Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ТЕЛЕГРАМ-БОТА ДЛЯ ГЕНЕРАЦІЇ КОНТЕНТУ НА ОСНОВІ ЗАПИТІВ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-3

Терно О.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Терно Олександру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка телеграм-бота для генерації контенту на основі запитів з використанням штучного інтелекту

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, документація Telegram Bot API та OpenAI API, інтегроване середовище розробки Visual Studio Code, програмне забезпечення Ngrok, фреймворк для розробки користувацького інтерфейсу Angular, відкрите середовище виконання Node.js, бібліотека взаємодії з телеграм-ботом GrammyJS.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд розробницьких технологій для реалізації телеграм-бота.
2. Огляд головних методів та алгоритмів втілення телеграм-бота.
3. Практична імплементація телеграм-бота.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми вибору бібліотеки для взаємодії з телеграм-ботом на різних мовах програмування, порівняння серверних технологій розробки, постановка задачі, процеси штучних моделей, архітектура телеграм-бота, лістинги реалізації застосунку, інструкція з використання, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-12.04.24	
3	Аналіз літератури з досліджуваної проблеми	13.04.24-15.04.24	
4	Аналіз розробницьких технологій	16.04.24-22.04.24	
5	Розробка методів та алгоритмів	23.04.24-07.05.24	
6	Програмна реалізація	08.05.24-19.05.24	
7	Оформлення пояснювальної записки	20.05.24-22.05.24	
8	Перевірка на плагіат	25.05.24	
9	Рецензування	26.05.24	
10	Підготовка презентації та доповіді	27.05.24-02.06.24	
11	Занесення роботи в електронний архів	03.06.24	
12	Попередній захист кваліфікаційної роботи	03.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Кобилін О.А._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 68 с., 3 табл., 38 рис., 1 дод., 32 джерела.

РОЗРОБКА ТЕЛЕГРАМ-БОТА, ГЕНЕРАЦІЯ КОНТЕНТУ ЗА ЗАПИТОМ, ШТУЧНИЙ ІНТЕЛЕКТ, НЕЙРОННА МЕРЕЖА, NODEJS, ANGULAR, OPENAI API, БІБЛІОТЕКА GRAMMYJS, NGROK, TELEGRAM MINI APPS.

Об’єктом роботи є комплексний телеграм-бот для генерації контенту за запитом з використання штучного інтелекту.

Метою роботи є розробка боту для генерації контенту за запитом з можливістю надавати потрібну інформацію за кнопками, генерувати текстову та аудіо відповідь користувачу на його запит за допомогою штучного інтелекту, а також взаємодіяти з вебінтерфейсом Telegram Mini Apps.

У результаті роботи проведено дослідження різних технологій для створення телеграм-ботів та взаємодії з ними, можливостей інтеграції до застосунку Telegram Mini Apps, методів обробки природної мови на базі моделі «gpt-3.5-turbo» та синтезу мовлення «tts-1», а також під час програмної реалізації застосовані актуальні технології та бібліотеки, як Node.js, GrammyJS, Angular, OpenAI API, Ngrok.

DEVELOPMENT OF A TELEGRAM BOT, CONTENT GENERATION BASED ON REQUEST, ARTIFICIAL INTELLIGENCE, NEURAL NETWORK, NODEJS, ANGULAR, OPENAI API, GRAMMYJS LIBRARY, NGROK, TELEGRAM MINI APPS.

The object of work is a comprehensive Telegram bot for content generation upon request using artificial intelligence.

The aim of the work is to develop a bot for content generation upon request with the ability to provide necessary information via buttons, generate textual and audio responses to user inquiries using artificial intelligence, and interact with the Telegram Mini Apps web interface.

As a result of the work, research has been conducted on various technologies for creating Telegram bots and interacting with them, integration possibilities into the Telegram Mini Apps application, natural language processing methods based on the «gpt-3.5-turbo» model and speech synthesis «tts-1», as well as relevant technologies and libraries such as Node.js, GrammyJS, Angular, OpenAI API, and Ngrok have been applied during software implementation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

NPL – Natural Language Processing (обробка природної мови)

ML – Machine Learning (машинне навчання)

TTS – Text-to-Speech (текст в мовлення)

GPT – Generative Pre-trained Transformer

AI – Artificial Intelligence (штучний інтелект)

DL – Deep Learning (глибоке навчання)

API – Application Programming Interface

IT – Information Technology (інформаційні технології)

QR – Quick Response (швидка відповідь)

FAQ – Frequently Asked Questions (часто задані питання)

DSL – Domain-Specific Language (мова спеціального призначення)

ES6 – ECMAScript 2015 (стандарт мови JS)

CS – Computer Science (комп’ютерні науки)

HTTPS – Hypertext Transfer Protocol Secure (захищений протокол передачі гіпертексту)

DDOS – Distributed Denial of Service (розподілена відмова у обслуговуванні)

BERT – Bidirectional Encoder Representations from Transformers

UI – User Interface

JS – JavaScript

JSON – JS Object Notation

OS – Operation System

IDE – Integrated Development Environment

SQL – Structured Query Language

NPM – Node Package Manager

JVM – Java Virtual Machine

SID – Security Identifier

GIL – Global Interpreter Lock

URL – Unified Resource Locator

DOM – Document Object Model

SPA – Single Page Application

IOS – IPhone Operating System

БД – база даних

ТГ – телеграм

ПО – предметна область

Стек – набір програмних технологій в рамках проєкту

ООП – об’єктно-орієнтоване програмування

ВСТУП

В повсякденному житті все частіше зіткаємося з тим, що в кожній сфері стає все більше вебсайтів чи застосунків. Звичайною практикою є замість квитка показувати QR-код, платити через онлайн-банкінг, навіть мати доступ до документів прямо з телефону. Інформаційні технології стали невід'ємною складовою життя більшості людей, всі стали швидше розвиватися, адаптуватися до потреб навколишніх реалій і тому з кожним днем стає все більше нових програмних продуктів.

Високий темп прогресу став причиною щось змінювати, щоб виділятися серед постійної конкурентності, рухатися вперед та становитися краще. Для цього був необхідний певний поштовх в сторону інновацій.

Сьогодні все більше компаній створюють своїх чат-ботів. По-перше, це зручно, функціонально, ефективно та вигідно. По-друге, відчиняє нові горизонти до експериментів, менш витратно з точки зору ресурсів у розробці, не потребує таких потужних обчислювальних систем і тому все частіше можна зустріти не тільки базових ботів, а й більш цікавих з можливостями застосування штучного інтелекту чи комп'ютерного зору у проєктах [1].

Один з найяскравіших представників, де є можливість створити свого бота – це Telegram. Він має багато бібліотек для різних мов програмування для взаємодії з ним. В ньому реалізовано множину функцій, які можна додати до свого бота та с кожним днем можливостей стає тільки більше. Постійна підтримка розробників та ріст користувачів самого месенджеру прямо натякають, що саме там сьогодні можна популяризувати новий продукт.

Телеграм-бот – це програмна реалізація заданих автоматизованих процесів, розроблених з метою надати користувачу змогу отримати необхідну йому опцію прямо у месенджері, завдяки взаємодії юзера через інтерфейс бота з серверною частиною телеграму, яка виступає посередником, між клієнтською стороною та розробленою програмою.

Основні задачі, які ставлять перед ботом:

– FAQ клієнтів або технічна підтримка;
– замовлення товару чи перевірка його статусу;
– рекламні оголошення, новини, інша інформація;
– зворотній зв’язок, оцінка якості послуги, коментарі тощо.

З точки зору універсальності, бот буде корисним у широкому спектрі застосування, буквально будь-де, починаючи з навчального ресурсу та закінчуючи крипто-валютним сервісом [2, 3]. З точки зору бізнес-процесів наявність власного телеграм-боту має багато переваг, таких як, автоматизація процесів, доступна масштабованість, легкість адаптації вже існуючих сервісів чи API. Бот може бути не тільки допоміжним інструментом, а й замінити повноцінний застосунок.

А також, не менш важливим є зручність єдиної екосистеми, коли можна через одного бота купити квитки на потяг, іншим забронювати готель, а потім все це надіслати приятелю з яким збираєтесь у подорож і все не виходячи з Telegram.

Отже, основним завданням кваліфікаційної роботи є аналіз, пошук рішення, яким має бути сучасний телеграм-бот, які функціональні можливості мають бути вбудовані в систему для створення відповідної програмної реалізації та надання кращого користувацького досвіду під час використання.

Актуальність даної теми полягає у створенні телеграм-боту, який спростить доступ до отримання необхідного контенту користувачу, шляхом автоматизації процесів та опрацюванням запитів штучним інтелектом. Бот має стати інструментом до якісного генерування текстового та аудіо матеріалу. А також підвищити ефективність та оптимізувати процеси потрібні юзеру в відповідному програмному продукті.

1 ОГЛЯД РОЗРОБНИЦЬКИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ТЕЛЕГРАМ-БОТА

1.1 Розробка телеграм-ботів

Цей розділ є оглядом найрозповсюджених технологій, бібліотек та фреймворків, які можна застосувати при розробці бота. Їх було ретельно досліджено та вибрано оптимальне рішення для використання в кваліфікаційній роботі. Вибір технологічного стеку залежав від запланованого функціоналу та специфічних деталей під час реалізації.

Створення телеграм-ботів – це процес, що передбачає в собі розробку програмного рішення, яке завдяки Telegram дає змогу взаємодіяти з користувачами та надавати їм автоматизовані опціїї отримання потрібного сервісу [4]. Зазвичай типовими можливостями є відправка повідомлень, інформації, файлів тощо, але можуть бути й більш складні задачі, наприклад магазин взуття, де товар можна додати у кошик, подивитися наявність та навіть замовити.

Під час розроблення ботів застосовують різноманітні набори технологій, які складаються з мов програмування та їх бібліотек, фреймворків чи API. Розробка застосунку такого виду має на увазі серверну програму, яка кожну мить готова надати свої послуги юзеру. Використовують такі мови, як Python, Node.js(JavaScript), Java, Ruby, Go тощо, разом з їх бібліотеками для праці з Telegram. Наприклад Telegraf для JS, детальніше наведено у таблиці 1.1. А також якщо бот потребує базу даних, то зазвичай – це MySQL або PostgreSQL чи MongoDB, Redis якщо потрібна NoSQL БД.

Після цього програмний код ТГ-бота, треба розмістити на постійно доступному сервері. Хостинг дає можливість безперервної роботи та безпеки даних. Декілька популярних – це Microsoft Azure, Amazon Web Services, Google Cloud Platform або платформи як Heroku для розгортання застосунків.

Таблиця 1.1 – Порівняльна таблиця популярних телеграм бібліотек з різних мов програмування

Мова програмування	**Бібліотека**	**Головні особливості**
Node.js(JavaScript)	GrammyJS	Легка в використанні, зручна в обробці команд та подій, достатньо гнучка та мобільна, має інтеграцію з Telegram Mini Apps, підтримка з сторони розробника
Node.js(JavaScript)	Telegraf	Має обширний набір функцій, підтримка вебхуків, достатньо зрозумілий синтаксис, має можливості роботи з middleware
Python	telebot	Інтуїтивно зрозуміла документація, багата на користувачів система, швидкодіюча
Python	aiogram	Підтримує асинхронність, постійна підтримка від розробників, великий функціонал, DSL для ботів
Java	TelegramBots	Інтеграція з Spring Framework
Ruby	Telegram-bot-ruby	Добре налагоджена робота з асинхронністю
Go	go-telegram-bot-api	Наявність middleware
Kotlin	ktor-telegram	DSL для обробників
C#	Telegram.Bot	Підвищена абстракція
PHP	telegram-bot/api	Локалізація

Кожний варіант наведений у таблиці має свої переваги та недоліки, але всі вони є цілком раціональним вибором для проєкту, отже вибирати треба залежно від потреб кожного юзер-кейсу індивідуально, але в першу чергу від мови на якій планується створення боту.

1.2 Розробка Telegram Mini Apps

Telegram Mini Apps – це невеликі інтерактивні вебзастосунки інтегровані у ботів месенджеру телеграм, безпосередньо в чаті, завдяки яким у розробників є можливість додавати більше нових потужних функцій та використовувати власні UI-дизайни.

Їх застосування може бути різноманітним, наприклад розроблення невеличких ігор в яких можна навіть, створити мультиплеєр для змагання з іншими користувачами або розробити сервіс з доставки їжі з власним інтерфейсом, можливістю вибрати потрібні позиції до кошику, потім заповнити форму з даними адреси доставки та оплатити замовлення через Apple або Google Pay за замовчуванням чи інтегрувати власного провайдера сплати. Приклад показано на рисунку 1.1.

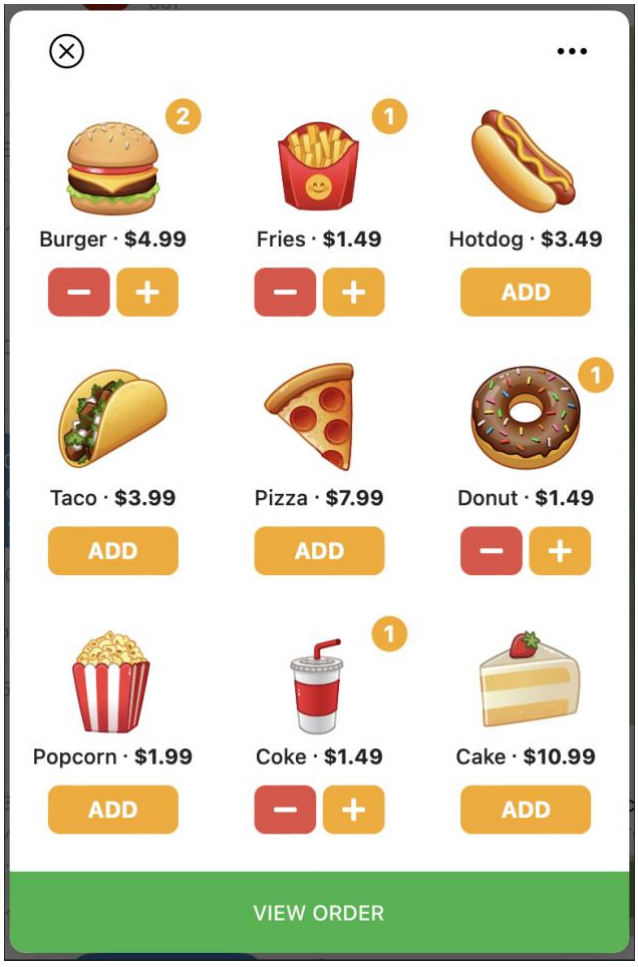


Рисунок 1.1 – Інтерфейс замовлення їжі на базі Telegram Mini Apps

Для написання інтерфейсів міні-вебзастосунків зазвичай використовують:

– HTML, CSS, JavaScript: стандартний набір для веброзробки, якого може бути цілком досить, зважаючи на масштаби та задачі проєкту [5];

– фреймворки на основі JS, які допомагають розробникам полегшити процес написання програмного коду, особливо коли справа стосується більш потужних програмних рішень, завдяки компонентному підходу, найпопулярнішими є:

1) React.js;
2) Vue.js;
3) Angular;
4) Svelte;
5) Ember.js;

– бібліотеки та UI компоненти: такі технології, як Tailwind CSS, Material UI, Bootstrap, Bulma тощо, для вже існуючих рішень та стилізації об’єктів на вебсторінці.

В свою чергу, клієнтська частина взаємодіє з серверною, яка була обрана під час створення телеграм-боту та за потреби з БД, наприклад SQLite для невеликих обсягів даних чи взагалі з JSON для мінімальних потреб системи.

Після чого Telegram Mini Apps застосунок має бути задеплоїним (розгорнутим) на хмарному сервісі для його інтеграції у боті. Наприклад завдяки зв’язці Docker та Git можна це зробити досить просто. Необхідність полягає в подальшому з’єднанні з Telegram, бо задля безпеки є обмеження і потрібно обов’язково мати протокол HTTPS.

В іншому разі, скористатися інструментами тунелювання локального серверу, такий варіант є оптимальним під час розробки, тестування та демонстрації програмного продукту без використання публічних серверів. Сервіс Ngrok може бути корисним в даному юз-кейсі для отримання URL-адреси через захищене з’єднання на локальний застосунок.

1.3 Аналіз серверних технологій

Для написання серверних програм на яких базуються боти, використовують багато різних мов програмування, тому хотілося б відмітити декілька технологій, які можна найчастіше зустріти як в комерційних проєктах, так і в аматорських.

Лідерами безсумнівно є Python, JS(Node.js) та Java. Тому після проведених досліджень потрібно детальніше розповісти про кожного представника окремо. Усі вони мають потужні фреймворки, завдяки їх використанню процес розробки стає продуктивнішим та масштабованішим. А кількість написаних строк коду з кожним днем росте.

1.3.1 Python

Python в контексті мови програмування – це потужний інструмент для надійного виконання поставлених задач, ідеальний вибір для серверних застосунків, в особливості для створення телеграм-ботів [6]. Є широкий вибір фреймворків, таких як Django або Flask. Кожного для створюється багато нових цікавих бібліотек через те, що він має широку популярність в спільноті розробників.

Крім вже перерахованих переваг, потрібно відмітити синтаксис, який дуже легко читається та дозволяє без труднощів перейти з іншої технології. Обробка одночасної, великої кількості запитів, також заслуга цієї мови.

Якщо трішки зануритися в історію Python, який з’явився в 1991 році, як тестовий варіант, його в першу чергу створювали зрозумілим, нескладним й досить універсальним інструментом розробки, що добре видно крізь роки. Вже в 1994 році, була випущена в світ перша його версія, де у дослідницькому центрі Нідерландів, працював Гвідо ван Россум. А ще через 6 років вже наступна, в якій реалізували концепт ООП та Unicode [7]. Тому

можна зазначити, що задумане у творця технології вдалося і в нинішній час ця мова має неймовірні можливості.

1.3.2 Node.js

Node – це JS-базоване середовище, яке здатно компілювати код без браузеру, над яким працював Р. Далем не так багато часу назад [8]. Саме 2009 став роком виходу на IT-ринок його винаходу, після чого суспільство побачило платформу для забезпечення мережевих програм за допомогою двигуну, який використовується в веббраузері Chrome від Google.

Ключовою опцією є подвійно-орієнтованість. Тому проблем з обробкою величезних об’ємів надсилаємих запитів там звісно немає. Суттєва кількість модулів та бібліотек, підтримуєма та оновлюєма весь час, надає можливості використання різних методів та підходів в написанні коду.

Далі, окрім вищезгаданого, іншою позитивною стороною є наявність власного менеджеру пакетів. NPM – є базою для управління залежностями проєкту, має найбільшу колекцію безкоштовних пакетних рішень для JavaScript. Допомагає економити час перевикористанням коду через створення власних рішень. Концепція роботи відображена на рисунку 1.2.

Розглянемо декілька основних фреймворків:

– NestJS зручний вибір побудови ефективних серверних застосунків з різними патернами програмування;

– Express.js насправді найрозповсюдженіший вибір через свою доступність, швидкість та надійність [9];

– Koa.js сучасний, концептуально базований на принципі генераторів ES6, який чистить та валідує код.

Тому за такий несуттєвий проміжок часу Node став ключовим гравцем в серверних застосунках, мікросервісах та розробленню API.

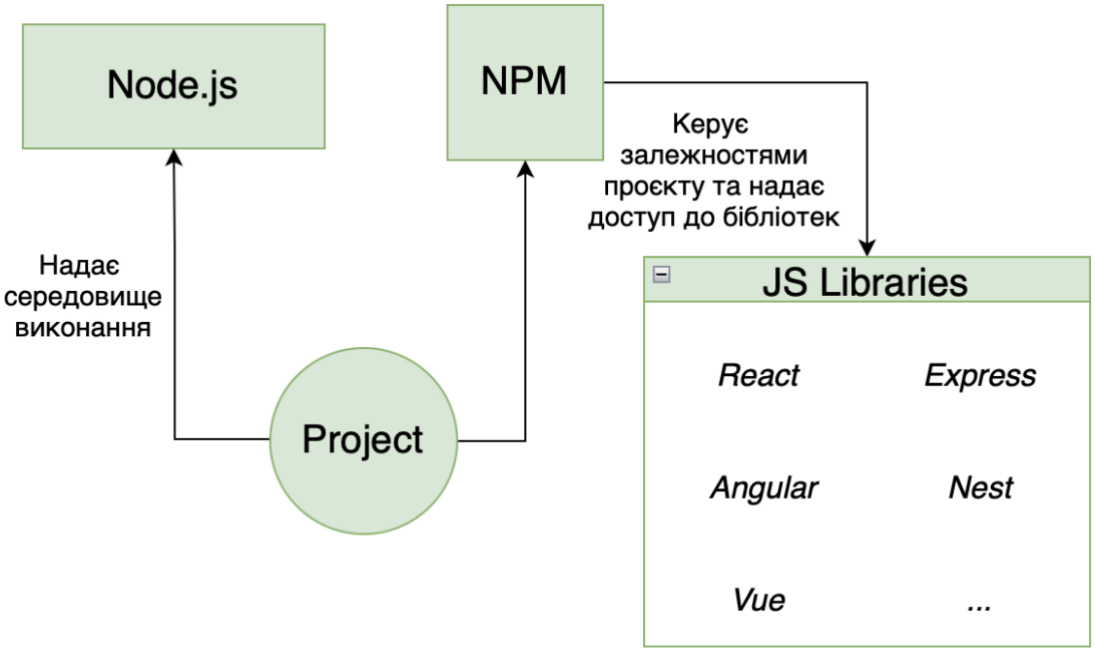


Рисунок 1.2 – Концепт використання NPM у Node.js проєкті

1.3.3 Java

Java – це ООП мова з вражаючим функціональними можливостями. Ключовою фішкою технології є здатність переносити на різні комп'ютерні системи не потребуючи зміни джерел коду: Windows, MacOS, Linux – кожна виконує без перешкод код Java, який конвертується JVM під потрібну операційну платформу [10]. Візуалізацію цього процесу можна побачити на рисунку 1.3.

Здатність до багатопоточності досягається інтегрованою особливістю технології, що розкриває багатоядерні процесори на весь їх потенціал. Є багато можливостей захисту інформації, а саме опції контролювання доступу, бібліотеки працюючі з криптографією або пам'яттю.

Різносторонній спектр екосистеми доводиться наявністю великої множини бібліотек та інших інструментів кодування. Java не з проста є однією з відоміших мов та широко використовується в серверних продуктах.

Spring Framework і Spring Boot, Apache, Dropwizard, Hibernate – деякі розповсюдженні фреймворки з нескінченної кількості існуючих.

Отже, завдяки різним методам розробки доступним у системі, наприклад роботою с кешуванням, з кластерами та потоками. Програми на Java відзначаються своєю сильною й ефективною можливістю справлятися з ростом навантажень та стабільно виконувати задані функції.

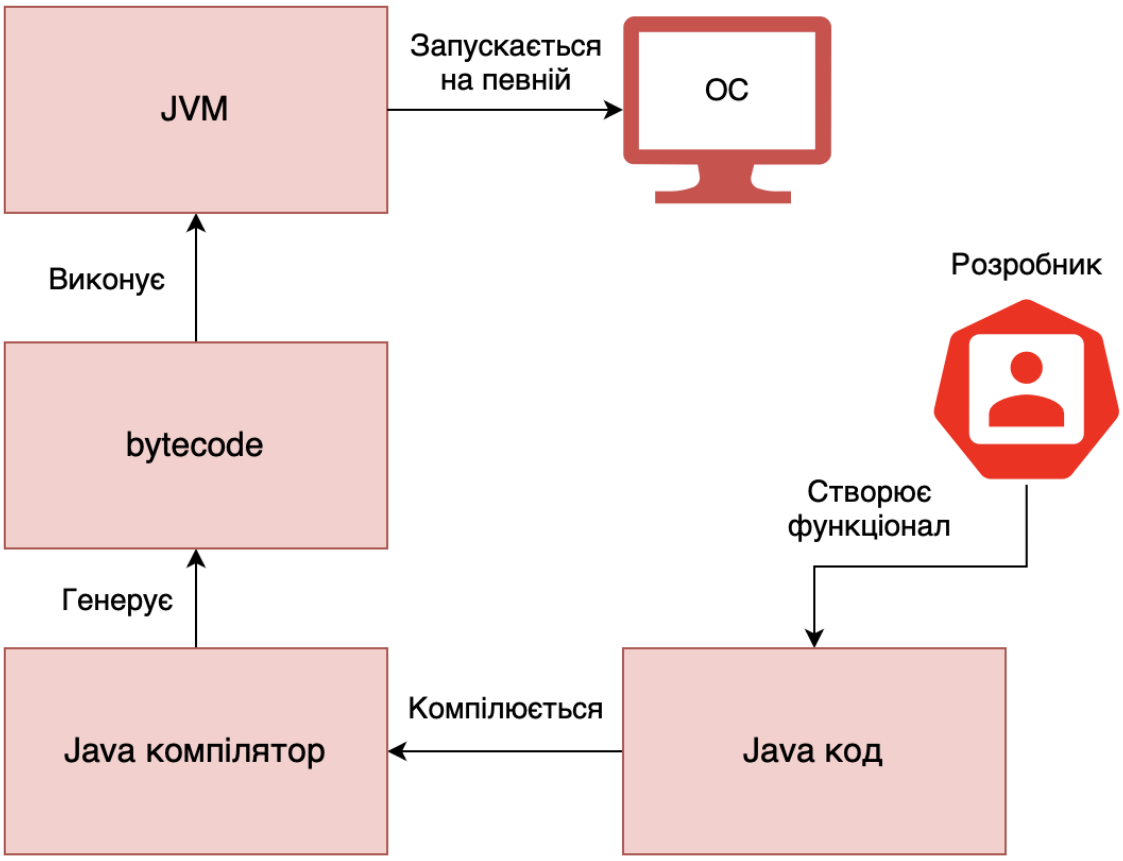


Рисунок 1.3 – Архітектура компіляції Java коду

1.3.4 Порівняння серверних технологій

Для підведення підсумку оглянемо таблицю 1.2, яка містить порівняння описаних раніше мов програмування. За критеріями з’ясуємо, який у Python, Node.js та Java рівень для того щоб обрати пріоритетну технологію для серверної частини телеграм-ботів.

Таблиця 1.2 – Порівняльна характеристика серверних технологій

Критерій порівняння	**Python**	**Node.js**	**Java**
1	**2**	**3**	**4**
Швидкодійність	Середня	Висока	Висока
Фреймворки	Присутні	Присутні	Присутні
Плагіни та інструменти розробки	Небагато	Багато	Багато
Переносимість	Менш легка	Легка	Легка
Підтримка	Розповсюд-жена	Розповсюд-жена	Розповсюд-жена
Розгортання	Середнє	Швидке	Повільне
Маштабованість	Середня	Висока	Висока
Безпека	Середня	Середня	Висока
Мультитопоточність	Тільки GIL	Має	Має

За підсумками, результати характеристики показують, що мови схожі в багатьох можливостях, кожна має власні особливості. Тому вибір пав на Node, бо там є усі якості необхідні для розробки ботів в месенджері Telegram.

1.4 Аналіз технологій веброзробки

Під час веброзробки, яка здійснюється для Telegram Mini Apps, але в цілому не відрізняється від інших вебпрограм, приміняють технології фронтенду, які треба роглянути для подальшого вибору в сторону одного з них.

Зазвичай – це React, Vue та Angular. Існує багато інших набираючих популярність варіантів, наприклад Ember або Svelte, але зараз не про них. Перераховані перспективні фреймворки є дійсно гарним вибором для

ознайомлення, які кожного дня розробники приміняють в своїх роботах. А зараз про трійку лідерів, тому треба дізнатися про головні моменти кожної технології з неї.

1.4.1 React.js

Компанія Facebook внесла колосальний вклад своїм інструментом розробки, який спочатку взагалі був призначений для закритого користування. Згодом React став настільки значним та надійним в IT-ком'юніті, що швидко заполонив ринок [11]. Навіть був адаптований для розроблення мобільних користувацьких інтерфейсів, за тими же методами й логікою, так винайшли – React Native.

Якщо виділити важливі моменти, то компонентний спосіб кодування, віртуальний рендер вебсторінок DOM, оптимізовані SPA застосунки з необхідністю тільки разового завантаженням початкового контента, а також JSX розширення, яке буквально є поєданням HTML і JavaScript коду в єдиний синтаксис – все це надає позитивний досвід програмування з цією вебтехнологією.

1.4.2 Vue.js

Актуальність – це про Vue. Він має досить невисокий поріг заходу, особливо з досвідом інших мов [12]. Синтаксичні конструкції зрозумілі та ясні. Висока інтерактивність та динамічність інтерфейсів. Компоненти з одного файлу, де шаблон, стилі та скрипт в купі органічно полегшують написання та читаємість.

Чудовий вибір для розробника будь-якого рівня кваліфікації, метою якого є ефективне і відзивчиве, сучасне програмне рішення.

1.4.3 Angular

Angular є TypeScript базованим фреймворком, який був випущений від всесвітньо відомої корпорації Google [13]. Завдяки строгої типізації більш захищений від помилок компіляції. Має обширний спектр інструментів для складних вебрішень одразу з встановлення. RxJS бібліотека в основі фреймворку, що застосовує реактивність в програмуванні. Вистроєна на спостерігачах, які ловлять зміни в моменті часу та надають можливість підстраюватися під оновлення станів у програмі та швидко реагувати на них. Технічно сильний вибір у розробці, для досвідчених спеціалістів, який підійде для масштабних ідей.

1.4.4 Вибір технології веброзробки

Для інтеграції Telegram Mini Apps у середовище телеграму, було вирішено, методом аналізу розглянутих фреймворків та бібліотек, використовувати Angular. Причиною цього стало те, що його кросплатформенні переваги, модульна архітектура та доступна шаблонізація будуть надзвичайно корисні у даній темі.

1.5 Постановка задачі

Таким чином, створення телеграм-бота для генерації контенту за запитом з використанням штучного інтелекту, є актуальною задачею в тенденціях швидкого росту попиту на автоматизацію процесів, а також бажанням делегувати обов’язки на моделі глибокого навчання для оптимізації часу та підвищення ефективності здобуття текстової та аудіо

інформації. Дана розробка також буде мати інтерфейс Telegram Mini Apps, що дозволить взаємодіяти користувачу з різними елементами вебвікон.

Об’єктом роботи є комплексний телеграм-бот для генерації контенту за запитом з використанням штучного інтелекту.

Метою роботи є розробка боту для генерації контенту за запитом з можливістю надавати потрібну інформацію за кнопками, генерувати текстову та аудіо відповідь користувачу на його запит за допомогою штучного інтелекту, а також взаємодіяти з вебінтерфейсом Telegram Mini Apps.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз технологій для створення ботів та методів взаємодії, безпосередньо в контексті Node.js бібліотеки GrammyJS;

– розробити способи інтеграції міні-вебзастосунків в телеграм та UI на Angular;

– реалізувати здатність використання Ngrok для тестування працездатності застосунку;

– реалізувати методи обробки природної мови та синтезу мовлення завдяки штучним моделям на базі OpenAI API.

2 ОГЛЯД ГОЛОВНИХ МЕТОДІВ ТА АЛГОРИТМІВ ВТІЛЕННЯ ТЕЛЕГРАМ-БОТА

2.1 Розгортання та початок роботи з Telegram Bots

В цьому розділі буде розглянуто методи та алгоритми використані під час виконання кваліфікаційної роботи:

– способи взаємодії з ботами на платформі;

– поняття та опис штучного інтелекту, машинного навчання тощо;

– алгоритми текстових та голосових AI моделей основаних на технологіях GPT та TTS;

– технічні методи генераторів рандомізації.

Почнемо з впровадження у систему, застосунку Telegram. Для цього треба звернутися до головного боту.

2.1.1 BotFather

BotFather є офіційним асистентом з справ по створенню, редагуванню або видаленню нових телеграм роботів. Співпраця з ним є інтуїтивно доступною та не потребує попереднього досвіду. Обмежень по відтворенню нових рішень для власних чи бізнес потреб немає, можна створювати скільки забажається.

Щоб запустити процес створення, треба пройти процедуру за наступним алгоритмом:

Крок 1. Завантажити Telegram з будь-якого доступного девайсу, він є кросплатформеним й наявний на IOS, Android телефонах, комп’ютерах з різними операційними системами, планшетах тощо. Чи скористатися вебверсією на сайті месенджеру з приставкою веб. Завдяки синхронізації та

клієнтським сесіям діалог завжди доступний і тому легко переходити з одного пристрою на інший, є змога авторизації через QR-код.

Крок 2. У стрічці пошуку знайти акаунт, що має псевдонім «@BotFather». Також в нього є відповідна галочка, що символізує його верифікацію від імені платформи для надання впевненості безпечності використання. Маємо звернутися до нього та розпочати чат, зазвичай кнопка або команда «/start».

Крок 3. Ознайомитися з великим списком його здатностей та в разі створення нового боту, вибрати «/newbot» (рис. 2.1). Після чого прийде повідомлення, де спочатку треба надати ім’я, а потім вигадати username, який повинен включати слово «Bot». Наприклад ExampleBot або example_bot.

Крок 4. Після відправки потрібної інформації, створення відбудеться, з’явиться опція переходу до приватного чату, за відповідним посиланням, на кшталт «t.me/ExampleBot» з знайомим в Крок 3 користувацьким ім’ям. Також прийде непублічний API ключ, у форматі набора символів, букв та чисел у хаотичному порядку та різних регістрах. Цей ключ не можна розповсюджувати, бо він відповідає за доступ до телеграм-боту.

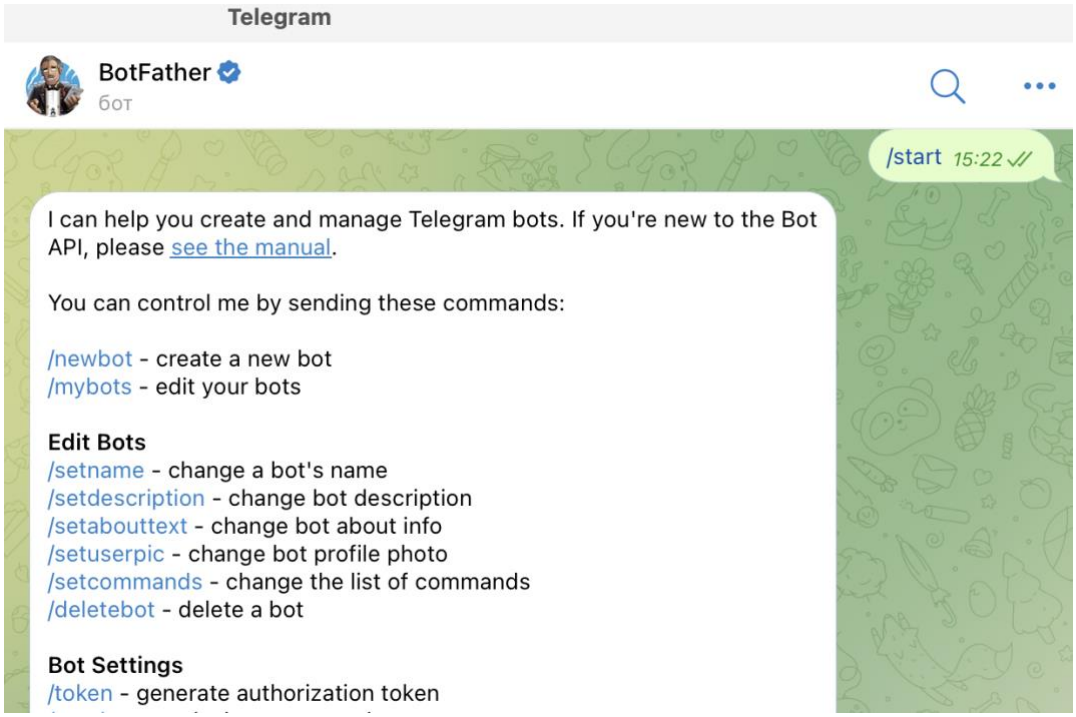


Рисунок 2.1 – Частковий список команд BotFather

2.1.2 Початок роботи

Тепер коли середовище для розробки створене, з цього моменту стає можливим кодування обробки команд та решти функцій. Наприклад, як побачена при використанні BotFather, команда старт.

Якщо Node та NPM вже встановлені, треба додати бібліотеку GrammyJS для комунікації з ботом. Для цього виконати «npm install grammy» до консолі директорії проєкту. Підключивши її, стає доступним відповідний синтаксис (лістинг 2.1). Зараз буде в нагоді API токен отриманий раніше.

Лістинг 2.1 Реалізація першочергового обробнику start:

```
const { Bot } = require('grammy');
const tgBot = new Bot('TELEGRAM_TOKEN');
bot.command('start', async (ctx) => {
await ctx.reply('Привіт, я телеграм-бот!');
});
tgBot.start();
```

Завдяки такому принципу й додається реагування на команди. На рисунку 2.2 зображено як це влаштовано.

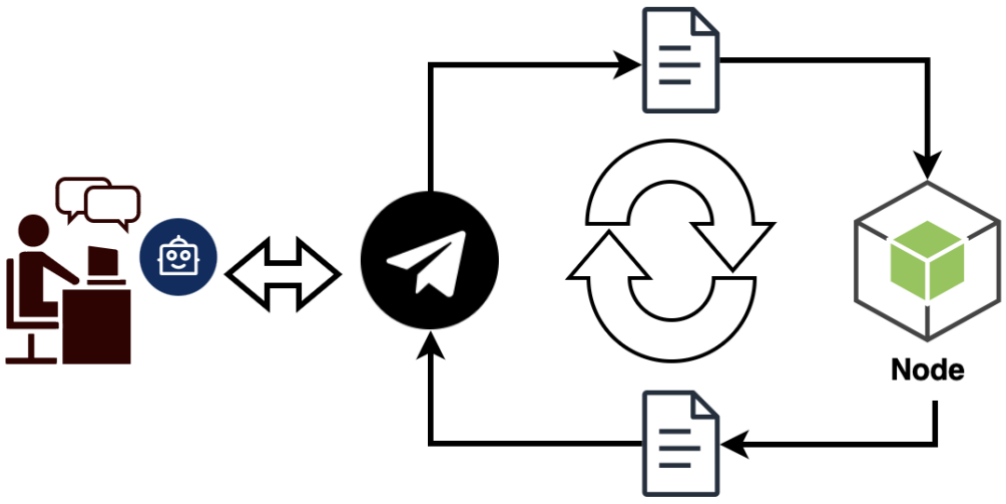
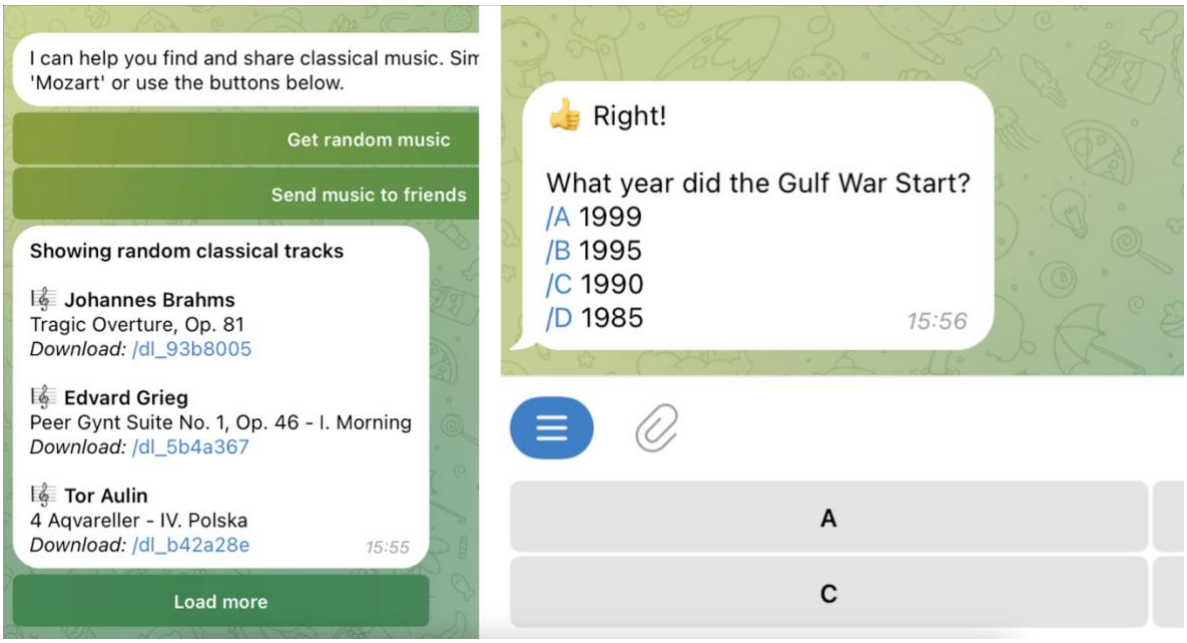


Рисунок 2.2 – Процес обробки команд

2.1.3 Способи користувацької взаємодії

Щоб зробити комфортний досвід вживання команд існує спосіб додання кастомних клавіатур. Це швидкий варіант для управління ТГ-ботом. Демонстрація Inline клавіатури на рисунку 2.3 а) та Reply на рисунку 2.3 б) відповідно.



а) б)

Рисунок 2.3 – Види клавіатур:
а) Inline Keyboard; б) Reply Keyboard

Розглянемо детальніше позитивні та негативні сторони кожної. З’ясуємо сценарії коли яку доречніше примінити.

Переваги інлайн введення:

– моментальність вибору: при цьому варіанті клавіатури дуже легко надати відповідь на опціональне питання;

– зручність розташування: одразу після читання повідомлень не треба прибирати погляд для пошуку кнопок;

– функціональність: швидкість заповнення форми при багаторівневому виборі.

Недоліки інлайн введення:

– обмеження кількості варіантів: доступно максимум шість додаваємих кнопок;

– незрозумілість інтерфейсу: оскільки місця для тексту не так багато, часто можна зустріти використання емодзі, які не завжди передають суть дії.

Переваги клавіатури відповіді:

– розмір поля для символів: завдяки масштабному розміру під полем вводу, є добра читаємість списку операцій;

– збереження відповіді: відображення попереднього вибору при переході на наступний список.

Недоліки клавіатури відповіді:

– займаний простір: інколи спливаюча клавіатура може заважати вводити текст;

– відсутність підтвердження: якщо натиснути кнопку випадково, вона одразу буде застосована і тому при розкладі, що це останній шаг перед застосуванням, є можливість помилкового вибору.

Отже, яку клавіатуру слід обирати залежить від потреб, Inline та Replay показують себе добре в поставлених до них задачах, незважаючи на несуттєві мінуси, а в тандемі зроблять застосунок набагато автоматизованішим.

2.2 Опис механіки штучних моделей

Для того щоб ретельніше підійти до огляду методів, треба поглибитись в загальні поняття щодо ПО застосованих алгоритмів галузі штучного інтелекту.

Після чого потрібно розглянути застосування, способи та концепції роботи, реалізацію коду та фактичні методи. Тобто, провести саме аналіз технічної сторони моделей роботи зі звуком та мовою.

2.2.1 Загальні відомості

Для початку зануримося в загальні поняття. Якщо казати про аналіз матеріалу, прагнення вибирати логічно та раціонально, можливість самовдосконалюватися, в контексті розділу CS, то одразу зрозуміємо, що мова йде про штучний інтелект. Агенти якого, намагаються наблизитись до людського сприйняття світу, вміння приймати рішення та бути схожими на своїх творців.

ШІ – це сучасна наука про створення програмних рішень, що зуміють імітувати модель розвитку. «Еволюція мозку» цих систем, нагадує дитину, яка пізнає світ з нуля, а потім на основі нової інформація за короткі інтервали часу робить впевнений крок вперед.

Деякі методи, які зустрічаються в багатьох розумних системах:

– комп’ютерне бачення (зір): система вчиться розуміти, що за об’єкт, який він, як його класифікувати, якого він кольору або параметрів для розпізнання на фото чи відео [14, 15];

– машинне навчання: спосіб стати краще, розвиток завдяки навчань з вчителем або без нього, чи усиленого пізнання. Ці методи ведуть штучну модель до здатності пізнавати нове через надані дані та робити певні прогнозування [16, 17];

– оброблення природної мови: підгалузь, завдяки якій комп’ютер читає та пише текст мовою зрозумілою людині. Генерує контент на вмінні акцентувати увагу на кодовому слові чи темі, коли аналізує повідомлення чи запит.

Використання таких інновацій дуже рухають індустрію, створюють середовище до нових продуктів з потужним функціональними опціями. Такі підрозділи, як глибоке навчання або інші, є дуже важливими винаходами нашого часу [18].

2.2.2 Текстова модель GPT

Generative Pretrained Transformer – transformer група NPL моделей, базованих на принципі Deep Learning, творцем якої є фірма OpenAI. Технологія GPT довела свою ефективність в генеруванні текстів на будь-які тематики, через що викликала величезний ажіотаж суспільства. Подивимось на структуру цієї моделі (рис. 2.4).

Рисунок 2.4 – Архітектура GPT моделі

Основні механізми, які лежать в складі моделей текстової класифікації:

– увага: дозволяє зосередитися на контексті, зрозуміти важливі моменти та головні слова;

– самонавчання: через величезну базу знань, яка прокручуються для засвоєння матеріалу, дозволяє мати гарний фундамент уявлення про граматику та синтаксис;

– самоувага: порівнює зв’язки слів між собою для надання пріоритетності в контексті словосполучень, речень або тексту загалом;

– індексування по позиції: формульне представлення, яким має бути правильне речення через вектори.

Проаналізуємо код практичного примінення штучної моделі gpt-3.5-turbo на рисунку 2.5.

```
import OpenAI from "openai";

const openai = new OpenAI();

async function main() {
  const completion = await openai.chat.completions.create({
    messages: [
      { role: "system", content: "You are a helpful assistant." },
      { role: "user", content: "Who won the world series in 2020?" },
      {
        role: "assistant",
        content: "The Los Angeles Dodgers won the World Series in 2020.",
      },
      { role: "user", content: "Where was it played?" },
    ],
    model: "gpt-3.5-turbo",
  });

  console.log(completion.choices[0]);
}
main();
```

Рисунок 2.5 – Лістинг застосування gpt-3.5-turbo на Node.js

Подібне застосування може бути корисним у багатьох сферах, де є бажання автоматизувати процес створення відповіді клієнту. Змінення ходу ведення бізнесу або просто сервісу залежить від інтелектуальних агентів, бо можливості до розумного способу технічної-підтримки або інструкції програмного продукту юзеру мають на меті зекономлять багато часу. В тому числі коштів, з боку компаній, завдяки делегуванню рутинних задач інтернет-роботам. Тому ідея цього коду полягає в демонстрації звернення до текстового асистенту з спорт питаннями.

Важливий принцип структури – це трансформерна архітектура (рис. 2.6), як у GPT або BERT (нейромережа від Google) для продуктивного конструювання текстових рядів через токенізацію [19].

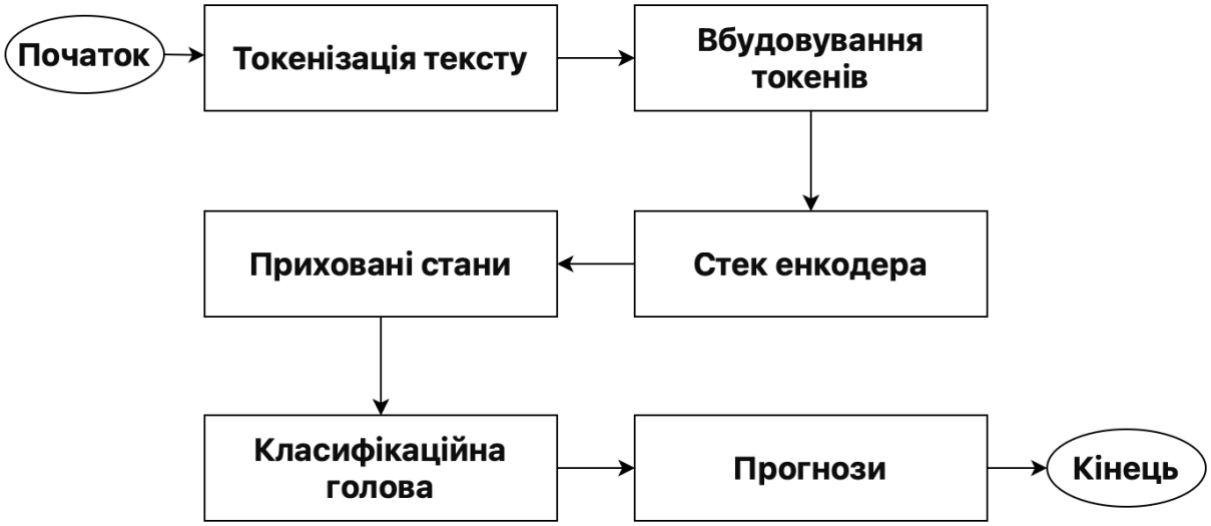


Рисунок 2.6 – Токенізація у структурі трансформеру

Спочатку все відокремлюється на юніти текстового представлення, так звані токени, конвертується у вектори, потім відформатована інформація після проходження через шари (де були етапи аналізу, нормального представлення), збирається в новий ряд токенів, який формує згенерований результат запиту [20].

Передбачення (прогнозування), інший важливий принцип. Функція виконує генерування, після того, коли зрозуміє слово, що буде далі. Залежно від методу, наприклад top-p sampling (nucleus), beam search, top-k sampling тощо, буде залежати й характер результату, його різнобарвність чи оригінальність.

Зазвичай, розповсюдженою технікою для підбору токена є Softmax. Алгоритм рахує вірогідність всіх вихідних елементів у векторі для подальшого використання при прогнозі, наприклад після процесу шарів для влучного збору речення.

2.2.3 Аудіо модель TTS

Text to Speech – загальний вид штучних моделей для конвертування текстового представлення в звуковий супровід. Включає різні аспекти, підтримка різних мов, акцентів, вимов та інших факторів, які наближають до відтворення людської дикції. Синтез голосу вбувається глибокими нейронними мережами, за рахунок адаптації звукових хвиль в голосові файли [21–23].

TTS технології ідеальна розробка для ніші аудіо-асистентів або озвучення продуктів, де не потребується професійна озвучка. Розглянемо на чому базується технічна складова (рис. 2.7).

Рисунок 2.7 – Процеси в TTS моделях

Головні концепції аудіо моделей:

– акустика: формування язикових конструкцій у векторне представлення;

– язикова модель: аналіз мовних особливостей, таких як вимова, дикція, акцент, інтонація тощо;

– голосова обробка: завдяки механізмів з попередніх пунктів, створюється озвучка текстового вихіднику, а потім проходять процеси тонких налаштувань звуку, фільтрації, обробки;

– вимова: тонкощі кожної підтримуваної мови вивчаються для відтворення наближеного до реалістичного голосового фрагменту на основі вхідного значення.

Роздивимось базову програмну реалізацію (лістинг 2.2) tts-1 технології.

Лістинг 2.2 Node код з синтезом мовлення tts-1:

```
const outputFile = path.resolve("./voice.wav");
async function voiceGenerate() {
  const audio = await openai.audio.speech.create({
    model: "tts-1",
    voice: "fable",
    input: "Який гарний день. Доречі, я телеграм-бот.",
  });
  const localBuffer = Buffer.from(await audio.arrayBuffer());
  await fs.promises.writeFile(outputFile, localBuffer);
};
voiceGenerate();
```

Цей застосунок озвучує вхідний інпут та зберігає файл з розширення .wav. Завдяки моделі від OpenAI є ряд тонких налаштувань, а саме вибір голосу, формату збереження, а також підтримка широкого об’єму мов різних країн.

Тому, існує широкий спектр сфер в яких можна скористатися інноваціями у розробці проєктів, такими як звуковий супровід. Інтегрувати які, буде не просто цікаво чи трендово, а дійсно корисно, наприклад для людей маючих проблеми із зором.

2.3 Методи рандомізації

Процес отримання числового або іншого значення випадковим образом, яке в свою чергу, має деякий процент вірогідності на своє генерування та не може бути відоме наперед, називається рандомізацією. Важливий інструмент для інформатики та науки в цілому.

2.3.1 Класифікація випадковості

Класифікуємо деякі типи примінення на прикладі таблиці 2.1.

Таблиця 2.1 – Види генерування рандомних значень

Тип випадковості	**Опис**	**Приклад**
1	**2**	**3**
Програмна	Необхідність отримати будь-яке число	Програмний код генератору випадкового числа
Імітаційна	Тестування наукових методів або підходів	Застосування на тваринах

Продовження таблиці 2.1

1	2	3
Природна	Без впливу зовнішніх факторів	Дощ в пустелі чи схожі нетипові явища
Штучна	Умови жеребкування через випадковість	Витягування найкоротшого сірнику
Комунікативна	Проведення збору інформації в натовпі	Опитування віку для оцінки середнього параметру

2.3.2 Математичний рандом в програмуванні

Зазвичай методи, які лежать в основі рандомізації технологій програмування вистроєні на концепції псевдовипадковості. Тобто згенеровані дані є не зовсім отриманими випадково, але на вид саме такі. Принцип їх порядку відтворення можна зрозуміти якщо дослідити алгоритми за якими вони генеруються.

Наприклад такою є JavaScript функція Math.random(). Що надає «вигадані» значення в проміжку 0 – 1 (не включаючи). Простий спосіб генератору від одиниці до ста (лістинг 2.3).

Лістинг 2.3 Реалізація генерування значення у нотації JS:

```
function getRandomValue(min, max) {
return  Math.floor(Math.random() * (max - min + 1) + min);
};
let currentValue = getRandomValue(1, 100);
```

2.3.3 Алгоритм бібліотеки random-js

Бібліотека random-js гнучке рішення з тонким налаштуванням, яке доступне завдяки установки через пакетний менеджер.

Головні відмінності від стандартних функцій генератору вигадування:

– доступність історії: наявні сіди (ідентифікатор створеного раніше показнику);

– потужність параметрів: підтримка нормального, неперервного та показникового розподілу;

– функціональність: щоб значення були непрогнозовані та унікальні в методах псевдогенерації застосовані ефективні алгоритми.

Один з таких, дуже розповсюджений в області рандомізації, а саме алгоритм Mersenne Twister:

Крок 1. Завдається стартовий вид (масив та індекс).

Крок 2. Певне значення обирається, після оброблюється внутрішніми розрахунковими діями (рис. 2.8).

Крок 3. При умові, що ряд закінчився, приймаються методи мутації елементів для їх змінення.

Крок 4. Генерується рандомне значення, якщо необхідно нове перехід до Крок 2.

Рисунок 2.8 – Розрахунковий алгоритм Мерсенна-Твіст

В результаті, якщо проєкт потребує сильного з технічної сторони методу з детальним налаштування генерації випадкових елементів, тоді ця бібліотека є максимально корисною в використанні. Алгоритми на яких базуються дійсно створюють надійну систему без повторень на дистанції, а також швидкодіючими можливостями роботи з великими об’ємами даних.

В разі досить простого застосунку можливо доцільніше звернутися до стандартного методу інтегрованого у мови програмування, який задовільнить потреби без підключення зовнішніх доповнень та виконає свої функції.

Подібні інструменти є важливим ресурсом до тестування структур даних, утворення процесів механіки поведінки у іграх або надійних ключів у крипто індустрії.

3 ПРАКТИЧНА ІМПЛЕМЕНТАЦІЯ ТЕЛЕГРАМ-БОТА

3.1 Вибір програмної області та засобів для досягнення мети

Під час виконання кваліфікаційної роботи було прийнято рішення реалізувати телеграм-бота, який має на меті допомагати з навчанням у сфері програмування та вивчати нові технології. Основні задачі – це перевірка знань методом тестування з різних видів питань, надання інформації або поради, яку бажає отримати користувач, завдяки асистенту на базі штучного інтелекту та UI для можливості подивитись roadmap конкретних технологій під час шляху вивчення.

Знання мають властивість забуватися, тому зручний інструмент, який завжди під рукою, гарний варіант мати спосіб повторювати матеріал. Також завдяки в основному контенту у вигляді тексту, не потребується мати дуже високу швидкість інтернет підключення, а заряд батареї буде витрачатися досить економно. Цей бот оптимальне рішення для підготовки до технічної бесіди під час влаштовування на роботу чи до іспиту з мови програмування.

IDE, яке слугувало засобом ініціалізації проєкту було Visual Studio Code. Це крос-платформне середовище написання коду від корпорації Microsoft. Сильні сторони, які роблять його популярним у ком’юніті:

– плагіни: багатий маркет з доповненнями для підвищення продуктивності чи комфорту;

– універсальність: доступ на всіх ОС та профілі для збереження налаштувань юзера через GitHub акаунт;

– ціна: точніше її відсутність, цей продукт безкоштовний;

– контроль над версіями: можливість зручного використання git;

– технології: наявність великого списку мов програмування.

А також аспект дизайну та зрозумілості з яким зазвичай не виникає проблем (рис. 3.1).

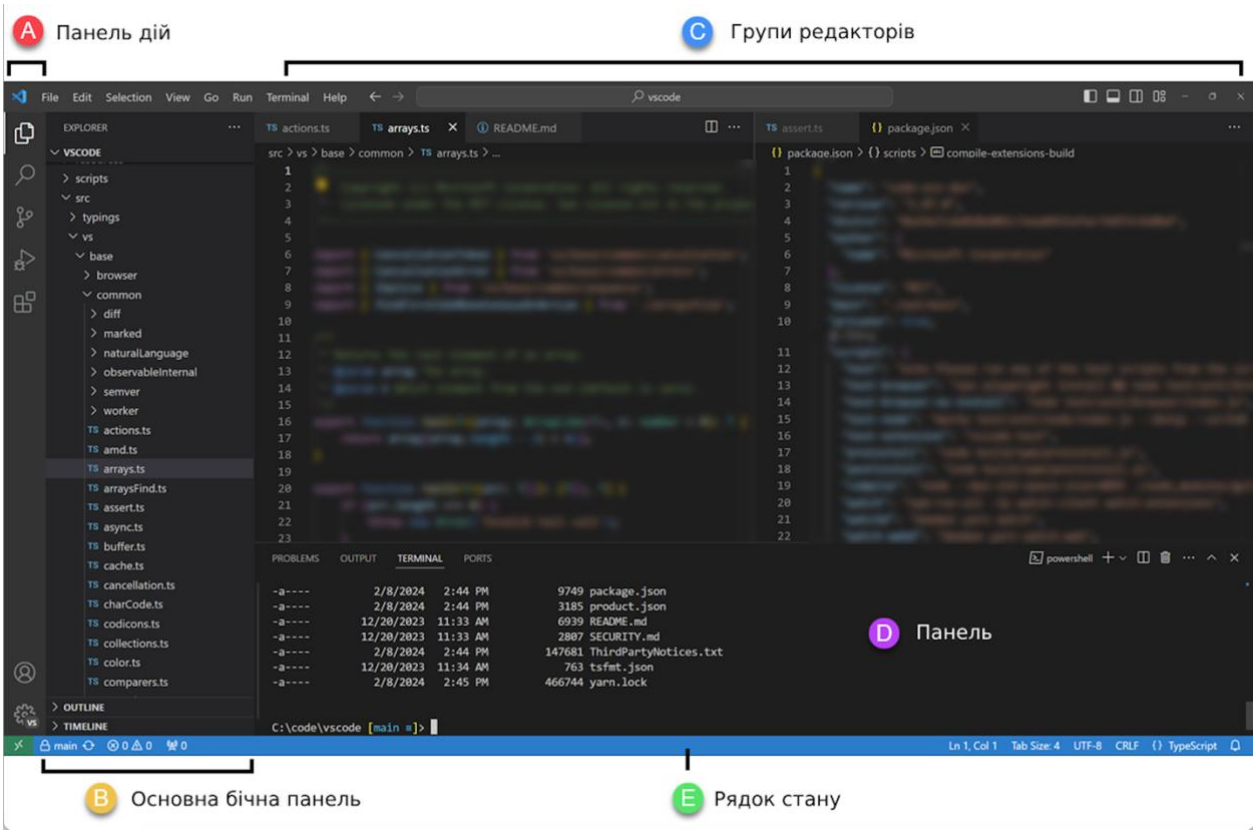


Рисунок 3.1 – Інтерфейс VS Code

Одним з важливих аргументів вибору технології Node.js стала наявність бібліотеки для взаємодії з месенджером телеграм під назвою GrammyJS. Яка в свою чергу вміє виконувати усі існуючі команди надані офіційним Telegram Bot API. В тому числі може приймати та оброблювати дані web_app_data, які потрібні для інтеграції функціоналу Telegram Mini Apps.

Node, який компілює бекенд JavaScript код, потужний та швидкодіючий інструмент, який в контексті застосунку допомагає з асинхронними та іншими потребами працездатності. Інша важлива річ, наявність способу легкої інсталяції залежностей проєкта різних версій завдяки NPM. Структура на рисунку 3.2, відображає наявні компоненти у системі.

Рисунок 3.2 – Інстальовані залежності проєкту

Серед яких, не згадані .env та nodemon. Де перший, зберігає API токени, щоб не відбулося їх компрометування, а другий, автоматизує процес внесення змін коду після збереження файлів до серверу.

Необхідності в використанні БД не було, тому тестові запитання збережені у JSON файлі (лістинг 3.1). Вони складають з себе два різновиди, усного характеру з можливістю перевірити себе та з кнопками вибору відповіді.

Лістинг 3.1 Приклад вигляду обох типів питань:

{
“id”: 3,
“question”: “Що таке стрілкова функція в JavaScript?”,
“isClosedType”: false,
“answer”: “Стрілкові функції – це …”
},
{
“id”: 16,
“question”: “Оператор для порівняння на строгу рівність?”,
“isClosedType”: true,
“variants”: [

{
“id”: 1,
“context”: “==”,
“isTrue”: false
},
{
“id”: 2,
“context”: “===”,
“isTrue”: true
}, ...]
}

Angular – сучасний фронтенд фреймворк, який дозволяє будувати застосунки за допомогою компонентів. Добре показує себе з відзивними односторінковими програмами. Також є зручний спосіб будувати маршрути до частин аплікації. Це робить дану технологію оптимальною для візуальних задач.

За допомогою підключення скрипту telegram-web-app є доступ до його методів та стилів (лістинг 3.2), що дає змогу повної свободи для створення інтерактивного інтерфейсу вебзастосунку в середині бота.

Лістинг 3.2 Приклад стилізування елементу під користувацьку тему:

```
body {
background: var(--tg-theme-bg-color);
color: var(--tg-theme-text-color);
}
button {
background: var(--tg-theme-button-color);
color: var(--tg-theme-button-text-color);
}
```

Такий спосіб дає змогу створення власного дизайнерського рішення, але з дотримуванням спільного дизайн-коду з ТГ для балансу візуального вигляду. При такому розкладі можна добитися результату однакової екосистеми власного брендингу та існуючого виду соціальної мережи. Що у результаті буде виглядати красиво, актуально та стильно.

Іншою суттєвою деталлю є, що в цьому кейсі при зміні світлого чи темного моду або теми, які змінюють кольори блоків всього месенджеру, стилі будуть автоматично адаптовуватися к стандартним палітрам, які вже підвлаштовані на приємні оку поєднання. На виході маємо уникнення теоретичної проблеми зі збігом однакових барв шрифту та фону через зміну інтерфейсу користувача в Telegram.

3.2 Розгортання та тестування сервісів

Для функціонування всіх здатностей телеграм-бота, потрібно налаштувати доступ у певних сервісах. OpenAI потрібен для працездатності аудіо та текст підказок від «розумного помічника», а Ngrok для розміщення вебчастини застосунку у мережі. Конкретніше буде оглянуто у відповідних пунктах.

3.2.1 OpenAI API

OpenAI є організацією розташованою у США, яка займається розвитком штучного інтелекту [24]. Їх ціль – пустити в маси нові технології, які будуть використовувати для покращення світу. Враховуючи пагубні фактори, новітні розробки дуже обережно досліджують перед розповсюдженням задля захисту людства. Завдяки їх платформі, де є змога отримати доступ до API, було впроваджено штучні моделі до проєкту.

Деякі види доступних моделей:

– DALL-E: генерування або редагування зображення [25, 26];

– Whisper: конвертування аудіо до тексту;

– TTS: озвучення тексту голосом наближеним до реалістичного;

– GPT: створення програмного коду чи іншої інформації за промптом;

– Moderation: аналізування на зв'язок неприйнятного типу контенту.

Щоб інтегрувати функціональні можливості, потрібно пройти відповідну процедуру:

Крок 1. Створити обліковий запис та підтвердити номер мобільного телефону.

Крок 2. Згенерувати новий ключ для проєкту в розділі API keys (на цьому етапі його вже можна додати до програмного коду застосунку).

Крок 3. Поповнити баланс платформи (для змоги користування сервісом потрібно мати кошти, але для нових акаунтів надається стартовий депозит в розмірі 5$ на строк протягом 3 місяців, яких для тестових засобів буде достатньо).

Крок 4. Ознайомитися з синтаксисом та прайс-курантом потрібної моделі в документації.

Крок 5. Перейти в розділ Usage (рис. 3.3) для огляду діаграм щодо кількості запитів та витрат на кожну штучну модель або усі разом за певний термін.

Рисунок 3.3 – Діаграми використання gpt-3.5-turbo у квітні

3.2.2 Платформа Ngrok

Ngrok є інструментом для тимчасового тунелювання локального серверу до публічної мережі інтернету. Ідеальний варіант для швидкого тестування продуктів завдяки своїй простоті та легкості виклику через термінал.

Сервіс має HTTPS, обов’язкову умову для додання TG Mini Apps, встановлену через турботу про конфіденційність та особисті дані. Таким чином, аспект безпеки налаштований приховувати IP-адресу, бо надається домен від платформи, тим самим трафік захищається від DDOS спроб нашкодити працездатності серверу. Ще Ngrok як посередник є більш надійним варіантом уникнення витіку інформації й адаптований для масштабних об’ємів навантаження задля безперебійного існування в мережі.

Що вміє Ngrok:

– демонстрація локальних рішень та застосунків, наприклад замовнику;

– тестування Webhooks, реагування на пробні запити [27];

– перевірка функціональності мобільних програм в контексті комунікації з бекендом, прямо з персонального комп’ютера програміста без хосту чи інших віддалених рішень.

Установка та перевірка дієздатності, здійснюється за наступним принципом:

Крок 1. Встановити на любій операційній системі, для MacOS «brew install ngrok/ngrok/ngrok» у консолі пристрою.

Крок 2. Поєднати з аунтифікатором (заздалегідь провести реєстрацію на сайті для його отримання) для цього «ngrok config add-authtoken <КЛЮЧ>».

Крок 3. Провести першочерговий запуск сервісу «ngrok http http://localhost:цифрове_значення», де після успішного виконання команди

відобразиться детальна інформація (рис. 3.4) та можна буде побачити спеціальний HTTPS URL.

Крок 4. Задати статичний домен, щоб кожен раз не оновлювати його, перейти у власний профіль Ngrok, створити його у відповідному розділі (Cloud Edge/Domains), а потім виконати «ngrok http цифрове_значення --domain символьне_значення.ngrok-free.app».

Крок 5. Далі (по бажанню) можна обмежити доступ, зробити вхід по OAuth чи Basic Auth, для підстрахування від випадкових відвідувань, детальніше у документації [28].

Крок 6. Після цього використати отримане особисте посилення для активації ТГ міні-застосунку у BotFather боті (рис. 3.5) та запустити інструмент, як у Крок 3, але з постійною вебадресою з Крок 4.

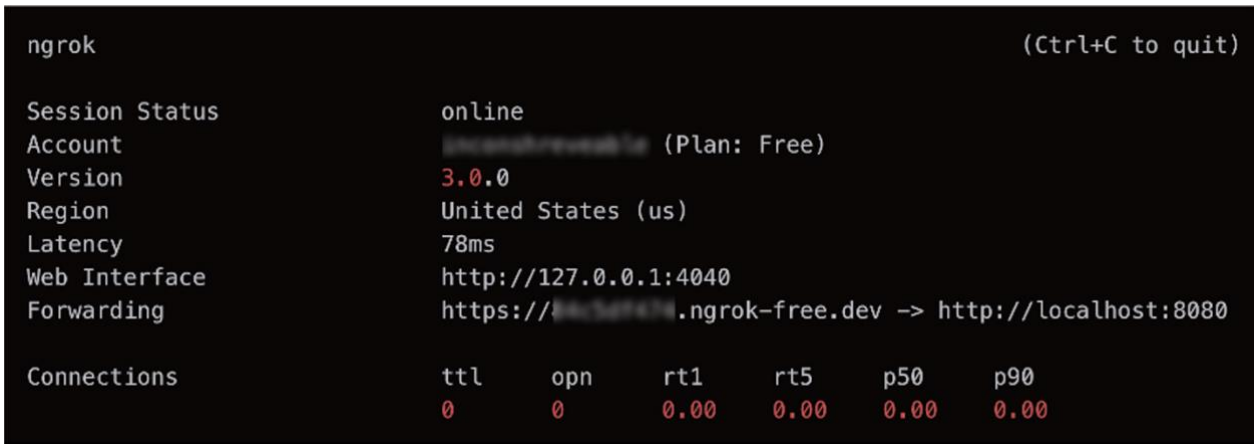


Рисунок 3.4 – Коротке зведення інформації щодо поточного стану

А також, щоб був доступний localhost, передчасно розпочати проєкт. Структура розподілена на дві технічні частини (в наступному пункті буде наведено у архітектурі програмної реалізації кваліфікаційної роботи), одна складається з самого бота, інша з вебзастосунком, зараз розглядається другий випадок. Якщо в синтаксисі Node, для цього необхідно виконати «npm start», то в нотації Angular CLI (Command Line Interface для виконання рядків цього середовища) – це «ng serve». Після чого програма почне білдиться (компілюватись) та стане локально доступною.

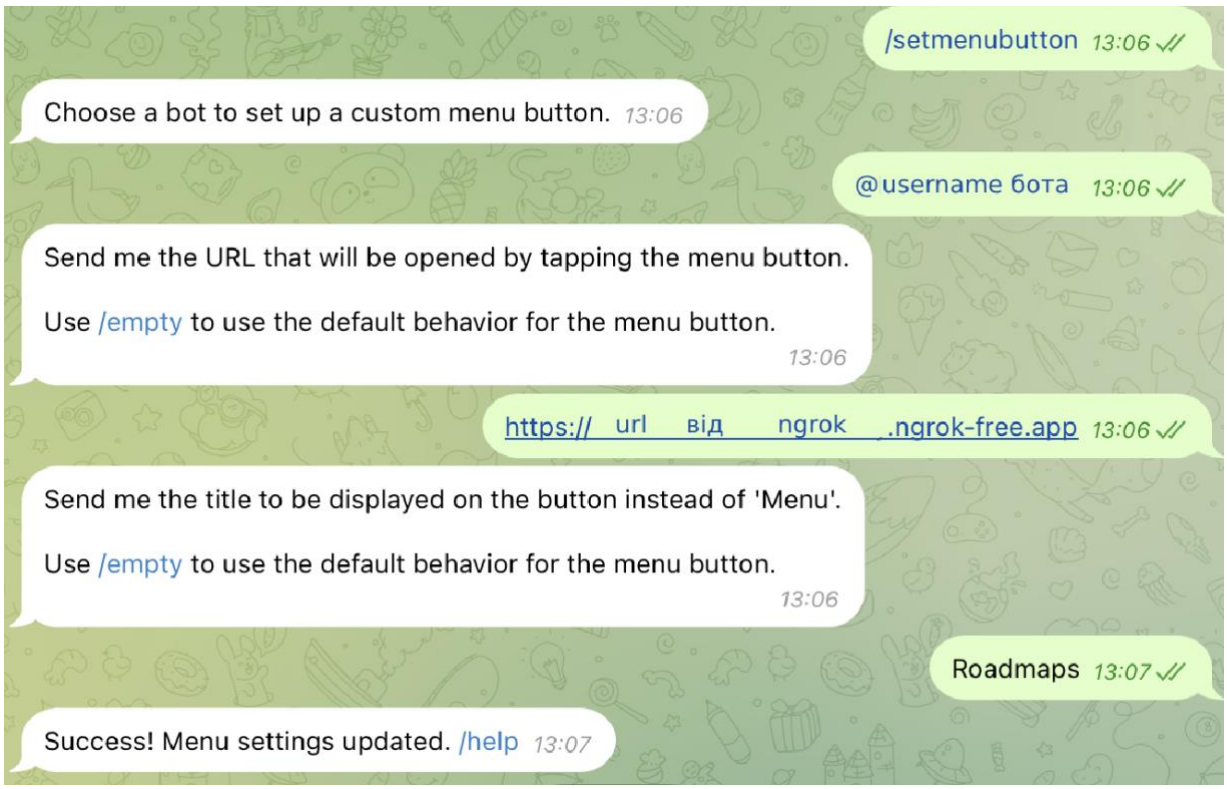


Рисунок 3.5 – Інтеграція Telegram Mini Apps у головному боті

3.3 Імплементація бота

Для розуміння всіх елементів структури було створено загальну архітектуру даної роботи (рис. 3.6). Кожен зображений об’єкт виконує свою функцію та є важливим при етапах роботи застосунку.

Кожна технологія є абстрактною, тобто вибрана з урахуванням особистих уподобань та цілей реалізації. За схемою можна легко змінити мову програмування, фреймворк чи бібліотеку та не втратити логіку бот-застосунку.

Таким чином, є багато можливих горизонтів для додавання нових механік у бота, а саме інші штучні моделі для нових сервісів або наприклад, СУБД (систему управління БД) для «кращої пам’яті» кожного гостя та його потреб.

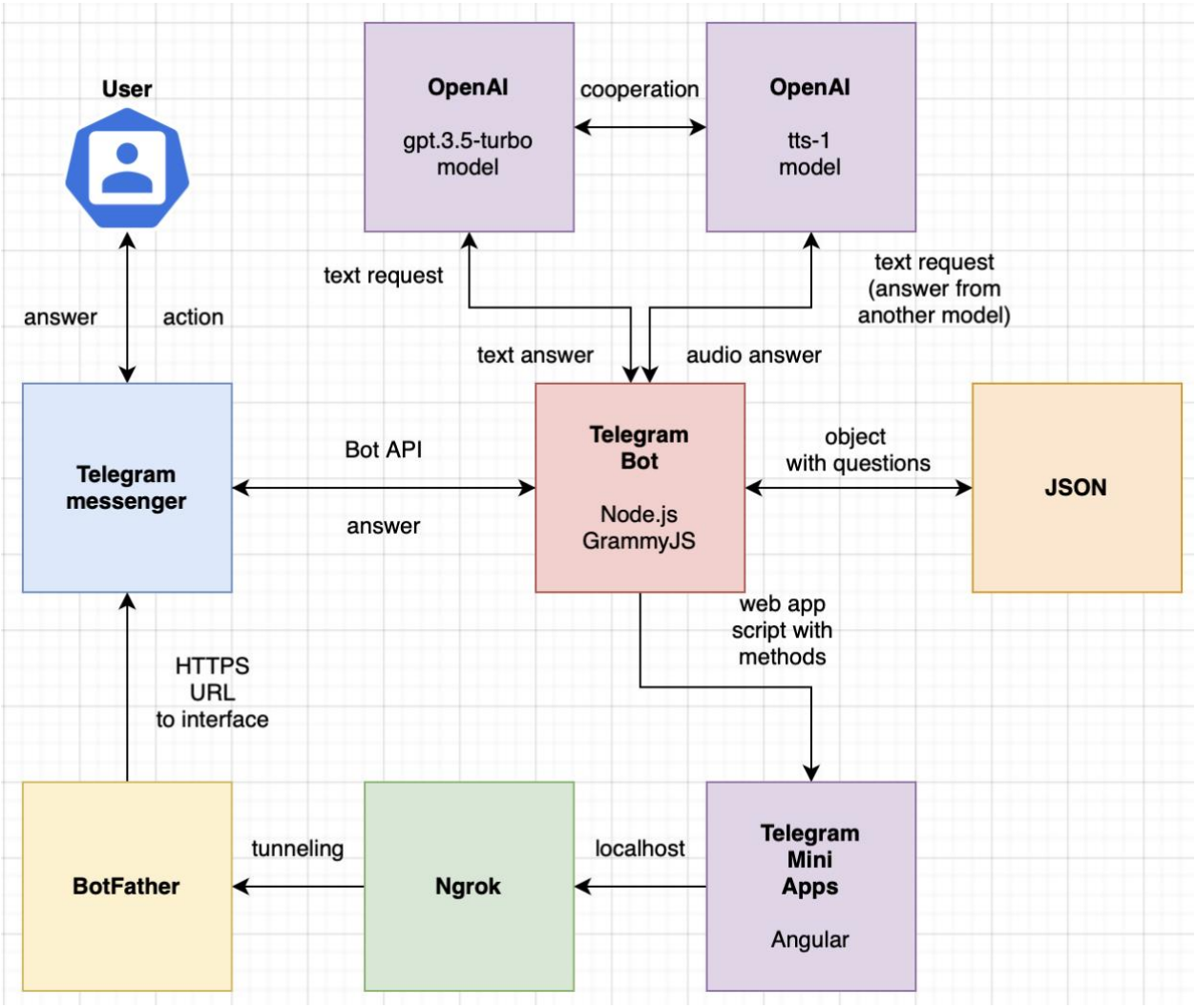


Рисунок 3.6 – Загальна архітектура телеграм-бота

Узагальнена характеристика щодо процесів:

– User: людина, яка використовує телеграм-бота;
– Telegram messenger: софт, яким користується клієнт;
– Telegram bot: середовище з надаваємими операціями;
– OpenAI: механізм зі штучним інтелектом;
– GPT-3.5: процедура генерування контенту;
– TTS-1: процедура подання штучного голосу;
– JSON: сховище даних;
– Mini Apps: інтерактивний інструмент взаємодії з UI;
– Ngrok: модуль для надання серверу публічності;
– BotFather: метод об’єднання з месенджером.

Один з етапів розробки є фактичне створення екземпляру нового бота та одразу моделей OpenAI (лістинг 3.3) через методи надаваємі технологією для звернень до ТГ-роботів Grammy, а саме конструктори «new Bot» та «new OpenAI».

Лістинг 3.3 Екземпляри об’єктів доданих конструкторів з бібліотек:

```
const tgBot = new Bot(process.env.TELEGRAM_TOKEN);
const openai = new OpenAI({
apiKey: process.env.OPENAI_TOKEN
});
const url = process.env.WEB_URL;
```

Самі токени з відповідними назвами зберігаються в .env файлі, а також публічне посилання на вебзастосунок для створення навігації. Цей код дає змогу почати додавати обробники подій та реагування на певні слова чи фрази або будь-які інші функції, наприклад клавіатуру з опціями.

Інший етап – це впровадження відловлювання можливих помилок для створювання кращого розробницького досвіду, лістинг на рисунку 3.7 [29]. Та команда старт, яка активізує бота.

```
bot.catch((err) => {
    const ctx = err.ctx;
    console.error(`Error while handling update ${ctx.update.update_id}:`);
    const e = err.error;

    if (e instanceof GrammyError) {
        console.error("Error in request:", e.description);
    } else if (e instanceof HttpError) {
        console.error("Could not contact Telegram:", e);
    } else {
        console.error("Unknown error:", e);
    }
});

bot.start();
```

Рисунок 3.7 – Лістинг відловлювання помилок

Далі, можливість звернутися до «розумного консультанта», який буде завдяки повідомленню із пізнавальною конструкцією «/help», а після прохання чи запит, надавати відповідь в чаті, а ще її промовляти через голосове сповіщення, лістинг можна побачити на рисунку 3.8 та 3.9.

```
else if (message.text && message.text.startsWith('/help')) {

  try {
      const prompt = message.text.replace('/help', '');

      const response = await openai.chat.completions.create({
          model: 'gpt-3.5-turbo',
          messages: [{ "role": "user", "content": prompt }],
          temperature: 0.8,
          n: 1,
      });

      if (!response || !response.choices || response.choices.length === 0) {
          throw new Error('Empty response from OpenAI API');
      }

      ctx.react('👍');
```

Рисунок 3.8 – Програмний код з генерацією контенту методом AI

```
async function generateVoice(text) {
    const mp3 = await openai.audio.speech.create({
        model: 'tts-1',
        voice: 'echo',
        input: text,
    });

    const speechFile = path.join(__dirname, 'speech.mp3');
    const buffer = Buffer.from(await mp3.arrayBuffer());

    await fs.promises.writeFile(speechFile, buffer);
    return speechFile;
}
```

Рисунок 3.9 – Програмний код з озвученням аудіо методом AI

Тепер, здатність взаємодіяти з методом web_app_data, який в свою чергу встановить зв'язок між серверною частиною та кодом з візуальним інтерфейсом. Тобто через сервер Telegram, буде об’єднано два під-проєкти для отримання спільних даних. Це приведе до умов для побудови функціоналу вебсторінки для зворотнього зв’язку, побажань або зауважень від юзерів (рис. 3.10).

```
bot.on('message', async (ctx) => {
    const message = ctx.message;

    if (message.web_app_data) {
        const data = message.web_app_data.data;

        if (data) {
            try {
                const jsonData = JSON.parse(data);

                const usernameFrom = message.from.username;
                const userFirstName = message.from.first_name;

                if (jsonData.feedback) {
                    await ctx.reply(`Відправлене повідомлення: <b>"${jsonData.feedback.trim()}"</b>
<i>від ${userFirstName} - @${usernameFrom}</i>`, { parse_mode: 'HTML' });
                } else {
                    await ctx.reply('Некоректні дані');
                }
            } catch (error) {
                await ctx.reply('Помилка з обробкою даних');
            }
```

Рисунок 3.10 – Лістинг для зворотнього зв’язку з об’єкту web_app_data

У цей момент, підключимо скрипт для обробки у Angular (лістинг 3.4) від офіційного TG клієнту. Це надасть доступ до об’єкту window.Telegram.WebApp. Який дозволить комунікувати з існуючими методами.

Лістинг 3.4 Підключення Web App скрипту:

<head>

<script src=https://telegram.org/js/telegram-web-app.js></script>

</head>

<body>

<app-root></app-root>

</body>

Приклад заповнення карток після переходу до відповідної технології з загального списку на рисунку 3.11. Це вміст сторінок на яких юзер може ознайомитись з шляхом вивченням мови програмування, оглянути базові факти про неї та перейти на вебсайт з практичними вправами.

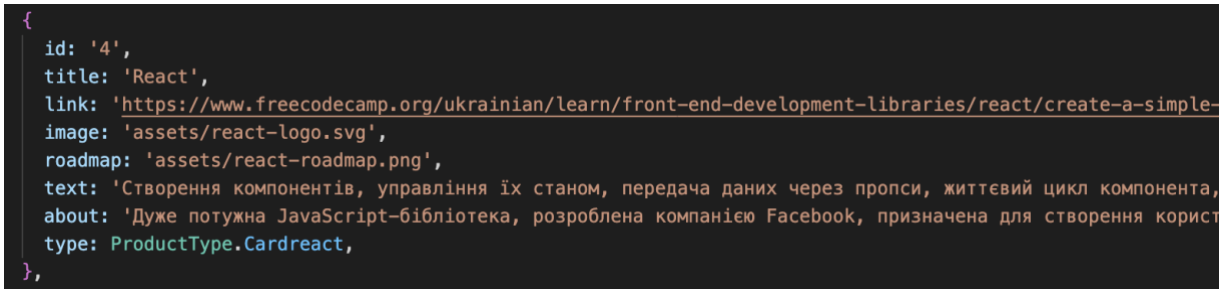


Рисунок 3.11 – Наповнення картки roadmap розділу

Іншою деталлю реалізації, було створення роутів для переходу по маршрутам Telegram Mini Apps UI. Вони надають змогу для переміщення до загального меню інтерфейсу з переліком контенту та сторінок з їх змістом. А також інша кнопка для виклику сторінки фідбеку. Наглядніше буде на лістингу 3.5.

Лістинг 3.5 Маршрутизація сторінок:

const routes: Router: [

{ path: '', component: ListPageComp },

{ path: 'info', component: InfoPageComp },

{ path: 'tech/:id', component: TechPageComp }

];

Також потрібно відзначити основний компонент, він залежно від роутів, буде мати у собі інші під-компоненти, які будуть оновлюватися

шляхом змінення розташування користувача та надавати актуальний зовнішній вигляд сторінки перебування. Це відбувається через «<router-outlet>» (рис. 3.12).

```
@Component({
    selector: 'app-root',
    standalone: true,
    imports: [CommonModule, RouterOutlet],
    template: `<router-outlet />`,
})
```

Рисунок 3.12 – Структура головного компоненту

Таким чином, налаштовуємо усі потрібні компоненти для фронтенду. В зображеному коді (рис. 3.13) задається склад змісту info сторінки, де параметр template, декоратора @Component утворює шаблон. Він містить форму, яка через ввод до інпуту буде виконувати метод події відповідного івенту.

```
@Component({
  selector: 'app-info',
  standalone: true,
  template: `
    <form class="middle view">
      <h1 class="lw-space">Зворотній зв'язок</h1>
      <textarea
        [value]="info()"
        (input)="handleSetting($event)"
        class="check-view"
      ></textarea>
    </form>
  `,
})
```

Рисунок 3.13 – Лістинг компоненту app-info

3.4 Користувацька інструкція з використання

Аби мати змогу використовувати усі бенефіти від платформи Telegram, не потребується робити нічого складного. По-перше, мати доступний телеграм-клієнт, наприклад завантажити з App Store чи Google Play для мобільної версії або скористатися їх вебсайтом для решти систем [30, 31]. По-друге, знайти у пошуковій стрічці «комплексного-помічника» за його ім’ям та зображенням профілю (рис. 3.14).

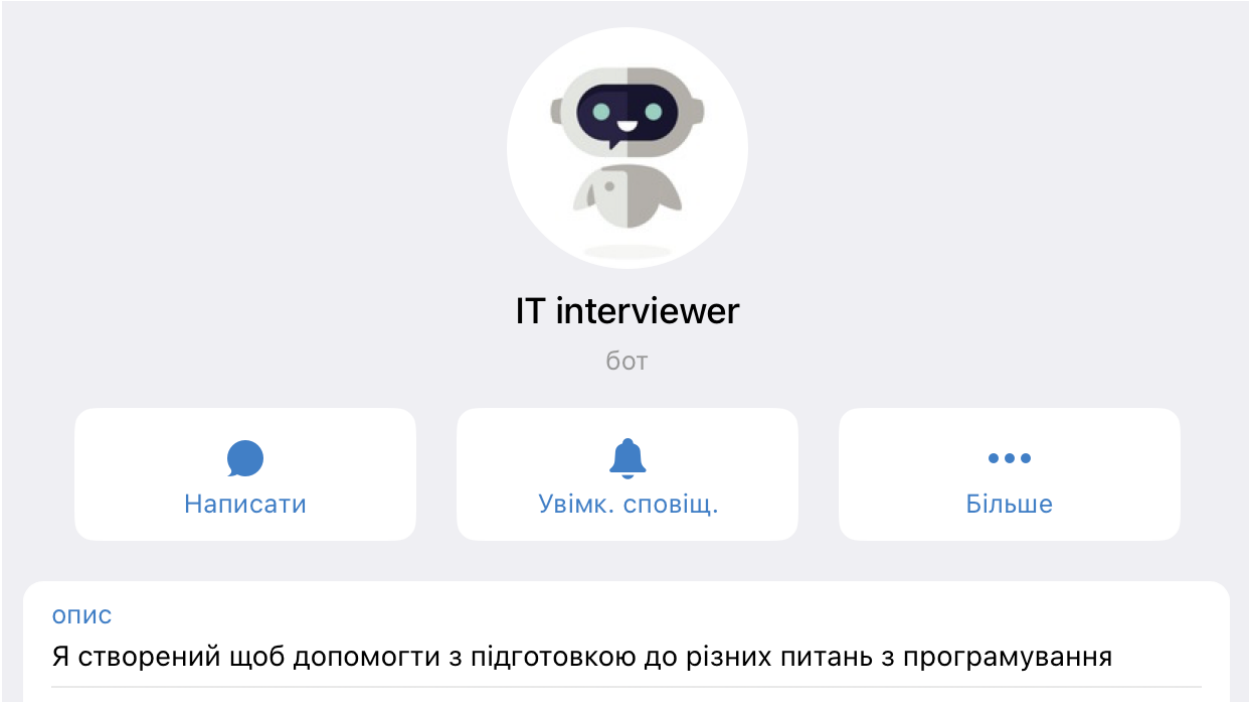


Рисунок 3.14 – Крок із знаходженням бота

Після чого, коли звернутися до нього, відобразиться стартова сторінка бесіди з однією кнопкою, яка активізує функціональні можливості та створить нову сесію для поточного користувача. Рисунок 3.15, ілюструє саме цей процес.

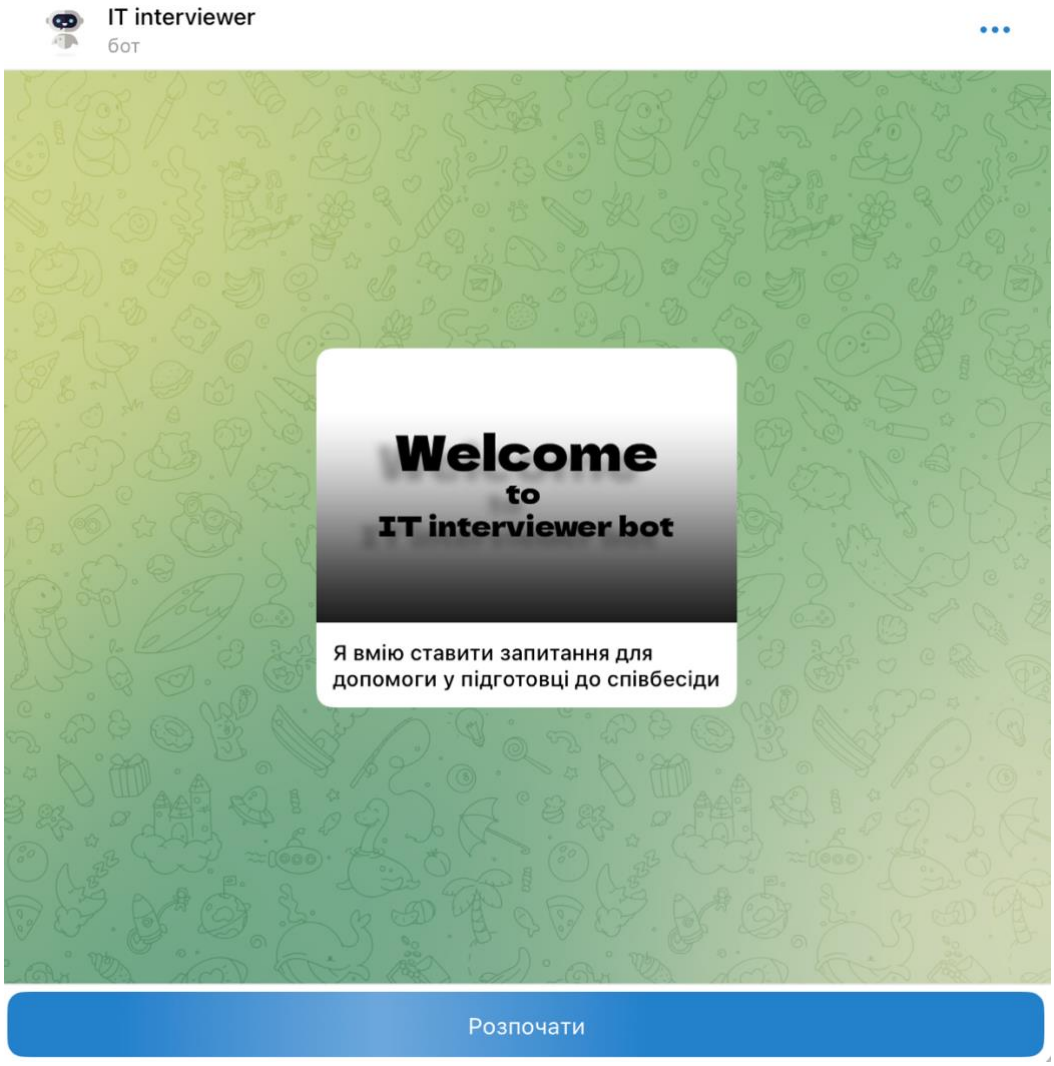


Рисунок 3.15 – Спосіб запуску телеграм-бота

Наступний етап, огляд отриманого вікна месенджеру після переведення в функціонуючий стан (рис. 3.16).

Тут можна побачити декілька основних моментів:

– отримані повідомлення, що свідчать про працездатність системи та розповідають, що саме можна зробити в першу чергу;

– клавіатура, яка надає «хот-кеї» для швидкого звернення до переліку функцій;

– запит до опції «Документація», який був успішно виконаний.

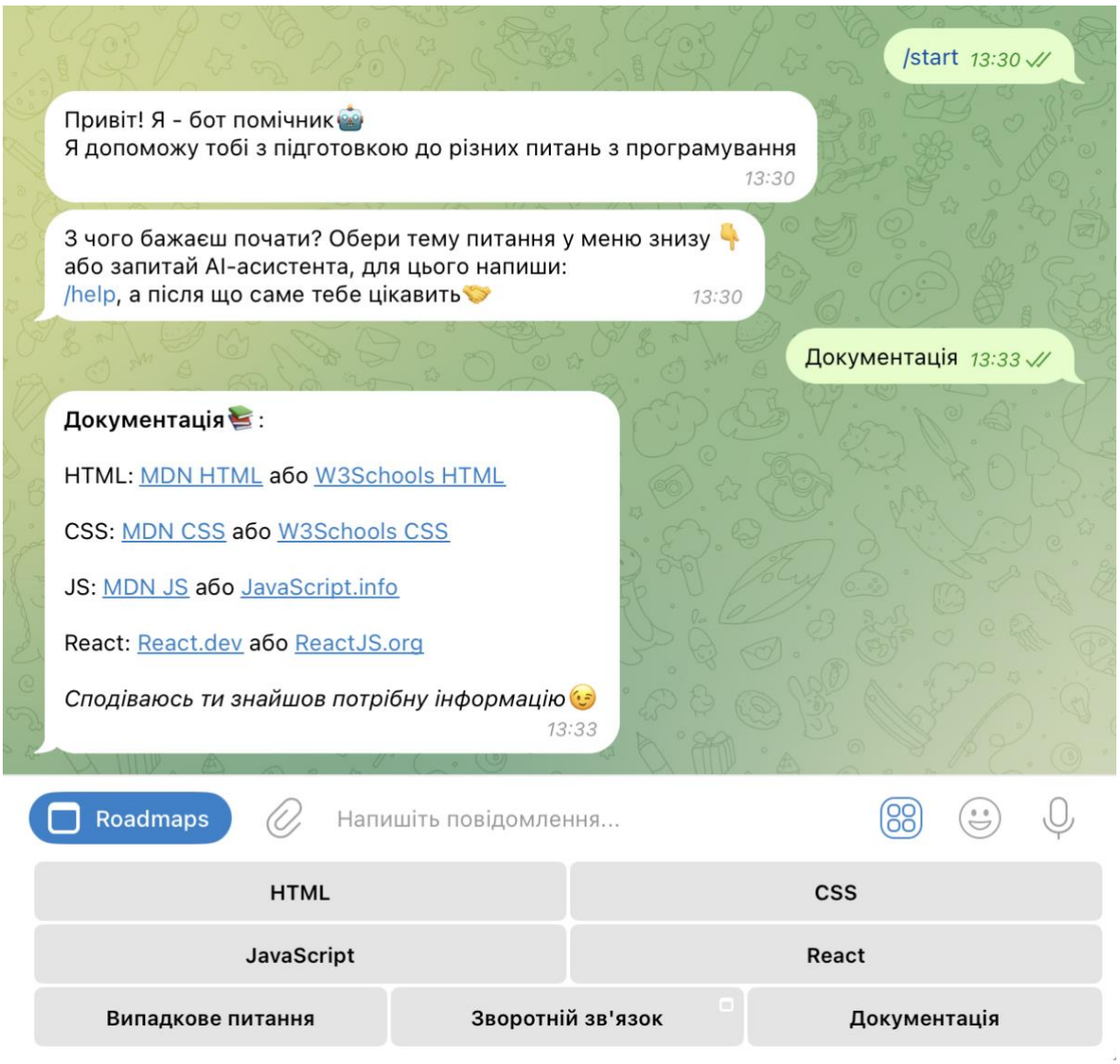


Рисунок 3.16 – Етап першої комунікації

Іншим кроком демонстрації здатностей, буде виклик із створеної клавіатури, категорії «React» і «Випадкове питання». Бот видав випадкові тести з цих тем, а ще й різних типів навчання.

Також в свою чергу, кнопка з рандомним запитанням – надала також реакт. Ілюстрація на рисунку 3.17, де в першому випадку є можливість обрати, що правильно, в іншому перевірити себе самостійно, а потім переглянути чи це вірно.

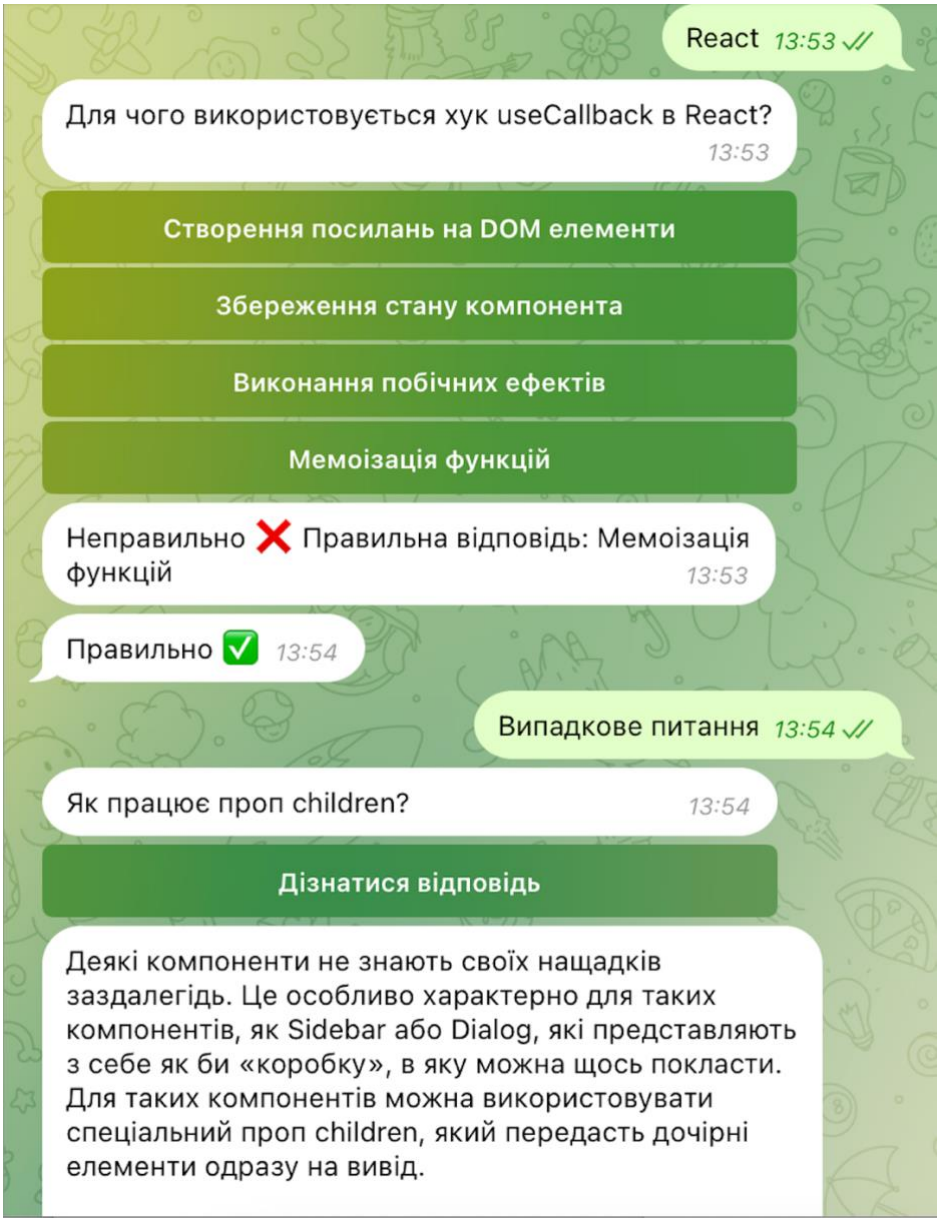


Рисунок 3.17 – Тестування перевірки знань

Зараз, створимо певний «промпт» до штучного інтелекту, який буде мати, те що цікаво дізнатися у своєму складі (рис. 3.18 та рис. А.1). Важливо на початку використати префікс «/help».

У відповідь отримаємо згенероване повідомлення з контентом, який був запитаний. А в додаток, звукове представлення цього ж повідомлення для зручності, бо зазвичай відповідь довгого характеру, який не завжди є змога комфортно прочитати, наприклад у транспорті.

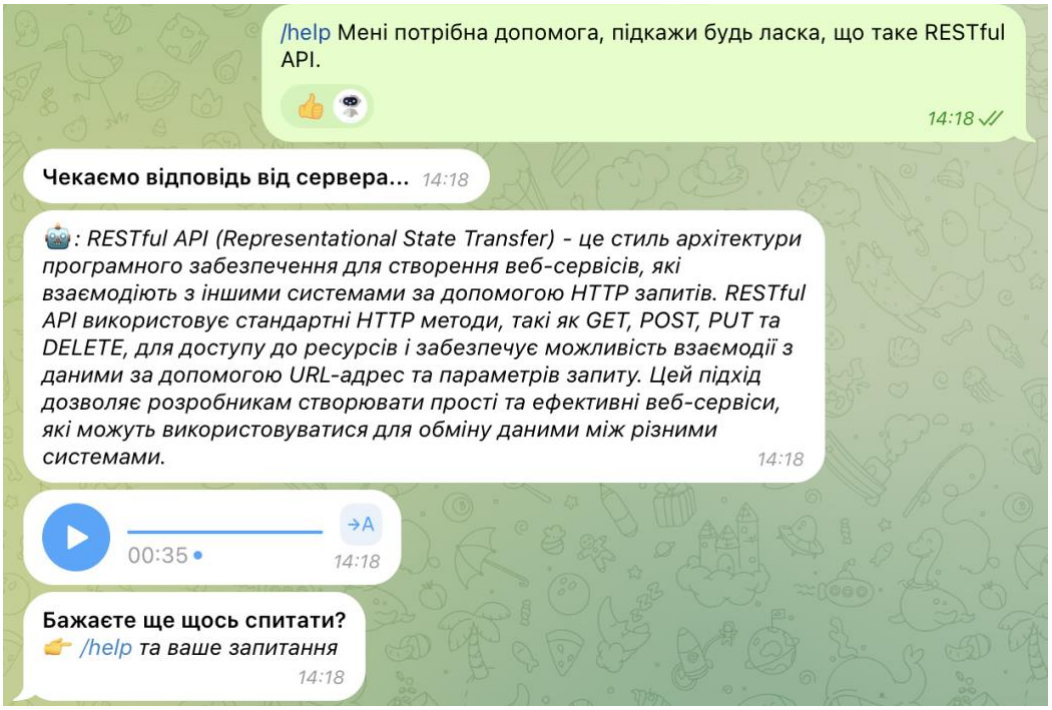


Рисунок 3.18 – Застосування інтелектуальних моделей

Далі, можна перейти до інтерфейсу Telegram Mini Apps. Спочатку, можливість повідомити юзеру про знайдений баг, друкарську помилку чи надати оцінку задоволеності від користування (рис. 3.19 та рис. 3.20).

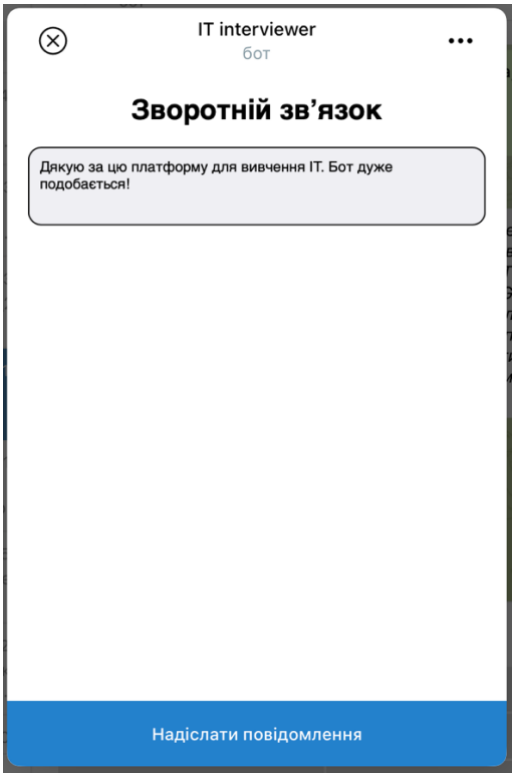


Рисунок 3.19 – Зворотній зв'язок користувачів

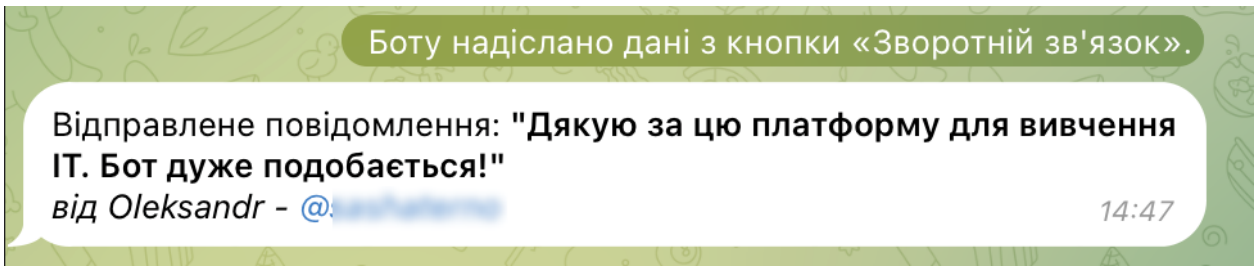


Рисунок 3.20 – Надіслане повідомлення від користувача

Тепер, час подивитись на інші варіанти цього інтерфейсу, які знаходяться вже не у кастомній клавіатурі, а у спеціальній Web App кнопці з назвою «Roadmaps». Там знаходиться список з технологіями, які в даний момент підтримуються застосунком. Якщо перейти до однієї з них, то там можна дізнатися чітку й лаконічну інформацію про неї та отримати посилання до матеріалу з базовою практикою. І звісно, побачити roadmap, де можна зрозуміти на які контрольні точки звертати увагу при вивченні. Приклади виборчого візуального вигляду на рисунках 3.21–3.23 та А.2.

Рисунок 3.21 – Список підтримуваних технологій на прикладі JS

Рисунок 3.22 – Список підтримуваних технологій на прикладі HTML

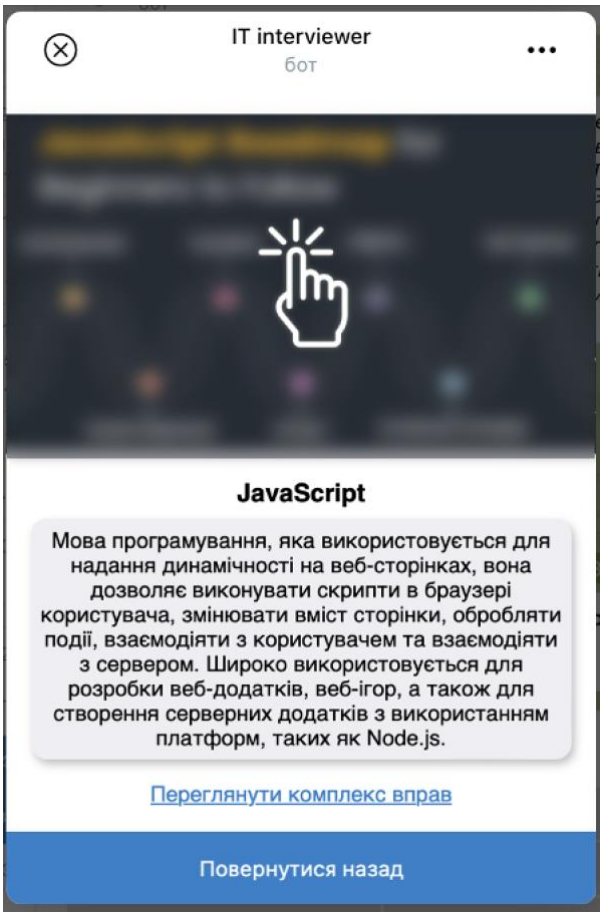


Рисунок 3.23 – Вид вибраної зі списку мови JS

Після ознайомлення є змога повернутися назад до загального меню, натиснувши по однойменній кнопці.

Якщо знову звернутися для детальнішого аналізу штучної аудіо моделі, то можна перевірити її якість. Завдяки функції конвертування голосових повідомлень до тексту (рис. 3.24), вбудованої опції телеграм для преміум користувачів (працює вона не ідеально на великих обсягах інформації). Можна побачити результат роботи інтегрованої моделі, який є тим самим, що і текстове представлення, яке спочатку надавав телеграм-бот.

Хоча кінцівка контексту була відсутня (зі сторони офіційної функції месенджеру Telegram, бо на рисунку 3.25 видно та у плеєрі чутно, наявність решти слів), напевно через об’ємний хронометраж, все рівно – це доказує точність реалізованого сервісу. Таким чином, підтверджено, що озвучення є на високому рівні та виконує своє завдання дуже добре.

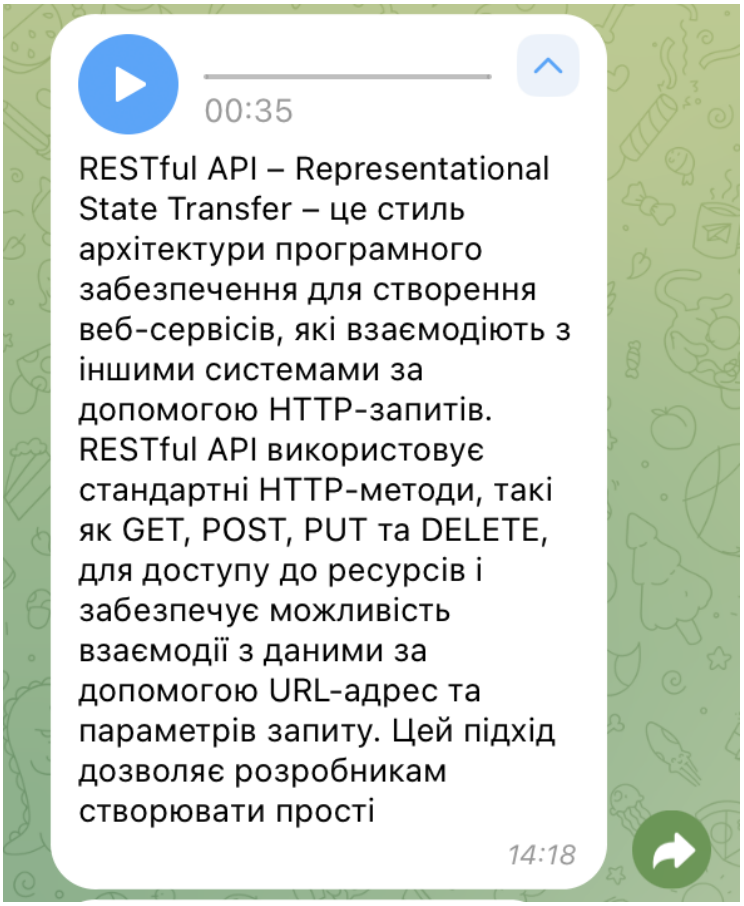


Рисунок 3.24 – Тестування якості моделі через вбудовану опцію TG

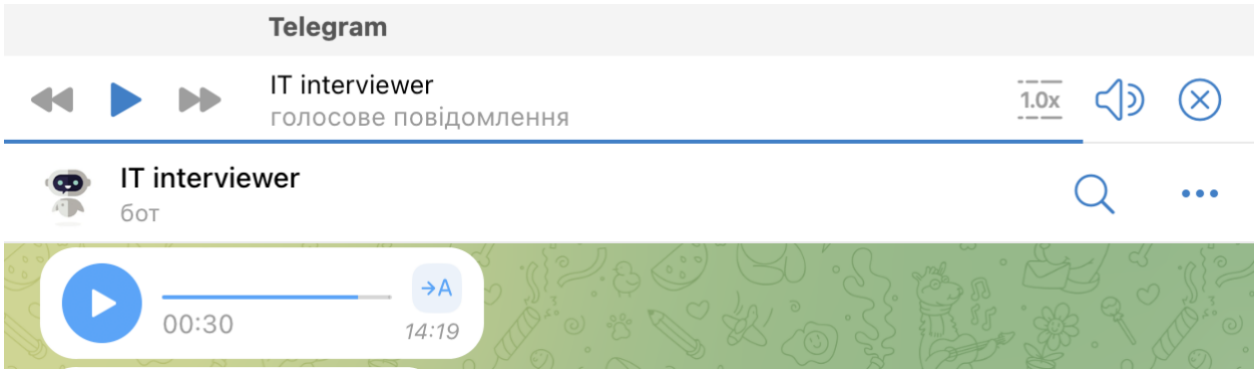


Рисунок 3.25 – Часові межі з рештою слів

Отже, телеграм-бот виконує поставлені йому комплексні задачі якісно і швидко. Використання розроблених функцій повинно принести користувачам задоволення, нові знання та корисні навички в сфері програмування.

ВИСНОВКИ

У рамках кваліфікаційної роботи було втілено практичну реалізацію телеграм-бота для генерації контенту на основі запитів з використанням штучного інтелекту.

Під час виконання розробки створено можливість відповідати на запити користувача через штучну модель gpt-3.5-turbo та озвучувати цю відповідь через аудіо-модель tts-1. Розроблено здатність надання випадкових питань через бібліотеку random-js. Здійснено інтерактивний інтерфейс на базі Telegram Mini Apps, який завдяки тунелюванню платформою Ngrok стає глобально доступним в мережі.

Результатом роботи є виконання запланованих цілей, таких як дослідження вибраних технологій, а саме NodeJS для серверної частини застосунку та бібліотеки GrammyJS для взаємодії з Telegram Bot API. Аналіз алгоритму отримання доступу до функціональних можливостей надаваємих OpenAI. Реалізація Web App UI завдяки фреймворку Angular та інтеграція у середовище боту через BotFather [32]. Режим перевірки знань за переліком питань JSON файлу, який має 4 різні теми та опцію одночасного їх примінення.

Підсумкова версія продукту – це корисний засіб отримання знань не тільки для початківців в області комп’ютерних наук, а й для досвідчених розробників, які також зможуть знайти щось для себе. Це може бути повторення теоретичних концепцій, ознайомлення з синтаксисом нової технології або ж звичайне бажання самовдосконалитись під час вивчення чогось корисного.

А в свою чергу, початківці отримують можливість понуритися в світ програмування з великим набором засобів та інструментів, які допоможуть зробити процес пізнання нового матеріалу набагато структурованішим, продуктивнішим та ефективнішим для засвоєння та використання в практичних умовах.

В майбутніх планах подальше масштабування проєкту, наприклад додання ширшого спектру підтримуваних технологій програмування до системи, підключення бази даних для неповторності запитань в клієнтській сесії, окрім умови, коли клієнт сам забажає почати тестування спочатку, інтегрування дошки лідерів для спортивної конкуренції між учасниками методом надання відміток за якісне проходження тестів та занесенням до топу тижня або місяця людей з кращими показниками, з подальшим розподіленням за певними лігами залежно від рівня знань, а також розміщення телеграм-бота на хмарному ресурсі для неперервного функціонування застосунку.

Результати роботи апробовано у вигляді тез доповіді під час XVIII Міжнародної науково-практичної конференції «Modern challenges: trends, problems and prospects development» [26].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гороховатський, В. О., & Творошенко, І. С. (2022). Аналіз багатовимірних даних за описом у формі множини компонент.

2. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2020). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In *Information and Communication Technologies in Education, Research, and Industrial Applications: 15th International Conference, ICTERI 2019, Kherson, Ukraine, June 12–15, 2019, Revised Selected Papers 15* (pp. 210-230). Springer International Publishing.

3. Gorokhovatskyi, V., & Baryshnikova, P. (2021). Features of distance education in the field of computer science in Ukraine.

4. Modrzyk, N. (2018). *Building telegram bots: develop bots in 12 programming languages using the telegram bot API*. Apress.

5. Ranjan, A., Sinha, A., & Battewad, R. (2020). *JavaScript for modern web development: building a web application using HTML, CSS, and JavaScript*. BPB Publications.

6. Ramalho, L. (2022). *Fluent python*. " O'Reilly Media, Inc.".

7. Lott, S. F., & Phillips, D. (2021). *Python Object-Oriented Programming: Build robust and maintainable object-oriented Python applications and libraries*. Packt Publishing Ltd.

8. Casciaro, M., & Mammino, L. (2020). *Node. js Design Patterns: Design and implement production-grade Node. js applications using proven patterns and techniques*. Packt Publishing Ltd.

9. Sharma, M. (2022). *Full Stack Development with MongoDB: Covers Backend, Frontend, APIs, and Mobile App Development using PHP, NodeJS, ExpressJS, Python and React Native*. BPB Publications.

10. Loy, M., Niemeyer, P., & Leuck, D. (2020). *Learning Java: An Introduction to Real-World Programming with Java*. O'Reilly Media.

11. Boduch, A., & Derks, R. (2020). *React and React Native: A complete hands-on guide to modern web and mobile development with React. js*. Packt Publishing Ltd.

12. Au-Yeung, J. (2021). *Vue. js 3 By Example: Blueprints to learn Vue web development, full-stack development, and cross-platform development quickly*. Packt Publishing Ltd.

13. Goldberg, J. (2022). *Learning TypeScript*. " O'Reilly Media, Inc.".

14. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description. *Advances in Electrical and Electronic Engineering*, *21*(1), 19-27.

15. Ibrahim, D. Y., Gorokhovatskyi, V., Tvoroshenko, I., & Mujahed, A. D. (2022). Classification of Images Based on a System of Hierarchical Features.

16. Lyashenko, V. V., Ahmad, M. A., Lyubchenko, V. A., & Kobylin, O. A. (2015). Methodology of Correlation Analysis in Solution of a Problem of Normalization of Projective Image Transformations.

17. Шафроненко, А. Ю., Бодянський, Є. В., & Руденко, Д. О. (2023). Модифікований рекурентний метод достовірної нечіткої кластеризації з використанням оптимізаційної процедури на основі косяків риб. *Системи обробки інформації*, (1 (172)), 92-96.

18. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57–70.

19. Rothman, D. (2022). *Transformers for Natural Language Processing: Build, train, and fine-tune deep neural network architectures for NLP with Python, Hugging Face, and OpenAI's GPT-3, ChatGPT, and GPT-4*. Packt Publishing Ltd.

20. Mashtalir, S. V., & Nikolenko, O. V. (2023). Data preprocessing and tokenization techniques for technical Ukrainian texts. *Applied Aspects of Information Technology*, *3*(6), 318-326.

21. Полубєхін, А., & Шафроненко, А. (2024). Огляд нейро-нечітких систем. *Матеріали конференцій МЦНД*, (19.01. 2024; Кривий Ріг, Україна), 328-329.

22. Bilonoh, B., Bodyanskiy, Y., Kolchygin, B., & Mashtalir, S. (2021, May). Tunable Activation Functions for Deep Neural Networks. In *International Scientific Conference "Intellectual Systems of Decision Making and Problem of Computational Intelligence"* (pp. 624-633). Cham: Springer International Publishing.

23. Руденко, Д., Лотвінова, В., & Безверха, Є. (2023). Навчання нейронної мережі в задачах обробки даних. *Collection of scientific papers «SCIENTIA»*, (September 22, 2023; Singapore, Singapore), 94-95.

24. Cotton, P. (2022). *Microprediction: building an open ai network*. MIT Press.

25. Lyubchenko, V., Matarneh, R., Kobylin, O., & Lyashenko, V. (2016). Digital image processing techniques for detection and diagnosis of fish diseases.

26. Terno O. (2024). Analysis and comparison of innovative artificial intelligence models for text-to-image synthesis, *Abstracts of XVIII International Scientific and Practical Conference «Modern challenges: trends, problems and prospects development», (May 07 – 10, 2024). Copenhagen, Denmark, pp. 283-285.*

27. Dunér, D., & Nilsson, M. (2020). Scalability of push and pull based event notification: A comparison between webhooks and polling.

28. Thorgersen, S., & Silva, P. I. (2021). *Keycloak-identity and access management for modern applications: harness the power of Keycloak, OpenID Connect, and OAuth 2.0 protocols to secure applications*. Packt Publishing Ltd.

29. da Costa, L. F. (2021). *Testing JavaScript Applications*. Simon and Schuster.

30. Iryna, T., & Maksym, K. (2021). Research results of functional, white box and smoke testing methods for mobile applications. *Trends in science and practice of today*, *5*, 418.

31. Dubnitskiy, V., Kobylin, A., & Kobylin, O. (2021). Виконання на мобільних пристроях арифметичних операцій з використанням аксіом класичного та нестандартного інтервального аналізу. *Advanced Information Systems*, *5*(3), 128-136.

32. Lohani, D. (2022). *Taking Flutter to the Web: Learn how to build cross-platform UIs for web and mobile platforms using Flutter for Web*. Packt Publishing Ltd.