МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, програмної інженерії та комп'ютерних наук

Пояснювальна записка

до випускної кваліфікаційної роботи
здобувача освітнього ступеню «магістр»

на тему: **РОЗРОБКА TELEGRAM – БОТА ДЛЯ ГРИ В ХРЕСТИКИ – НОЛІКИ**

Виконав: студент 2 курсу магістратури,
групи ІКК – 2.1
спеціальності
121 «Інженерія програмного забезпечення»
Долгополов Артур Олександрович
Керівник
к.т.н., доцент кафедри ІТ Горбачов В.Е.
Рецензент
к.т.н., доцент кафедри КН О. П. Русу

Одеса – 2023 р.

ДОВІДКА

кафедри інформаційних технологій про виконану магістерську роботу

студента 2 курсу магістратури ФКПІ та групи ІКК 2.1

Долгополова Артура Олександровича

на тему: Розробка Telegram – бота для гри в хрестики –ноліки

Висновок нормоконтролер: оформлено згідно вимог МГУ. Пояснювальна записка до кваліфікаційної роботи викон з незнач. порушеннями ДСТУ

Нормоконтролер викладач каф ІТ 15.12.2023 Кіміліщина І.В.
(науковий ступінь, вчене звання, посада) (підпис, дата) (і. б. прізвище)

Висновок відповідального за наявність академічного плагіату: згідно сертифікату ID 1015695764 унікальність роботи підтверджено.

Відповідальна особа викл каф ІТП 15.12.2023 Кіміліщина І.В.
(науковий ступінь, вчене звання, посада) (підпис, дата) (і. б. прізвище)

Попередня експертиза (захист) магістерської роботи
(бакалаврської роботи чи магістерської роботи)

студ. Долгополов А.О. проведена "12" 12 2023р.
(прізвище і.б.)

Висновки:
Магістерська робота виконана у повному обсязі. В роботі проведено аналіз існуючих ігор на вивчено можливості оптимізації та ускладнення гри в хрестики-нолики. Розроблено Telegram-бот спрямований на гру. Магістерська робота відповідає вимогам випускних маг. робіт та рекомендується до захисту.

Члени комісії

(підпис) д.т.н, проф Сергієвська І.В.
(науковий ступінь, вчене звання, посада, прізвище і.б.)

(підпис) к.т.н, доц Григорієва М.І.
(науковий ступінь, вчене звання, посада, прізвище і.б.)

(підпис) к.т.н, доц Горбачов О.Е.
(науковий ступінь, вчене звання, посада, прізвище і.б.)

МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, програмної інженерії та комп'ютерних наук
Кафедра інформаційних технологій
Освітній ступінь магістр
Галузь знань 12 Інформаційні технології
Спеціальність 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
к.т.н., доц
Григор'єва Т.І.
"15" 09 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Долгополова Артура Олександровича

1. Тема Розробка Telegram – бота для гри в хрестики – нуліки
керівник роботи к.т.н., доц., Горбачов Віктор Едуардович
затверджені наказом закладу вищої освіти від 25.09.2023 № 1957
2. Строк подання студентом роботи 11.12.2023 р.

3. Вихідні дані до роботи науково-методичні розробки вітчизняних та зарубіжних авторів, матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки
Три розділу – завдання виконуються за допомогою мови Python та Telegram
Розділ 1: Визначення та роль ботів у месенджерах, зокрема Telegram
Розділ 2: Проектування та розробка Telegram бота на мові програмування Python
Розділ 3: Оптимізація та розширення функціоналу

5. Перелік графічного матеріалу (з зазначенням обов'язкових креслень)
Слайд 1 – Вступна частина
Слайд 2 – Характеристики та унікальність
Слайд 3 – Технічні деталі та оптимізація
Слайд 4 – Програмна архітектура
Слайд 5 – Майбутні перспективи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 25.09.2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Визначення та роль ботів у месенджерах, зокрема Telegram.	27.09.2023 – 04.10.2023	вик
2	Аналіз існуючих технологій розробки чат-ботів та їх застосування в ігрових сценаріях	04.10.2023 – 18.10.2023	вик
3	Обгрунтування вибору мови програмування Python для розробки бота.	20.10.2023 – 03.11.2023	вик
4	Детальний опис архітектури бота та його основних компонентів.	06.11.2023 – 20.11.2023	вик
5	Впровадження алгоритмів оптимізації для покращення швидкодії гри та відгуку бота.	22.11.2023 – 29.11.2023	вик
6	Додавання можливостей для гри з іншими користувачами через Telegram.	01.12.2023 – 14.12.2023	вик
7			
8			

Студент _____ Долгополов А.О.
(підпис)

Керівник роботи _____ Горбачов В.Е.
(підпис)

РЕФЕРАТ

Пояснювальна записка до магістерської роботи: 69 сторінка, 10 рисунків, 9 джерел.

РОЗРОБКА, TELEGRAM – БОТ, PYTHON, ГРА, ХРЕСТИКИ – НОЛІКИ, ІНТЕРАКТИВНЕ СПІЛКУВАННЯ, МЕСЕНДЖЕР, ПРОГРАМУВАННЯ, ЧАТ – БОТ.

Об'єктом розгляду є розробка Telegram-бота на мові програмування Python для гри в хрестики-нолики. Головною метою дослідження є створення інтерактивного ігрового середовища для користувачів та оптимізація процесу взаємодії з ботом для поліпшення ігрового досвіду.

В ході виконання роботи був розроблений Telegram-бот, спрямований на гру в хрестики-нолики, і вивчено існуючі технології для вдосконалення функціоналу гри. Проведено аналіз існуючих ігор та виявлено можливості оптимізації та удосконалення гри в хрестики-нолики.

Результати розробки включають створений бот, а також висвітлення основних переваг та можливостей гри через Telegram, які можуть бути використані різними категоріями користувачів, від простих гравців до групових команд чи організацій.

ABSTRACT

Explanatory note to the master's thesis: 69 pages, 10 figures, 9 sources.

DEVELOPMENT, TELEGRAM BOT, PYTHON, GAME, TIC-TAC-TOE, INTERACTIVE COMMUNICATION, MESSENGER, PROGRAMMING, CHAT BOT.

The object of consideration is the development of a Telegram bot in the Python programming language for playing tic-tac-toe. The main goal of the research is to create an interactive gaming environment for users and optimize the process of interaction with the bot to improve the gaming experience.

In the course of the work, a Telegram bot aimed at playing tic-tac-toe was developed and existing technologies were studied to improve the game's functionality. We analyzed existing games and identified opportunities to optimize and improve the tic-tac-toe game.

The development results include the created bot, as well as highlighting the main advantages and features of the game via Telegram, which can be used by different categories of users, from simple players to group teams or organizations.

ВІДГУК КЕРІВНИКА

магістерської роботи студента Долгополова А. О.
на тему: Розробка telegram-бота для гри в хрестики-нолики

Сучасні месенджери, такі як Telegram, стали невід'ємною частиною нашої повсякденної комунікації. У цьому контексті використання чат – ботів набуває дедалі більшої важливості, надаючи зручні та ефективні засоби взаємодії з користувачами. Однак, незважаючи на їх широке поширення, розробка чат – ботів, спеціалізованих для конкретних сценаріїв, залишається актуальним завданням.

Метою даного дослідження є створення та аналіз Telegram – бота, орієнтованого на ігровий сценарій на прикладі гри в хрестики-нулики, бібліотека якого оснащена інтелектуальними алгоритмами, що робить її не просто ботом, а інтелектуальним супутником у грі.

Студент Долгополов А. О. добре виконав завдання до ВКР. Робота проводилася значною мірою самостійно. Графік консультацій не порушувався. Завдання на ВКР виконано повністю. Необхідні для цього розрахунки проведені.

При оформленні пояснювальної записки використовувались комп'ютерні технології.

Під час виконання магістерської роботи студент Долгополов А. О. засвоїв методи вимірювань, показав уміння користуватись навчальною та технічною літературою, ставити та розв'язувати дослідницькі задачі.

Магістерська робота відповідає вимогам до випускних кваліфікаційних робіт магістрів та заслуговує оцінки «добре».

Студент Долгополов А. О. заслуговує присвоєння кваліфікації магістр з інженерії програмного забезпечення за заявленою спеціальністю 121 Інженерія програмного забезпечення.

Керівник
к.т.н., доцент кафедри ІТ _____ В. Е. Горбачов

РЕЦЕНЗІЯ

на магістерську роботу студента Долгополова А. О.
на тему: Розробка telegram-бота для гри в хрестики-нуліки

Магістерська робота виконана на 69 сторінках текстової частини та містить три розділи згідно з завданням на магістерську роботу.

У магістерській роботі студента Долгополова А. О. в ході виконання роботи був розроблений Telegram-бот, спрямований на гру в хрестики-нуліки, і вивчено існуючі технології для вдосконалення функціоналу гри. Проведено аналіз існуючих ігор та виявлено можливості оптимізації та удосконалення гри в хрестики-нуліки.

В роботі Долгополова А. О. показав достатню теоретичну підготовку.

Пояснювальна записка й графічні матеріали виконані охайно й відповідно до вимог ЄСКД, оформлення демонстраційних слайдів якісне.

Зауваження:

- не достатньо розглянуті можливості розширення гри;

Але названі недоліки не знижують цінності виконаної роботи.

Магістерська робота студента Долгополова А. О. відповідає вимогам до випускних кваліфікаційних робіт магістрів та заслуговує оцінки «добре».

Студент Долгополов А. О. заслуговує присвоєння кваліфікації магістр з інженерії програмного забезпечення за заявленою спеціальністю 121 Інженерія програмного забезпечення.

Рецензент
к.т.н., доцент каф. КН  О. П. Русу

UNICHECK
by Turnitin

Ім'я користувача:
Анна Серединко

ID перевірки:
1016010075

Дата перевірки:
15.12.2023 19:26:24 MSK

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
17.12.2023 19:04:33 MSK

ID користувача:
100001433

Назва документа: Магістерська робота. Долгополов

Кількість сторінок: 74 Кількість слів: 12326 Кількість символів: 95161 Розмір файлу: 838.73 KB ID файлу: 1015695764

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.43%
Схожість

Найбільша схожість: 2.57% з джерелом з Бібліотеки (ID файлу: 1015695769)

4.23% Джерела з Інтернету	701	Сторінка 76
2.8% Джерела з Бібліотеки	78	Сторінка 79

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2
Підозріле форматування	15 сторінок

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

PM – Private Message – особисті повідомлення, у контексті бота це діалог сам – на – сам із користувачем, а не група/канал.

Chat – Чат – загальна назва для особистих повідомлень, груп, супергруп і каналів.

Update – Апдейт – будь – яка подія з цього списку: повідомлення, редагування повідомлення, колбек, інлайн – запит, платіж, додавання бота в групу тощо.

Handler – Хендлер – асинхронна функція, яка отримує від диспетчера/роутера черговий апдейт і обробляє його.

Dispatcher – Диспетчер – об'єкт, що займається отриманням апдейтів від Telegram з подальшим вибором хендлера для обробки прийнятого апдейта.

Router – Роутер – аналогічно диспетчеру, але відповідає за підмножину множини хендлерів. Можна сказати, що диспетчер – це кореневий роутер.

Filter – Фільтр – вираз, який зазвичай повертає True або False і впливає на те, буде викликаний хендлер чи ні.

Middleware – Мідлвар – прошарок, який вклинюється в обробку апдейтів.

NPC – Non – player character – неігровий персонаж або неіграбельний персонаж.

HTML5 (англ. HyperText Markup Language, version 5) — мова для структурування та представлення вмісту всесвітньої павутини.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

PM – Private Message – особисті повідомлення, у контексті бота це діалог сам – на – сам із користувачем, а не група/канал.

Chat – Чат – загальна назва для особистих повідомлень, груп, супергруп і каналів.

Update – Апдейт – будь – яка подія з цього списку: повідомлення, редагування повідомлення, колбек, інлайн – запит, платіж, додавання бота в групу тощо.

Handler – Хендлер – асинхронна функція, яка отримує від диспетчера/роутера черговий апдейт і обробляє його.

Dispatcher – Диспетчер – об'єкт, що займається отриманням апдейтів від Telegram з подальшим вибором хендлера для обробки прийнятого апдейта.

Router – Роутер – аналогічно диспетчеру, але відповідає за підмножину множини хендлерів. Можна сказати, що диспетчер – це кореневий роутер.

Filter – Фільтр – вираз, який зазвичай повертає True або False і впливає на те, буде викликаний хендлер чи ні.

Middleware – Мідлвар – прошарок, який вклинюється в обробку апдейтів.

NPC – Non – player character – неігровий персонаж або неіграбельний персонаж.

ВСТУП

Сучасні месенджери, такі як Telegram, стали невід'ємною частиною нашої повсякденної комунікації. У цьому контексті використання чат – ботів набуває дедалі більшої важливості, надаючи зручні та ефективні засоби взаємодії з користувачами. Однак, незважаючи на їх широке поширення, розробка чат – ботів, спеціалізованих для конкретних сценаріїв, залишається актуальним завданням.

Платформа Telegram є миттєвим і безпечним додатком для обміну повідомленнями, функціонально порівнянним з SMS або електронною поштою. Однак, на відміну від звичайних текстових повідомлень, вона також надає можливість надсилання різноманітних мультимедійних даних, таких як фотографії, відео та файли будь – якого формату. Крім того, є функціонал для організації групових чатів з місткістю до 200 000 учасників, читання каналів і проведення відео – та аудіо дзвінків.

Особливістю є можливість використання платформи на необмеженій кількості пристроїв, де кожен з них міститиме повний набір ваших контактів, чатів та історію листування. Додатки Telegram доступні як для мобільних пристроїв, так і для персональних комп'ютерів, а також підтримуються веб – версією в браузері. Цей широкий спектр доступу та функціональних можливостей робить Telegram зручним і гнучким засобом спілкування на різних платформах.

Чат – боти надають можливість надання послуг та обслуговування клієнтів цілодобово, без необхідності підтримання людського персоналу на пункті приймання замовлень або в службі підтримки. Ці автоматизовані системи також гарантують конфіденційну та безпечну взаємодію з користувачами, що робить їх оптимальним інструментом для обробки чутливої інформації.

Особлива популярність ботів на платформі Telegram зумовлена її великою аудиторією та високим рівнем захисту даних. Ці боти можуть використовуватися для різних цілей, чи то автоматизація бізнес – процесів, чи то надання новин та оновлень, а також навчання або розвага користувачів. Їх функціональність і

універсальність роблять їх невід'ємним інструментом у різних сферах, де потрібна ефективна і бездоганна взаємодія з публікою.

Метою даного дослідження є створення та аналіз Telegram – бота, орієнтованого на ігровий сценарій, а саме, на гру в хрестики – нулики. У сучасному інформаційному суспільстві, де увага користувачів є цінним ресурсом, створення захопливих та інтерактивних форм спілкування через месенджери стає ключовим завданням розробників.

В даному дослідженні будемо розглядати основні аспекти та виклики, пов'язані із створенням Telegram бота для гри в хрестики – ноліки, а також вивчати можливості застосування мови програмування Python для досягнення цієї мети.

Актуальність цього дослідження підкреслюється не тільки зростаючим інтересом до чат – ботів, а й необхідністю створення інноваційних та ефективних ігрових додатків у контексті месенджерів. У цьому світлі, представлення бота для гри в хрестики – нулики в Telegram є перспективним напрямком, який потребує глибокого розуміння принципів розроблення чат – ботів та їхнього застосування в ігрових сценаріях.

Дослідження міститиме аналіз наявних технологій розроблення чат – ботів, а також обговорення принципів гри в хрестики – нулики з урахуванням їхнього відображення в програмному забезпеченні. Під час роботи буде презентовано та розроблено Telegram – бот мовою програмування Python, що дасть змогу детально розглянути процеси проектування та взаємодії бота з користувачами.

1 ВИЗНАЧЕННЯ ТА РОЛЬ БОТІВ У МЕСЕНДЖЕРАХ, ЗОКРЕМА TELEGRAM

Бот в месенджері Telegram – це автоматизований обліковий запис, який взаємодіє з користувачами за допомогою програмних алгоритмів. Це віртуальний чат – супутник, здатний виконувати різні функції, такі як надсилання повідомлень, відповіді на запитання, обробка команд, отримання та надсилання інформації, виконання завдань та інше. Боти в Telegram можуть використовуватися для різноманітних цілей, включаючи отримання новин, гри, роботу з послугами, використання API та багато іншого. Вони можуть бути створені і управлятися розробниками або компаніями для полегшення взаємодії з користувачами у чат – форматі.

У наш час чат – боти стали невід'ємною частиною месенджерів, відіграючи важливу роль у забезпеченні автоматизованої взаємодії між користувачами та програмами. У Telegram ці віртуальні агенти виконують різноманітні функції, удосконалюючи процес спілкування.

За допомогою спеціального API сторонні розробники можуть створювати «ботів» – спеціальні акаунти, керовані програмами [1 ; 2]. Типові боти відповідають на спеціальні команди в персональних і групових чатах, також вони можуть здійснювати пошук в інтернеті або виконувати інші завдання, застосовуються в розважальних цілях або в бізнесі.

У вересні 2015 року Павло Дуров заявив про швидку появу можливостей монетизації та розміщення реклами в ботах.

18 травня 2017 року для ботів було представлено платіжне API [3]. Щоб користувачі могли протестувати цю функцію, командою Telegram був створений тестовий бот, який пропонує купити «Машину часу» (гроші з користувачів не стягувалися).

Боти також використовують ігрову платформу Telegram, яка використовує HTML5, тому ігри завантажуються за запитом у міру необхідності, як звичайні веб – сторінки. Ці ігри підходять для iPhone починаючи з моделі 4 і свіжіших, а також для Android – пристроїв починаючи з версії 4.4 і вище.

З червня 2021 року користувачі можуть використовувати нове меню бота, щоб надсилати команди, перебуваючи в чаті. У квітні 2022 року боти отримали можливість налаштування інтерфейсів відповідно до теми програми, а також вбудоване завантаження сторінок. Тепер інтерфейси можуть бути змінені навіть під час взаємодії.

Боти в месенджері Telegram виконують різноманітні функції і відіграють важливу роль у забезпеченні зручного та функціонального взаємодії користувачів. Ось деякі з їх ключових ролей:

– створення швидкого і простого доступу до інформації, створюючи персоналізовані інтерактивні сценарії та підтримуючи безперервну взаємодію;

– створення зручного та інтерактивного середовища для взаємодії з користувачами. Призначені для спрощення спілкування та надання відповідей на запитання, що може бути особливо важливим у випадках, коли користувачі мають конкретні запитання або потребують допомоги;

– надання різноманітної інформації користувачам, такої як новини, погода, рекомендації чи інша актуальна інформація. Вони можуть автоматично надсилати користувачам сповіщення про події, які їх цікавлять;

– виконання різних операцій безпосередньо в месенджерах. Це може включати покупки, бронювання, отримання послуг та інші дії, які раніше вимагали виходу з чату та використання окремих додатків чи веб – сайтів;

– підтримки клієнтів, вирішення їхніх проблем та відповіді на питання. Автоматизована підтримка через чат – ботів дозволяє швидше та ефективніше реагувати на запитання користувачів;

– служити як засіб для гейміфікації та розваг. Вони здатні організовувати різні ігри, вікторини або надавати розважальний контент, щоб залучити та розважити аудиторію, и навіть багатокористувацькі ігри, збагачуючи досвід користувачів у месенджері.

– дозволяти підприємствам проводити маркетингові кампанії та надавати рекламні послуги через новий, більш особистий канал взаємодії з користувачами;

Чат – боти в месенджерах стають потужним інструментом для забезпечення різноманітних сервісів та покращення комунікації, а їх роль продовжує розширюватися в контексті розвитку технологій та вимог користувачів.

Розглянемо еволюцію ролі чат – ботів у сучасному спілкуванні та їх вплив на різноманітні сфери, включаючи бізнес, освіту та розваги:

1. Від інформаційного агента до персонального асистента. Спочатку чат – боти використовувалися головним чином як інформаційні агенти, що відповідали на запитання та надавали користувачам необхідну інформацію. З плином часу їх роль еволюціонувала, і вони стали персональними асистентами, здатними виконувати більше завдань та взаємодіяти з користувачем у більш інтерактивний спосіб.

2. Вплив на бізнес. Чат – боти стали ефективним інструментом для підтримки бізнес – процесів. Вони використовуються для ведення діалогу з клієнтами, обробки замовлень, вирішення проблем та навіть для проведення опитувань. Їхня взаємодія з бізнес – системами робить їх невід'ємною частиною стратегії впровадження технологій у бізнесі.

3. Використання в освіті. Чат – боти знаходять своє місце в освіті, допомагаючи студентам здобувати знання, відповідаючи на питання, а також виконуючи роль інтерактивних навчальних асистентів. Вони можуть стимулювати інтерес до навчання шляхом використання ігрових елементів та інтерактивних завдань.

4. Розваги та відпочинок. У розважальній сфері чат – боти впроваджуються для створення інтерактивних ігор, конкурсів та розважальних віджетів. Вони стають частиною стрімко змінюваного світу розваг, де відбувається інтеграція між віртуальним та реальним життям.

5. Повсякденне використання. Чат – боти стають необхіднішим елементом повсякденного життя, де вони можуть служити для організації подій, нагадування про плани, відслідковування фінансів, та навіть надавати емоційну підтримку.

Розглядаючи різні аспекти ролі чат – ботів в сучасному спілкуванні, ми можемо визначити їхню важливість і вплив на різні сфери життя.

1.2 Аналіз існуючих технологій розробки чат – ботів та їх застосування в ігрових сценаріях

В рамках даного розділу проводиться аналіз сучасних технологій розробки чат – ботів та їх потенціального використання у контексті ігрових сценаріїв. Обґрунтовується актуальність дослідження в контексті постійного розвитку інформаційних технологій та зростання зацікавленості користувачів у взаємодії з інтерактивними системами. Розглядаються ключові аспекти технічного впровадження чат – ботів, а також визначаються перспективи їх застосування в сучасних ігрових середовищах. Особлива увага приділяється вивченню принципів та можливостей, які надають існуючі технології для створення чат – ботів, зокрема в контексті їх інтеграції в ігрові процеси та оптимізації користувацького досвіду.

Розглянемо різноманітні технології, що використовуються для створення чат – ботів, і визначимо їхні переваги та недоліки:

1. Платформи для розробки. Основні платформи та середовища, які використовуються для створення чат – ботів, включаючи Telegram Bot API, Facebook Messenger API, Microsoft Bot Framework та інші. Звернемо увагу на можливості, які ці платформи надають розробникам для створення та розгортання ботів.

Чат – боти використовують різноманітні платформи для свого створення та ефективної взаємодії з користувачами. Ось детальний огляд ключових платформ для розробки чат – ботів:

Telegram Bot API – це одна з найпопулярніших платформ для розробки чат – ботів. Вона надає розробникам набір інструментів для створення ботів в Telegram, забезпечуючи можливості відправки повідомлень, обробки команд, роботи з мультимедіа та взаємодії з групами;

Facebook Messenger API: дозволяє створювати ботів для Facebook Messenger, що дозволяє взаємодіяти з користувачами на платформі Facebook. Вона надає розробникам засоби для відправки текстових повідомлень, зображень, відео та інших мультимедійних елементів;

Microsoft Bot Framework: ця платформа надає інтегрований набір інструментів для створення чат – ботів, які можуть працювати в Microsoft Teams, Skype та інших платформах. Microsoft Bot Framework підтримує використання мов програмування C# та Node.js.;

Dialogflow (Google) – це інструмент для створення розумних чат – ботів з можливістю розпізнавання природної мови (NLP). Він підтримує інтеграцію з різними месенджерами та дозволяє створювати конвертаційні агенти швидко та ефективно;

Wit.ai (Facebook): який належить Facebook, використовується для створення чат – ботів з використанням природної мови. Це дозволяє розробникам використовувати інтеграцію з багатьма месенджерами та іншими платформами;

Botpress – це відкрите програмне забезпечення для розробки чат – ботів. Воно надає розробникам розширені можливості для створення ботів з використанням Node.js та вбудованими інструментами для аналізу діалогів.

Враховуючи різні платформи, розробники можуть обирати ту, яка найбільше підходить їхнім потребам та забезпечить оптимальні можливості для розробки та взаємодії з користувачами.

2. Мови програмування. Розглянемо різні мови програмування, які використовуються для розробки чат – ботів. Python, JavaScript, Java, та інші мови мають свої переваги та обмеження. Визначимо, які мови є більш популярними та ефективними в різних випадках.

Вибір мови програмування визначається завданнями, специфікою проекту та зручністю для розробника. Ось детальний огляд деяких популярних мов програмування для створення чат – ботів:

Python одна з найпопулярніших мов для розробки чат – ботів. Вона відзначається зрозумілістю синтаксису, великою кількістю бібліотек для обробки природної мови (NLTK, spaCy), а також готових фреймворків, таких як Django чи Flask.

JavaScript (Node.js), зокрема в його виконавчому середовищі Node.js, широко використовується для розробки серверних додатків, включаючи чат – ботів. Node.js

забезпечує швидку та ефективну обробку вводу – виводу, що особливо важливо для роботи в реальному часі.

Java є мовою, яку часто вибирають підприємства для розробки великих та стабільних систем. З використанням фреймворків, таких як Spring, Java може бути ефективною мовою для створення чат – ботів.

C# мова програмування використовується в Microsoft Bot Framework для створення чат – ботів для платформи Microsoft. Її велика популярність в універсальному програмуванні дозволяє розробникам легко вивчати та використовувати її для розробки ботів.

Ruby, разом з фреймворком Ruby on Rails, може бути використаний для розробки чат – ботів. Простий та чистий синтаксис Ruby полегшує швидкість розробки, що робить його привабливим вибором для деяких розробників.

PHP може бути використаний для створення простих чат – ботів, особливо для веб – додатків. Існує кілька бібліотек та фреймворків, які дозволяють легко інтегрувати чат – ботів в PHP – додатки.

Вибір мови програмування залежить від потреб проекту, досвіду розробника та технічних особливостей обраної платформи чат – бота.

3. Фреймворки та бібліотеки.

Обговоримо різноманітні фреймворки та бібліотеки, які використовуються для розробки чат – ботів. Вони можуть включати в себе Botpress, Rasa, Microsoft Bot Framework, і інші. Зазначимо, які можливості ці інструменти надають для розширення функціоналу ботів.

Фреймворки та бібліотеки грають ключову роль в розробці чат – ботів, допомагаючи розробникам швидше та ефективніше створювати функціональні та потужні боти. Ось огляд деяких з найбільш популярних фреймворків та бібліотек:

1) Aiogram – це бібліотека для роботи з Telegram API, яка підтримує асинхронний код. Вона надає можливість створювати асинхронні функції та обробляти події паралельно.

2) Python – Telegram – bot – це популярна бібліотека для розробки чат – ботів для платформи Telegram на мові програмування Python. Вона надає зручний

інтерфейс для взаємодії з Telegram Bot API та включає в себе ряд інструментів для роботи з повідомленнями, клавіатурами, зображеннями, та іншими можливостями Telegram. Має добре документований код та офіційну документацію, яка детально пояснює всі можливості та способи використання бібліотеки. Крім того, у випадку питань чи проблем, спільнота користувачів активно допомагає на GitHub.

3) Botpress – це відкрите програмне забезпечення для створення чат – ботів з використанням Node.js. Його особливість – це можливість використання плагінів для розширення функціоналу та вбудовані інструменти для аналізу діалогів, що полегшує розробку складних ботів.

4) Rasa – це фреймворк для розробки чат – ботів з використанням природної мови. Його ключовою особливістю є можливість розгортання ботів локально та безпека обробки даних, що робить його популярним для проектів, де приватність є пріоритетом.

5) Microsoft Bot Framework – це інструмент, який надає засоби для створення чат – ботів, які можуть працювати в різних месенджерах. Він підтримує використання мов програмування C# та Node.js та має вбудовані інструменти для створення розумних ботів.

6) Dialogflow – це інструмент від Google для створення ботів з використанням природної мови. Він має вбудовану підтримку для інтеграції з різними месенджерами та платформами, а також можливості роботи з різними мовами.

7) Wit.ai, належить Facebook, є інструментом для створення ботів з використанням природної мови. Він визначається своєю простотою використання та можливістю розпізнавання широкого спектру команд.

Ці фреймворки та бібліотеки спрощують розробку чат – ботів, дозволяючи розробникам швидше та ефективніше створювати розумні та інтерактивні агенти.

4. Інтеграція зі сторонніми сервісами зі сторонніми сервісами, такими як бази даних, API сторонніх сервісів, системи штучного інтелекту, та інші.

Інтеграція чат – ботів зі сторонніми сервісами розширює їхні можливості та полегшує взаємодію з користувачами. Огляд деяких можливостей інтеграції зі сторонніми сервісами:

1) API та веб – сервісами для отримання та надсилання інформації. Це може включати доступ до баз даних, інформації про погоду, фінансових даних та інших джерел;

2) для комерційних ботів важливо мати можливість приймати платежі. Інтеграція з платіжними системами, такими як Stripe, PayPal чи іншими, дозволяє чат – ботам надавати послуги та приймати оплату в реальному часі;

3) інтегруватися з системами управління відносинами з клієнтами (CRM) для збереження та отримання інформації про користувачів. Це дозволяє ботам надавати персоналізовані послуги та зберігати історію взаємодії;

4) чат – боти можуть бути пов'язані з акаунтами користувачів у соціальних мережах, щоб надавати персоналізовану інформацію та взаємодію. Інтеграція з Facebook, Twitter, Instagram чи іншими соціальними мережами забезпечує розширення аудиторії.

5) інтеграція з системами штучного інтелекту та аналітики дозволяє чат – ботам використовувати передові алгоритми для аналізу даних та надання користувачам більш інтелектуальних відповідей;

6) чат – боти можуть взаємодіяти з електронною поштою для надсилання сповіщень, підтвердження реєстрації чи отримання даних. Це особливо корисно для підтримки та нагадувань.

Інтеграція зі сторонніми сервісами розширює функціонал чат – ботів та робить їх більш універсальними в інтеракції з користувачами в різних контекстах.

5. Оптимізація продуктивності. Важлива для забезпечення ефективної та швидкої взаємодії з користувачами. Розглянемо стратегії та методи оптимізації продуктивності:

1) хешування результатів запитань (збереження результатів) часто використовуваних запитань та відповідей у кеші може значно зменшити час

відповіді. Це особливо ефективно для статичних запитань, де відповіді не змінюються часто.

2) використання швидких та оптимізованих алгоритмів для обробки природної мови дозволяє ботам швидше розуміти та відповідати на запитання користувачів. Вдосконалення алгоритмів NLP (Natural Language Processing) може покращити точність та швидкість взаємодії. Основна ідея поля NLP полягає в тому, щоб надати комп'ютерам здатність розуміти, інтерпретувати і відповідати на людську мову в спосіб, зрозумілий для людини.

3) використання асинхронної обробки запитань дозволяє ботам обробляти кілька запитань одночасно, забезпечуючи швидку реакцію та високу продуктивність в періоди великого навантаження. Асинхронна обробка запитань означає, що комп'ютер може обробляти різні завдання або запитання одночасно, не чекаючи завершення кожного запитання перед переходом до наступного. Це як у випадку, коли ви розмовляєте з декількома людьми одночасно та відповідаєте на їх питання без очікування, коли кожна бесіда завершиться.

У контексті програмування це означає, що комп'ютер може виконувати різні завдання паралельно, що може покращити швидкодію програм та забезпечити ефективне використання ресурсів. Асинхронність дозволяє виконувати інші дії, поки чекається відповідь на конкретне завдання, що сприяє більш ефективному використанню часу та ресурсів.

Asyncio – вбудована бібліотека Python для асинхронного програмування. Вона дозволяє вам створювати асинхронні функції , які можна використовувати для обробки декількох завдань одночасно.

Ці інструменти дозволяють створювати асинхронні боти, які можуть обробляти кілька запитань одночасно, забезпечуючи високу продуктивність у періоди інтенсивного навантаження;

4) оптимізація запитів до сторонніх сервісів грає важливу роль, враховуючи, що багато чат – ботів взаємодіють зі сторонніми сервісами. Мінімізація затримок під час обміну інформацією сприяє швидкій взаємодії.

Оптимізація запитів до сторонніх сервісів, визначається як систематична та методологічна робота над зменшенням часу та ресурсів, витрачених на виконання запитів до зовнішніх сервісів у комп'ютерних системах. Цей підхід включає в себе розгляд оптимальних стратегій взаємодії з віддаленими серверами, зокрема обмеження кількості та частоти запитів, використання хешування для збереження та прискорення доступу до результатів попередніх запитів, використання асинхронних технік для паралельного виконання запитів та інших стратегій для мінімізації затрат на взаємодію зі зовнішніми ресурсами.

Оптимізація запитів грає важливу роль у забезпеченні ефективної та продуктивної роботи комп'ютерних систем, особливо в умовах інтенсивного використання мережевих ресурсів та залежності від зовнішніх сервісів;

5) для ботів, які використовують зображення та мультимедійний контент, оптимізація цих елементів для швидкого завантаження та обробки є важливою. Використання стиснених форматів та асинхронного завантаження може покращити продуктивність;

6) важливо встановити системи моніторингу та аналізу продуктивності для виявлення слабких місць та можливостей для оптимізації. Це дозволяє швидко реагувати на проблеми та покращувати продуктивність чат – бота з часом.

Оптимізація продуктивності забезпечує надійну та швидку взаємодію з користувачами, покращуючи вагому експертність бота та загальне враження від використання.

Для моніторингу та аналізу продуктивності чат – бота в Telegram існують різноманітні інструменти та підходи. Нижче наведені деякі з них:

1. Bot Analytics [4] – це платформа для аналізу та візуалізації даних з Telegram – ботів. Вона дозволяє відстежувати статистику використання бота, аналізувати взаємодію з користувачами та отримувати звіти про ефективність.

2. Google Analytics [5] – для відстеження взаємодії користувачів з веб – версією вашого бота, якщо така наявна. Встановлення аналітичного коду на веб – сайт, який використовується для чат – бота, дозволяє отримати важливі дані.

3. Використовування інструментів моніторингу серверів, такі як Prometheus

[6] (або Datadog [7], для відстеження ресурсів, використовуваних ботом. Це допоможе виявити можливі проблеми з продуктивністю або надмірне використання ресурсів.

4. Використовуйте інструменти моніторингу трафіку, такі як Wireshark [8] або NGINX Amplify [9], для відстеження величини запитів, їх часу виконання та відповідей бота.

6. Генерація звітів за допомогою бібліотек Python, такі як Matplotlib [10] або Pandas [11] , для аналізу та візуалізації даних, зібраних з логів та інших джерел.

Вибір конкретних інструментів може залежати від стеку технологій та конкретних потреб проекту при розробці чат – ботів.

Застосування технологій в ігрових сценаріях.

Застосування наявних технологій розроблення чат – ботів у контексті ігрових сценаріїв зроблено на технологіях і методах, що використовуються для створення чат – ботів, і розгляді їхньої застосовності та ефективності в ігрових сценаріях. Розглянемо основні тенденції в розробці чат – ботів, а також практичні рекомендації для успішної інтеграції чат – ботів в ігрові сценарії.

З розвитком технологій штучного інтелекту та обробки природної мови (Natural Language Processing, NLP) стали активно застосовуватися в різних сферах, включно з розробкою чат – ботів. У контексті ігрових сценаріїв, чат – боти надають унікальні можливості для поліпшення взаємодії гравців з ігровим світом і збагачення ігрового досвіду.

Використання методів машинного навчання та NLP є ключовим у створенні чат – ботів, здатних розуміти та генерувати природну мову. Такі технології дають змогу ботам адаптуватися до стилю спілкування кожного гравця, покращуючи персоналізацію та автентичність взаємодії.

Розробка ефективних діалогових систем, заснованих на алгоритмах генерації відповідей і управлінні контекстом, відіграє важливу роль у створенні чат – ботів для ігор. Це дає змогу ботам підтримувати довгі та складні діалоги, не втрачаючи зв'язку з попередніми повідомленнями.

Чат – боти можуть бути використані для створення більш цікавих і непередбачуваних сценаріїв в іграх. Шляхом аналізу вподобань і поведінки гравців боти можуть пропонувати персональні квести, адаптувати рівні складності та надавати індивідуальні сюжетні лінії.

Чат – боти здатні забезпечити більш природну та гнучку взаємодію з ігровим світом. Гравці можуть використовувати текстові команди для управління персонажем, взаємодії з NPC – персонаж в іграх, який не перебуває під контролем гравця. У комп'ютерних іграх поведінка таких персонажів визначається програмно. У настільних рольових іграх неігровим персонажем керує майстер (ведучий гри) і виконання завдань, створюючи глибокий та інтуїтивно зрозумілий ігровий досвід.

Тенденції та майбутні напрямки, є інтеграція чат – ботів з віртуальною реальністю, що створить більш реалістичну взаємодію гравців з ігровим світом, що більш залучає.

Чат – боти надають безліч можливостей для поліпшення ігрового досвіду і взаємодії з ігровим світом. Однак, для досягнення максимальної ефективності, розробники повинні активно стежити за новими тенденціями та інноваціями в галузі штучного інтелекту та NLP, щоб інтегрувати їх у розроблювані ігрові проекти.

1.3 Обговорення основних принципів гри в хрестики – ноліки та їх відображення в програмному забезпеченні

Хрестики – нулики – це класична настільна гра для двох гравців, у якій вони по черзі розміщують свої символи (хрестик або нулик) на квадратному полі 3 на 3 клітини або більшого розміру (аж до «нескінченного поля»). Перший гравець, який зможе побудувати лінію зі своїх символів (хрестиків або нуликів) по вертикалі, горизонталі або діагоналі, перемагає. Нижче наведено детальніші правила гри:

1. Ігрове поле:

– гра ведеться на квадратному полі, зазвичай розміром 3х3 комірки;

– кожна клітинка може бути порожньою або містити символ одного з гравців: хрестик (X) або нулик (O).

2. Ходи гравців:

– гравці роблять ходи по черзі, починаючи з першого гравця (зазвичай хрестиків);

– кожен хід полягає в розміщенні свого символу в будь – якій порожній клітинці на ігровому полі.

3. Мета гри:

– метою кожного гравця є побудова лінії з трьох своїх символів по вертикалі, горизонталі або діагоналі;

4. Перемога:

– якщо гравцеві вдається побудувати лінію з трьох своїх символів, він оголошується переможцем;

– лінія може бути вертикальною, горизонтальною або діагональною;

5. Нічия:

– якщо всі комірки заповнені, і жоден із гравців не побудував лінію, гра завершується як нічия;

6. Черговість ходів:

– гравці продовжують робити ходи доти, доки не настає перемога або нічия.

7. Перший хід:

– часто перший хід робить гравець, який використовує хрестики.

8. Перегравання:

– після завершення гри гравці можуть вибрати перегравання для початку нової партії.

Ці основні правила забезпечують простоту і зрозумілість гри в хрестики – нулики. Вона залишається популярною не тільки у фізичному форматі, а й у цифрових реалізаціях, зокрема у вигляді ботів у месенджерах, таких як Telegram.

Стратегії та тактики гри.

Гра в хрестики – нулики надає гравцям просту, але глибоку стратегічну задачу. Ось деякі основні стратегії і тактики, які гравці можуть використовувати для досягнення перемоги або запобігання поразки, та важливість центральних клітин, блокування ходів супротивника і створення загроз для перемоги.

1. Основний контроль центру:

– заняття центральних осередків ігрового поля дає гравцеві більше можливостей для побудови ліній по горизонталі, вертикалі та діагоналі.

– почати гру з центральної комірки зазвичай вважається одним із найкращих ходів.

2. Захист кутів:

– захоплення кутів поля робить складнішою для противника побудову ліній, оскільки кутові осередки входять у кілька потенційних ліній одночасно.

– захоплення кутів також створює додаткові можливості для побудови своїх ліній.

3. Блокування противника:

– гравець може використовувати блокування, щоб заважати противнику будувати свої лінії.

– якщо противник будує загрозливу лінію, гравець може вставити свій символ, щоб розірвати її.

4. Створення загроз:

– створення загрози – розміщення символу так, щоб у наступному ході можна було побудувати лінію.

– це змушує противника реагувати і блокувати загрозу, що може відкрити нові можливості для інших ліній.

5. Уникнення очевидних пасток:

– уникнення розміщення символу поруч із двома вже зайнятими комірками противника, які утворюють загрозу.

– гравець повинен обмірковувати свої ходи, щоб не дати противнику легкі можливості для перемоги.

6. Підтримка гнучкості:

– гравець повинен будувати свої лінії так, щоб вони були максимально гнучкими і могли розвиватися в кілька напрямків одночасно.

– гнучка стратегія робить гравця менш передбачуваним і важчим для противника.

7. Пастки та блеф:

– іноді гравець може свідомо створювати пастки, застерігаючи противника від блокування лінії, яка насправді не загрожує.

– блеф може ввести противника в оману і створити додаткові можливості для перемоги.

Ефективне поєднання цих стратегій і тактик, а також грамотне реагування на ходи противника, роблять гравця успішним у хрестики – нулики. Складність полягає в тому, щоб знайти баланс між нападом і обороною, передбачаючи ходи противника і будуючи адаптивні стратегії.

Врахування особливостей гри в програмному забезпеченні.

Розробляючи програмне забезпечення для гри в хрестики – нулики через чат – бота, важливо враховувати не тільки класичні правила гри, а й адаптувати їх до цифрового формату та взаємодії з користувачем через месенджер, такий як Telegram. Нижче представлено низку особливостей, які слід врахувати:

1. Інтерфейс введення та виведення:

– програма повинна надати зручний та інтуїтивно зрозумілий інтерфейс для введення команд і відображення поточного стану гри;

– у текстовому чаті необхідно використовувати зрозумілі команди і ясні повідомлення для взаємодії з користувачем.

2. Обробка команд користувача – чат – бот повинен акуратно обробляти команди, що надходять від користувача. Це може включати в себе розпізнавання текстових команд, узгодження з правилами гри і генерацію адекватних відповідей.

3. Штучний інтелект – якщо бот має штучний інтелект, він має вміти ухвалювати обґрунтовані рішення, засновані на стратегіях і тактиках. Це може включати в себе аналіз поточного стану гри і вибір оптимальних ходів.

4. Програма має бути стійкою до помилок, таких як некоректні команди або несподівані ситуації. Важливо передбачити коректні відповіді та повідомлення про помилки.

5. Для підтримки можливості перегравання або відновлення гри після перерви, бот повинен уміти зберігати поточний стан гри і відновлювати його за необхідності.

6. Залежно від можливостей месенджера, бот може використовувати графічні елементи, емодзі або інші візуальні засоби для відображення ігрового поля та поточного ходу.

7. Якщо передбачено обмеження часу на хід, бот має вміти ефективно керувати часом, щоб не допустити тривалих затримок у грі.

8. Бот має правильно оцінювати стан гри й оголошувати перемогу, нічию або продовження гри відповідно до правил.

Врахування цих особливостей допоможе створити програмне забезпечення, яке ефективно адаптується до цифрового формату та забезпечує задовільний досвід гри для користувачів через чат – бот у месенджері.

Адаптація під гру з ботом.

Адаптація гри в хрестики – нулики під бота охоплює кілька ключових аспектів, які забезпечують зручність, інтерес і ефективну взаємодію з користувачами. Ось докладний опис цього процесу:

1. Бот повинен надати гнучкий і легко сприймається інтерфейс для взаємодії з користувачами. Це включає в себе інтуїтивно зрозумілі команди, легкість введення і зрозумілі повідомлення.

2. Бот повинен надавати доброзичливі та інформативні повідомлення про поточний стан гри, виконані ходи та результати. Це створює позитивний ігровий досвід.

3. Бот може автоматично аналізувати ходи гравців із використанням алгоритмів штучного інтелекту, що забезпечує складніший і адаптивніший рівень гри.

4. Надавати підказки та допомогу гравцям, наприклад, пояснення стратегій, пропозицію найкращих ходів або виведення правил гри за запитом.

6. Має підтримувати можливість перегравання, а також збереження поточного стану гри, щоб користувачі могли повернутися до неї в будь – який час.

7. Якщо передбачено обмеження часу на хід, бот повинен ефективно керувати часом, попереджати гравців про швидке завершення часу і надавати їм можливість прийняти рішення.

8. Бот може збирати дані про статистику ігор, надавати зворотний зв'язок про ігровий досвід і використовувати цю інформацію для поліпшення своєї роботи.

Адаптація під гру з ботом спрямована на створення приємного, цікавого та задовільного досвіду для користувачів, забезпечуючи при цьому баланс між динамікою гри та зручністю взаємодії з програмою.

Обробка та аналіз ходів.

Обробка та аналіз ходів у програмному забезпеченні для гри в хрестики – нулики є важливим етапом в розробці ігрового бота. Гра, яка на перший погляд може здатися простою, вимагає від розробників вдосконалених алгоритмів обробки кроків та аналізу гри з метою надання користувачам високоякісного інтерактивного досвіду. У цьому розділі розглянемо ключові аспекти обробки та аналізу ходів у програмному забезпеченні для гри в хрестики – нулики, звертаючи увагу на важливість оптимізації алгоритмів та покращення геймплейу для задоволення користувачів.

1. Бот повинен акуратно обробляти команди, що надходять від гравців. Це передбачає розпізнавання текстових команд, перевірку їхньої коректності та відповідність правилам гри.

2. Після приймання команди на здійснення ходу, бот повинен оновлювати ігрове поле відповідно до цього ходу. Візуальне представлення ігрового поля може бути оновлено для відображення останніх змін.

3. Після кожного ходу бот повинен перевіряти, чи не з'явився переможець. Перевірка наявності переможця – це перевірка рядків, стовпців і діагоналей на наявність однакових символів. У разі виявлення перемоги, бот має оголосити результат і завершити гру.

4. Перевірка наявності нічиєї, якщо всі комірки на ігровому полі заповнені, і переможець не оголошений, бот повинен визначити, що гра закінчилася нічиєю.

5. Аналіз стратегій гравців, якщо бот має штучний інтелект, він повинен аналізувати стратегії і тактики гравців. Це може включати в себе оцінку поточного становища і вибір оптимальних ходів для досягнення перемоги або запобігання поразки.

6. Після завершення гри бот має виводити результат (перемога, поразка, нічия) і, за необхідності, статистику за ходами і часом.

7. Бот може зберігати історію ходів і результатів ігор для подальшого аналізу або можливості перегравання.

8. У разі виникнення помилок, як некоректні ходи або непередбачувані ситуації, бот має коректно та інформативно обробляти ці сценарії, надаючи зрозумілі повідомлення про помилки.

Обробка й аналіз ходів важливі для створення функціонального й інтелектуального чат – бота для гри в хрестики – нулики, який забезпечує захопливий і цікавий ігровий досвід для користувачів.

Візуальне представлення гри та інтерактивна взаємодія.

Візуальне представлення гри відіграє важливу роль у створенні привабливого та зрозумілого інтерфейсу для користувачів. У контексті чат – бота в месенджері, такому як Telegram, візуалізація може здійснюватися в текстовому форматі або з

використанням графічних елементів, залежно від можливостей месенджера. Нижче наведено докладний опис візуального представлення гри:

1. Бот може використовувати текстове подання для відображення поточного стану гри, використовуючи числові та літерні позначення для клітинок і символів гравців.

Приклад:

```
 1 | 2 | 3
-----------
 4 | X | 6
-----------
 7 | 8 | O
```

2. Месенджер підтримує використання емодзі та графічних елементів, тому бот може візуалізувати хрестики й нулики.

Приклад:

⭕ | ❌ | ⭕

❌ | ⭕ | ❌

⭕ | ❌ | ⭕

3. Під час кожного ходу гравця бот може динамічно оновлювати текстове або графічне представлення ігрового поля для відображення останніх змін, створюючи інтерактивний досвід.

4. Можливості месенджера можуть дозволяти додавати інтерактивні елементи, як кнопки для здійснення ходу або вибору клітини.

5. Бот може надавати зворотний зв'язок про ходи гравців, підсвічуючи змінені клітини або виводячи повідомлення про вчинені дії.

Приклад: «Гравець X зробив хід у клітину 5.»

6. Після завершення гри бот може стисло оформляти результати, оголошувати перемогу, нічию або поразку, і надавати можливість перегратися.

Інтерактивна взаємодія є ключовим аспектом створення захопливого та приємного досвіду гри з чат – ботом. Елементи інтерактивної взаємодії в грі в хрестики – нулики включають:

1. Використання кнопок або команд для забезпечення легкого і зручного введення ходів.

Приклад:

Виберіть номер клітинки для ходу:

[1] [2] [3]
[4] [5] [6]
[7] [8] [9]

2. Після кожного ходу бот може динамічно оновлювати текстове або графічне представлення ігрового поля.

Приклад: «Гравець X зробив хід у клітинку 5.»

3. Підтримка текстових команд для взаємодії, таких як «Зроби хід у клітинку 3» або «Переграти».

Приклад:

Введіть номер клітини для ходу: [3]

4. Зворотний зв'язок і підсвічування змінених клітин.

Приклад: «Хід успішно здійснено! Поточний стан гри:»

5. Інтерактивні елементи, як кнопки для спрощення введення команд і полегшення взаємодії.

Приклад:

Виберіть ваш хід:

❌ | [2] | ⭕
---
[4] | ❌ | ⭕

⭕ | [8] | ❌

6. Підтримка перегравання, яка надає опцію почати нову гру без виходу з чату.

Приклад:

Гру завершено. Хочете зіграти ще раз? [Так] [Ні]

Інтерактивна взаємодія забезпечує легкість використання, зручність і задоволення від гри з чат – ботом, роблячи процес взаємодії цікавим і приємним для користувачів.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА TELEGRAM БОТА НА МОВІ ПРОГРАМУВАННЯ PYTHON

Детально розглянемо етапи проектування та розробки Telegram бота на мові програмування Python, враховуючи кращі практики у використанні інструментів, які забезпечують надійність, безпеку та зручність взаємодії. Аналізуючи приклади та впроваджуючи передові методики роботи з API Telegram, можна вивчити можливості створення функціонально багатогранних та ефективних ботів для різноманітних завдань. Такий підхід дозволяє не лише оволодіти конкретними технічними навичками, а й зрозуміти основні принципи проектування і розробки чат – ботів, що є важливим кроком у напрямку вдосконалення технологічної культури та впровадження інновацій в області інтерактивних інтерфейсів.

Проектування та розробка ботів на Python в цьому контексті визначається не тільки швидкістю та ефективністю мови програмування, але і широким спектром бібліотек, які сприяють інтеграції з іншими технологіями.

Проектування та розробка чат – ботів на платформі Telegram, зокрема на мові програмування Python, є актуальною та перспективною темою в контексті розвитку інтерактивних систем заснованих на штучному інтелекті. За останні роки зацікавленість у розробці та використанні чат – ботів значно зросла завдяки їхньому потенціалу полегшити комунікацію між користувачами та програмними системами.

В сучасному інформаційному суспільстві, на фоні стрімкого розвитку цифрових технологій та штучного інтелекту, створення ігрових додатків, зокрема ботів, стає не лише актуальною задачею, але й невід'ємною частиною розважального та технологічного простору. У цьому контексті виникає необхідність проектування та розробки Telegram бота, який відтворює класичну гру в хрестики – нуліки.

Гра хрестики – нуліки, відома своєю простотою та стратегічністю, виступає ідеальним полем для впровадження та вдосконалення алгоритмів штучного інтелекту в середовищі месенджера Telegram. Проектування та розробка Telegram

бота для цієї гри на мові програмування Python не лише визначається потребою у створенні забавного розважального інструменту, але і є інноваційним кроком у використанні сучасних технологій для створення інтерактивних ігор в онлайн середовищі.

2.1 Обґрунтування вибору мови програмування Python для розробки бота

Python – мова програмування, створена Гвідо ван Россумом наприкінці 1980х років, яка з плином часу стала однією з найпопулярніших і найзручніших мов для розробки програм. Її синтаксис простий і читабельний, що робить її привабливою для новачків та експертів. Python активно розвивається, і його екосистема охоплює безліч бібліотек і фреймворків для різноманітних додатків.

Ключовою особливістю Python є високий рівень абстракції, що дає змогу програмістам фокусуватися на розв'язанні завдань, не піклуючись про безліч низькорівневих деталей. Ця особливість робить Python чудовим вибором для розробки ігор для месенджера Telegram.

Ігри для Telegram вимагають гнучкості та легкості взаємодії з користувачем. Python забезпечує зручність у створенні інтерактивних і адаптивних користувацьких інтерфейсів. Багатство бібліотек і фреймворків у мові Python також надає розробникам потужні інструменти для опрацювання даних і логіки гри.

Завдяки активній спільноті розробників і відкритому вихідному коду, Python постійно оновлюється та вдосконалюється, що робить його надійним вибором для створення сучасних додатків, таких як ігри для месенджера Telegram.

Обґрунтування вибору мови програмування Python базується на ряді факторів, які забезпечують ефективну і якісну розробку Telegram бота для гри в хрестики – нуліки:

1. Читабельний синтаксис. Python володіє простим і лаконічним синтаксисом, що полегшує розробку та читання коду. Це особливо важливо для проектів, які включають багато логіки та взаємодіють з API.

2. Багата екосистема бібліотек і фреймворків. Python має велику кількість бібліотек і фреймворків, спеціалізованих на розробці чат – ботів та ігор. Наприклад, бібліотека aiogram спрощує взаємодію з Telegram API.

3. Підтримка штучного інтелекту (AI). Python є однією з найпопулярніших мов для розробки проектів з обробкою природної мови (NLP) та іншими аспектами штучного інтелекту. Це важливо для розуміння та обробки команд користувачів.

4. Широке використання у галузі розробки ботів. Python широко використовується для створення чат – ботів завдяки своїй простоті і зручності в розробці. Це дозволяє залучити широке коло розробників з різних напрямків.

5. Підтримка асинхронного програмування. Python версії 3.5 і вище надає вбудовану підтримку асинхронного програмування, що є важливим для оптимізації взаємодії бота з користувачами та API, забезпечуючи швидкий відгук на дії.

6. Крос – платформеність. Розроблений на Python код може бути запущений на різних операційних системах без значних змін, що робить його крос – платформеним і гнучким для використання в різних середовищах.

7. Активна спільнота та документація. Python має велику та активну спільноту розробників, що забезпечує доступ до численних ресурсів, форумів та документації для вирішення проблем і підтримки в процесі розробки.

Обґрунтування вибору Python зумовлене його зручністю, ефективністю та підтримкою широкого спектру інструментів, необхідних для успішної реалізації Telegram бота для гри в хрестики – ноліки.

2.2 Детальний опис архітектури бота та його основних компонентів

У цьому розділі розглядається архітектура бота, його структура та основні компоненти. Описуються модулі, класи, їхні взаємозв'язки, а також загальний потік даних у додатку.

Використовувані технології: Хмарний месенджер Telegram, Visual Studio Code, мова програмування Python версії 3.12.1, бібліотека Aiogram.

Розробка будь – якої програми починається з підготовки середовища.

Для початку встановлюємо VS Code. Завантажуємо інсталятор із сайту, запускаємо, встановлюємо. За замовчуванням середовище вже готове до роботи, але рекомендується встановити додаткові розширення для Python.

Після встановлення VS Code, створюємо на комп'ютері папку для проекту та структуру (рисунок 1), назвемо її Telegram_bot_tic_tac_toe. Вкажемо у VS Code шлях до нашої папки.

Наш бот буде розділений на кілька логічних частин – файлів, створимо файли. Можна писати весь код в одному файлі – він так само працюватиме, однак налагодження та пошук потрібної функції чи класу стане вкрай незручним.

Структура файлів для нашого Telegram – бота:

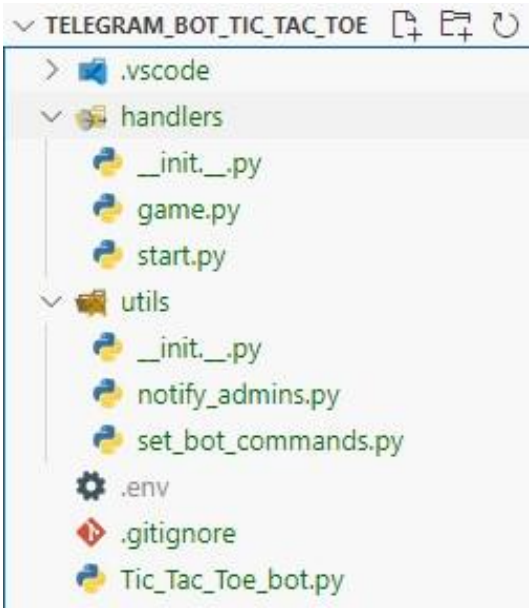

Рисунок 1. Папка проекту та його структура.

Це базова структура, і я можу налаштовувати її залежно від конкретних потреб проекту. Для кожного обробника (handler) у директорії handlers ми можемо створювати піддиректорії або використовувати додаткові модулі, якщо це зручно.

Директорія handlers для наших обробників команд і подій. Кожен файл може містити обробники для конкретного функціоналу бота. Наприклад, game.py може містити обробники для гри в хрестики – нулики, а start.py – обробник команди /start.

Файл «__init__.py» у директоріях Python використовується для визначення, що дана директорія є пакетом, а також для виконання ініціалізації під час імпорту пакета. У більшості випадків, цей файл залишається порожнім, але якщо нам потрібно виконати якісь додаткові дії під час імпорту пакета, можна додати код у цей файл.

Важливо, щоб цей файл був присутній у кожній директорії, яку ми хочемо розглядати як пакет. В іншому випадку Python може не розпізнати директорію як пакет, і ви можете зіткнутися з проблемами імпорту, якщо це необхідно.

У цьому коді ми використовуємо бібліотеку Aiogram для створення бота в Telegram. Команди /start і /newgame ініціалізують нову гру і відправляють поточний стан дошки в чат. Крім того, весь код асинхронний, що відповідає стандартам Aiogram.

Встановлення бібліотеки aiogram.

Почнемо зі створення базового каркаса для гри в хрестики – нулики в месенджері Telegram мовою Python. Для цього ми будемо використовувати бібліотеку aiogram.

Термінологія – введемо деякі терміни, щоб надалі не плутатися:

Встановимо цю бібліотеку через консоль, за допомогою команди pip install aiogram (рисунок 2):

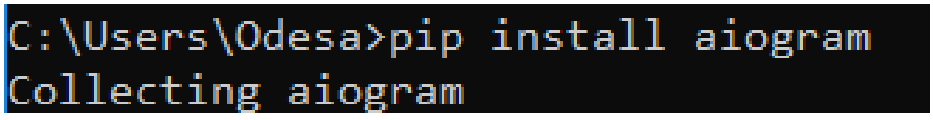

Рисунок 2

Напишемо код у файл game.py, він міститиме основну логіку для гри хрестики – нулики. Нижче наведено приклад коду.

Файл game.py (рисунок 3, додаток В) у цьому файлі міститься код, що обробляє дії, пов'язані з грою. Наприклад, функції для ходу гравця, перевірки умов перемоги та оновлення стану гри.

1. Отримання значень змінної ADMIN_IDS з файлу .env:

– Здесь імпортується бібліотека os та load_dotenv для роботи з оточенням віртуального середовища.

– ADMIN_IDS – це список адміністраторських ідентифікаторів, які визначаються у файлі .env. Ці ідентифікатори можуть бути використані для ідентифікації адміністраторів бота чи виконавців певних дій.

2. Клас TicTacToe для гри в «хрестики – нулики» визначає клас TicTacToe, який представляє гру в «хрестики – нулики». У конструкторі класу створюється ігрова дошка та встановлюється поточний гравець.

Метод print_board виводить ігрову дошку на екран у зручному форматі.

Метод make_move здійснює хід гравця на позначену позицію, перевіряючи коректність ходу.

Метод check_winner перевіряє, чи є переможець в рядках, стовпцях або по діагоналі гри.

Метод is_board_full визначає, чи заповнена вся ігрова дошка, що вказує на нічийний стан гри.

3. Створення об'єкту гри та передача значень ADMIN_IDS. Створюється об'єкт гри game з класу TicTacToe, і значення ADMIN_IDS передаються при його створенні. Це означає, що адміністраторські ідентифікатори можуть бути використані для адміністративного керування грою чи ботом.

```python
import os
from dotenv import load_dotenv

# Завантаження значень змінної ADMIN_IDS з файлу .env
ADMIN_IDS = os.getenv("ADMIN_IDS", "").split(",")

class TicTacToe:
    def __init__(self):
        # Ініціалізація гри: створення ігрової дошки та поточного гравця
        self.board = [" "] * 9
        self.current_player = "X"

    def print_board(self):
        # Виведення ігрової дошки на екран
        for i in range(0, 9, 3):
            row = " | ".join(self.board[i:i + 3])
            print(row)
            if i < 6:
                print("---------")

    def make_move(self, position):
        # Здійснення ходу гравця на позначену позицію
        if 1 <= position <= 9 and self.board[position - 1] == " ":
            self.board[position - 1] = self.current_player
            self.current_player = "O" if self.current_player == "X" else "X"
            return True
        else:
            return False

    def check_winner(self):
        # Перевірка наявності переможця в рядках, стовпцях та по діагоналі
        for i in range(0, 9, 3):
            if self.board[i] == self.board[i + 1] == self.board[i + 2] != " ":
                return self.board[i]

        for i in range(3):
            if self.board[i] == self.board[i + 3] == self.board[i + 6] != " ":
                return self.board[i]

        if self.board[0] == self.board[4] == self.board[8] != " ":
            return self.board[0]

        if self.board[2] == self.board[4] == self.board[6] != " ":
            return self.board[2]

        return None

    def is_board_full(self):
        # Перевірка, чи заповнена вся ігрова дошка
        return " " not in self.board

# Створення об'єкту гри з передачею значень ADMIN_IDS
game = TicTacToe(ADMIN_IDS)
```

Рисунок 3. Файл game.py – обробляє дії, пов'язані з грою

Файл start.py – це програма для бота, який грає в гру «хрестики – нулики» з користувачем (Рисунок 4, додаток Г).

Давайте подивимося на основні елементи:

1. Код починається імпортом необхідних бібліотек і класів для роботи з Telegram (наприклад, Bot, Dispatcher, types). Зчитується токен бота з файлу .env.

Токен – це унікальний ідентифікатор бота, отриманий при створенні його в Telegram через BotFather. Ваш токен залишається конфіденційним, тому він читається з файлу, що не включений у відкритий код. Ініціалізується бот та його диспетчер.

2. Створюється об'єкт гри «хрестики – нулики» з класу TicTacToe (завантаженого з іншого файлу game.py). Визначаються обробники для команди /start і текстових повідомлень, які складаються лише з цифр. Кожен обробник викликає відповідну функцію для обробки повідомлення. Наприклад, команда /start виводить вітальне повідомлення та стартовий стан гри, а введене число використовується як хід у грі. Після кожного ходу виводиться стан гри, і гра перевіряється на перемогу або нічию.

```python
import logging
from aiogram import Bot, Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.types import ParseMode
from aiogram.utils import executor
from game import TicTacToe
from dotenv import load_dotenv
import os

# Отримання значення токену бота з файлу .env
API_TOKEN = os.getenv("API_TOKEN")

# Ініціалізація бота та диспетчера
bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)

# Створення об'єкта гри
game = TicTacToe()

# Обробник команди /start
@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    # Відправлення вітального повідомлення і відображення ігрової дошки
    await message.answer("Привіт! Давай зіграємо в хрестики-нулики.")
    game.print_board()
    await message.answer("Хід гравця X. Введи число від 1 до 9, щоб поставити свій символ.")

# Обробник повідомлень, що складаються лише з цифр
@dp.message_handler(lambda message: message.text.isdigit())
async def make_move(message: types.Message):
    # Намагаємося зробити хід відповідно до введеного числа
    position = int(message.text)
    if game.make_move(position):
        game.print_board()
        winner = game.check_winner()
        if winner:
            await message.answer("Вітаю! Переміг гравець {}.".format(winner))
            game.reset()
        elif game.is_board_full():
            await message.answer("Нічия! Дошка заповнена.")
            game.reset()
        else:
            await message.answer("Хід гравця {}.".format(game.current_player))
    else:
        await message.answer("Некоректний хід. Спробуй ще раз.")

# Запуск бота
if __name__ == '__main__':
    from aiogram import executor
    executor.start_polling(dp, skip_updates=True)
```

Рисунок 4. Файл start.py – це програма для бота

Розглянемо далі директорію utils для допоміжних модулів. Наприклад, файли notify_admins.py і set_bot_commands.py можуть містити функції для сповіщень адміністратора і встановлення команд бота відповідно.

Код в файлі notify_admins.py (рисунок 5, додаток Д) використовує бібліотеку «aiogram» для надсилання сповіщень адміністраторам бота при його запуску та завершенні роботи. Давайте розглянемо основні елементи:

1. Імпорт бібліотек та змінних:

– from aiogram import types – імпортуються необхідні класи та типи з бібліотеки Aiogram для роботи з Telegram API.

– from handlers.game import ADMIN_IDS – імпортуються ідентифікатори адміністраторів з файлу game.py, що розташований в директорії handlers.

2. Функція `on_startup_notify`:

– async def on_startup_notify(dp) – асинхронна функція, яка викликається при запуску бота (on_startup). Призначена для надсилання сповіщень адміністраторам про готовність бота.

– for admin in ADMIN_IDS – проходження по списку ідентифікаторів адміністраторів.

– await dp.bot.send_message (admin, «Бот був запущений і готовий до використання!»). Надсилається повідомлення кожному адміністратору про успішний запуск бота.

– except types.ChatNotFound – обробляється випадок, якщо чат з адміністратором не знайдено, тобто бот не може відправити повідомлення.

3. Функція on_shutdown_notify:

– async def on_shutdown_notify(dp) – асинхронна функція, яка викликається при завершенні роботи бота (on_shutdown). Призначена для надсилання сповіщень адміністраторам про завершення роботи бота.

– for admin in ADMIN_IDS – проходження по списку ідентифікаторів адміністраторів.

– await dp.bot.send_message (admin, «Бот завершив свою роботу.»): Надсилається повідомлення кожному адміністратору про завершення роботи бота.

– except types.ChatNotFound – обробляється випадок, якщо чат з адміністратором не знайдено, тобто бот не може відправити повідомлення.

Цей код додає функціональність надсилання сповіщень адміністраторам при запуску та завершенні роботи бота, що може бути корисним для моніторингу та відладки.

```python
# Імпорт необхідних класів та типів з бібліотеки Aiogram
from aiogram import types

# Імпорт ідентифікаторів адміністраторів з файлу handlers/game.py
from handlers.game import ADMIN_IDS

# Функція для надсилання сповіщень адміністраторам при запуску бота
async def on_startup_notify(dp):
    # Цикл для кожного адміністратора в списку ADMIN_IDS
    for admin in ADMIN_IDS:
        try:
            # Надсилання повідомлення адміністратору про успішний запуск бота
            await dp.bot.send_message(admin, "Бот був запущений і готовий до використання!")
        except types.ChatNotFound:
            # Обробка винятку, якщо чат з адміністратором не знайдено
            print(f"Чат з користувачем {admin} не знайдено, сповіщення не відправлено.")

# Функція для надсилання сповіщень адміністраторам при завершенні роботи бота
async def on_shutdown_notify(dp):
    # Цикл для кожного адміністратора в списку ADMIN_IDS
    for admin in ADMIN_IDS:
        try:
            # Надсилання повідомлення адміністратору про завершення роботи бота
            await dp.bot.send_message(admin, "Бот завершив свою роботу.")
        except types.ChatNotFound:
            # Обробка винятку, якщо чат з адміністратором не знайдено
            print(f"Чат з користувачем {admin} не знайдено, сповіщення про завершення не
            відправлено.")
```

Рисунок 5. Файл notify_admins.py – сповіщення адміністраторам бота при його запуску та завершенні роботи

Далі файл set_bot_commands.py (рисунок 6, додаток Е) код в ньому встановлює команди для нашого Telegram – бота при його запуску. Давайте подивимося, як це відбувається:

1. Імпорт необхідних класів та типів:

– from aiogram import types – імпорт необхідних класів та типів з бібліотеки Aiogram для роботи з Telegram API.

2. Функція set_commands:

– async def set_commands(dp) – асинхронна функція, яка встановлює команди для бота. Параметр dp вказує на диспетчер бота;

– commands = [...] – створення списку команд бота. У цьому прикладі є три команди: "/start" для запуску бота, "/help" для отримання довідки та "/play" для початку гри в хрестики – нулики.

– await dp.bot.set_my_commands(commands) – встановлення цих команд для бота за допомогою методу set_my_commands. Команди відображатимуться, коли користувач введе "/".

3. Функція on_startup:

– async def on_startup(dp): Асинхронна функція, яка викликається при запуску бота (on_startup);

– await set_commands(dp): Виклик функції set_commands для встановлення команд при ініціалізації бота.

Отже, цей код дозволяє легко визначити та змінювати команди нашого бота, щоб користувачі могли легко спілкуватися з ботом через команди в чаті.

```python
# Імпорт необхідних класів та типів з бібліотеки Aiogram
from aiogram import types

# Асинхронна функція для встановлення команд для бота
async def set_commands(dp):
    # Ось список команд бота. Ви можете змінити його відповідно до ваших потреб.
    commands = [
        types.BotCommand("start", "Запустити бота"),
        types.BotCommand("help", "Отримати довідку"),
        types.BotCommand("play", "Розпочати гру в хрестики-нулики"),
    ]

    # Встановлюємо команди для бота за допомогою методу set_my_commands
    await dp.bot.set_my_commands(commands)

# Асинхронна функція, яка викликається при ініціалізації бота
async def on_startup(dp):
    # Викликаємо функцію set_commands для встановлення команд при запуску бота
    await set_commands(dp)
```

Рисунок 6. Файл set_bot_commands.py – команди для нашого Telegram – бота при його запуску

Надалі файл Tic_Tac_Toe_bot.py (Додаток Ж) містить налаштування бота, обробники команд, ініціалізацію гри, а також функції для відображення поточного стану гри та повідомлень адміністратору при запуску та завершенні бота.

Структура коду розділена на кілька частин:

1. Імпорт бібліотек та класів:

– import logging – імпорт модулю для обробки логування;

– from aiogram import Bot, Dispatcher, types – імпорт класів із бібліотеки Aiogram для роботи з Telegram API;

– from aiogram import executor – імпорт функції для запуску асинхронного виконання коду;

– from handlers import dp – імпорт об'єкта dp з файлу handlers;

– from utils.notify_admins import on_startup_notify – імпорт функції on_startup_notify з файлу notify_admins у папці utils;

– from utils.set_bot_commands import set_default_commands – Імпорт функції set_default_commands з файлу set_bot_commandsi у папці utils.

2. Налаштування бота:

– API_TOKEN = ... визначення токену для доступу до Telegram API;

– BOARD_SIZE = 3 – визначення розміру ініціалізованої дошки для гри.

3. Ініціалізація диспетчера та бота:

– bot = Bot(token=API_TOKEN) – створення об'єкту бота з використанням отриманого токену;

– dp = Dispatcher(bot): Створення об'єкту диспетчера, який використовується для обробки повідомлень.

4. Логування:

– logging.basicConfig(level=logging.INFO) – налаштування рівня логування на рівень INFO.

5. Логіка гри в хрестики – нулики (Рисунок 9):

– start_game() – функція для ініціалізації нової гри та обнуління гри на дошці;

– on_start(message) – обробник команди /start, який викликає start_game та вітає користувача;

– on_new_game(message) – обробник команди /newgame, який викликає start_game та виводить поточний стан гри;

– display_board(message) – функція для відправлення поточного стану гри на дошці в чат.

– @dp.message_handler(commands=['start'])` і @dp.message_handler (commands=[newgame') – декоратори, які встановлюють обробники команд /start та /newgame.

6. Запуск бота:

– if __name__ == __main__ – умова, яка перевіряє, чи файл є головним (не імпортований з іншого файлу);

– on_startup_notify(dp) – сповіщення адміністраторів при старті бота;

– set_default_commands(dp) – встановлення стандартних команд для бота;

– executor.start_polling(dp, on_startup=on_startup_notify): Запуск бота з використанням polling, передаючи функцію on_startup_notify для обробки стартових подій.

Файл .env для зберігання змінних оточення, таких як токен бота, ідентифікатори адміністраторів та інші конфіденційні дані. Також обов’язково додати .env до .gitignore, щоб уникнути публікації конфіденційної інформації в нашому сховищі.

Файл .gitignore для вказівки файлів і директорій, які не повинні потрапляти під версійний контроль Git. Включимо .env, щоб уникнути публікації конфіденційних даних.

В даному розділі була надана детальна інформація про архітектуру бота, його структуру та ключові компоненти. Ми розглянули модулі, класи, їх взаємозв'язки та загальний потік даних у додатку. Архітектура бота розроблена з використанням мови програмування Python версії 3.12.1 та бібліотеки Aiogram для роботи з хмарним месенджером Telegram. Процес розробки виконувався у середовищі Visual Studio Code.

В розділі наведено деталізований опис кожного компонента бота, включаючи його модулі, функції та класи. Також надана інформація про взаємодію компонентів та передачу даних між ними. Розглянуті ключові етапи обробки команд користувачів, відповіді бота та взаємодії з ігровим двигуном.

Використані технології, такі як месенджер Telegram, мова програмування Python та бібліотека Aiogram, обрані на підставі їхньої актуальності та ефективності для створення функціонального та високопродуктивного бота.

Усі ці аспекти архітектури допомагають створити добре структурований, ефективний та розширюваний бот для гри в хрестики –ноліки в середовищі Telegram.

2.3 Розгляд процесу взаємодії бота з користувачами та обробки їхніх введених команд

Користувачі можуть взаємодіяти із ботом, висилаючи текстові команди з вказівкою номера клітинки для виконання ходу. Бот обробляє команди, перевіряє їх на коректність і здійснює необхідні дії, такі як оновлення ігрового поля та відправлення відповіді користувачеві.

Користувачі взаємодіють з ботом, вводячи текстові команди. Команда для виконання ходу в грі може виглядати, наприклад, як "/ход 2,2", де координати вказують на конкретну клітину на ігровому полі. Команди також можуть містити додаткові параметри чи інформацію:

1. Модуль обробки команд перевіряє введений текст на наявність команд та параметрів. Використовуються регулярні вирази чи інші методи для виділення команди та інформації з тексту.

2. Після розпізнавання команди, проводиться валідація введених даних. Це може включати перевірку коректності введених координат чи інших параметрів, а також перевірку дозволених ходів в даному контексті.

3. Отримані від користувача команди передаються ігровому двигуну для обробки. Він визначає логіку гри, оновлює стан ігрового поля, перевіряє наявність переможця чи нічиєї.

4. Після обробки команди ігровий двигун оновлює стан гри. Результат (наприклад, зміни на ігровому полі) передається модулю користувацького інтерфейсу. Зміни відображаються в чаті користувача, а бот може відправляти повідомлення з результатами ходу чи підсумками гри.

Бот може взаємодіяти з користувачами не лише в грі, але й через інші команди, що надають доступ до різних функцій (наприклад, команда для початку нової гри, відображення правил тощо).

Бот може бути розроблений так, щоб розпізнавати різні формати введення команд, зробивши його більш гнучким та зручним для користувачів.

Система логування може використовуватися для реєстрації введених команд, аналізу їх використання та виявлення можливих проблем чи покращень у взаємодії з користувачами.

Цей процес взаємодії дозволяє боту ефективно спілкуватися з користувачами, обробляти їхні команди та забезпечувати гармонійну гру в хрестики – ноліки.

2.4 Використання вбудованих функцій Telegram API для зручної взаємодії з платформою

Розглянемо використання вбудованих функцій Telegram API для оптимальної і ефективної взаємодії з платформою. Зокрема, буде розглянуто вбудовані методи та можливості, які надає Telegram API, для спрощення розробки та покращення функціональності бота.

Вбудовані функції Telegram API:

1. Отримання, відправлення та обробка текстових повідомлень. Можливість обробки різноманітних медіа – файлів: фото, відео, документи тощо.

2. Створення і відправлення вбудованих клавіатур для зручного взаємодії з користувачами.

3. Реєстрація та використання команд бота для керування та запуску функціональності.

4. Зручне отримання та обробка інформації про користувачів, включаючи їх ідентифікатори, імена тощо.

5. Використання клавіш відповіді для структурованого та інтерактивного спілкування з користувачами.

Переваги використання вбудованих функцій:

1. Простота інтеграції. Вбудовані функції API дозволяють швидко та легко інтегрувати бота з Telegram, зменшуючи час на створення базового функціоналу.

2. Використання офіційних методів API забезпечує стабільну та надійну роботу бота, адаптовану до змін в Telegram API.

3. Наявність вбудованих інструментів дозволяє розширювати та кастомізувати функціонал бота, створюючи більш гнучкий інтерфейс.

4. З використанням вбудованих функцій бот може легко використовувати нові функції, які надаються Telegram, оновлюючи свій функціонал.

Приклад використання вбудованих функцій зображені на рисунку 7, Додаток З.

```python
from aiogram import Bot, types
from aiogram.dispatcher import Dispatcher

from utils.notify_admins import on_startup_notify

# Ініціалізація бота та диспетчера
bot = Bot(token="TOKEN")
dp = Dispatcher(bot)

# Відправка текстового повідомлення
@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    await message.answer("Ласкаво просимо! Для розпочатку введіть /help.")

# Взаємодія з клавіатурою відповіді
@dp.message_handler(commands=['help'])
async def help_command(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    buttons = ["Інструкції", "Контакти"]
    keyboard.add(*buttons)

    await message.answer("Оберіть опцію:", reply_markup=keyboard)

# Обробка вибору з клавіатури
@dp.message_handler(lambda message: message.text in ["Інструкції", "Контакти"])
async def handle_choice(message: types.Message):
    await message.answer(f"Ви обрали: {message.text}")

# Запуск бота
if __name__ == '__main__':
    from aiogram import executor
    from handlers import dp

    executor.start_polling(dp, on_startup=on_startup_notify)
```

Рисунок 7. Використання вбудованих функцій Telegram API

У цьому прикладі продемонстровано використання вбудованих функцій Telegram API для взаємодії з користувачем, відправки повідомлень, використання клавіатури відповіді

3 ОПТИМІЗАЦІЯ ТА РОЗШИРЕННЯ ФУНКЦІОНАЛУ

Цей розділ фокусується на покращенні швидкодії гри, вдосконаленні відгуку бота та розширенні можливостей взаємодії з користувачами. Розглянемо кожен підрозділ детальніше.

3.1 Впровадження алгоритмів оптимізації для покращення швидкодії гри та відгуку бота

У даному розділі розглянемо важливий аспект розробки бота – впровадження алгоритмів оптимізації для значущого покращення швидкодії гри та відгуку бота на введені команди користувачів.

Актуальність оптимізації в контексті гри в хрестики – нулики зі зростанням популярності ботів у месенджерах та ігор на їхній основі, виникає великий попит на забезпечення швидкодії та плавності взаємодії з користувачами. Особливо це стосується ігрових ботів, де швидка реакція на хід користувача є ключовим фактором задоволення гравців.

Вибір та впровадження алгоритмів оптимізації:

1. Використання ефективних алгоритмів для обробки введених команд користувачів та оновлення стану гри дозволяє зменшити час виконання операцій та покращити відгук бота.

2. Покращення алгоритмів визначення переможця, перевірка на нічию та забезпечення логіки гри без зайвого обчислення дозволяє зменшити обчислювальне навантаження та збільшити продуктивність.

3. Застосування асинхронних операцій в коді дозволяє боту паралельно виконувати завдання, такі як обробка кількох гравців чи одночасне оновлення стану гри.

4. Використання хешування результатів попередніх обчислень (наприклад, стану гри) дозволяє уникнути повторного обчислення та прискорити реакцію на події.

5. Розробка механізмів, які дозволяють боту ефективно працювати при різних рівнях навантаження, може включати автоматичне масштабування ресурсів, динамічну оптимізацію та інші стратегії, спрямовані на адаптацію до змін у роботі бота.

6. Запровадження системи моніторингу продуктивності дозволить вчасно виявляти можливі точки оптимізації. Збір та аналіз даних про використання ресурсів під час взаємодії з користувачами може служити основою для подальших покращень.

Впровадження алгоритмів оптимізації є критичним етапом розробки гри в хрестики – нулики через месенджер Telegram. Це дозволяє досягти значущого покращення швидкодії та відгуку бота, забезпечуючи плавну та задовільну гру для користувачів.

3.2 Додавання можливостей для гри з іншими користувачами через Telegram

Можливості з іншими користувачами через Telegram може бути важливим розширенням функціоналу бота. Ось кілька кроків та ідей, як це можна реалізувати:

1. Введення системи лобі, де користувачі можуть обирати опонентів або приєднуватися до існуючих ігор, дозволить створити зручний механізм для гри з іншими користувачами.

Створення системи лобі – це важливий елемент для організації ігор та спілкування між гравцями в боті. Така система надає користувачам можливість обирати опонентів, створювати нові ігри та приєднуватися до існуючих лобі. Розглянемо кроки, необхідні для реалізації системи лобі:

– визначити структуру об'єкту, який буде представляти лобі. Цей об'єкт може містити інформацію про гравців, налаштування гри (розмір поля, правила тощо) та інші важливі параметри;

– розробити команди для створення нового лобі, перегляду існуючих лобі, приєднання до лобі та інших відповідних операцій. Кожна команда може взаємодіяти з вищезгаданим об'єктом лобі та запускати відповідні дії;

– врахувати можливість зберігання інформації про лобі в базі даних. Це дозволить зберігати стан лобі навіть при перезапуску бота і забезпечить постійну доступність ігор для гравців;

– надати гравцям зручний спосіб перегляду інформації про існуючі лобі. Це може бути реалізовано через текстові повідомлення, кнопки або інші елементи інтерфейсу;

– забезпечити можливість гравцям обирати опонентів для гри. Це може бути здійснено за допомогою команд або взаємодії з інтерфейсом бота;

– розробити механізми для захисту приватності гравців та забезпечення безпеки участі в іграх. Наприклад, ідентифікація користувачів, обмеження доступу та інші заходи безпеки;

– врахувати можливість відправлення повідомлень та сповіщень про події в лобі, такі як нова гра, приєднання гравця чи зміни у налаштуваннях гри;

– забезпечити асинхронну обробку дій в системі лобі для ефективної взаємодії з багатьма гравцями одночасно та уникнення блокувань;

– провести тестування системи лобі, виявити можливі помилки та вдосконаліть функціонал на основі отриманих результатів;

Створення системи лобі дозволить гравцям легко організовувати та приєднуватися до ігор, покращить соціальний аспект взаємодії та забезпечить більш високий рівень зручності при грі в бота.

2. Реалізація запрошень – це додавання можливості для користувачів надсилати запитання на гру іншим користувачам через бота. Це може бути виконано за допомогою команди або кнопки, яка ініціює запрошення.

Реалізація запрошень в контексті бота може бути спрямований на організацію та впровадження механізму запрошень для залучення нових гравців в гру, роблячи процес взаємодії більш динамічним та зацікавлюючим.

Реалізація запрошень може бути структурована за різними концепціями, наприклад нижче розглянуть більш детально.

Структура та зміст включають в себе визначення концепції запрошень та їхнього впливу на гру, обговорення мети впровадження системи запрошень та розкриття переваг для гравців та бота від використання запрошень.

Архітектура та компоненти включають в себе опис структури системи запрошень та визначення ключових компонентів, таких як запрошення, статус запрошення, інтерфейс для взаємодії з запрошеннями тощо.

Створення та відправка запрошень включає в себе опис процесу створення запрошення та розгляд варіантів відправлення запрошень через команду, кнопку чи інший інтерфейс.

Прийняття та відхилення запрошень охоплює опис логіки для обробки прийняття та відхилення запрошень, а також визначення взаємодії гравців після прийняття запрошення.

Реалізація системи сповіщень та повідомлень для гравців про отримання чи відхилення запрошення.

Взаємодія із запрошеннями в інтерфейсі включає опис інтерфейсу взаємодії гравців із запрошеннями, а також реалізацію кнопок чи інших елементів інтерфейсу для зручного оброблення запрошень.

Забезпечення безпеки та контролю за використанням системи запрошень, уникнення можливих абузів.

Опис логіки гри при використанні запрошень, як вони впливають на стан інших гравців, чи можливість виходу з гри тощо.

Розгляд процесу тестування системи запрошень, виявлення та виправлення можливих помилок.

Аналіз та покращення включає аналіз результатів впровадження системи запрошень, а також визначення можливих напрямків для подальших покращень та розвитку цієї функціональності.

Введення системи запрошень в гру забезпечить нові можливості для взаємодії гравців та створить стимули для активної участі в ігровому процесі.

3. Створення системи лобі дозволяє гравцям спілкуватися та взаємодіяти перед початком гри. Цей розділ детально описує впровадження функціоналу для лобі гри, включаючи створення та приєднання до лобі, можливість обрати параметри гри (наприклад, розмірність поля чи правила гри), а також систему керування гравцями в лобі. Крім того, він розглядає можливості взаємодії з іншими гравцями через Telegram, такі як відправка запрошень та обробка їх.

Треба розглянути впровадження різноманітних налаштувань та можливостей для гравців перед початком гри. Це включає в себе можливість вибору параметрів гри, налаштування правил, комунікацію між гравцями (окремий канал для чату), а також інші функції, які роблять ігрове лобі більш гнучким та персоналізованим.

4. Взаємодія з друзями включає в себе можливість грати з друзями або учасниками того ж самого чату, використовуючи вбудовані функції Telegram для ідентифікації та взаємодії з користувачами.

5. Додавання механізму сповіщень, який інформує гравця, коли його опонент зробив хід, сприяє покращенню взаємодії та збільшенню зацікавленості в грі.

6. Введення системи рейтингу або статистики гравців, таблиць досягнень, топ учасників, може стимулювати змагальний дух і додатково залучити користувачів до участі в іграх.

7. Розгляд можливостей, які Telegram API пропонує для взаємодії між користувачами у групових чатах, групах та приватних повідомленнях, може допомогти оптимально використати функціонал месенджера для організації гри.

8. Надання інструкцій та довідкового матеріалу для користувачів щодо можливостей гри з іншими користувачами підвищить зручність та доступність функціоналу.

Додавання цих можливостей дозволить розширити соціальний та груповий аспект взаємодії з ботом, що сприятиме залученню більшого числа користувачів.

3.3 Розгляд можливостей розширення гри та взаємодії з користувачами за допомогою додаткових функцій та модулів

Розглянемо перспективи розширення гри та взаємодії з користувачами. Аналізуються можливості впровадження нових елементів у гру та додаткових функцій для покращення взаємодії з ботом.

1. Розширення гри включає в себе огляд можливостей для додавання нових елементів, таких як різні режими гри, розміри ігрового поля, нові правила тощо. Паралельно з цим, проводиться визначення процесу додавання нових елементів та оцінка їхнього впливу на ігровий процес.

2. Нові функції для користувачів включають розгляд можливостей для впровадження нових команд та функціональностей, які покращать взаємодію користувачів з ботом. Описується процес взаємодії з новими функціями та надаються відомості про їхні переваги для гравців.

3. Використання додаткових модулів обговорює можливості підключення додаткових модулів чи бібліотек для розширення функціоналу бота. Також визначається процес інтеграції та користування зовнішніми модулями для поліпшення роботи бота..

4. Система розширень розглядає можливості створення для бота, що дозволить розробникам легко додавати новий функціонал. Також включає опис інтерфейсу та логіки роботи цієї системи.

5. Логування та аналіз використання, розглядається як важливість системи логування для відстеження використання нових функцій та модулів, а також визначаються інструменти аналізу та методи збору статистики щодо взаємодії користувачів із розширеними можливостями гри.

6. Безпека та контроль якості розглядаються через обговорення заходів щодо забезпечення безпеки при впровадженні нових функцій та модулів, а також визначаються методи контролю якості для перевірки працездатності та надійності розширень.

7. Підтримка та документація включають розгляд питань підтримки користувачів у разі виникнення проблем із новим функціоналом та визначення необхідності створення документації для розробників та користувачів щодо використання розширень.

8. Перспективи та майбутні напрямки включають розгляд можливостей для подальшого розвитку та вдосконалення розширених функцій гри та взаємодії з користувачами, а також визначення перспективних напрямків для майбутньої роботи над ботом та його функціональністю.

Зазначені можливості охоплюють не лише технічні аспекти, а й практичні вигоди для гравців та бота.

Проведено огляд різноманітних напрямків розширення гри, включаючи додавання нових елементів, функцій та режимів гри. Зазначено важливість підходу до реалізації нового функціоналу, який не лише розширює можливості гравців, але й зберігає збалансованість гри.

Також розглянуті переваги додаткових функцій для користувачів, включаючи нові команди та інтерактивні можливості, які покращують взаємодію з ботом. Важливим елементом є визначення та імплементація системи розширень для зручного та ефективного додавання нового функціоналу.

Звернута увага на важливість логування та аналізу використання нових функцій для подальшого вдосконалення та вирішення можливих проблем. Безпека та контроль якості визначені як ключові аспекти при впровадженні розширень.

Зазначено важливість підтримки та створення документації для розробників та користувачів, а також виявлено потенційні напрямки подальшого розвитку та покращень у функціоналі бота. Розділ надає чіткий нарис можливих шляхів для розширення та вдосконалення бота в майбутньому.

ВИСНОВОК ТА РЕКОМЕНДАЦІЇ

В ході дослідження та розробки Telegram бота для гри в хрестики –ноліки на мові програмування Python було проведено аналіз ролі та значення ботів у месенджерах, зосереджено увагу на існуючих технологіях розробки ботів та їх можливостях в ігрових сценаріях. Описано основні принципи гри та їх реалізацію в програмному забезпеченні, включаючи детальну архітектуру бота та взаємодію з користувачами через вбудовані функції Telegram API.

У процесі оптимізації та розширення функціоналу були впроваджені алгоритми для покращення швидкодії гри та відгуку бота. Додатково була досліджена можливість гри з іншими користувачами через Telegram, розглянуті можливості розширення гри та покращення взаємодії з користувачами за допомогою додаткових функцій та модулів.

Наукові та практичні висновки цього дослідження відкривають нові можливості для розробки ігрових ботів у месенджерах, забезпечуючи оптимальну швидкість та різноманітний функціонал для користувачів. Результати дослідження вказують на значущий внесок у розвиток інтерактивних ігрових додатків на платформі Telegram, а також на потенціал вдосконалення взаємодії користувачів із ботом через впровадження нових функцій та модулів.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАНЬ

1. https://telegram.org/blog/bot-revolution
2. https://core.telegram.org/bots
3. https://telegram.org/blog/payments
4. https://botanalytics.co/
5. https://analytics.google.com/
6. https://prometheus.io/
7. https://www.datadoghq.com/
8. https://www.wireshark.org/
9. https://amplify.nginx.com/
10. https://matplotlib.org/
11. https://pandas.pydata.org/
12. Джессоп А. «Python Telegram Bot Quickstart». – 2021. Packt Publishing.
13. Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API Paperback – January 1, 2019.
14. «Python Crash Course, 2nd Edition: A Hands – On, Project – Based Introduction to Programming 3nd Edition». – 2021.
15. «Hands – On Chatbot Development with Alexa Skills and Amazon Lex» Sam Williams, 2020.
16. «Reinforcement Learning: An Introduction» Second edition, in progress, Richard S. Sutton and Andrew G. Barto, – 2014.
17. Офіційна документація Telegram API.

Додаток В Файл game.py – обробляє дії, пов'язані з грою

```
import os
from dotenv import load_dotenv

# Завантаження значень змінної ADMIN_IDS з файлу .env
ADMIN_IDS = os.getenv("ADMIN_IDS", "").split(",")

class TicTacToe:
    def __init__(self):
        # Ініціалізація гри: створення ігрової дошки та
поточного гравця
        self.board = [" "] * 9
        self.current_player = "X"

    def print_board(self):
        # Виведення ігрової дошки на екран
        for i in range(0, 9, 3):
            row = " | ".join(self.board[i:i + 3])
            print(row)
            if i < 6:
                print(" - - - - - - - - - ")

    def make_move(self, position):
        # Здійснення ходу гравця на позначену позицію
    if 1 <= position <= 9 and self.board[position - 1] == " ":
            self.board[position - 1] = self.current_player
            self.current_player = "O" if self.current_player ==
"X" else "X"
```

```
            return True
        else:
            return False
    def check_winner(self):
        # Перевірка наявності переможця в рядках, стовпцях та по
діагоналі
        for i in range(0, 9, 3):
            if self.board[i] == self.board[i + 1] ==
self.board[i + 2] != " ":
                return self.board[i]

        for i in range(3):
            if self.board[i] == self.board[i + 3] ==
self.board[i + 6] != " ":
                return self.board[i]

      if self.board[0] == self.board[4] == self.board[8] != " ":
            return self.board[0]

      if self.board[2] == self.board[4] == self.board[6] != " ":
            return self.board[2]

        return None
    def is_board_full(self):
        # Перевірка, чи заповнена вся ігрова дошка
        return " " not in self.board

# Створення об'єкту гри з передачею значень ADMIN_IDS
game = TicTacToe(ADMIN_IDS)
```

Додаток Г Файл start.py – це програма для бота, який грає в гру «хрестики – нулики» з користувачем

```
import logging
from aiogram import Bot, Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.types import ParseMode
from aiogram.utils import executor
from game import TicTacToe
from dotenv import load_dotenv
import os

# Отримання значення токену бота з файлу .env
API_TOKEN = os.getenv("API_TOKEN")

# Ініціалізація бота та диспетчера
bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)

# Створення об'єкта гри
game = TicTacToe()

# Обробник команди /start
@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    # Відправлення вітального повідомлення і відображення ігрової дошки
    await message.answer("Привіт! Давай зіграємо в хрестики - нулики.")
```

```
    game.print_board()
    await message.answer("Хід гравця X. Введи число від 1 до 9,
щоб поставити свій символ.")

# Обробник повідомлень, що складаються лише з цифр
@dp.message_handler(lambda message: message.text.isdigit())
async def make_move(message: types.Message):
    # Намагаємося зробити хід відповідно до введеного числа
    position = int(message.text)
    if game.make_move(position):
        game.print_board()
        winner = game.check_winner()
        if winner:
            await message.answer("Вітаю! Переміг гравець
{}.".format(winner))
            game.reset()
        elif game.is_board_full():
            await message.answer("Нічия! Дошка заповнена.")
            game.reset()
        else:
            await message.answer("Хід гравця
{}.".format(game.current_player))
    else:
        await message.answer("Некоректний хід. Спробуй ще раз.")

# Запуск бота
if __name__ == '__main__':
    from aiogram import executor
    executor.start_polling(dp, skip_updates=True)
```

Додаток Д Файл notify_admins.py – сповіщення адміністраторам бота при його запуску та завершенні роботи

```
# Імпорт необхідних класів та типів з бібліотеки Aiogram
from aiogram import types

# Імпорт ідентифікаторів адміністраторів з файлу handlers/game.py
from handlers.game import ADMIN_IDS

# Функція для надсилання сповіщень адміністраторам при запуску бота
async def on_startup_notify(dp):
    # Цикл для кожного адміністратора в списку ADMIN_IDS
    for admin in ADMIN_IDS:
        try:
            # Надсилання повідомлення адміністратору про успішний запуск бота
            await dp.bot.send_message(admin, "Бот був запущений і готовий до використання!")
        except types.ChatNotFound:
            # Обробка винятку, якщо чат з адміністратором не знайдено
            print(f"Чат з користувачем {admin} не знайдено, сповіщення не відправлено.")

# Функція для надсилання сповіщень адміністраторам при завершенні роботи бота
async def on_shutdown_notify(dp):
```

```
    # Цикл для кожного адміністратора в списку ADMIN_IDS
    for admin in ADMIN_IDS:
        try:
            # Надсилання повідомлення адміністратору про
завершення роботи бота
            await dp.bot.send_message(admin, "Бот завершив свою
роботу.")
        except types.ChatNotFound:
            # Обробка винятку, якщо чат з адміністратором не
знайдено
            print(f"Чат з користувачем {admin} не знайдено,
сповіщення про завершення не відправлено.")
```

Додаток Е файл set_bot_commands.py – команди для нашого Telegram – бота при його запуску

```
# Імпорт необхідних класів та типів з бібліотеки Aiogram
from aiogram import types

# Асинхронна функція для встановлення команд для бота
async def set_commands(dp):
    # Список команд бота. Ми можемо змінити його відповідно до наших потреб.
    commands = [
        types.BotCommand("start", "Запустити бота"),
        types.BotCommand("help", "Отримати довідку"),
        types.BotCommand("play", "Розпочати гру в хрестики - нулики"),
    ]

    # Встановлюємо команди для бота за допомогою методу set_my_commands
    await dp.bot.set_my_commands(commands)

# Асинхронна функція, яка викликається при ініціалізації бота
async def on_startup(dp):
    # Викликаємо функцію set_commands для встановлення команд при запуску бота
    await set_commands(dp)
```

Додаток Ж Файл Tic_Tac_Toe_bot.py – налаштування бота, обробники команд, ініціалізацію гри

```
# Імпорт необхідних бібліотек та класів
import logging
from aiogram import Bot, Dispatcher, types
from aiogram import executor
from handlers import dp
from utils.notify_admins import on_startup_notify
from utils.set_bot_commands import set_default_commands

# Токен бота для доступу до Telegram API
API_TOKEN = '5985426988:AAFMt4nnRcXzgHVO-CRpnDBby0i-anWqhTk'

# Розмір ініціалізованої дошки для гри
BOARD_SIZE = 3

# Ініціалізація гри: створення ініціалізованої дошки для
крестики-нулики
board = [[' ' for _ in range(BOARD_SIZE)] for _ in
range(BOARD_SIZE)]

# Ініціалізація бота та диспетчера
bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)

# Налаштування рівня логування
logging.basicConfig(level=logging.INFO)
```

```
def start_game():
    """Ініціалізація нової гри, обнулення гри на дошці."""
    global board
    board = [[' ' for _ in range(BOARD_SIZE)] for _ in
range(BOARD_SIZE)]

async def on_start(message: types.Message):
    """Обробник команди /start."""
    start_game()
    await message.answer('Для початку нової гри використовуйте
/newgame.')

async def on_new_game(message: types.Message):
    """Обробник команди /newgame."""
    start_game()
    await display_board(message)

async def display_board(message: types.Message):
    """Відправляє поточний стан гри на дошці в чат."""
    global board
    # Формування рядка для відображення гри на дошці в чаті
    board_str = '\n'.join([' '.join(row) for row in board])
    # Відправка повідомлення з відформатованою дошкою
    await message.answer(board_str,
parse_mode=types.ParseMode.MARKDOWN)

@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    """Обробник команди /start."""
```

```
    await on_start(message)

@dp.message_handler(commands=['newgame'])
async def new_game(message: types.Message):
    """Обробник команди /newgame."""
    await on_new_game(message)


if __name__ == '__main__':
    from handlers import dp
    from utils.notify_admins import on_startup_notify
    from utils.set_bot_commands import set_default_commands

    # Сповіщення адміністраторів при старті бота
    on_startup_notify(dp)
    # Встановлення стандартних команд для бота
    set_default_commands(dp)

    # Запуск бота з використанням polling
    executor.start_polling(dp, on_startup=on_startup_notify)
```

Додаток З Використання вбудованих функцій Telegram API

```
# Імпорт необхідних бібліотек та класів
import logging
from aiogram import Bot, Dispatcher, types
from aiogram import executor
from handlers import dp
from utils.notify_admins import on_startup_notify
from utils.set_bot_commands import set_default_commands

# Токен бота для доступу до Telegram API
API_TOKEN = '5985426988:AAFMt4nnRcXzgHVO – CRpnDBby0i – anWqhTk'

# Розмір ініціалізованої дошки для гри
BOARD_SIZE = 3

# Ініціалізація гри: створення ініціалізованої дошки для
крестики – нулики
board = [[' ' for _ in range(BOARD_SIZE)] for _ in
range(BOARD_SIZE)]

# Ініціалізація бота та диспетчера
bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)

# Налаштування рівня логування
logging.basicConfig(level=logging.INFO)

def start_game():
```

```
    """Ініціалізація нової гри, обнулення гри на дошці."""
    global board
    board = [[' ' for _ in range(BOARD_SIZE)] for _ in
range(BOARD_SIZE)]

async def on_start(message: types.Message):
    """Обробник команди /start."""
    start_game()
    await message.answer('Для початку нової гри використовуйте
/newgame.')

async def on_new_game(message: types.Message):
    """Обробник команди /newgame."""
    start_game()
    await display_board(message)

async def display_board(message: types.Message):
    """Відправляє поточний стан гри на дошці в чат."""
    global board
    # Формування рядка для відображення гри на дошці в чаті
    board_str = '\n'.join([' '.join(row) for row in board])
    # Відправка повідомлення з відформатованою дошкою
    await message.answer(board_str,
parse_mode=types.ParseMode.MARKDOWN)

@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    """Обробник команди /start."""
    await on_start(message)
```

```python
@dp.message_handler(commands=['newgame'])
async def new_game(message: types.Message):
    """Обробник команди /newgame."""
    await on_new_game(message)

if __name__ == '__main__':
    from handlers import dp
    from utils.notify_admins import on_startup_notify
    from utils.set_bot_commands import set_default_commands

    # Сповіщення адміністраторів при старті бота
    on_startup_notify(dp)
    # Встановлення стандартних команд для бота
    set_default_commands(dp)

    # Запуск бота з використанням polling
    executor.start_polling(dp, on_startup=on_startup_notify)
```