекомендацій поєднує **гнучкість**, **адаптивність** та **швидкодію**, забезпечуючи користувачам *Bot_film* релевантний та сучасний контент без потреби у складних запитах. Це підвищує залучення аудиторії та задоволення від користування ботом.

2.3. Забезпечення збереження вподобань користувачів і історії переглядів

Персоналізований підхід до формування рекомендацій вимагає зберігання та обробки даних про кожного користувача, що взаємодіє з Telegram-ботом *Bot_film*. Одним із ключових компонентів системи є надійна база даних, яка дозволяє зберігати вподобання, історію переглядів і динаміку змін інтересів користувачів.

Структура даних користувача

Для кожного користувача в базі даних **MongoDB** створюється окремий документ, який містить такі поля:

- **user_id** — унікальний Telegram-ідентифікатор користувача;
- **movies** — масив фільмів;
- **preferred_genres** — динамічно сформований список улюблених жанрів;

Технологія зберігання та обробки

База даних MongoDB обрана завдяки своїй:

- **гнучкій схемі**, що дозволяє легко змінювати або розширювати формат даних без потреби у складних міграціях;
- **високій продуктивності** при обробці великих обсягів неструктурованої або слабо структурованої інформації;
- **масштабованості**, що дозволяє підтримувати зростання кількості користувачів без втрати продуктивності.

Збереження здійснюється через Node.js-сервер, який опрацьовує запити користувачів і здійснює CRUD-операції (Create, Read, Update, Delete) з базою даних у режимі реального часу.

2.4. Архітектура системи, налаштування бази даних і інтеграція з API

Архітектура Telegram-бота *Bot_film* побудована за **модульною багаторівневою структурою**, що забезпечує **масштабованість**, **гнучкість у розширенні функціоналу** та **ефективне обслуговування запитів користувачів**. Такий підхід дозволяє ізольовано оновлювати окремі модулі без порушення загальної роботи системи.

Основні компоненти архітектури:

1. **Клієнтська частина (Telegram):**
   - Взаємодія з користувачами здійснюється через Telegram-інтерфейс.
   - Використовуються стандартні кнопки (Inline/Reply), меню, а також **Telegram Web Apps** для відображення візуального контенту.
   - Підтримується обробка повідомлень, callback-запитів та Web App-подій.
2. **Серверна частина (Node.js):**
   - Реалізована за допомогою фреймворку **Express.js**.
   - Виконує логіку обробки запитів, формує відповіді, керує сесіями користувачів.
   - Організовано окремі модулі для:
   - • взаємодії з Telegram API;
   - • обробки запитів до бази даних;
   - • викликів зовнішніх API;
   - • обчислення рекомендацій.
3. **База даних (MongoDB):**
   - Документно-орієнтована модель зберігання.
   - Використовуються колекції: users, movies, history, genres, sessions.
   - Реалізована індексація за user_id, movie_id, genre для пришвидшення пошуку.
4. **Зовнішні API:**

- **TMDb API** – основне джерело інформації про фільми (описи, актори, постери).
- **IMDb API** – додаткове джерело рейтингів і рецензій.

Взаємодія компонентів (загальна схема):

1. Користувач надсилає запит → повідомлення обробляється Telegram API.
2. Сервер (Node.js) отримує запит через Webhook, аналізує його зміст.
3. Якщо потрібна інформація про фільм — здійснюється виклик відповідного API (TMDb, IMDb або YouTube).
4. Отримані дані зберігаються або оновлюються в MongoDB.
5. Бот надсилає користувачеві відповідь у зручному форматі (повідомлення, кнопки, Web App).

Приклад структури документа в MongoDB (колекція users):

```
{
 "user_id": 123456789,
 "username": "cinemalover",
 "liked_movies": [550, 238, 680],
 "watched_history": [
 { "movie_id": 550, "date": "2025-04-25", "rating": 9 }
 ],
 "preferred_genres": ["Drama", "Thriller"],
 "last_active": "2025-05-26T10:34:00Z"
}
```

Особливості інтеграції з API:

- **REST-запити** виконуються за допомогою бібліотеки axios (або fetch) з обробкою помилок.
- **Асинхронна обробка запитів** (async/await) дозволяє уникати блокування потоків.
- Для кожного API реалізовано окремий модуль (наприклад, tmdbService.js, imdbService.js), що спрощує підтримку та тестування.

Переваги обраної архітектури:

- **Гнучкість** — легко додавати нові функції (наприклад, рекомендації за настрієм або часом доби).
- **Масштабованість** — можлива міграція на мікросервісну структуру при зростанні навантаження.
- **Розширюваність** — підтримка нових джерел даних (інших API, AI-модулів).

У результаті реалізована архітектура забезпечує **стабільну роботу системи**, **високу швидкодію** та **можливість персоналізованої взаємодії з кожним користувачем**. Поєднання Telegram, Node.js, MongoDB та зовнішніх API створює технічно цілісну та зручну платформу для кіноманів.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

3.1. Тестування роботи бота на різних пристроях і платформах

Для забезпечення стабільної та ефективної роботи Telegram-бота *Bot_film* було проведено комплексне тестування його функціоналу на різноманітних пристроях та програмних середовищах. Метою тестування стало виявлення можливих помилок, перевірка сумісності з платформами Telegram та забезпечення однаково високої якості користувацького досвіду незалежно від типу пристрою.

Тестовані платформи:

- **Мобільні пристрої** (Android, iOS) – через офіційні мобільні додатки Telegram;
- **Планшети** – тестування адаптації інтерфейсу до більших екранів;
- **Персональні комп'ютери** – через Telegram Desktop;
- **Telegram Web** – перевірка Web App-функціональності, включаючи інтерактивні компоненти та 3D-інтерфейс.

Основні етапи тестування:

1. **Функціональне тестування**:

- Перевірка запуску та коректної відповіді бота на всі типи команд.
- Тестування кнопок, меню, inline-інтерфейсу.
- Робота системи рекомендацій на основі історії переглядів та вподобань.

2. **Інтерфейсне тестування (UI/UX)**:

- Перевірка візуального відображення повідомлень, кнопок та Web App-елементів.
- Адаптивність інтерфейсу до різних розмірів екранів.

3. **Тестування продуктивності**:

- Оцінка швидкості реакції бота на запити.
- Тестування навантаження при одночасній взаємодії з кількома користувачами.

4. **Тестування стабільності**:

- Повторювані дії користувача (перезапуск команд, повторний запит інформації).
- Перевірка роботи при нестабільному інтернет-з'єднанні.

5. **Тестування безпеки**:

- Валідація введених користувачем даних.
- Перевірка обробки некоректних або підозрілих запитів.
- Обмеження доступу до технічних даних та внутрішніх маршрутів API.

Результати тестування:

- Бот **успішно функціонує** на всіх перевірених платформах.
- **Інтерфейс Web App** коректно відображається як у мобільному Telegram, так і в десктопній версії.
- Всі ключові функції — отримання рекомендацій, перегляд опису фільмів, збереження вподобань — працюють **без збоїв**.
- Виявлені незначні помилки (наприклад, некоректне відображення кнопок у старій версії Telegram Desktop) були **усунені в процесі оптимізації**.

Висновок:

Проведене тестування підтвердило **стабільну, кросплатформену роботу Telegram-бота "Bot_film"**. Реалізовані функції повністю відповідають поставленим цілям, а інтерфейс залишається інтуїтивно зрозумілим і зручним для користувачів незалежно від типу пристрою. Це створює **універсальний інструмент для рекомендацій фільмів**, доступний широкій аудиторії кіноманів.

3.2. Виявлення та усунення помилок у логіці, інтерфейсі та візуалізації

У процесі тестування та вдосконалення Telegram-бота *Bot_film* було виявлено низку недоліків, пов’язаних із логікою обробки запитів, відображенням інтерфейсних елементів на різних пристроях, а також із коректністю візуалізації інтерактивних компонентів, зокрема 3D-графіки у Telegram Web Apps.

Основні типи помилок, що були виявлені:

1. **Логічні помилки**:
   - Неправильне оброблення запитів користувача за умов неповних або нетипових даних.
   - Некоректне формування списків рекомендацій у випадку відсутності достатньої кількості даних про вподобання.
   - Подвійна або повторна рекомендація вже переглянутих фільмів.

2. **Інтерфейсні помилки (UI/UX)**:
   - Неправильне відображення кнопок або елементів меню у певних версіях мобільного Telegram.
   - Перенасичення інтерфейсу елементами при малому розмірі екрану.
   - Невідповідність стилів між мобільною та десктопною версією Telegram Web.
3. **Помилки візуалізації та продуктивності**:
   - Збої у відображенні інтерактивної 3D-графіки в Telegram Web Apps при повільному інтернет-з’єднанні.
   - Затримки при рендерингу каталогів фільмів через великі обсяги зображень.
   - Неправильне масштабування вмісту на планшетах або екранах із високою щільністю пікселів.

Проведені заходи з усунення помилок:

1. **Оптимізація логіки обробки запитів**:
   - Вдосконалено механізм обробки неповних або неоднозначних запитів.
   - Реалізовано фільтрацію та перевірку дублікатів у рекомендаціях.
   - Додано механізм fallback-повідомлень з пропозицією уточнити запит.
2. **Поліпшення інтерфейсу**:
   - Адаптовано кнопки та меню під різні типи пристроїв, використано адаптивну верстку.
   - Оптимізовано дизайн інтерфейсу для кращої читабельності та навігації.
   - Уніфіковано стилі для всіх платформ Telegram.
3. **Оптимізація 3D-візуалізації**:
   - Скорочено обсяг графічних ресурсів, реалізовано завантаження за потребою (lazy loading).
   - Підвищено продуктивність через використання WebGL-бібліотек з адаптивною продуктивністю.
   - Додано перевірку сумісності браузера перед запуском візуалізації.

Результат:

Після внесення змін та повторного циклу тестування **було забезпечено стабільну, плавну та інтуїтивно зрозумілу роботу бота на всіх типах пристроїв**. Значно покращено користувацький досвід, знижено кількість помилок при взаємодії, а інтерфейс став доступним і зручним для широкого кола користувачів, включно з тими, хто вперше використовує подібні системи.

3.3. Аналіз відповідності функціоналу цілям і очікуванням користувачів

Після завершення розробки, інтеграції та тестування Telegram-бота *Bot_film* було проведено аналіз ступеня відповідності реалізованого функціоналу початковим цілям проєкту, а також очікуванням кінцевих користувачів. Основною метою було створення зручного, інформативного та персоналізованого інструмента для пошуку фільмів і отримання рекомендацій у режимі реального часу.

Для об’єктивної оцінки функціоналу було проведено:

- анкетування серед цільової аудиторії (кіноманів віком від 16 до 35 років),
- збір зворотного зв’язку через інтерфейс бота,
- аналіз поведінки користувачів на основі анонімізованих логів взаємодії.

Основні висновки за результатами аналізу:

1. **Відповідність цілям проєкту**:
   - Усі ключові завдання, поставлені на етапі проєктування (надання рекомендацій, інтеграція з базами даних фільмів, підтримка Web Apps та 3D-інтерфейсу), були успішно реалізовані.
   - Система забезпечує **персоналізований підхід** до кожного користувача, ґрунтуючись на його вподобаннях та історії переглядів.
2. **Оцінка зручності інтерфейсу**:
   - Більшість користувачів позитивно оцінили **зрозумілу навігацію** через кнопки й меню.
   - Інтерактивний 3D-інтерфейс отримав високу оцінку серед користувачів, які використовують Telegram Web Apps або десктопну версію Telegram.

3. **Якість рекомендацій**:

- Рекомендації визнано релевантними, з достатньою глибиною добору фільмів навіть за мінімального обсягу даних про вподобання.
- Окремо відзначено **актуальність** інформації завдяки інтеграції з TMDb, IMDb та YouTube API.

4. **Виявлені побажання та пропозиції**:

- Деякі користувачі висловили побажання щодо **розширення списку жанрів**, зокрема додавання піджанрів, авторського кіно та аніме.
- Була озвучена ідея додати функціонал створення "власного списку до перегляду", а також **сповіщення про новинки** за улюбленими жанрами.

Загальна оцінка:

Функціонал Telegram-бота *Bot_film* повністю відповідає поставленим цілям проєкту. Бот забезпечує **інформативний, персоналізований та візуально привабливий** досвід взаємодії з кіноконтентом. Високий рівень користувацького задоволення та активна взаємодія з ботом підтверджують ефективність обраних технічних і дизайнерських рішень. Зібрані побажання будуть враховані в наступних ітераціях розвитку системи.

ВИСНОВКИ

У результаті виконання проєкту з розробки Telegram-бота "Bot_film" було досягнуто поставлених цілей та виконано всі заплановані завдання. Основною метою проєкту було створення функціонального, зручного та сучасного інструменту для рекомендації фільмів у форматі месенджер-бота, що дозволяє користувачеві швидко знаходити фільми за власними вподобаннями, спираючись на персоналізовані алгоритми. Розроблений бот враховує інтереси користувача, реалізує інтеграцію з зовнішніми джерелами інформації, має привабливий інтерфейс і є доступним у використанні на будь-якому пристрої з Telegram.

Перш за все, у процесі роботи над проєктом було проведено глибокий аналіз сучасного ринку Telegram-ботів, орієнтованих на рекомендацію фільмів. Було виявлено, що більшість таких ботів мають вузький функціонал і не використовують сучасні інтерфейсні рішення, що негативно впливає на користувацький досвід. Як правило, рекомендації в подібних ботах обмежуються випадковим вибором або загальними добірками за жанрами, без урахування особистих вподобань користувачів. Деякі боти взагалі не містять інтеграції з перевіреними джерелами інформації про фільми, що знижує точність та актуальність видачі.

Зважаючи на ці недоліки, було сформульовано завдання створити Telegram-бота нового типу — такого, що забезпечуватиме якісну персоналізацію, зручність у використанні, візуальну привабливість та інформативність. Актуальність цього завдання обґрунтовується високим попитом на подібні сервіси серед користувачів Telegram, зростанням споживання відеоконтенту через стрімінгові платформи, а також розвитком технологій автоматизованих рекомендацій.

Для реалізації проєкту було обрано платформу Telegram, яка надає широкі можливості для створення ботів, включно з інтеграцією через Web Apps. Основною мовою програмування став Node.js — одна з найпопулярніших платформ для розробки серверної логіки. Вона забезпечує стабільну роботу в

режимі реального часу, високу продуктивність і зручність при інтеграції з API сторонніх сервісів. Для зберігання даних про користувачів, їхні вподобання та історію взаємодії було використано нереляційну базу даних MongoDB. Такий вибір зумовлений необхідністю гнучкої структури зберігання інформації та швидкої обробки запитів.

Однією з ключових особливостей бота є реалізація алгоритму персоналізованих рекомендацій. На основі аналізу жанрових уподобань, історії вибраних фільмів, частоти запитів та інших факторів бот формує списки фільмів, що найбільш відповідають інтересам конкретного користувача. Для цього було реалізовано комбінацію контентного фільтрування та базових методів колаборативної фільтрації. Такий підхід дозволяє враховувати як об’єктивні параметри фільмів, так і поведінкові патерни користувачів, що значно підвищує точність рекомендацій.

Крім того, важливим елементом стало створення зручного, інтуїтивно зрозумілого інтерфейсу. Telegram-бот підтримує меню, швидкі кнопки, можливість перегляду фільмів у форматі 3D-каталогу завдяки Telegram Web Apps. Це дозволяє створити ефект повноцінного додатку всередині месенджера, де користувач може вільно обирати фільми, дізнаватись про них деталі та переглядати трейлери.

Інтеграція з зовнішніми API, зокрема TMDb (The Movie Database) та IMDb, дозволила отримати повну й актуальну інформацію про фільми: назви, описи, рейтинги, постери, рік випуску, акторський склад тощо. Це значно підвищує якість взаємодії, дозволяючи користувачу отримувати інформацію миттєво та з достовірних джерел. Зі сторони реалізації така інтеграція дозволила автоматизувати процес оновлення бази даних без необхідності ручного додавання нових фільмів.

Особливу увагу в процесі розробки було приділено тестуванню. Бот проходив функціональне, модульне та інтеграційне тестування на різних пристроях, операційних системах та версіях Telegram. У процесі тестування було виявлено низку недоліків, зокрема проблеми з некоректним відображенням деяких

елементів у Web Apps, затримки при обробці запитів за повільного інтернет-з'єднання, а також неповна обробка деяких виняткових ситуацій. Усі виявлені недоліки були усунуті, після чого бот продемонстрував стабільну роботу, швидкий відгук та коректне відображення на всіх тестованих пристроях.

Результати роботи свідчать про доцільність створення Telegram-бота "Bot_film" як зручного сервісу для користувачів, які шукають фільми для перегляду. Бот має низку переваг над аналогами, серед яких:

- персоналізовані рекомендації, що підвищують релевантність пропозицій;
- сучасний 3D-інтерфейс, що створює новий рівень користувацького досвіду;
- інтеграція з провідними джерелами інформації про фільми;
- доступність та зручність використання в Telegram;
- підтримка швидкого реагування завдяки Node.js та MongoDB.

Узагальнюючи отримані результати, можна зазначити, що створений продукт повністю відповідає сучасним вимогам до зручності, функціональності та дизайну цифрових сервісів. "Bot_film" є не лише ефективним інструментом для пошуку фільмів, але й може слугувати прикладом якісної реалізації персоналізованих систем у форматі месенджер-ботів.

Значний потенціал проєкту також полягає у його подальшому розвитку. Серед можливих напрямів удосконалення можна виділити:

- впровадження системи авторизації користувачів та синхронізації між пристроями;
- розширення рекомендацій за рахунок машинного навчання або нейромереж;
- підтримку перегляду новин кіноіндустрії та анонсів нових релізів;
- реалізацію можливості оцінювання фільмів та обміну думками між користувачами;
- інтеграцію з потоковими сервісами (наприклад, YouTube, Netflix тощо) для прямого переходу до перегляду.

У підсумку, можна зробити висновок, що Telegram-бот "Bot_film" є успішно реалізованим проєктом, що відповідає сучасним вимогам користувачів та має значний потенціал для подальшого розвитку. Його функціонал, інтерфейс та технічна реалізація є прикладом ефективного поєднання простоти використання, персоналізації та сучасного дизайну. Проєкт може бути корисним не лише кіноманам, а й розробникам, які цікавляться створенням якісних Telegram-сервісів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Іванов І. І. Розробка чат-ботів: теорія та практика / І. І. Іванов. — Київ: Наукова думка, 2021. — 280 с.

2. Петров П. П. Алгоритми рекомендацій у сучасних інформаційних системах / П. П. Петров // Інформатика і соціальні технології. — 2020. — Вип. 12. — С. 45–53.

3. Сидоренко О. М. Основи JavaScript для розробки веб-додатків / О. М. Сидоренко. — Харків: ХНУ, 2019. — 320 с.

4. Node.js Documentation // URL: https://nodejs.org/en/docs/ (дата звернення: 23.05.2025).

5. MongoDB Manual // URL: https://docs.mongodb.com/manual/ (дата звернення: 23.05.2025).

6. Telegram Bot API Documentation // URL: https://core.telegram.org/bots/api (дата звернення: 24.05.2025).

7. TMDb API Documentation // URL: https://developers.themoviedb.org/3/getting-started/introduction (дата звернення: 24.05.2025).

8. YouTube Data API Overview // URL: https://developers.google.com/youtube/v3 (дата звернення: 24.05.2025).

9. IMDb API Documentation // URL: https://imdb-api.com/ (дата звернення: 25.05.2025).

10. Netflix Recommendations: Algorithms and Applications / J. Doe // Journal of Streaming Technology. — 2022. — Vol. 5, No. 2. — P. 120–135.

11. Козак В. В. Особливості побудови Telegram-ботів з 3D інтерфейсами / В. В. Козак // Вісник Національного університету. — 2023. — Вип. 7. — С. 88–95.

12. Савченко І. О. Основи роботи з Three.js / І. О. Савченко. — Львів: Львівська політехніка, 2020. — 210 с.

13. Романюк А. Б. Персоналізовані системи рекомендацій / А. Б. Романюк // Сучасні інформаційні технології. — 2021. — № 3. — С. 15–25.

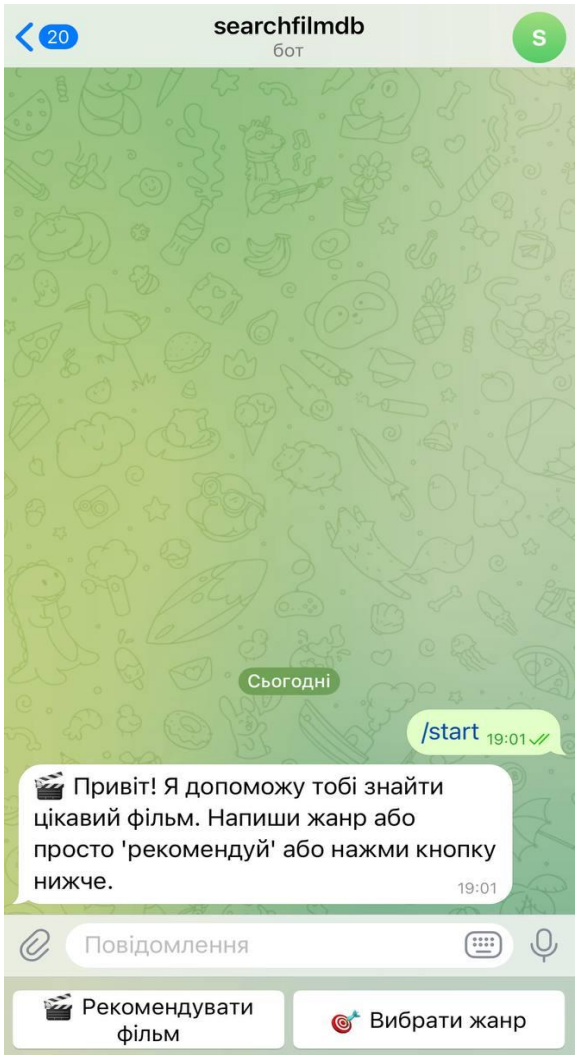
14. MovieMate Bot // URL: https://t.me/moviematebot (дата звернення: 25.05.2025).

15. IMDb Bot // URL: https://t.me/IMDb_Bot (дата звернення: 25.05.2025).

ДОДАТКИ

Додаток А. Скріншоти інтерфейсу розробленого бота

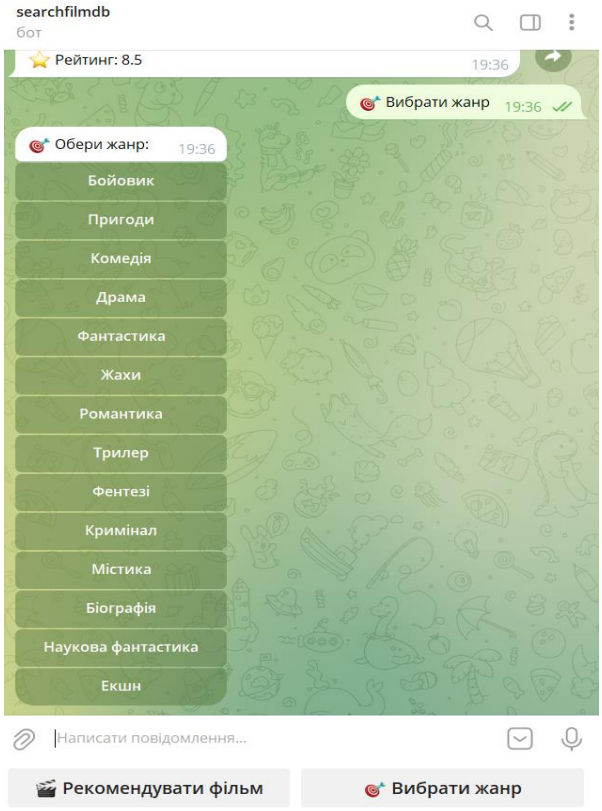
1. Екран привітання та початок взаємодії з бо



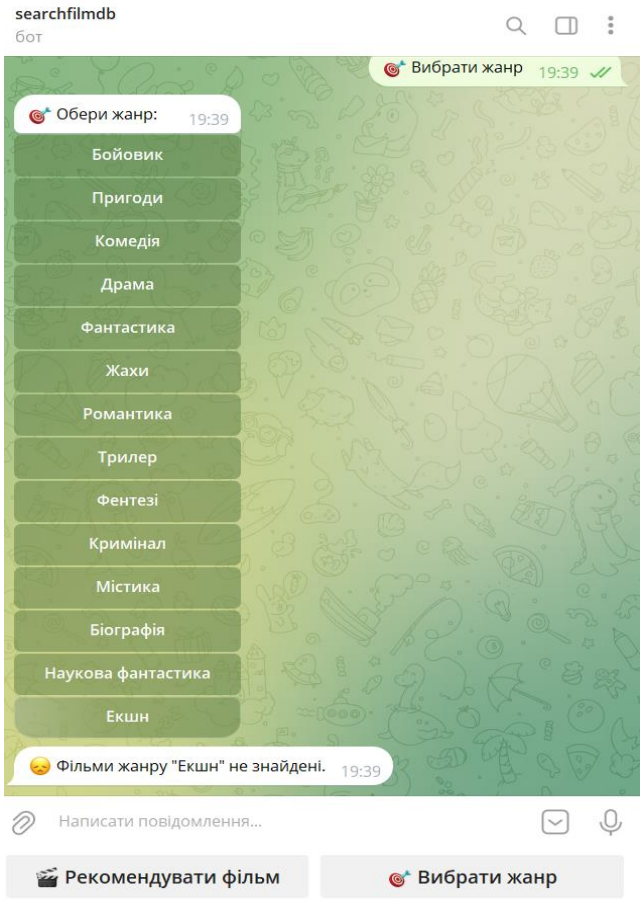
2. Головне меню глядача

3. Запит на рекомендацію фільму

4. Список доступних жанрів для глядачів.



5 Якщо фільму такого жанру не знайдено


searchfilmdb
бот
Вибрати жанр 19:39
Обери жанр: 19:39
Бойовик
Пригоди
Комедія
Драма
Фантастика
Жахи
Романтика
Трилер
Фентезі
Кримінал
Містика
Біографія
Наукова фантастика
Екшн
Фільми жанру "Екшн" не знайдені. 19:39
Написати повідомлення...
Рекомендувати фільм
Вибрати жанр

Додаток В. Фрагменти коду та конфігураційні файли

1. Вітальне повідомлення, та кнопка «Рекомендувати фільм».

```
if (text === "/start") {

bot.sendMessage(chatId, "🎬 Привіт! Я допоможу тобі знайти цікавий фільм. Напиши жанр або
просто 'рекомендуй' або нажми кнопку нижче.", mainMenu);

} else if (text.includes("рекомендуй") || text === "🎬 Рекомендувати фільм") {

const randomMovie = await Movie.aggregate([{ $sample: { size: 1 } }]);

const movie = randomMovie[0];



if (!movie) {

return bot.sendMessage(chatId, "☹ В базі даних на жаль поки немає таких фільмів.");

}

bot.sendPhoto(chatId, movie.poster, {

caption: `📽 *${movie.title}* (${movie.year})\n\n${movie.description}\n\n⭐ Рейтинг:
${movie.rating}`,

parse_mode: "Markdown"
```

2. Команда «Вибрати жанр».

```
else if (text === "🎯 Вибрати жанр") {
        const chatId = msg.chat.id;
        const genres = [
            /// Список жанрів, доступних для вибору.
            { name: "Бойовик", value: "Бойовик" },
            { name: "Пригоди", value: "Пригоди" },
            { name: "Комедія", value: "Комедія" },
            { name: "Драма", value: "Драма" },
            { name: "Фантастика", value: "Фантастика" },
            { name: "Жахи", value: "Жахи" },
            { name: "Романтика", value: "Романтика" },
            { name: "Трилер", value: "Трилер" },
            { name: "Фентезі", value: "Фентезі" },
            { name: "Кримінал", value: "Кримінал" },
            { name: "Містика", value: "Містика" },
            { name: "Біографія", value: "Біографія" },
            { name: "Наукова фантастика", value: "Наукова фантастика" },
```

```
        { name: "Екшн", value: "Екшн" }
];

const keyboard = genres.map((g) => [
{ text: g.name, callback_data: `genre_${g.value}` }
]);

bot.sendMessage(chatId, "🎯 Обери жанр:", {
reply_markup: {
inline_keyboard: keyboard
}
//Обробка вибору жанру.

bot.on("callback_query", async (query) => {
const chatId = query.message.chat.id;
const genre = query.data?.replace("genre_", "");

console.log("🔎 Запит на жанр:", genre);

try {
const movies = await Movie.aggregate([
{ $match: { genre: genre } },
{ $sample: { size: 1 } }
]);

if (!movies.length) {
return bot.sendMessage(chatId, `😞 Фільми жанру "${genre}" не знайдені.`);
}

const movie = movies[0];

bot.sendPhoto(chatId, movie.poster, {
caption: `📼 *${movie.title}* (${movie.year})\n\n${movie.description}\n\n☆ Рейтинг: ${movie.rating}`,
parse_mode: "Markdown"
});

} catch (error) {
bot.sendMessage(chatId, "❌ Помилка при пошуку фільму за жанром.");
console.error("❌ MongoDB genre error:", error.message);
}
});
```

3. Команда /add «Назва фільму», доступ до команди мають адміністратори.

```
else if (text.startsWith("/add ")) {
const title = text.replace("/add ", "").trim();

if (msg.from.id !== ADMIN_ID) {
console.log("No access")
return bot.sendMessage(chatId, "❌ У тебе немає доступу до цієї команди.");
}
bot.sendMessage(chatId, `🔍 Пошук фільму: *${title}*`, { parse_mode:
"Markdown" });

const result = await addMovieByTitle(title);

if (result.success) {
const movie = result.movie;
bot.sendPhoto(chatId, movie.poster, {
caption: `✅ *${movie.title}* (${movie.year}) додано до бази.`,
parse_mode: "Markdown"
});
console.log("✅ Фільм успішно доданий");
} else {
bot.sendMessage(chatId, `⚠️ Не вдалося додати фільм: ${result.reason}`);
```

```
  // Якщо текс не підходить під жодну умову – виводиться повідомдення про помилку.
else {
        bot.sendMessage(chatId, "❓ Не зрозумів тебе. Напиши /start або 'рекомендуй'.");
    }
    console.log("Bot starting");
 })
```

Додаток Г. Схеми і таблиці щодо візуалізації аналізу

Назва поля	Тип даних	Опис
id	INTEGER / ObjectId	Унікальний ідентифікатор фільму
user-id	INTEGER / String	Telegram ID користувача, що зробив запит
movie-title	TEXT	Назва фільму
year	String/Number	Рік виходу фільму
genre	String або Array	Жанр фільму (наприклад: «Комедія», «Драма»
description	String	Опис фільму
rating	Number	Рейтинг фільму

Таблиця Г.1 – Структура таблиці appointments

Додаток Д. Узагальнена схема алгоритму функціонування чат-боту