МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»_____,

освітньо-професійної програми «Інформаційні технології проектування»_____

на тему: Телеграм-бот для розвитку логічного мислення__

_____

Здобувачки групи ІТ-01Кириченко Альбіни Володимирівни
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

__ _____ Альбіна КИРИЧЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник ____ _к.т.н., доцент, Юлія ПАРФЕНЕНКО_____

(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) _____ (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В. о. зав. кафедри ІТ

Світлана ВАЩЕНКО

«_____» _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Кириченко Альбіни Володимирівни

1 Тема роботи Телеграм-бот для розвитку логічного мислення

керівник роботи *Парфененко Юлія Вікторівна, к.т.н., доцент*,

затверджені наказом по університету від « *07* » травня 2024 р. №0482-VI

2 Строк подання студентом роботи « 26 » травня 2024 р.

3 Вхідні дані до роботи

Технічне завдання на розробку телеграм-бота для розвитку логічного мислення

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Вступ, аналіз предметної області, моделювання телеграм-боту, розробка телеграм-боту, тестування телеграм-боту, висновки

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Постановка задачі (актуальність розробки, об'єкт та предмет дослідження, мета, задачі для досягнення мети, вимоги до створюваного програмного продукту)
Аналіз предметної області (огляд останніх досліджень і публікацій, аналіз існуючих продуктів-аналогів, вибір засобів реалізації)
Моделювання телеграм-боту (діаграма IDEF0 на концептуальному рівні, декомпозиція функціональної моделі 1го рівня, діаграма варіантів використання, логічна модель бази даних)

Розроблення телеграм-боту (створення телеграм-бота завдяки BotFather, створення основного функціоналу, процес додавання бази даних)
Результати роботи (вигляд меню бота, робота модуля швидкої математики, робота модуля Судоку, робота модуля Нонограм, вигляд і робота адміністративної панелі)
Тестування телеграм-боту (тест-кейси мануального тестування)
Висновки
Апробація (публікування в ІМА-2024)

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Оформлення технічного завдання	20.11 – 25.11.2023	виконано
2	Проведення аналізу предметної області та постановка задачі	01.12 – 15.12.2023	виконано
3	Оформлення планування робіт	18.12 – 28.12.2023	виконано
4	Проведення етапу моделювання	29.12.2023 – 09.01.2024	виконано
5	Проведення етапу розробки	10.01 – 16.05.2024	виконано
6	Проведення тестування	17.05 – 04.06.2024	виконано
7	Оформлення пояснювальної записки	11.06 – 25.05.2024	виконано

Студент _____ Альбіна КИРИЧЕНКО
(підпис)

Керівник роботи _____ к.т.н., доц. Юлія ПАРФЕНЕНКО
(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Телеграм-бот для розвитку логічного мислення».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 18 найменувань, додатків. Загальний обсяг роботи − 140 сторінок, у тому числі 66 сторінок основного тексту, 2 сторінки списку використаних джерел, 74 сторінки додатків.

Актуальність роботи грунтується на виявлених проблемах при аналізі предметної області. Зокрема це низький рівень розвитку мислення у сучасному суспільстві та монотонність розваг, які вказують на недостатній інтелектуальний стимул для розумової діяльності. Це спричиняється через малопривабливі форми навчання та недостатність розвивальних інструментів у месенджерах.

Мета роботи: розробка багатофункціонального телеграм-бота з привабливим оформленням та зручним інтерфейсом, призначеного для розвитку мислення користувачів месенджеру під час періодів неформального відпочинку, який буде вирізнятися своєю доступністю та різнонаправленністю у питанні способів розвитку та підтримки розумових здібностей.

Додатково: публікація в ІМА-2024 на тему «Аналіз популярності функціоналу додатків для розвитку мислення на основі наївного байєсівського класифікатора».

Ключові слова: Телеграм, бот, розвиток мислення, Python.

ЗМІСТ

ВСТУП

Сучасне життя вимагає ефективних методів розвитку мислення та управління часом, оскільки прискорений ритм інформаційного потоку породжує потребу у нових засобах саморозвитку. Стрес та втома, спричинені цими умовами, роблять актуальною потребу в нових способах саморегуляції та розвитку. Інформаційні технології можуть відігравати ключову роль у цьому процесі. Розробка інтелектуального бота, спрямованого на розвиток мислення та використання часу, може стати не лише засобом розваги, але й інструментом активного розвитку в періоди відпочинку.

Об’єкт – інформаційне забезпечення розвитку логічного мислення. Предмет – телеграм-бот для розвитку логічного мислення.

Метою кваліфікаційної роботи бакалавра є розробка багатофункціонального телеграм-бота з привабливим оформленням та зручним інтерфейсом, призначеного для розвитку мислення користувачів месенджеру під час періодів неформального відпочинку, який буде вирізнятися своєю доступністю та різнонаправленністю у питанні способів розвитку та підтримки розумових здібностей. Бот буде поєднувати головоломки, цікаві факти, вправи для швидкої математики та вивчення англійської слів рівня A1, так як все це – способи стимулювання роботи мозку, і інструменти розвитку мислення.

Для досягнення мети проекту необхідно виконати наступні задачі:

- визначити актуальність роботи, дослідити предметну область;
- провести аналіз аналогів телеграм-ботів та додатків;
- визначити функціональні можливості програмного продукту;
- визначити технології, які будуть використані для реалізації програмного продукту;
- розробити структуру та зовнішній вигляд телеграм-бота;
- реалізувати структуру та функціонал телеграм-бота;
- виконати тестування телеграм-бота.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Сучасний світ стикається з неабиякими викликами, пов’язаними із стрімким розвитком технологій та зміною соціокультурних тенденцій. В умовах постійного темпу життя та великого обсягу інформації, питання розвитку мислення та ефективного використання часу стають особливо актуальними. Ці аспекти впливають на якість життя і працездатність людей, тож дослідження в даній області набуває важливості та обґрунтованості.

Логічне мислення – це процес, під час якого особа використовує принципи та правила логіки для аналізу, розуміння і вирішення завдань. Цей термін визначає основний аспект в психології та когнітивних науках [1].

Розвиток логічного мислення може бути спричиненим різними факторами, такими як вивчення математики, філософії та вирішення складних завдань. Також, використання логічних головоломок, ігор та тренувань сприяє у зміцненні та розширенні цієї навички.

Узагальнюючи, логічне мислення є суттєвим компонентом розумового розвитку людини, дозволяючи аналізувати та розуміти складні проблеми в систематичний та послідовний спосіб. Це відіграє ключову роль у розвитку критичного мислення, прийнятті рішень та раціональному підході до прийняття рішень у різних сферах життя.

Логічне мислення надає основу для критичного мислення, створюючи систематичний та послідовний підхід до розуміння та аналізу інформації. Критичне мислення, в свою чергу, розширює цей підхід, додаючи оцінювання, аргументацію та прийняття обґрунтованих рішень на основі логічних принципів. Обидві навички спільно сприяють створенню компетентного та аналітичного мислителя.

Саме тому для суспільства, зокрема роботодавців, ці навички є важливими критеріями, про що вказують на рисунку 1.1 – дослідження Фонду молодих австралійців за 2016 рік [2].

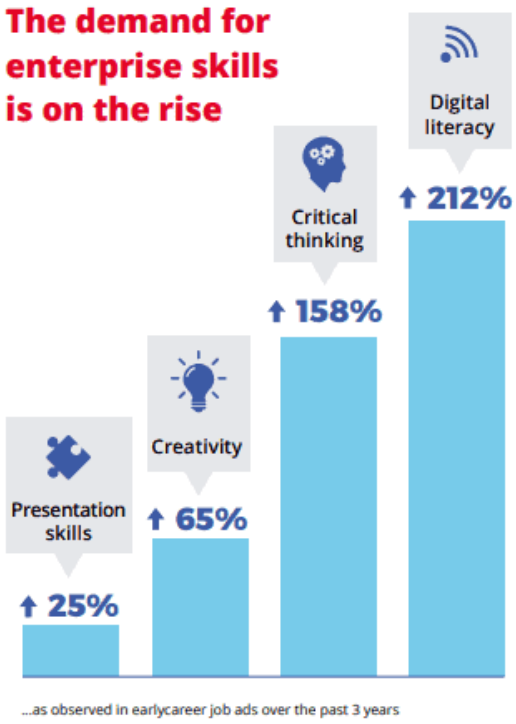


Рисунок 1.1 – Статистика попиту на навички критичного мислення

Наразі майже не залишилося людей, які б не користувалися тим чи іншим месенджером. Спираючись на джерела [3-6] – популярність Телеграм зростає, що можна бачити з рисунку 1.2 та 1.3, а отже його використання як бази для поширення бота – буде мати успішні наслідки.

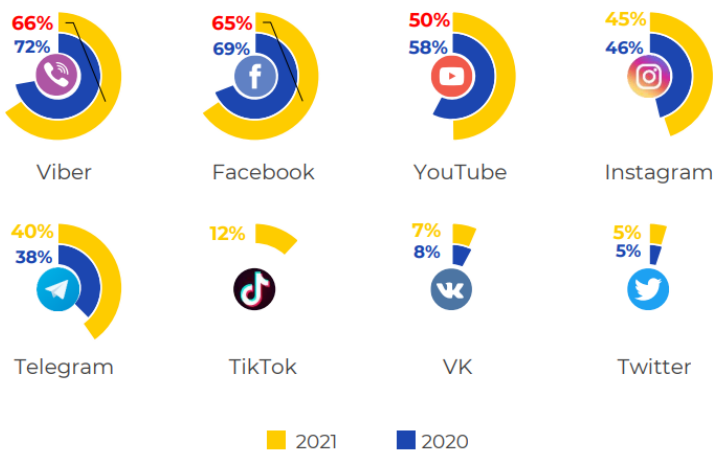


Рисунок 1.2 – Користування соціальними мережами, 2020-2021 роки

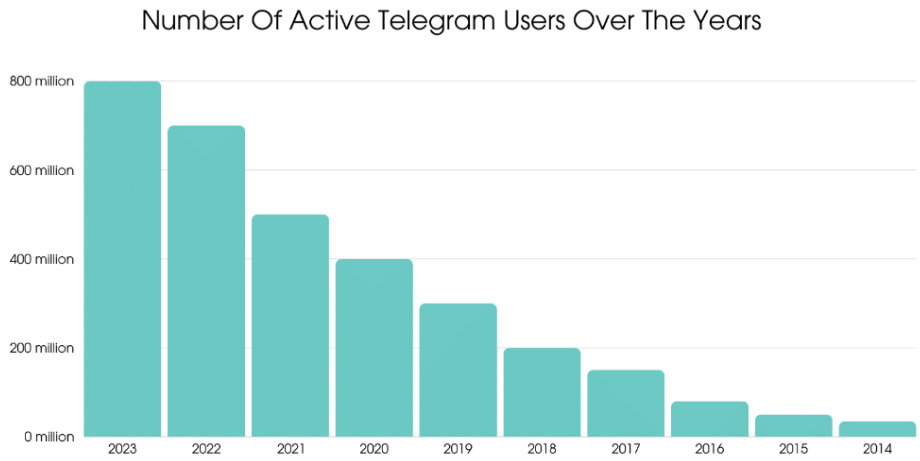


Рисунок 1.3 – Статистика активних користувачів Телеграм з 2014 до 2023 року

З появою та стрімким розвитком штучного інтелекту, технологічний прогрес ставить нові завдання, а саме – створення інноваційних рішень для покращення взаємодії людини з технологією. Серед важливих напрямків цього вектору вирізняється використання чат-ботів, які завдяки штучному інтелекту здатні не лише розуміти, а й ефективно спілкуватися з користувачами на більш продуктивному рівні.

Популярність використання чат-ботів серед користувачів з кожним роком зростає, зокрема для таких цілей, як швидке отримання докладної відповіді на певні питання чи запити. Про що можна пересвідчитися з рисунку 1.4. Аналізуючи

джерела [7-12], визначено активне використання чат-ботів у сфері маркетинга та бізнесах, відзначається важливість їх ролі в процесі діджиталізації. Доведена ефективність ботів і для шкільного навчання [13].

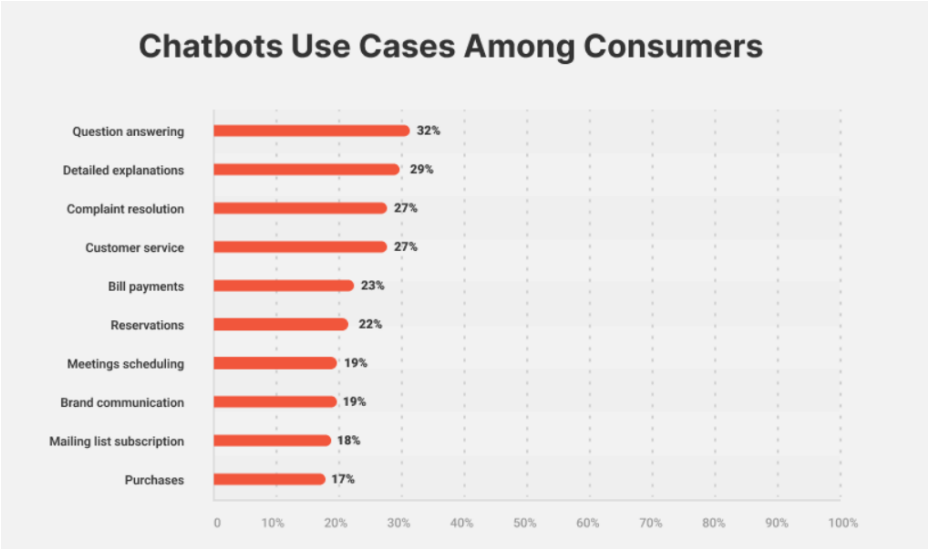


Рисунок 1.4 – Причини використання чат-ботів серед користувачів

Розумові здібності потребують постійного тренування для ефективного розвитку. Мозок, як вкрай пластичний орган, реагує на різноманітні розумові виклики. Отже, для досягнення оптимального розвитку інтелекту та мислення важливо забезпечувати йому щоденне навантаження [14].

Зокрема рекомендується наступне:

- читання художньої літератури: читання сприяє стимуляції всіх відділів мозку, включаючи уяву, розвиваючи когнітивні функції, пам’ять, уяву, абстрактне та понятійне мислення;
- вирішення логічних задач: займання логічними іграми, а також вирішення кросвордів і головоломок, сприяє розвитку мозкових здібностей. Навіть короткочасне щоденне тренування призводить до помітних результатів;
- розвиток математичного мислення: розв’язання математичних задач сприяє активізації мозкових процесів та покращенню швидкості мислення;
- вивчення іноземних мов: вивчення різних мов розширює способи мислення та формує нові перспективи.

Всі ці дії становлять не лише засіб розвивання мозку, а й спосіб впливу на особистий інтелектуальний рівень, зберігання гостроти розуму та підтримки активного мислення в повсякденному житті.

Головоломки – один з найстаріших жанрів ігор, але досі залишається популярним, про що свідчать рисунки 1.4 та 1.5. Це означає, що ігри, які стимулюють роботу мозку – цікавлять аудиторію. Отже, Судоку та Нонограм, які відносяться до цієї категорії та функціоналу телеграм-бота, будуть привертати увагу користувачів до результату проекта.

Рисунок 1.5 – Кількість додатків за жанрами у Google Play та їх відсоток від загального завантаження на момент грудня 2022 року

Рисунок 1.6 – Тенденція зміни кількостей завантажень з Google Play ігор категорії «Головоломки» за період грудня 2022 – листопада 2023

Серед великої кількості ігор даного жанру – одними з найпопулярніших залишаються Sudoku та Nonogram, завдяки одночасній простоті правил та

різноманітності рівнів від простих до більш складних. Про це свідчить те, що вони очолюють списки рекомендованих ігор (рис. 1.7), а також мільйонні значення завантажень додатків (рис. 1.8).

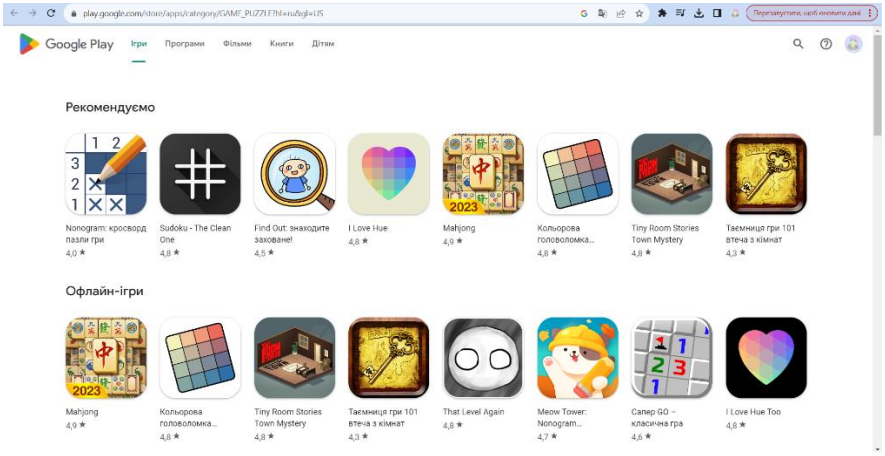


Рисунок 1.7 – Рекомендовані ігри жанру Головоломки в Google Play

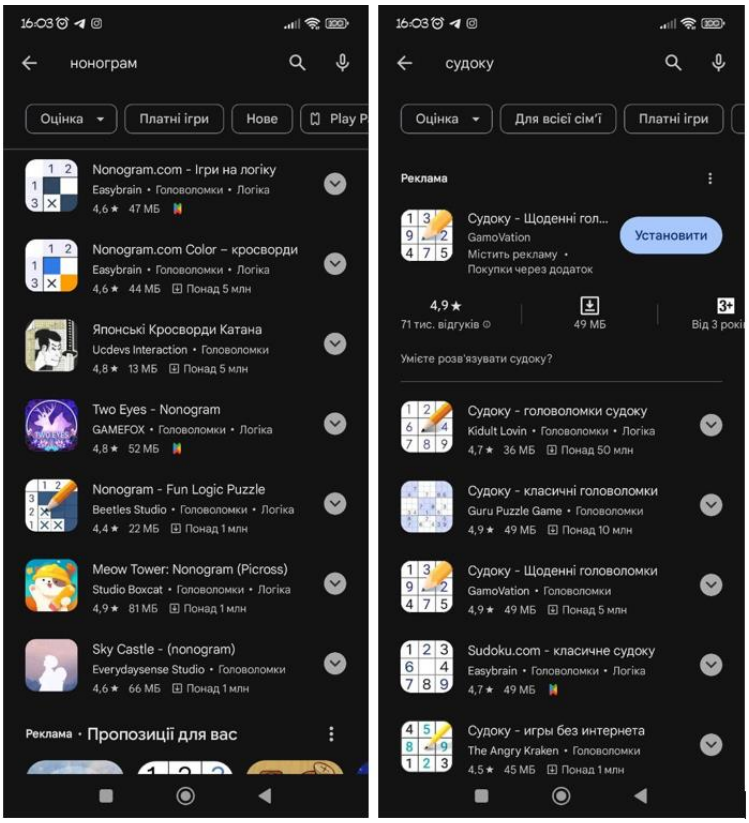


Рисунок 1.8 – Кількість завантажень і різноманіття ігор Sudoku та Nonogram у Google Play

1.2 Аналіз програмних продуктів - аналогів

Телеграм-боти вже перестали бути чимось новим для користувачів мережі Інтернет, особливо для користувачів даного месенджеру. Різноманітні телеграм-боти користуються великою популярністю, про що свідчать рейтинги в джерелі [15]. Але вони вузькоспеціалізовані під конкретну ситуацію, зокрема для розсилок, реклами, завантаження чи обробки даних. Здебільшого, замовники телеграм-ботів мають на меті їх комерційне використання, тобто для продажів чи популяризації товарів, а от працюючих ботів навчального спрямування – одиниці.

Проте існують окремі телеграм-канали, додатки та сайти, які виконують подібну мету, а отже є конкуруючими для продукту проекту.

Таким чином, для визначення вимог майбутнього програмного продукту було проведено дослідження існуючих аналогів:

- телеграм-бота «EasyWord»;
- телеграм-канала «Мозковий тренер | Логіка»;
- гри з вільним доступом у Google Play «Судоку – щоденні головоломки»;
- гри з вільним доступом у Google Play «Nonogram.com – ігри на логіку».

Телеграм-бот «EasyWord» – саме приклад навчального телеграм-боту. У нього вузька спеціалізація – вивчення англійських слів, проте широкий функціонал для цього: доступні опції прослуховування звучання слова, виведення всіх вивчених слів, проходження тестування знань. Проте весь спектр можливостей – відкритий лише при платній підписці. Також варто зазначити – наявність навчального відео роботи бота, при його першому запускі. Зовнішній вигляд оформлення повідомлень від бота – приємний, зручний, застосовуються смайлики для візуального відокремлення блоків інформації. У цьому можна переконатися на рисунку 1.9.

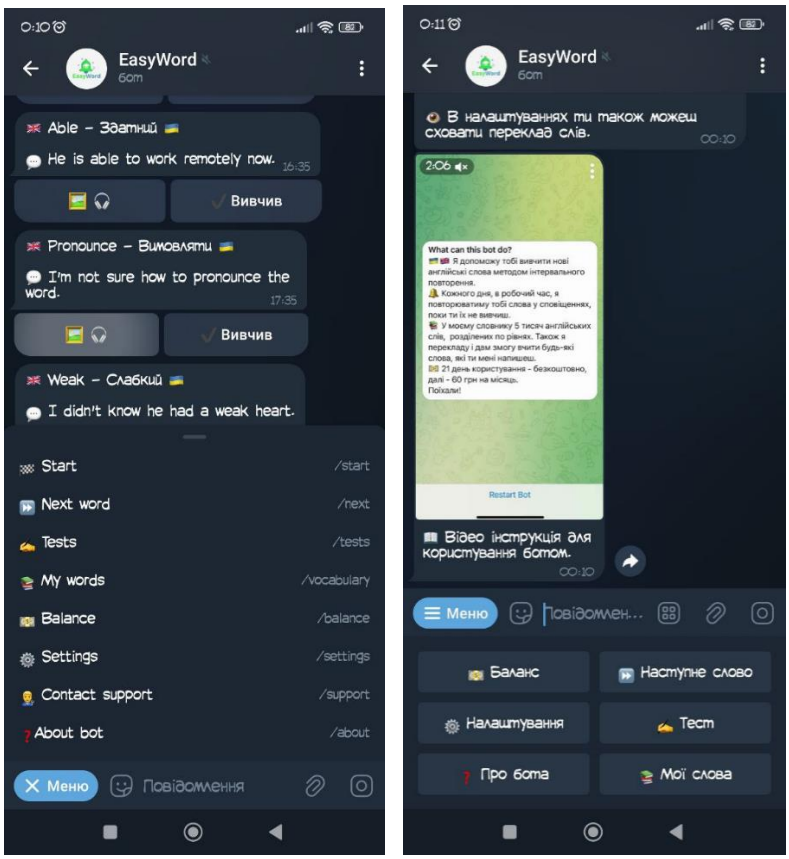


Рисунок 1.9 – Вигляд телеграм-боту «EasyWord»

Телеграм-канал «Мозковий тренер | Логіка» – приклад розвиваючого мислення каналу, завдяки періодичним постам з картинками логічних задач та цікавих фактів. Проте з тою ж періодичністю викладається реклама інших каналів, що псує загальне враження. Особливого оформлення немає, проте відмічу функціонал постів з задачами – наявність кнопок з різними відповідями на задачу. При натисканні на одну з них – вискакує вікно з вказанням чи правильно була дана відповідь, і при потребі – поясненням до вирішення. Побачити перечислене можливо на рисунку 1.10.

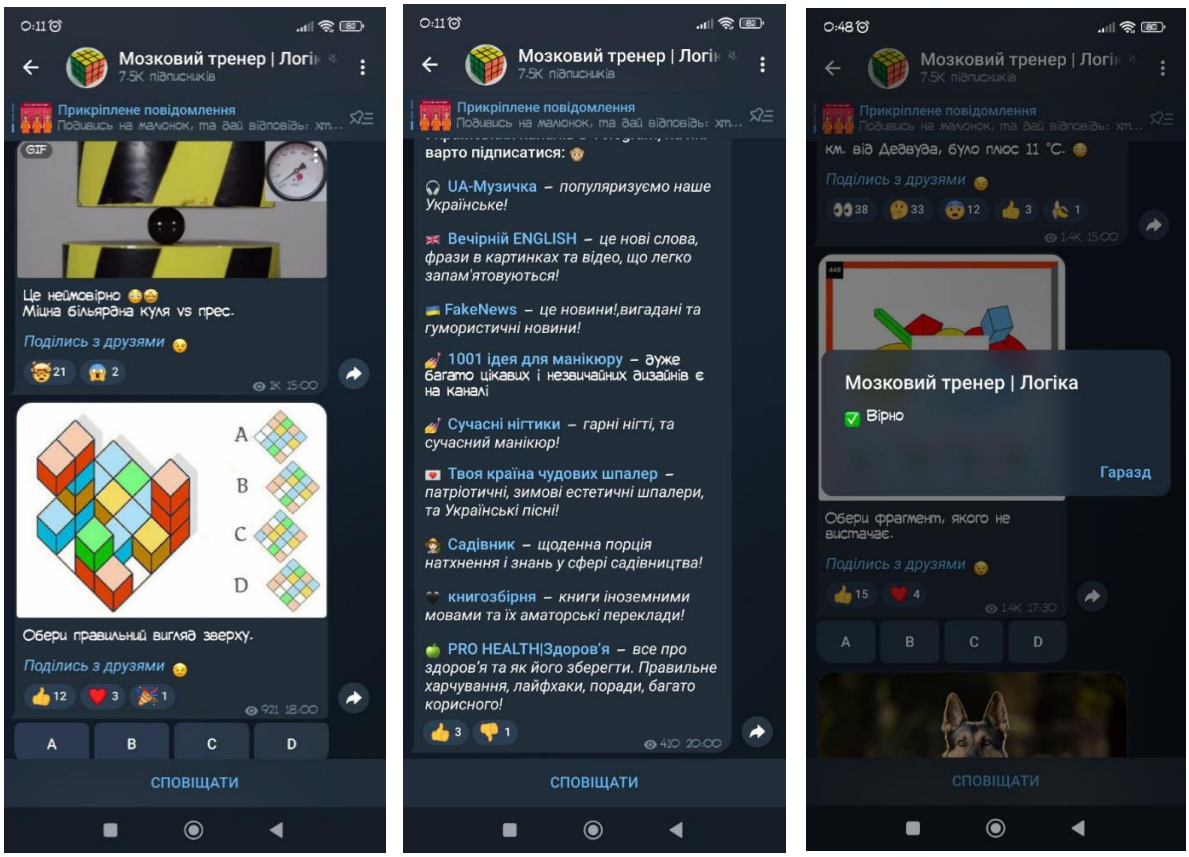


Рисунок 1.10 – Вигляд телеграм-каналу «Мозковий тренер | Логіка»

Гра з вільним доступом у Google Play «Судоку – щоденні головоломки» – безкоштовна, проте її потрібно окремо завантажити з Google Play. Завдяки простому і зручному інтерфейсу розібратися як працювати з нею не складно: правила гри – класичне судоку. Є поділ на рівні складності, а також класичні допоміжні інструменти для їх проходження: підказки, які можливо отримати або за ігрову валюту(монетки), або за рекламу. Процес гри відбувається у вигляді занесення цифр на поле: при правильному розташуванні – число стає синім, при неправильному червоним – що спрощує процес розв'язування задачі. У грі присутня платна частина – за реальні кошти – можливо купувати ігрову валюту і прибирати рекламу. Вигляд додатку представлений на рисунку 1.11.

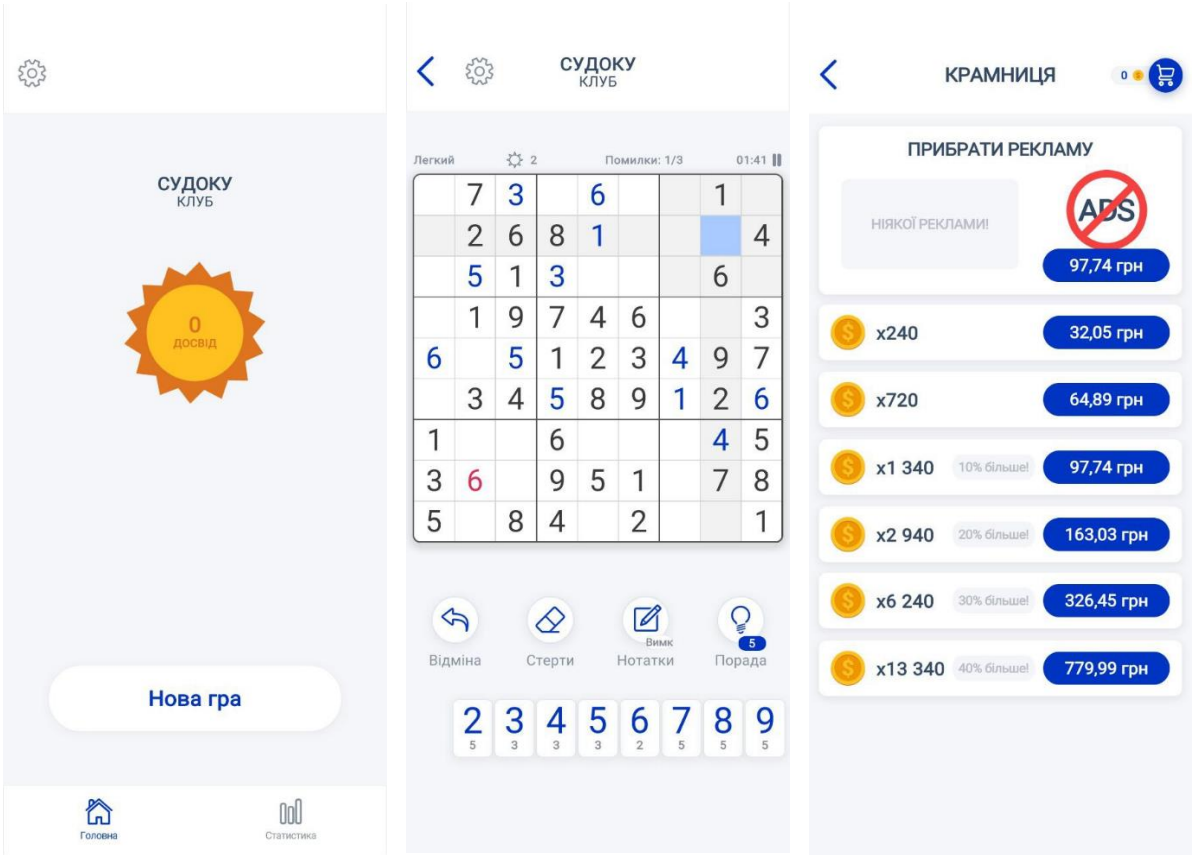


Рисунок 1.11 – Вигляд додатку «Судоку – щоденні головоломки»

Гра з вільним доступом у Google Play «Nonogram.com – ігри на логіку» – абсолютно безкоштовна, проте її потрібно окремо завантажити з Google Play. Інтерфейс простий, виконаний в світлих тонах, є різні рівні складності та періодично з’являються тимчасові івенти – для зацікавлення проведення часу користувачів у додатку. При першому запускі гри – запускається навчання, де пояснюється як саме варто замальовувати клітинки та користуватися інструментами, щоб завершити рівень. Варто зазначити, що є бібліотека всіх пройдених рівнів. Присутня реклама, для відключення якої – необхідно сплатити певну реальну грошову суму. Як виглядає гра – можна бачити на рисунку 1.12.

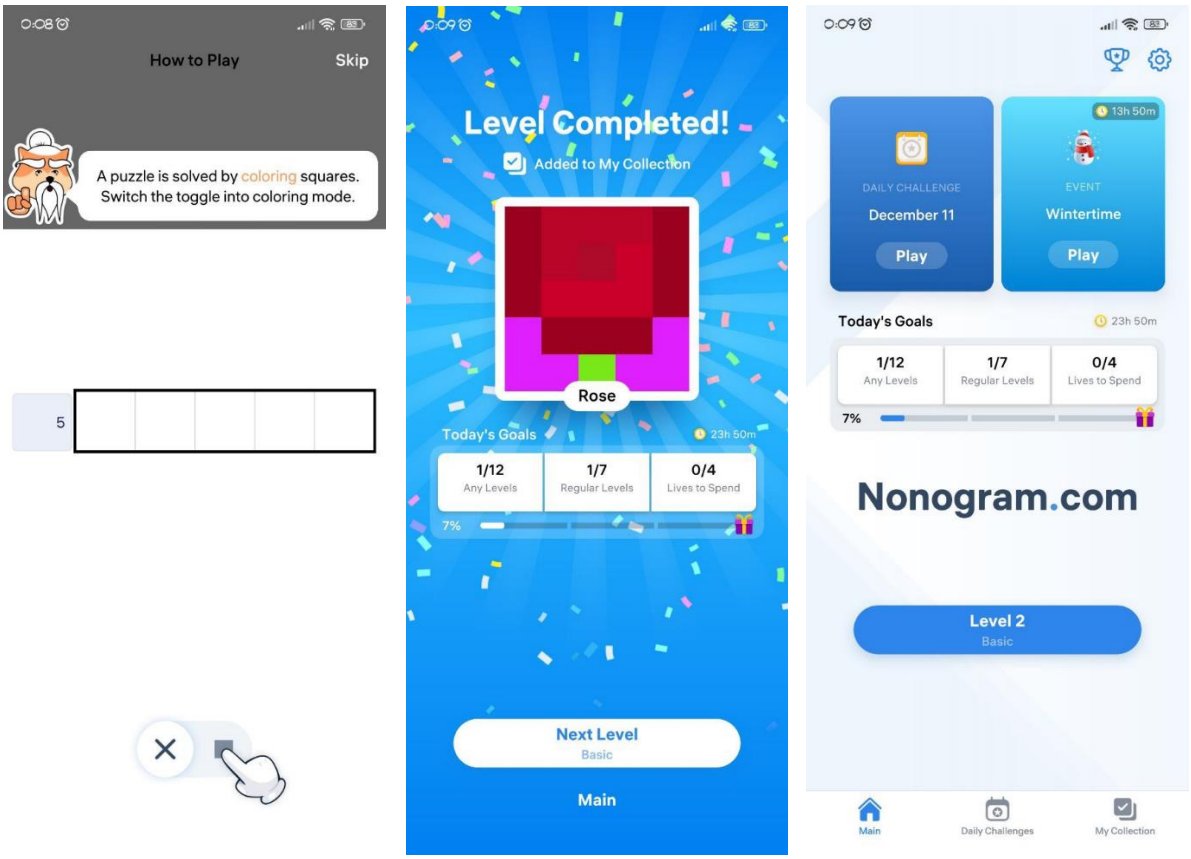


Рисунок 1.12 – Вигляд телеграм-боту «Nonogram.com – ігри на логіку»

Після детального аналізу аналогів засобів розвитку мислення, а саме додатків, телеграм-боту та телеграм-каналу було визначено їх переваги та недоліки. Його результати представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця характеристик аналогів

Аналог / Характеристика	«EasyWord»	«Мозковий тренер | Логіка»	«Судоку – щоденні головоломки»	«Nonogram.com – ігри на логіку»
Зрозумілий інтерфейс	+	+	+	+
Привабливий дизайн	+	–	+	+
Наявність навчальної частини	+	–	–	+

Продовження табл. 1.1

Аналог / Характеристика	«EasyWord»	«Мозковий тренер | Логіка»	«Судоку – щоденні головоломки»	«Nonogram.com – ігри на логіку»
Необхідність додаткового завантаження	–	–	+	+
Поєднання різних засобів розвитку мислення	–	+	–	–
Наявність сторонньої реклами	–	+	+	–
Наявність системи рівнів	–	–	+	+
Наявність платної частини	+	–	+	+

Дані з таблиці 1.1 надають змогу під час розробки звернути увагу на необхідні характеристики, які обов’язково потрібно втілити, а також на недоліки, які варто подолати. Насамперед – це потреба в зрозумілому для користувача інтерфейсі – такому, щоб користувач інтуїтивно розумів як користуватися і що потрібно натискати, при цьому дизайн повинен бути максимально привабливим, щоб зацікавити у подальшому використанні. По-друге, варто потурбуватися про навчальну частину при першому запускі бота, щоб людина відразу могла зрозуміти як ним користуватися. Найважливіший аспект для розроблюваного продукту – це поєднання різних засобів розвитку мислення, а серед аналогів це було представлено лиш у телеграм каналі, і то вузько, тоді як інші варіанти – були спеціалізовані на одному напрямку. Звичайно, виключаємо необхідність додаткового завантаження – адже бот буде в доступі у кожного користувача месенджера Телеграм. Варто взяти до уваги характеристику системи рівнів, так як бот повинен бути цікавим для продовження роботи з ним, а от наявність сторонньої реклами, призводить до протилежного ефекту – отже це потрібно виключити. Останній пункт – наявність

платної частини – буде гальмуючим для поширення та зацікавлення аудиторії в перші місяці, отже цього теж не буде додано, натомість – необхідність у грошовій підтримці теж з’являться, адже використання хостингу для розміщення бота – не безкоштовне. Таким чином буде додано пункт про можливість добровільного донату для користувачів на розвиток проекту.

1.3 Постановка задачі

Метою даного дослідження є розробка багатофункціонального телеграм-бота з привабливим оформленням та зручним інтерфейсом, призначеного для розвитку мислення користувачів месенджеру під час періодів неформального відпочинку, який буде вирізнятися своєю доступністю та різнонаправленністю у питанні способів розвитку та підтримки розумових здібностей.

Основні вимоги до створюваного програмного продукту є наступними:

- створити навчальну функцію для ознайомлення роботи з ботом при початку роботи з ним;
- організувати можливість запуску математичних тестів не високої складності;
- організувати можливість запуску режиму вивчення англійських слів та їх перекладу рівня A1;
- створити гру «Судоку» безпосередньо у боті з різними рівнями складності;
- створити гру «Нонограм» безпосередньо у боті з різними рівнями складності;
- забезпечити наявність системи рівнів як заохочення користування ботом;
- створити систему добровільного донату для розвитку проекту.

Для розроблення телеграм-боту необхідно:

- визначити функціональні можливості програмного продукту;
- спроектувати модель та структуру телеграм-бота;
- визначити зовнішній вигляд телеграм-бота;
- обрати технології для розробки телеграм-бота;
- створити прототип телеграм-боту;
- реалізувати структуру телеграм-боту;
- розробити функціонал телеграм-боту для розвитку мислення користувачів;
- виконати тестування телеграм-боту.

Вимоги до проекту в цілому, структури телеграм-боту, видів забезпечення та функціонування системи описані у технічному завданні на розробку проекту (додаток А).

Для розробки телеграм-боту, спрямованого на розвиток логічного мислення, вирішено використати сучасні технології, що забезпечують ефективність та функціональність продукту.

Обрання мови програмування Python для забезпечення швидкої, ефективної та стабільної розробки та функціонування телеграм-боту є стратегічно обґрунтованим рішенням з кількох ключових причин:

- висока читабельність та простота використання: Python славиться своєю простотою в освоєнні та використанні. Велика кількість простих та зрозумілих конструкцій мови дозволяє розробникам швидко і ефективно реалізовувати функціонал;
- широкий спектр бібліотек: Python має велику кількість готових бібліотек, що значно полегшує розробку. Для телеграм-ботів, наприклад, існують спеціалізовані бібліотеки, такі як Telebot, які надають готові інструменти для взаємодії з Телеграм API;
- активна спільнота та підтримка: Python є однією з найпопулярніших мов програмування, і він користується великою активною спільнотою. Це означає, що завжди є доступ до знань, розробників та ресурсів для вирішення будь-яких проблем чи питань;

– можливість використання великої кількості API: Python підтримує багато бібліотек для взаємодії з різноманітними API. Це особливо важливо для телеграм-ботів, які взаємодіють з Телеграм API.

Для обрання офіційного бота @BotFather для створення та керування телеграм-ботом теж є кілька важливих причин:

– зручний інтерфейс: бот надає зручний та інтуїтивно зрозумілий інтерфейс для налаштування параметрів бота. Використання цього бота робить процес створення та конфігурування бота простим та зрозумілим;

– швидке налаштування: бот дозволяє швидко налаштувати основні параметри бота, такі як ім'я, опис, а також отримати токен, необхідний для взаємодії з Телеграм API. Це значно прискорює процес старту проекту;

– офіційний інструмент: Як офіційний бот від Телеграм, @BotFather є надійним та безпечним інструментом для створення та керування ботами. Використання офіційних інструментів сприяє дотриманню стандартів та правил платформи;

– підтримка Телеграм API: бот надає можливість отримати необхідний токен для бота, який потім можна використовувати для взаємодії з Телеграм API. Це робить його важливим етапом для реалізації функціоналу та взаємодії бота з користувачами;

– оновлення та підтримка: Оскільки @BotFather є частиною офіційного месенджеру Телеграм, він регулярно оновлюється та підтримується, що забезпечує актуальність та стабільність його функцій.

Також обрання бази даних MySQL для створення проекту, пояснюється наступними чинниками:

– потужність та ефективність в обробці великого обсягу даних: забезпечує швидку та безперебійну роботу телеграм-боту, особливо в умовах високої активності користувачів Телеграм;

– надійність та стабільність: критичний аспект для постійної доступності та коректної роботи бота, забезпечуючи задоволення користувачів та позитивний досвід використання;

– широкий спектр інструментів для адміністрування та моніторингу: забезпечує зручне управління базою даних та ефективне виявлення та вирішення проблем у реальному часі.

2 МОДЕЛЮВАННЯ ТЕЛЕГРАМ-БОТУ

2.1 Структурно-функціональне моделювання процесу розробки боту

Функціональне моделювання процесу розробки телеграм-бота для розвитку логічного мислення в нотації IDEF0 представлене на рисунку 2.1.

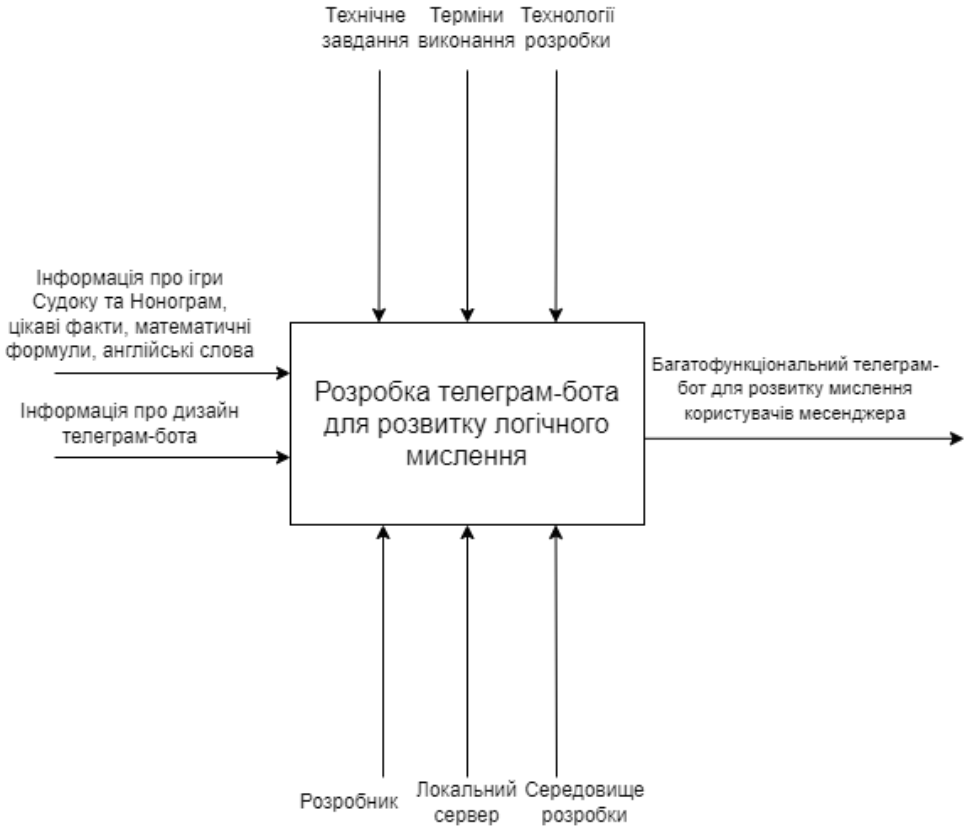


Рисунок 2.1 – Контекстна діаграма процесу розробки телеграм-бота нотації IDEF0

IDEF0 – методологія функціонального моделювання. Використовується для створення функціональної моделі, що відображає структуру і функції системи, а також потоки інформації і матеріальних об'єктів, що зв'язують ці функції [16].

На діаграмі по центру розміщений прямокутник – який позначає головний процес – «Розробка телеграм-бота для розвитку мислення».

Стрілки вказують на дані або матеріальні об'єкти, що пов'язані з функціями, і не представляють потоку або послідовності подій. Вони ілюструють, які дані чи об'єкти повинні надходити на вхід до функції, щоб вона могла виконуватися.

Стрілки входу (зліва) – відображають дані або об'єкти, які змінюються під час виконання роботи. Конкретно – це інформація про ігри Судоку та Нонограм, а також цікаві факти, математичні формули, англійські слова.

Стрілка виходу (справа) – показує дані або об'єкти, які формуються в результаті виконання роботи. На діаграмі – це багатофункціональний телеграм-бот для розвитку мислення користувачів месенджера, який охоплює різні напрямки розвитку мозку, такі як головоломки, математику, дізнання нових іноземних слів та цікавих фактів.

Стрілки управління (вхід зверху) – представляють правила та обмеження, згідно з якими здійснюється робота. На діаграмі це технічне завдання, терміни виконання та технології обробки.

Стрілки механізму (вхід знизу) – ілюструють ресурси, необхідні для виконання роботи, які не змінюються в процесі виконання. Зокрема, це розробник, локальний сервер та середовище розробки.

Контекстна діаграма показує лише загальний вигляд процесу, але не дає повного уявлення про його послідовність виконання. Щоб зрозуміти, як процес виконується, необхідно декомпозувати діаграму. Декомпозиція – це розбиття процесу на більш дрібні частини, які називаються задачами. Задачі можуть виконуватися послідовно або паралельно. Декомпозицію можна повторювати, доки не буде досягнута необхідна ступінь деталізації.

Декомпозиція функціональної моделі розробки телеграм-боту для розвитку мислення представлена на рисунку 2.2.

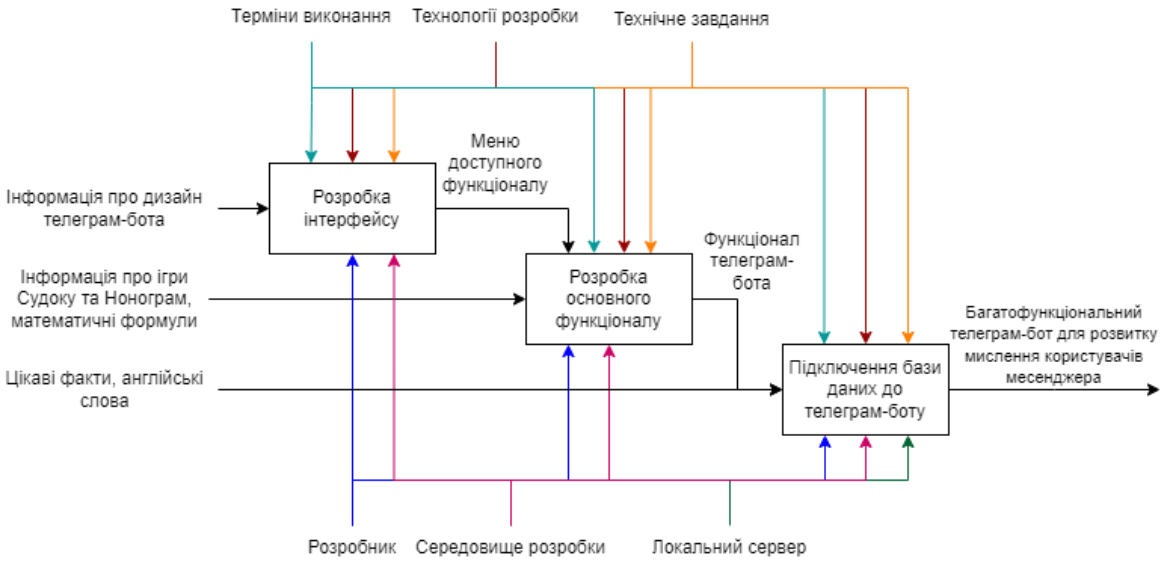


Рисунок 2.2 – Декомпозиція функціональної моделі процесу розробки телеграм-бота

На рисунку 2.3 представлена детальна декомпозиція другого рівня процесу «Розробка основного функціоналу».

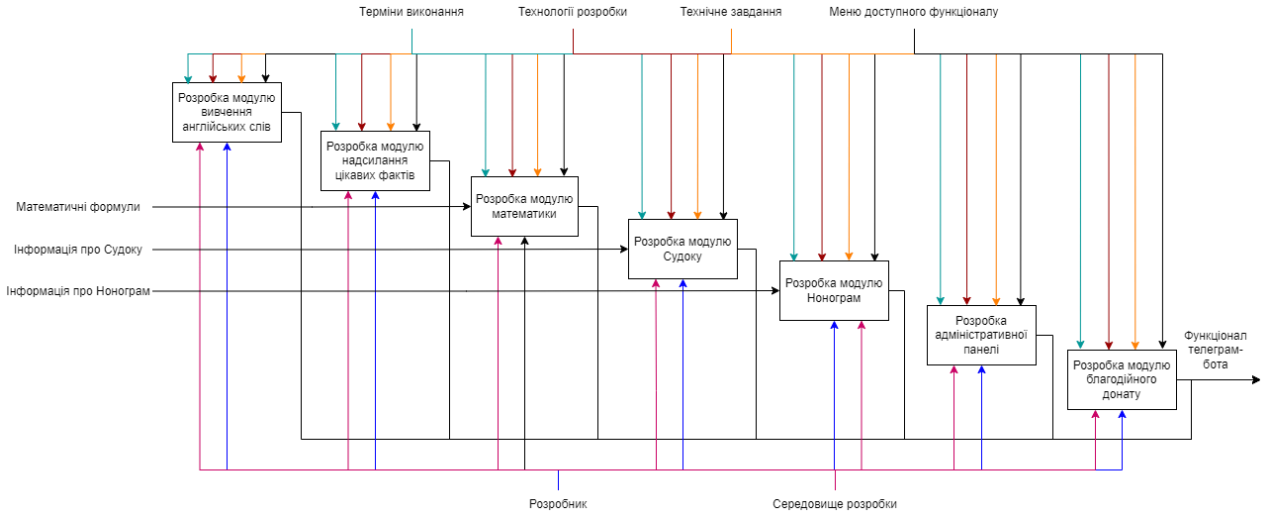


Рисунок 2.3 – Декомпозиція процесу «Розробка основного функціоналу»

Потрібно зазначити, що вихід, що отримується внаслідок виконання процесу «Розробка інтерфейсу», а саме «Меню доступного функціоналу» є елементом управління для наступного процесу «Розробка основного функціоналу». А вихід

цього процесу – «Функціонал телеграм-бота» є вхідними даними для процесу «Підключення бази даних до телеграм-боту».

На декомпозиції процесу «Розробка основного функціоналу» усі підпроцеси розроблюються незалежно один від одного і на виході дають «Функціонал телеграм-боту».

2.2 Структурно-функціональне моделювання на концептуальному рівні

Щоб показати використання телеграм-бота з точки зору користувача – була побудована діаграма в нотації IDEF0 на концептуальному рівні (рисунок 2.4).

Рисунок 2.4 – Контекстна діаграма в нотації IDEF0 на концептуальному рівні

Головний процес – «Розвиток логічного мислення ігровим методом». На вхід подаються – потреба у розвитку логічного мислення та текстове повідомлення від користувача.

У результаті виконання роботи отримується виконане завдання засобу розвитку мислення. Якщо конкретніше, то це або вирішення рівня гри Судоку або Нонограм, або проходження математичного тесту, або отримання нової інформації: цікавого факту чи англійського слова.

Правила та обмеження, згідно з якими здійснюється робота – це специфіка роботи чат-ботів та меню функціоналу телеграм-бота.

Ресурсами, які не змінюються в процесі, але є необхідними для виконання роботи, є користувач, бот, телеграм та модулі розвитку мислення.

Декомпозиція функціональної моделі розвитку логічного мислення ігровим методом представлена на рисунку 2.5. Головний процес був розбитий на три підпроцеси: запуск телеграм-бота, вибір засобу розвитку мислення та використання засобу розвитку мислення.

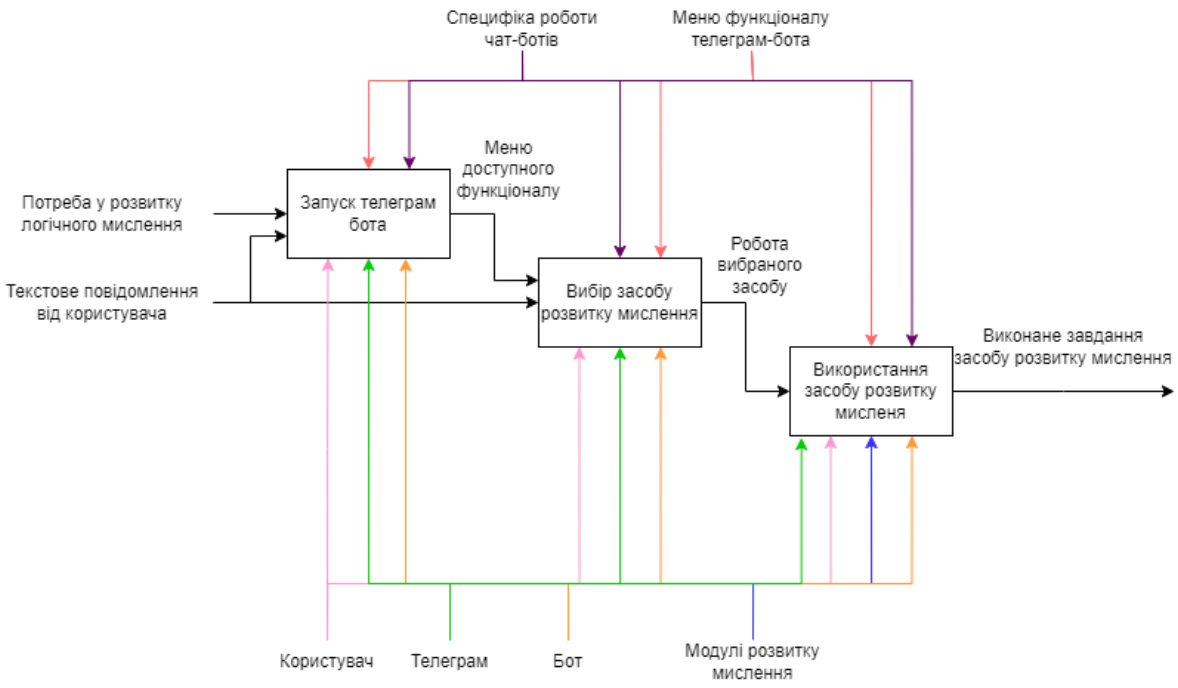


Рисунок 2.5 – Декомпозиція процесу розвитку логічного мислення ігровим методом

2.3 Моделювання варіантів використання боту

Для досягнення цілей функціонування спочатку будується модель у формі діаграми варіантів використання (use-case diagram), яка описує функціональне призначення системи. Діаграма варіантів використання є вихідною концептуальною моделлю системи в процесі її проектування та розробки.

Діаграма варіантів використання в нотації UML представлена на рис. 2.6.

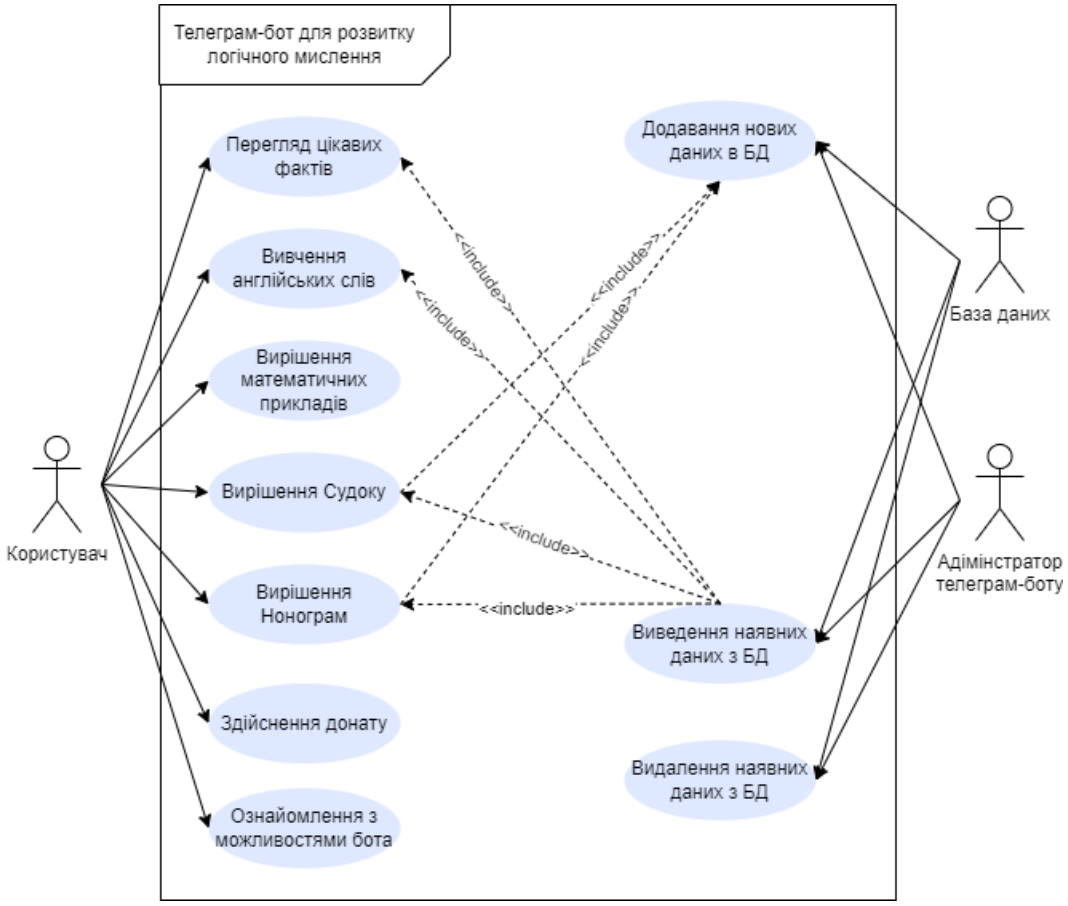


Рисунок 2.6 – Діаграма варіантів використання

На діаграмі представлені 3 актори: Користувач, База Даних та Адміністратор телеграм-боту.

Можливі види використання боту у ролі “Користувач”:

1. перегляд цікавих фактів – можливість запуску відповідного модулю телеграм-боту, коли бот надсилатиме рандомні цікаві факти, з тих, що містяться у базі даних;

2. вивчення англійських слів – можливість запуску відповідного модулю телеграм-боту, коли бот надсилатиме рандомні англійські слова, з тих, що містяться у базі даних, а також їх переклад і відповідне зображення;

3. вирішення математичних прикладів – можливість запуску відповідного модулю телеграм-боту, коли бот надсилатиме певні математичні приклади різної складності і чекатиме відповіді користувача. Потім звірятиме правильність отриманого результату, і надішле наступну задачу. В кінці тесту – бот вкаже на помилки;

4. вирішення «Судоку» – можливість запуску відповідного модулю телеграм-боту, коли бот надішле поле для гри з можливістю користувачу розставляти цифри у пусті комірки на полі;

5. вирішення «Нонограм» – можливість запуску відповідного модулю телеграм-боту, коли бот надішле поле для гри з можливістю користувачу зафарбовувати пусті комірки для отримання картинки;

6. здійснення благодійного донату – бот надішле відповідні реквізити для того, щоб користувач міг підтримати проект фінансово;

7. ознайомлення з можливостями боту – бот надішле повідомлення з інструкцією роботи з ним.

Можливі види використання боту у ролі “Адміністратор телеграм-боту”:

1. додавання нових даних в базу даних – можливість через адміністративну панель додати необхідну інформацію, яка буде використовуватися у модулях функціоналу боту;

2. виведення наявних даних з бд – можливість через адміністративну панель вивести список користувачів, які користувалися ботом та були занесені у базу даних;

3. видалення наявних даних з бд – можливість через адміністративну панель видалити конкретного користувача.

База даних задіяна у всіх процесах, окрім «Ознайомлення з можливостями бота», «Здійснення донату» та «Вирішення математичних прикладів».

2.4 Моделювання логічної моделі бази даних

Логічна модель даних - це абстрактне представлення даних, яке не залежить від конкретного програмного забезпечення або апаратного забезпечення. Вона використовується для розуміння структури даних, які зберігаються та обробляються в системі. Логічна модель даних є важливою частиною процесу аналізу та проектування систем, оскільки вона допомагає розробникам створювати ефективні та якісні системи.

На рисунку 2.7 представлена ER-діаграма телеграм-боту для розвитку мислення.

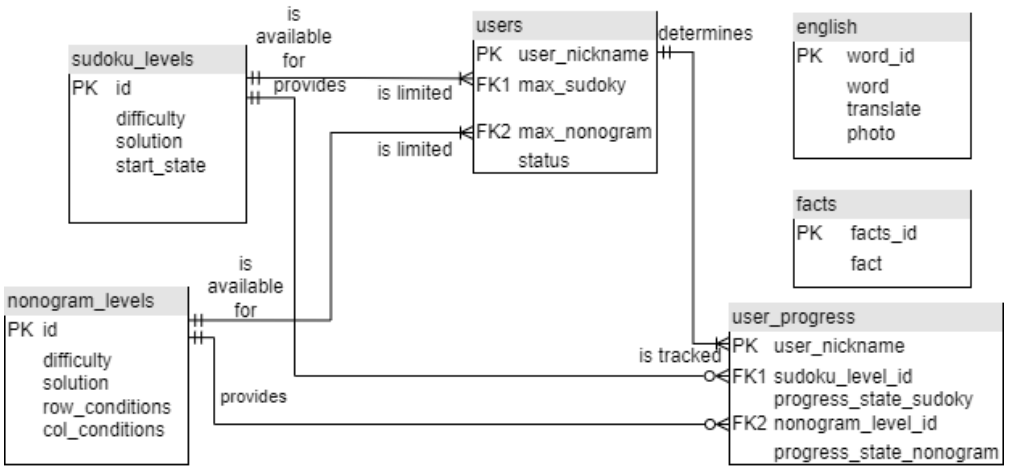


Рисунок 2.7 – Логічна модель бази даних для телеграм-боту

Усього в базі даних міститься 6 таблиць:

1. sudoku_levels(рівні судоку) – містить атрибути id(номер рівня – первинний автозаповнюваний ключ), difficulty(складність рівня), solution(рішення поля для гри), start_game(поле для початку гри);

2. nonogram_levels(рівні нонограм) – містить атрибути id(номер рівня – первинний автозаповнюваний ключ), difficulty(складність рівня), solution(рішення поля для гри), row_conditions(інформація про те, скільки клітинок у кожному рядку має бути заповнено) , col_conditions(інформація про те, скільки клітинок у кожному стовпці має бути заповнено);

3. users(користувачі) – містить атрибути user_nickname(первинний ключ – визначається нікнеймом користувача у месенджері), max_sudoky(максимально доступний рівень для проходження гри Судоку), max_nonogram(максимально доступний рівень для проходження гри Нонограм), status(чи є користувач адміністратором);

4. english(англійська) – містить атрибути word_id(автозаповнюваний первинний ключ), word(слово англійською), translate(переклад цього слова українською), photo(ілюстрація до цього слова);

5. facts(факти) – містить атрибути facts_id(автозаповнюваний первинний ключ), fact(текстова інформація, яка є цікавим фактом).

6. user_progress(прогрес користувача) – містить атрибути user_nickname(первинний ключ – визначається з таблиці users), sudoku_level_id(рівень судоку, який проходить користувач), progress_state_sudoky(поле судоку користувача) , nonogram_level_id(рівень нонограм, який проходить користувач), progress_state_nonogram(поле нонограм користувача).

Потрібно зазначити, що таблиці english та facts – не пов’язані з іншими, тому що виконують роль сховищ для інформації, яка використовується в відповідних модулях розвитку мислення. Інші таблиці – пов’язані між собою для відстеження прогресу користувача в іграх Нонограм та Судоку, і виконання обмеження доступу до рівнів. Таблиці sudoku_levels та nonogram_levels зберігають інформацію про рівні судоку та нонограм відповідно. Таблиця user_progress відстежує поточний прогрес користувача в іграх, зберігаючи інформацію про те, на якому рівні знаходиться користувач і в якому стані знаходяться ігрові поля для кожної гри.

Таким чином, зв'язок між таблицями забезпечує комплексне управління даними про користувачів та їх прогрес у іграх, дозволяючи ефективно керувати доступом до ігрових рівнів і відстежувати досягнення користувачів.

3 РОЗРОБКА ТЕЛЕГРАМ-БОТУ

3.1 Архітектура телеграм-боту

Загальна структура боту передбачає, що користувач надсилає повідомлення в бот через Телеграм-клієнт, надалі отримані дані обробляються відповідними модулями та користувачу надсилається відповідь.

Для побудови бота можна використати логіку MVC архітектури (рис. 3.1).

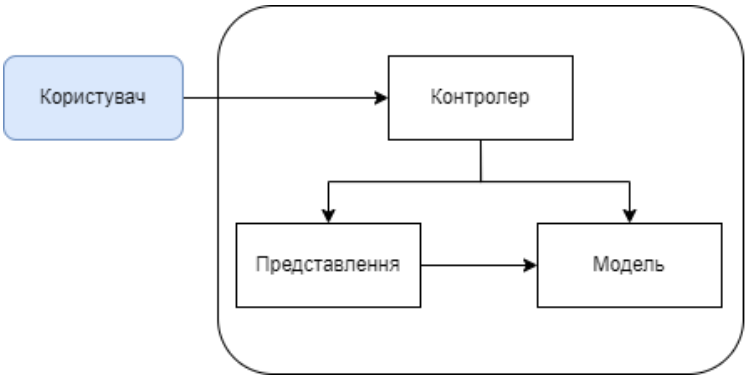


Рисунок 3.1 – Схема архітектури MVC

MVC (Model-View-Controller) архітектура є популярним підходом до розробки програмного забезпечення, який сприяє розділенню відповідальностей між різними компонентами системи. У контексті Телеграм-бота для розвитку мислення, MVC архітектура дозволяє структуровано організувати код, підвищуючи його зрозумілість і підтримуваність. Модель (Model) відповідає за обробку та збереження даних, таких як прогрес користувачів у іграх, факти та англійські слова. Представлення (View) забезпечує візуальне відображення інтерфейсу користувача через Телеграм-клієнт, включаючи текстові повідомлення. Контролер (Controller) обробляє вхідні дані від користувача, координуючи взаємодію між моделлю та представленням. Такий підхід забезпечує модульність, що полегшує розробку, тестування та розширення функціональності бота, а також підвищує ефективність розробки та підтримки системи.

Основні переваги MVC архітектури:

– розділення відповідальностей: кожен компонент (модель, представлення, контролер) виконує свої функції, що дозволяє легко управляти логікою, інтерфейсом та обробкою даних;

– модульність: спрощення підтримки та розширення системи завдяки незалежній розробці одного компоненту від іншого;

– повторне використання: можливість повторно використати компоненти в інших проектах або контекстах, що прискорює розробку та сприяє ефективному використанню ресурсів;

– тестування: полегшення знаходження та виправлення помилок, завдяки окремому тестуванню кожного елемента.

Архітектура телеграм-бота для розвитку логічного мислення зображена на рисунку 3.2.

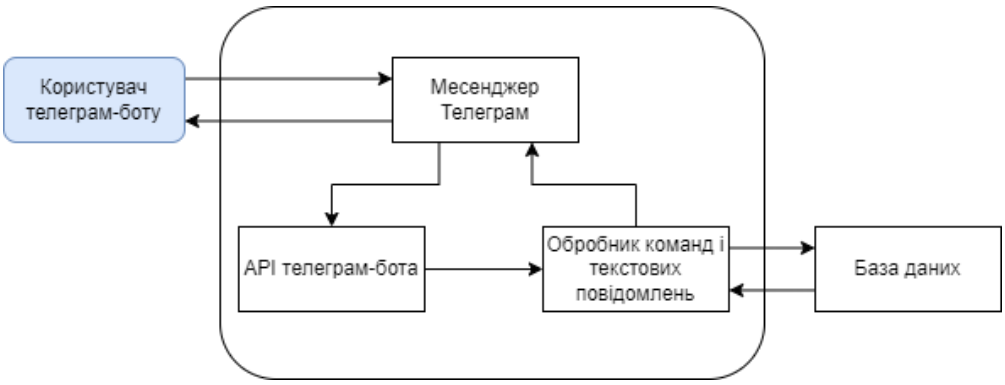


Рисунок 3.2 – Схема архітектури телеграм-боту

Елементи на даній схемі: View (Представлення): Месенджер Телеграм; Controller (Контролер): API телеграм-бота; Model (Модель): Обробник команд і текстових повідомлень. Таким чином, спочатку користувач надсилає повідомлення до месенджеру, далі Телеграм передає його до API телеграм-бота. API телеграм бота обробляє повідомлення і передає його до обробника команд та текстових повідомлень. Обробник взаємодіє з базою даних, формує відповідь і відправляє її назад до месенджеру. Телеграм відправляє відповідь користувачу.

3.2 Реалізація телеграм-боту

Спочатку необхідно було створити новий обліковий запис телеграм-бота та отримати його API токен, який використовуватиметься для інтеграції з Телеграм і надсилання та отримання повідомлень. Для цього використано офіційний бот BotFather. Завдяки його вбудованому інтерфейсу процес створення нового бота є простим і зрозумілим: користувач надає боту назву та унікальне користувацьке ім'я, після чого BotFather генерує унікальний API токен, який можна використовувати для взаємодії з Телеграм API. Надалі встановлено фото профілю бота (зображення було згенеровано штучним інтелектом – Adobe Firefly). Процес можна відстежити на рисунку 3.3.

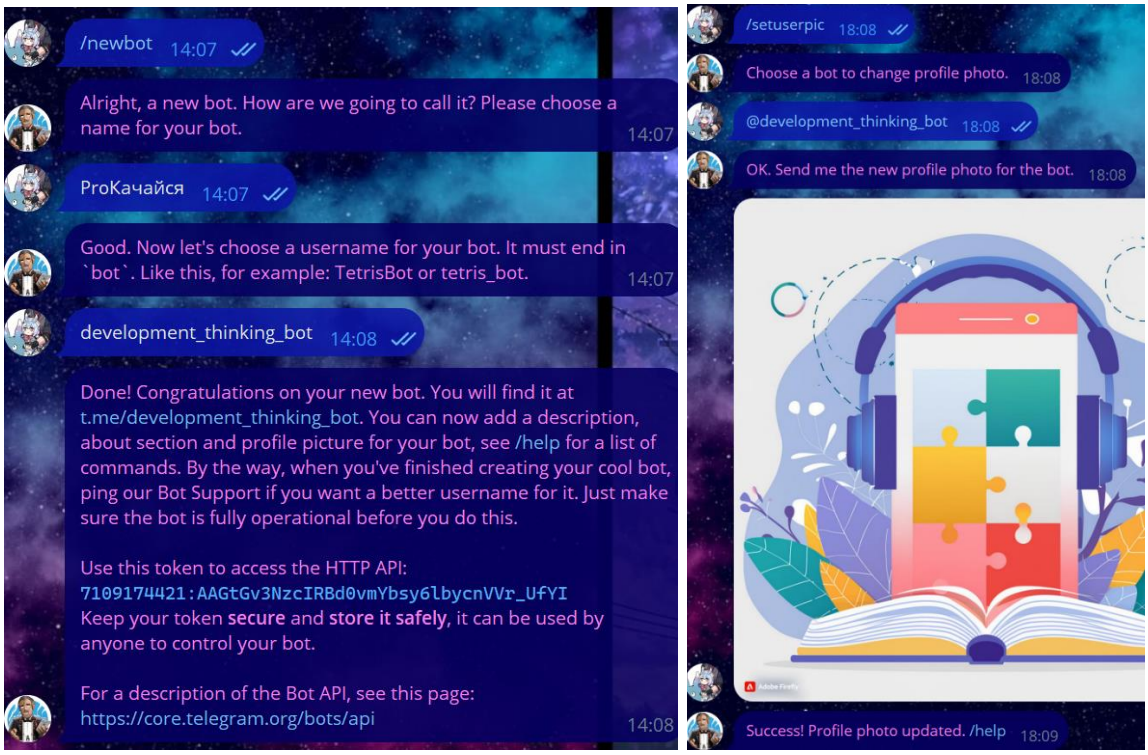


Рисунок 3.3 – Створення нового телеграм-бота, призначення йому назви та користувацького імені. Отримання API та встановлення фото профілю бота

Подальша реалізація телеграм-боту виконувалася в декілька етапів: розробка інтерфейсу, розробка основного функціоналу та підключення бази даних.

Інтерфейс для користувача вирішено зробити як дві варіації меню: через вибір текстової команди та через натискання кнопок. Текстове меню повинно мати весь перелік можливих команд, доступних користувачеві: ознайомитися з ботом, благодійно задонатити, дізнатися цікавий факт, вивчати англійську, швидка математика, грати судоку, грати нонограм. Воно було створено завдяки боту BotFather (рисунок 3.4).

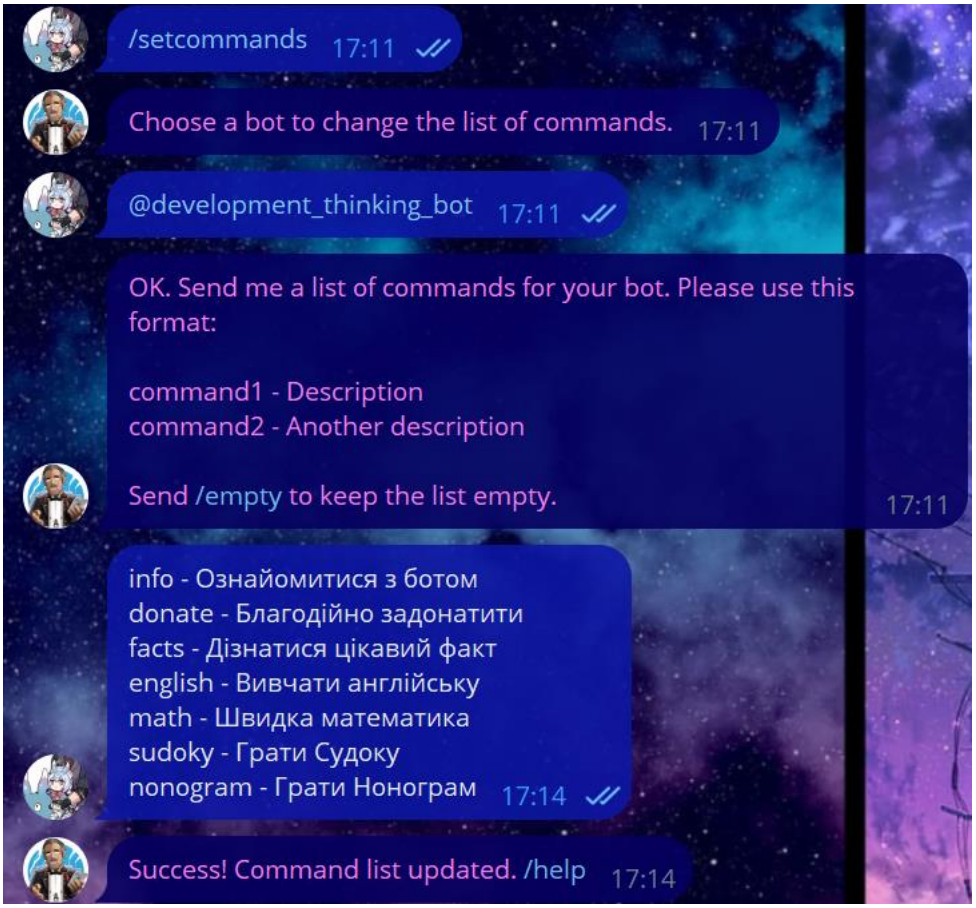


Рисунок 3.4 – Створення меню текстових команд

Створення кнопкового меню було виконано кодом. Використовувалося мова програмування Python, середовище розробки – PyCharm Community Edition. Для керування ботом на платформі телеграм – використовується бібліотека Telebot.

Спершу було вказано токен телеграм-бота:

```
import telebot
# Вказуємо токен нашого бота
bot = telebot.TeleBot('7109174421:AAGtGv3NzcIRBd0vmYbsy6lbycnVVr_UfYI')
```

Надалі створено обробник команди /start, яка виконується для початку роботи користувача з телеграм-ботом. Передбачається виведення текстового повідомлення та першого кнопкового меню. Використовуючи методи бібліотеки telebot, було створено кнопкове меню за допомогою клавіатури ReplyKeyboardMarkup. Кожна кнопка створюється за допомогою класу KeyboardButton і додається до клавіатури за допомогою методу add та row:

```
@bot.message_handler(commands=['Start','Старт','старт','noadmin'])
def handle_start(message):
    markup = types.ReplyKeyboardMarkup()  # Визначаємо змінну markup тут
    btn1 = types.KeyboardButton('Ознайомитися з ботом')
    btn2 = types.KeyboardButton('Почати розвивати мислення')
    btn3 = types.KeyboardButton('Благодійно задонатити')
    markup.add(btn1)
    markup.row(btn2)
    markup.row(btn3)
    bot.send_message(message.chat.id, f'Яким буде твій наступний крок?', reply_markup=markup)
```

Подібним чином було створено друге кнопкове меню, яке повинне замінювати перше, після натискання кнопки «Почати розвивати мислення» і містити шість інших кнопок: дізнатися цікавий факт, вивчати англійську, швидка математика, грати судоку, грати нонограм, в попереднє меню. Вигляд меню текстових команд – на рисунку 3.5. Вигляд кнопкових меню на рисунку 3.6.

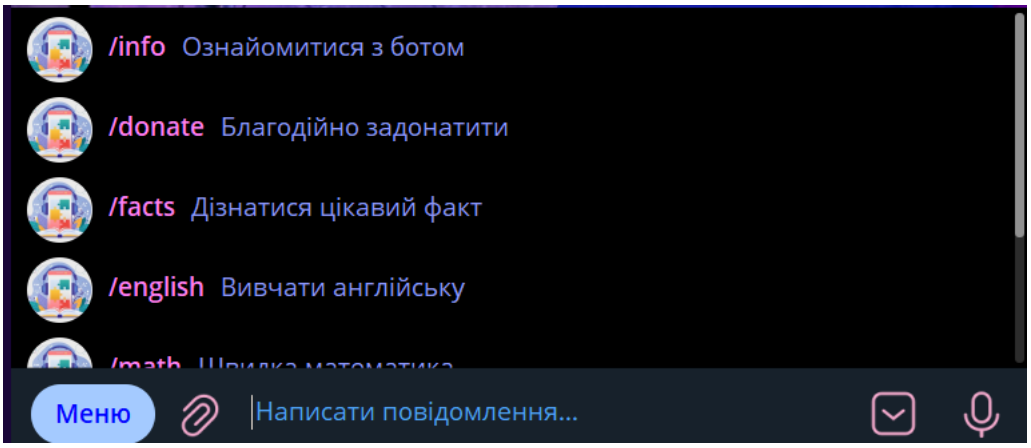


Рисунок 3.5 – Вигляд меню текстових команд

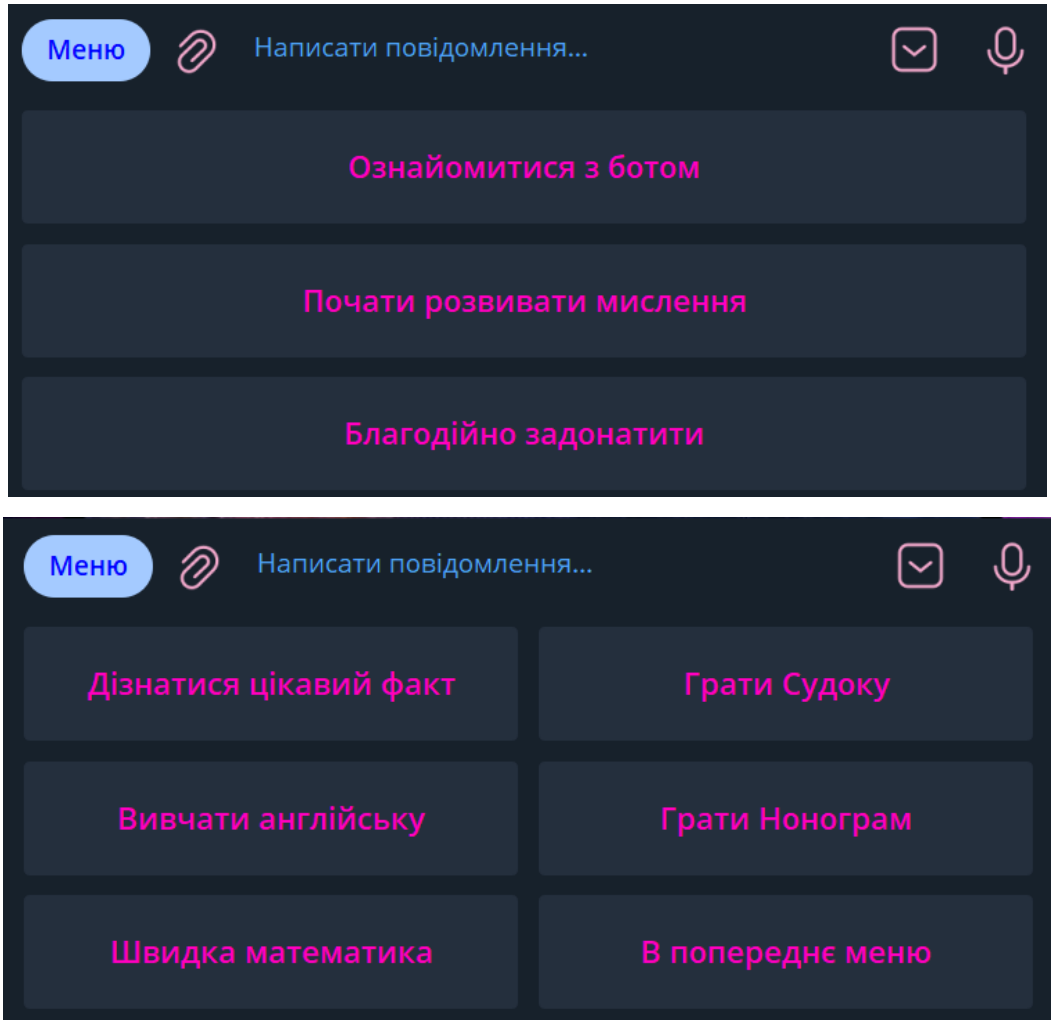


Рисунок 3.6 – Вигляд кнопкового меню

Основний функціонал телеграм-бота – це його модулі для розвитку мислення та адміністративна панель. Тому цей етап виконувався у декілька підетапів:

– розробка модулю вивчення англійських слів;

- розробка модулю надсилання цікавих фактів;
- розробка модулю математики;
- розробка модулю Судоку;
- розробка модулю Нонограм;
- розробка адміністративної панелі.

Кожен модуль запускається після надсилання від користувача відповідної команди чи текстового повідомлення з меню (текстове повідомлення автоматично надсилається після натискання на кнопку і відповідає підпису в кнопці).

Для запуску кожного модуля створена окрема функція. Передбачено, що бот обробляє надіслане від користувача повідомлення і відповідно реагує: при відповідності повідомлення до одного з пунктів меню, запускає функцію.

Наприклад, функції для модулів цікавого факту та англійської:

```
def interesting_facts(chat_id, cursor):
    cursor.execute("SELECT fact FROM facts ORDER BY RAND() LIMIT 1")
    fact = cursor.fetchone()
    bot.send_message(chat_id, f"Факт: *{fact}*", parse_mode='Markdown')
def english_word(chat_id, cursor):
    cursor.execute("SELECT word, translate, photo FROM english ORDER BY RAND() LIMIT 1")
    word, translate, photo_url = cursor.fetchone()
    bot.send_message(chat_id, f"Нове слово: *{word}*\nПереклад: {translate}", parse_mode='Markdown')
    bot.send_photo(chat_id, photo_url)
```

Частина коду, в якій ці функції запускаються:

```
elif message.text == 'Дізнатися цікавий факт':
    interesting_facts(chat_id, conn.cursor())
elif message.text == 'Вивчати англійську':
    english_word(chat_id, conn.cursor())
```

Модулі "Цікаві факти" та "Вивчення англійської мови" для телеграм-бота були створені з використанням Python та бібліотеки для роботи з базами даних. Для

вибірки даних використовується реляційна база даних, де SQL-запити дозволяють отримувати випадкові записи.

Модуль "Цікаві факти" витягує випадковий факт з бази даних і надсилає його користувачу у вигляді повідомлення, використовуючи Телеграм API. Це додає елемент несподіванки та цікавості. Модуль "Вивчення англійської мови" працює аналогічно, витягуючи випадкове англійське слово разом з його перекладом та зображенням. Це повідомлення та фото також відправляються користувачу, роблячи процес навчання інтерактивним та візуально привабливим.

Модуль швидкої математики для телеграм-бота був створений з використанням Python і декількох спеціалізованих бібліотек для забезпечення інтерактивності та різноманітності питань. Основні бібліотеки включають random для генерації випадкових чисел та вибору питань, math для обчислення математичних функцій, таких як факторіали, тригонометричні функції та логарифми, а також ast для безпечного оцінювання математичних виразів, введених користувачами.

Процес тестування реалізовано у вигляді серії з 10 питань різної складності, які послідовно задаються користувачеві. Після кожного питання бот очікує відповіді від користувача та перевіряє її на правильність. Питання включають прості обчислення, складні обчислення, тригонометричні функції, логарифми, факторіали, похідні та формули скороченого множення. Кожне питання має визначену кількість балів, що дозволяє підраховувати загальний результат користувача. У випадку неправильних відповідей бот зберігає інформацію про помилки і надає її користувачу наприкінці тестування, що допомагає у подальшому вдосконаленні знань.

Гра подається у текстовому форматі, де кожне питання супроводжується умовою та, при необхідності, варіантами відповідей. На початку тестування – користувач отримує інструктаж. Це робить тестування інтуїтивно зрозумілим та зручним для користувача. Бот також підтримує можливість зупинити тестування в будь-який момент. Цей модуль не лише розвиває математичні навички, але й додає елемент інтерактивності та зацікавленості до навчання.

При створенні модуля судоку, одним з ключових аспектів було забезпечення збереження і відновлення прогресу гри користувачів. Для цього використано бібліотеку mysql.connector, яка дозволяє здійснювати підключення до MySQL бази даних, виконувати SQL-запити та маніпулювати даними.

Модуль реалізовано таким чином, щоб користувачі могли почати нову гру або продовжити збережену. Для початку нової гри користувач вибирає рівень, доступний на основі його попереднього прогресу. Дані про рівні зберігаються в таблиці sudoku_levels, а прогрес користувачів – у таблиці user_progress. Кожного разу, коли користувач розв'язує головоломку, його прогрес оновлюється у базі даних. Важливим моментом було створення інтуїтивного інтерфейсу, де гра подається у вигляді таблиці з емодзі, що полегшує сприйняття та взаємодію через телеграм-бота. Зокрема це реалізовано наступним чином:

```
def format_sudoku_board(board):
    emoji_digits = {
        '1': '1⃣', '2': '2⃣', '3': '3⃣',
        '4': '4⃣', '5': '5⃣', '6': '6⃣',
        '7': '7⃣', '8': '8⃣', '9': '9⃣',
        '0': ' □ '
    }
    formatted_board = ''
    for i in range(0, 81, 9):
        row = board[i:i+9]
        formatted_row = ''.join([emoji_digits[digit] for digit in row])
        formatted_board += formatted_row + '\n\n'
    return formatted_board
```

Користувач може взаємодіяти з грою за допомогою текстових команд, де він вводить координати і значення для заповнення судоку. Реалізація цієї взаємодії включає обробку введених даних та відповідну валідацію, що забезпечує коректність та відповідність правилам гри. У випадку правильного вводу, бот оновлює і відображає новий стан гри, а також зберігає прогрес у базі даних. Якщо

користувач правильно розв'язує всю головоломку, йому відкривається новий рівень, що стимулює подальший розвиток логічного мислення.

Модуль нонограм для телеграм-бота був створений з використанням Python та кількох спеціалізованих бібліотек. Для взаємодії з базою даних застосовувалась бібліотека mysql.connector, яка забезпечує виконання SQL-запитів для збереження та завантаження прогресу гри користувачів. Бібліотека Pillow (PIL) використовувалась для створення та обробки зображень нонограм, дозволяючи генерувати та відправляти користувачам візуальне представлення гри. Бібліотека telebot забезпечувала інтерактивність через створення та обробку інлайн-кнопок, дозволяючи користувачам взаємодіяти з ігровою дошкою.

Функція малювання поля для гри виглядає наступним чином:

```
def create_nonogram_image(board, row_conditions, col_conditions):
    size = len(board)
    cell_size = 30  # розмір кожної комірки
    margin = 50  # поле для умов
    img_size = size * cell_size + margin
    img = Image.new('RGB', (img_size, img_size), color='white')
    draw = ImageDraw.Draw(img)
    font = ImageFont.load_default()
    for y in range(size):
        for x in range(size):
            fill = 'black' if board[y][x] == 1 else 'white'
            draw.rectangle(
                [margin + x * cell_size, margin + y * cell_size, margin + (x + 1) * cell_size,
                 margin + (y + 1) * cell_size],
                fill=fill,
                outline='gray'
            )
    # малюємо числа для рядків
    for i, condition in enumerate(row_conditions):
```

```
        draw.text((5, margin + i * cell_size + 10), condition, fill='black', font=font)
    # малюємо числа для стовпців
    for i, condition in enumerate(col_conditions):
        draw.text((margin + i * cell_size + 10, 5), condition, fill='black', font=font)
    return img
```

Гра нонограм реалізована у вигляді зображення, де кожна клітинка може бути зафарбованою або порожньою. Умови зафарбування клітинок (для рядків та стовпців) відображаються на полях зображення. Гравець взаємодіє з грою через інлайн-кнопки, натискаючи на клітинки для зміни їхнього стану. Прогрес гри зберігається в базі даних, дозволяючи гравцям продовжувати гру з того місця, де вони зупинилися. Важливим аспектом є можливість автоматичного збереження та завантаження прогресу, що забезпечує безперервність ігрового процесу.

Створення адміністративної панелі включало в себе кілька ключових етапів. Починаючи з налаштування з'єднання з базою даних, використовуючи бібліотеку mysql.connector, та імпортування необхідних модулів, таких як re для роботи з регулярними виразами та telebot для взаємодії з Телеграм API. Створення функцій для керування користувачами включало в себе реалізацію операцій, таких як перегляд списку користувачів, видалення користувача, додавання нового рівня Нонограм або Судоку, додавання нових англійських слів та цікавих фактів. Кожна з цих функцій використовувала механізми обробки повідомлень з подальшою реакцією на них.

Графічний інтерфейс адміністративної панелі був реалізований за допомогою клавіатур Телеграм, які надають зручний спосіб вибору опцій. Кожна кнопка відповідала певній функції адміністратора, такі як перегляд користувачів, видалення, додавання нових рівнів гри, англійських слів чи цікавих фактів.

Вигляд адміністративної панелі зображено на рисунку 3.7.

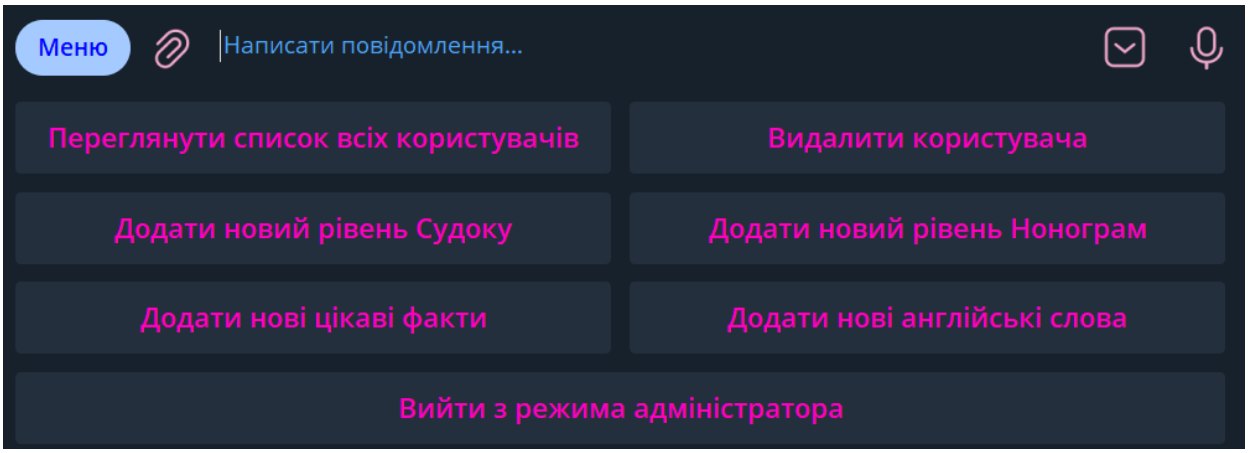


Рисунок 3.7 – Вигляд адміністративної панелі

Для забезпечення з'єднання з базою даних використовується бібліотека mysql.connector. Функція connect_to_database() відповідає за створення з'єднання з базою даних. У функції використовується блок try-except для обробки можливих помилок під час з'єднання. В разі успішного з'єднання вказані параметри, такі як ім'я хоста, користувач, пароль та назва бази даних, передаються в об'єкт з'єднання. Якщо підключення відбувається успішно, функція повертає об'єкт з'єднання для подальшого використання.

Функція з’єднання з базою даних:

```
def connect_to_database():
    try:
        conn = mysql.connector.connect(
            host='localhost',
            user='root',
            password='28072003',
            database='prokachaisya_bot'
        )
        return conn
    except mysql.connector.Error as e:
        print("Помилка підключення до бази даних:", e)
```

Даний етап реалізації бота – найважливіший, тому що він впливає на коректну роботу попередньо розробленого функціоналу, оскільки модулі головоломок, англійської та цікавих фактів, а також адміністративна панель – отримують дані з бази даних.

3.3 Вигляд реалізованого телеграм-боту

Виконавши реалізацію, було отримано повністю коректно функціонуючий бот. При запуску бота новим користувачем, введенням команди /start – користувач отримує привітання та бачить перше меню(рисунок 3.8).

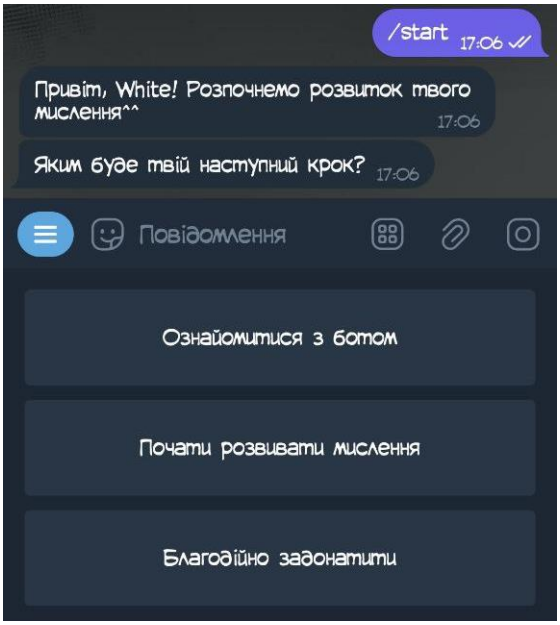


Рисунок 3.8 – Початок роботи з ботом

Після натискання на кнопку «Ознайомитися з ботом» користувач отримує повідомлення про призначення бота (рис. 3.9). Після натискання на «Благодійно задонатити» – повідомлення з подякою та номером карти (рис. 3.10).

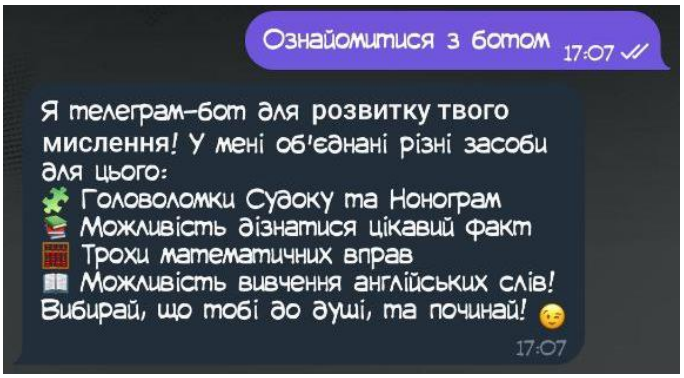


Рисунок 3.9 – Реакція бота на натискання кнопки «Ознайомитися з ботом»

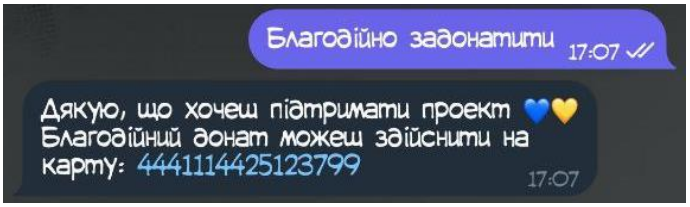


Рисунок 3.10 – Реакція бота на натискання кнопки «Благодійно задонатити»

Після натискання на кнопку «Почати розвивати мислення» користувачу надсилається текстове повідомлення, і відбувається зміна меню: п’ять кнопок для запуску засобів розвитку мислення та кнопка повернення до попереднього.

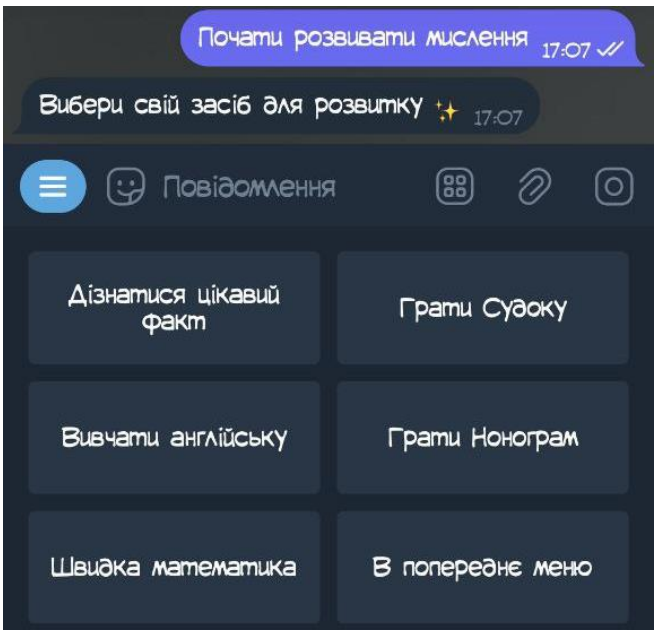


Рисунок 3.11 – Реакція бота на натискання кнопки «Почати розвивати мислення». Меню засобів розвитку мислення

Після натиснення на кнопку «Дізнатися цікавий факт» запускається відповідна функція, і користувачу виводиться повідомлення з рандомним цікавим фактом, який міститься в базі даних.

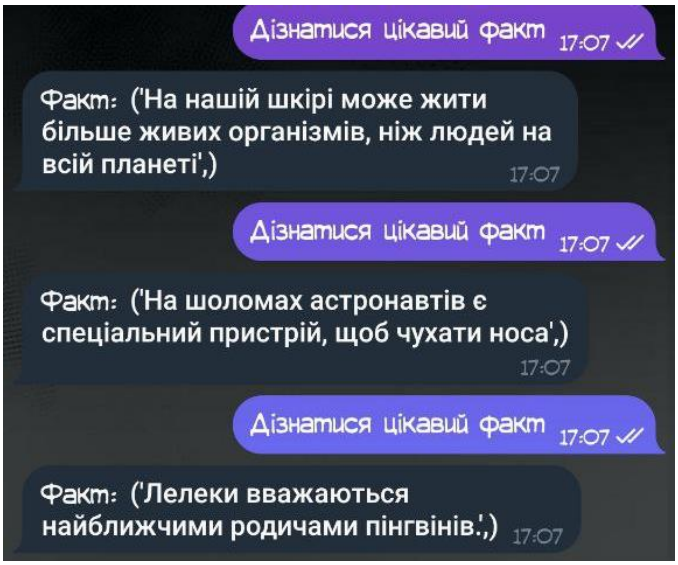


Рисунок 3.12 – Реакція бота на натискання кнопки «Дізнатися цікавий факт»

Коли користувач натискає «Вивчати англійську», бот запускає відповідну функцію і з бази даних зчитується рандомний запис, який потім виводиться користувачу у вигляді: англійське слово – його український переклад – відповідна картинка, що ілюструє це слово.

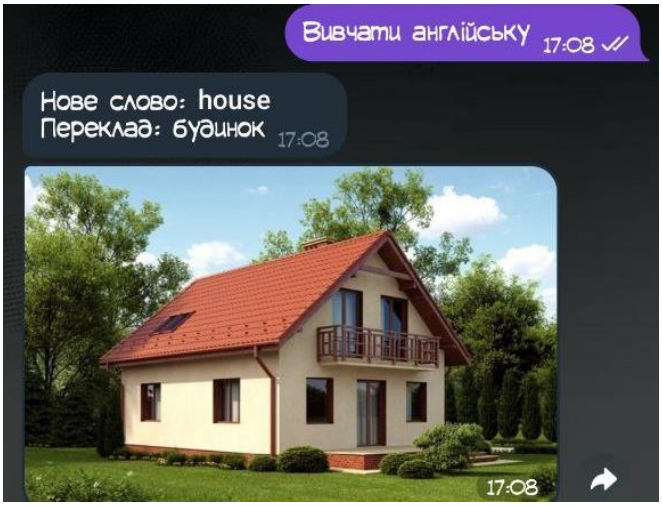


Рисунок 3.13 – Реакція бота на натискання кнопки «Вивчати англійську»

Натискаючи кнопку «Швидка математика», користувач активує відповідний модуль: запускається функція для тесту математичних знань користувача. Спочатку виводиться повідомлення про правила. Потім бот надсилає перше питання і очікує відповідь користувача. В кінці тестування бот вказує відсоток успішності проходження тесту(питання оцінюються в різну кількість балів, в залежності від складності), а також, якщо були помилки, то бот на них вказує. Також попереднє меню – замінене на кнопку «Закінчити і вийти».

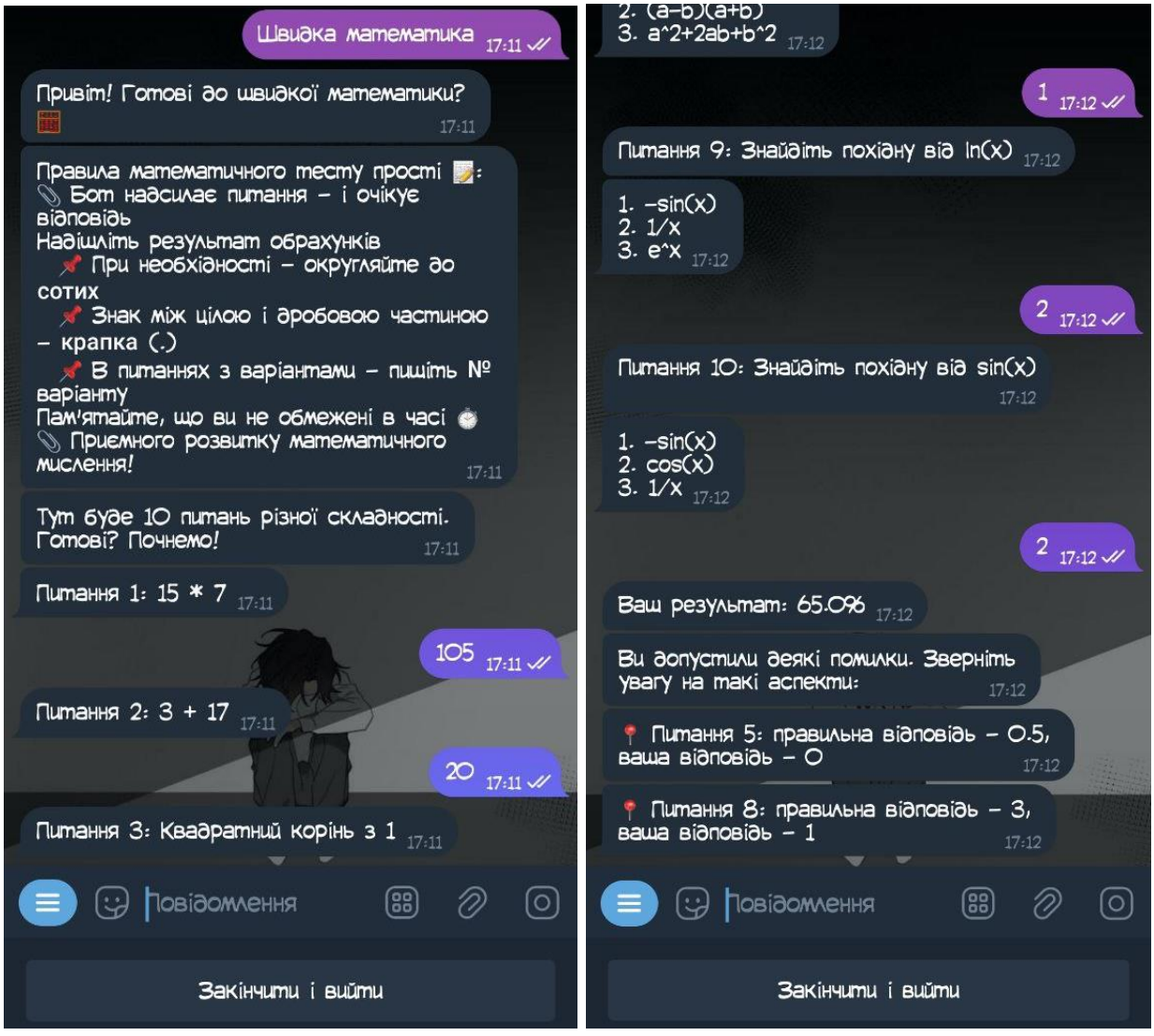


Рисунок 3.14 – Реакція бота на натискання кнопки «Швидка математика»

Після натискання на кнопку «Грати Судоку» користувачу виводиться повідомлення «Оберіть гру Судоку», а попереднє меню замінюється на меню

вибору (рисунок 3.15). Якщо користувач обирає «Продовжити останню гру» – то перевіряється таблиця user_progress на дані про незакінчену гру користувача. Якщо вона дійсно є – то користувачу надсилається поле, яке він вирішував раніше, інакше – надсилається повідомлення про те, що збереженої гри немає (рисунок 3.16).

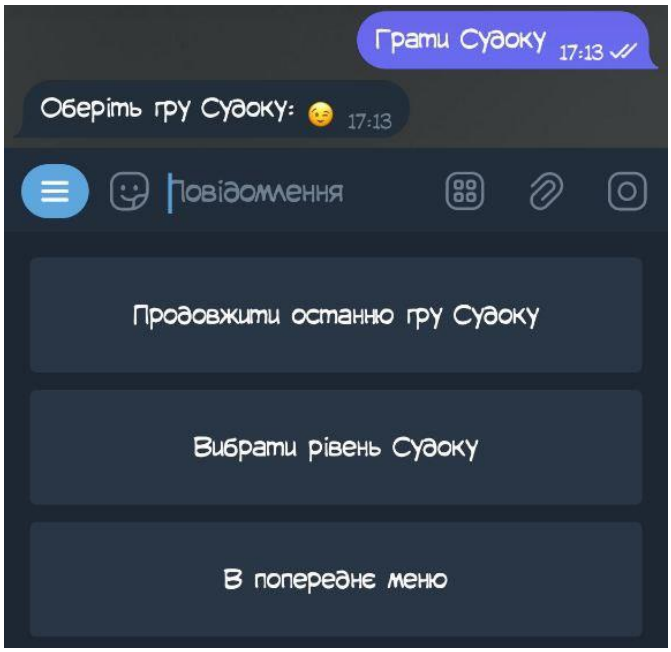


Рисунок 3.15 – Реакція бота на натискання кнопки «Грати Судоку»

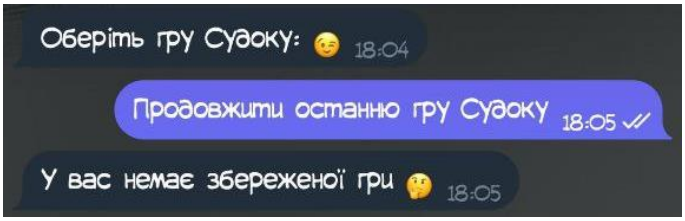


Рисунок 3.16 – Реакція бота, коли збереженої гри Судоку немає

Якщо користувач обирає «Вибрати рівень Судоку», то бот інформує користувача про максимально доступний для нього рівень. Якщо користувач вводить доступний номер рівня, то йому надсилається поле для гри саме того рівня, але якщо користувач хоче почати недоступний рівень, то бот вказує на це та запускає останній відкритий. Також попередні кнопки замінюються кнопкою «Зберегти прогрес і вийти». Цей функціонал показано на рисунку 3.17.

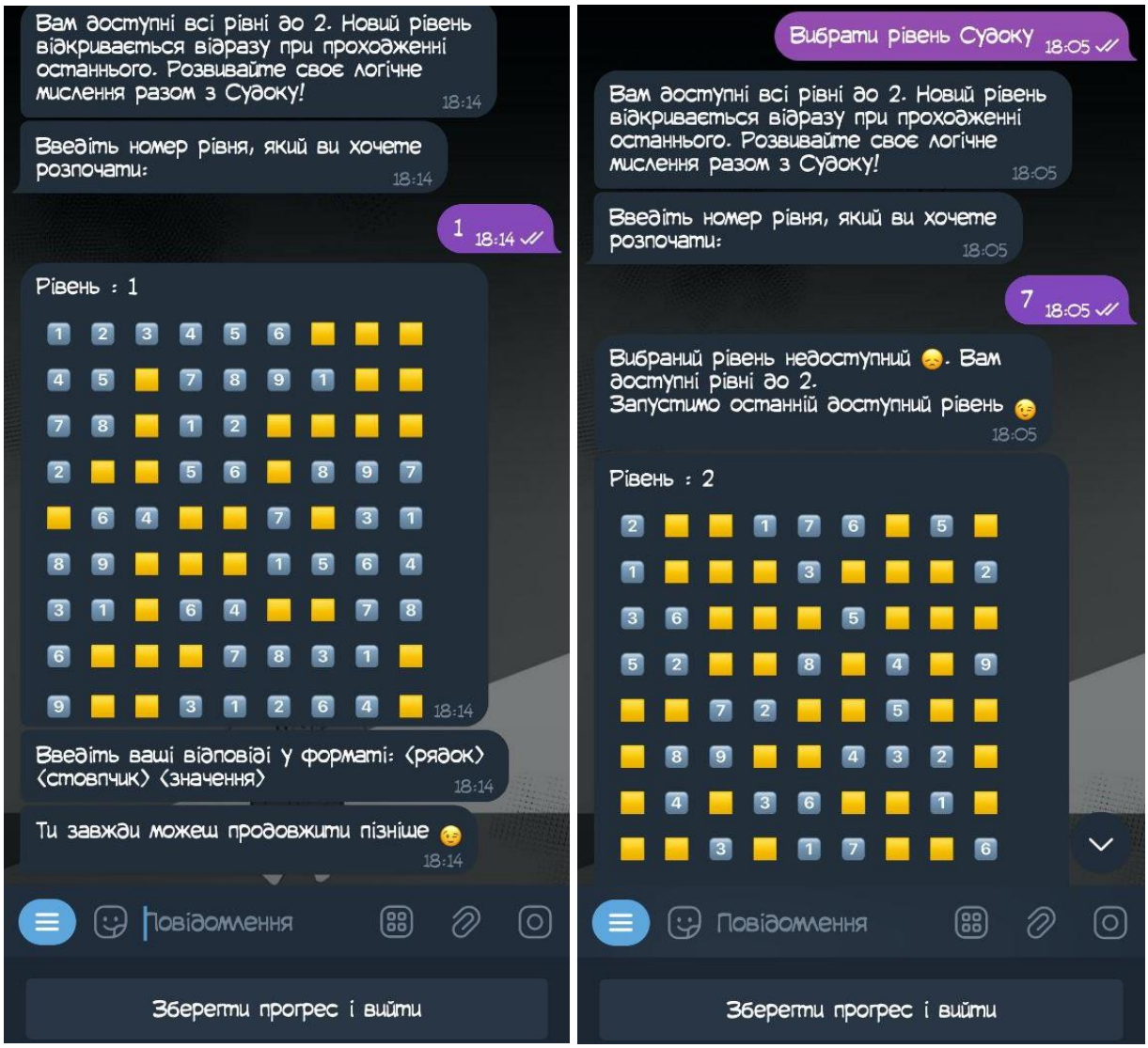


Рисунок 3.17 – Вибір рівня Судоку користувачем

Сам процес гри Судоку (рисунок 3.18) відбувається наступним чином: бот надсилає користувачу поле для гри та просить ввести координати клітинки та значення, яке там повинно бути у форматі <рядок> <стовпчик> <значення>. Після отримання відповіді, перевіряється чи це значення повинне там стояти. Якщо так, то користувачу надсилається нове зображення поля, вже з заповненою вказаною коміркою. Інакше користувачу надсилається повідомлення про помилку. Коли все поле заповнено, користувачу надсилається повідомлення-вітання з картинкою. Якщо ще є нові невідкриті рівні, то значення максимально доступного рівня Судоку користувача збільшується, інакше бот повідомляє, що усі рівні, що є в системі були вирішені і скоро будуть додані нові.

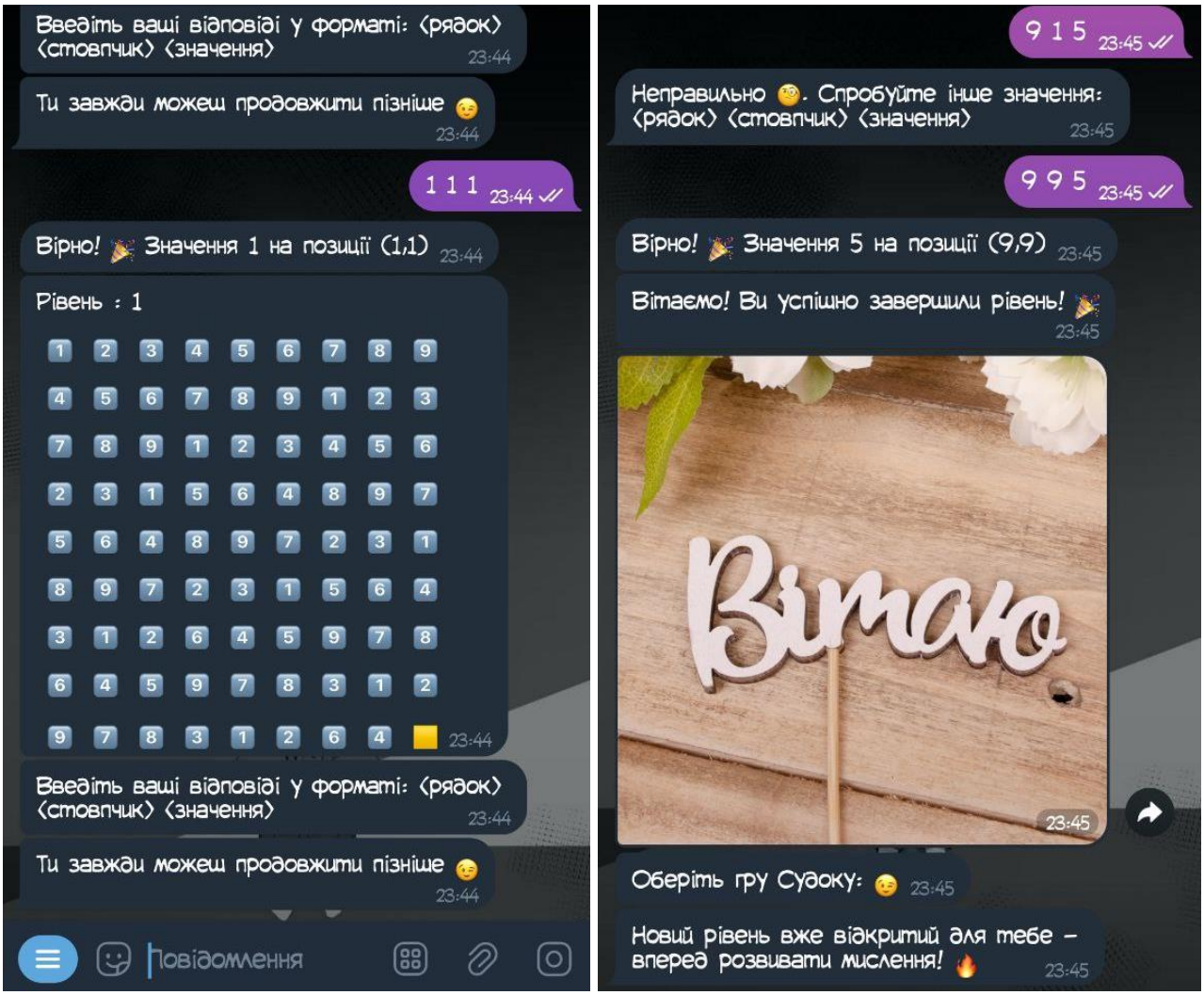


Рисунок 3.18 – Процес гри Судоку

Після натискання на кнопку «Грати Нонограм» користувачу виводиться повідомлення «Оберіть гру Нонограм», а попереднє меню замінюється на меню вибору (рисунок 3.19). Якщо користувач обирає «Продовжити останню гру», то перевіряється таблиця user_progress на дані про незакінчену гру користувача. Якщо вона дійсно є – то користувачу надсилається поле, яке він вирішував раніше, інакше – надсилається гра першого рівня (рисунок 3.20).

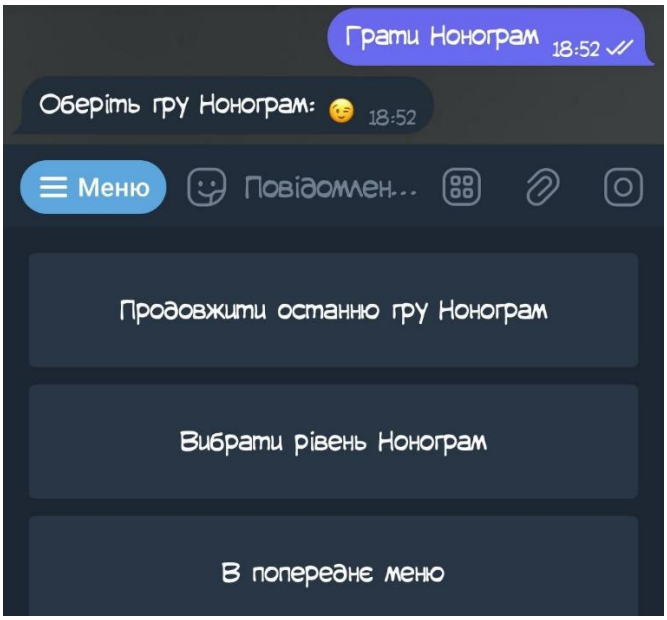


Рисунок 3.19 – Меню для гри Нонограм

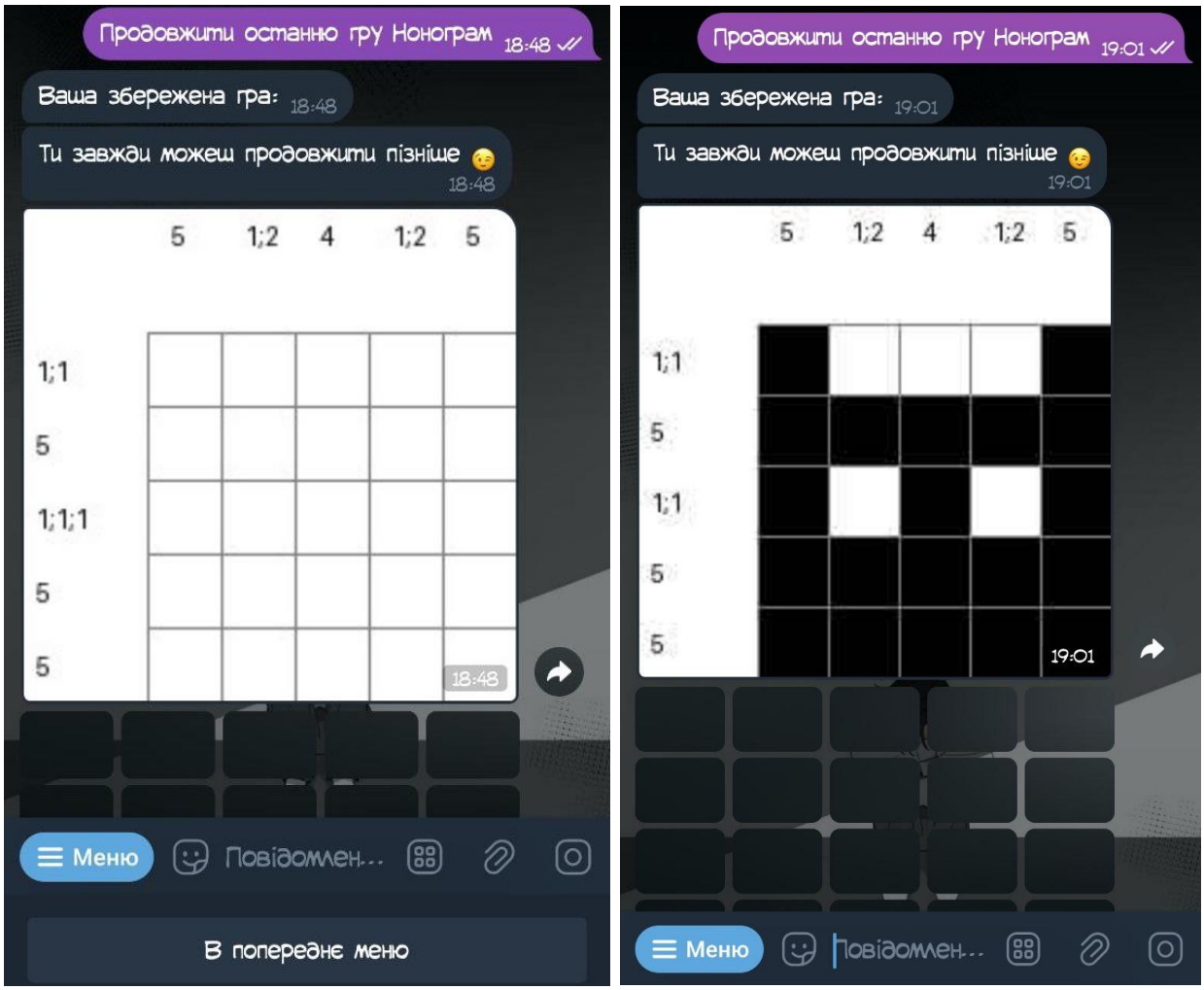


Рисунок 3.20 – Продовження гри Нонограм у боті

Поле для гри Нонограм виглядає як зображення з білими клітинками. Зверху і зліва від клітинок знаходяться підписи: скільки клітинок у кожному рядку та стовпцю, відповідно, має бути зафарбовано. Під полем знаходяться кнопки, при натисканні на які – на полі зафарбовується відповідна клітинка. Поле для гри зображено на рисунку 3.21.

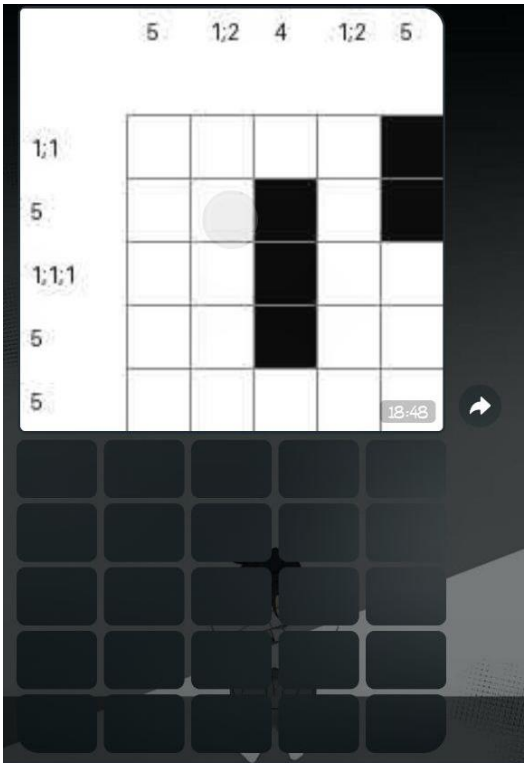


Рисунок 3.21 – Поле для гри Нонограм

Коли користувач вибирає «Вибрати рівень Нонограм», то бот виводить повідомлення про максимально доступний рівень та просить користувача ввести бажаний рівень. Якщо введене значення відповідає діапазону відкритих рівнів, то саме поле для гри цього рівня завантажується, інакше завантажується останній доступний рівень. Після завершення рівня, користувачу виводиться повідомлення-вітання про відкриття нового рівня, або про проходження всіх доступних рівнів, або про те, що тренування ніколи не буває зайвим(якщо це був не останній відкритий рівень). Кнопка «В попереднє меню» – повертає в стартове меню. Це показано на рисунку 3.22.

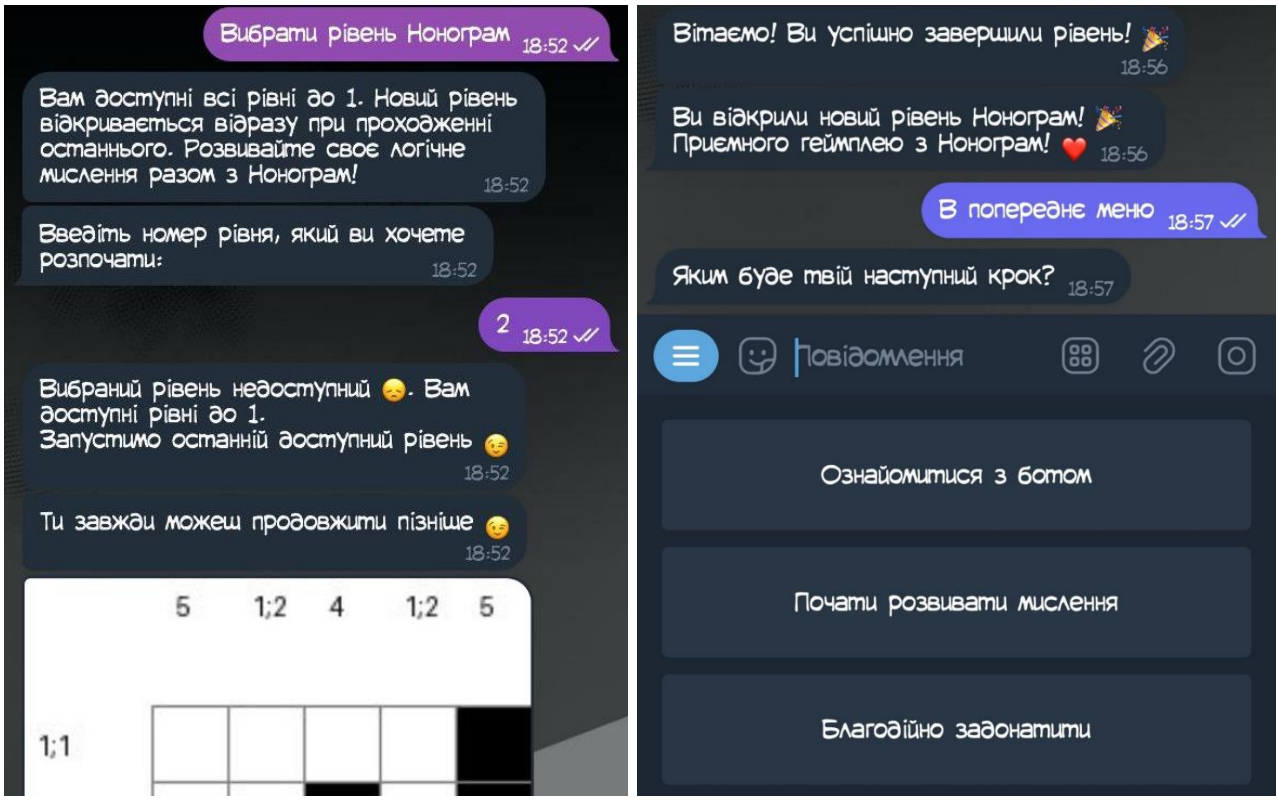


Рисунок 3.22 – Вибір рівня Нонограм. Повідомлення про завершення рівня

Доступ до адміністративної панелі – відкривається лише після введення команди /admin (ця команда не перелічена у жодному з меню, і тому звичайний користувач не зможе її випадково запустити). Бот перевіряє, чи має користувач статус «admin» у базі даних, потім відправляє користувачу відповідне повідомлення. Якщо користувач дійсно є адміністратором, то попереднє меню замінюється на меню адміністративної панелі.

Це показано на рисунках 3.23 та 3.24.

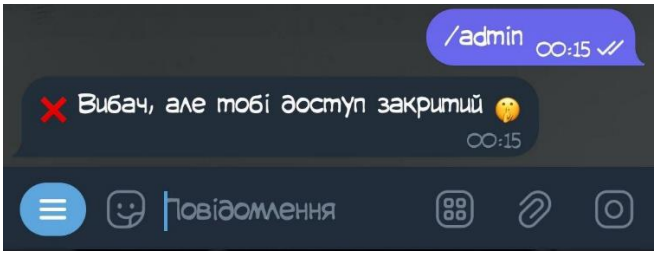


Рисунок 3.23 – Реакція бота, коли користувач без доступу намагається відкрити адміністративну панель

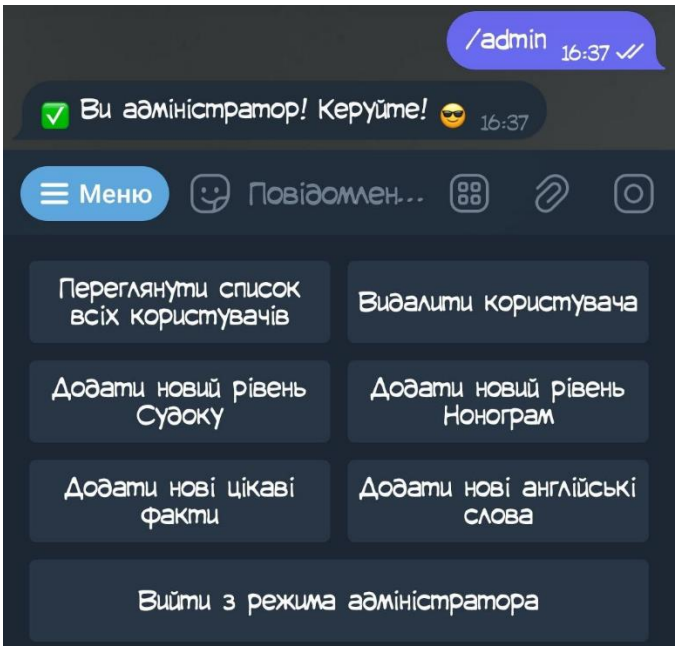


Рисунок 3.24 – Відкриття адміністративної панелі після введення команди для користувача з статусом адміністратора

Додавання нових даних з адімністративної панелі у базу даних відбувається у форматі питання – відповідь. Бот просить вказати необхідну інформацію, перевіряє її на коректність значення і відправляє відповідне повідомлення для користувача.

Оскільки цікавий факт – це текст, то особливих перевірок не передбачено. Приклад додавання нового факту на рисунку 3.25.

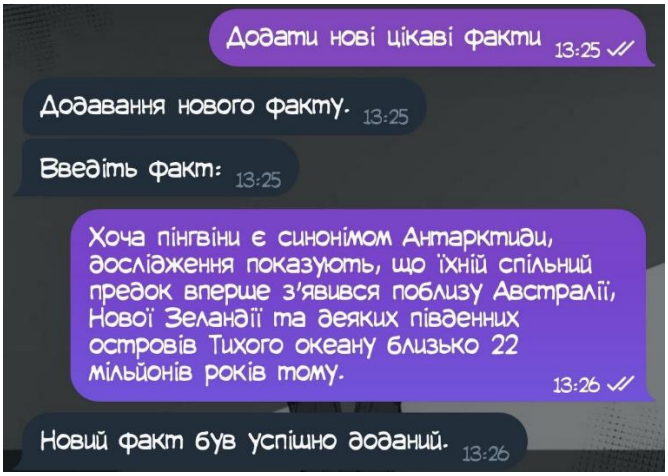


Рисунок 3.25 – Додавання нового цікавого факту до бази даних

Додавання нових англійських слів відбувається за тим же алгоритмом, проте додається перевірка на те, що слово повинне містити лише букви. При введенні посилання на зображення – перевіряється правильність введення на початку (http:// або https://) та в кінці(одне з значень: .jpg, .jpeg, .png, .gif).

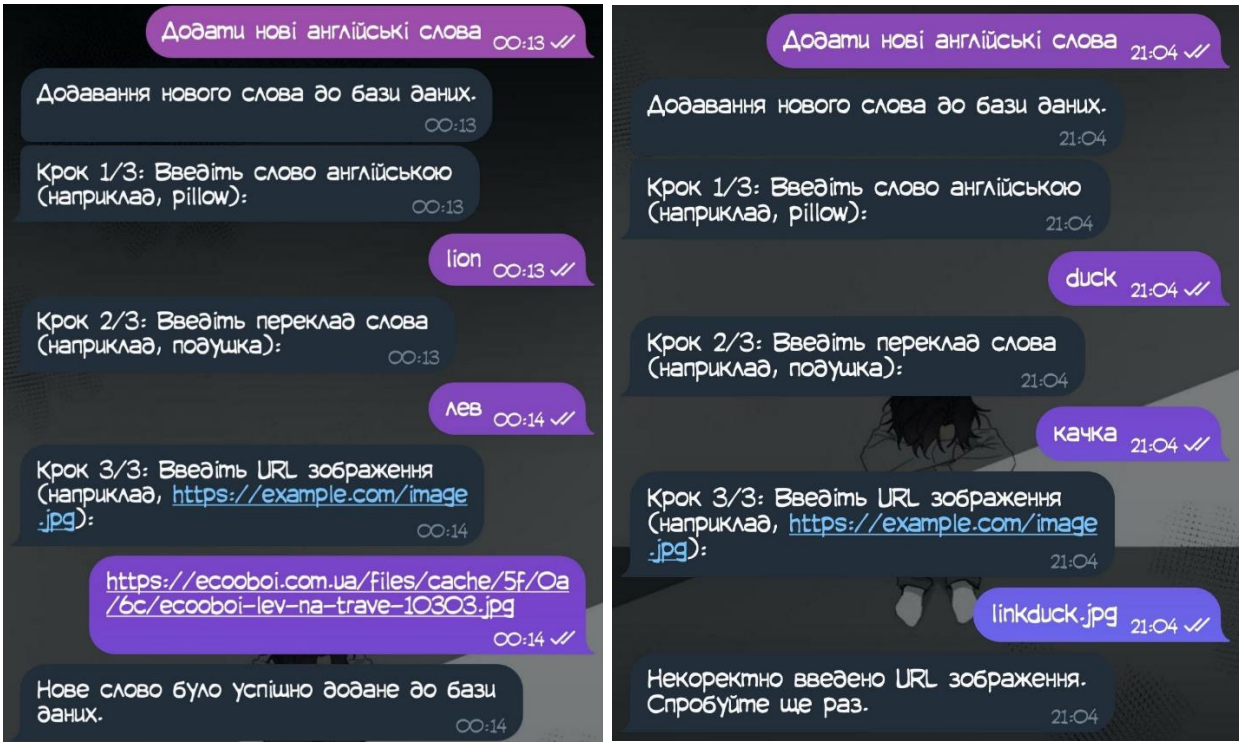


Рисунок 3.26 – Додавання нового англійського слова в базу даних

Процес додавання нового рівня Нонограм зображений на рисунку 3.27. Спочатку бот запитує рівень складності рівня: hard, medium або easy. Якщо користувач коректно ввів одне із значень, то надалі бот просить надіслати рішення поля: це послідовність з 0 та 1, де 0 – це біла клітинка, а 1 – чорна. Потім дані для заповнення рядків, а після – стовпчиків. Бот вказує в якому форматі мають бути надіслані дані, тому якщо адміністратор вводить їх інакше, ніж це передбачено – бот надсилає повідомлення про те, що новий рівень не буде доданий, і процес додавання завершується. Якщо всі дані були введені правильно – буде додано новий запис у базу даних у відповідну таблицю.

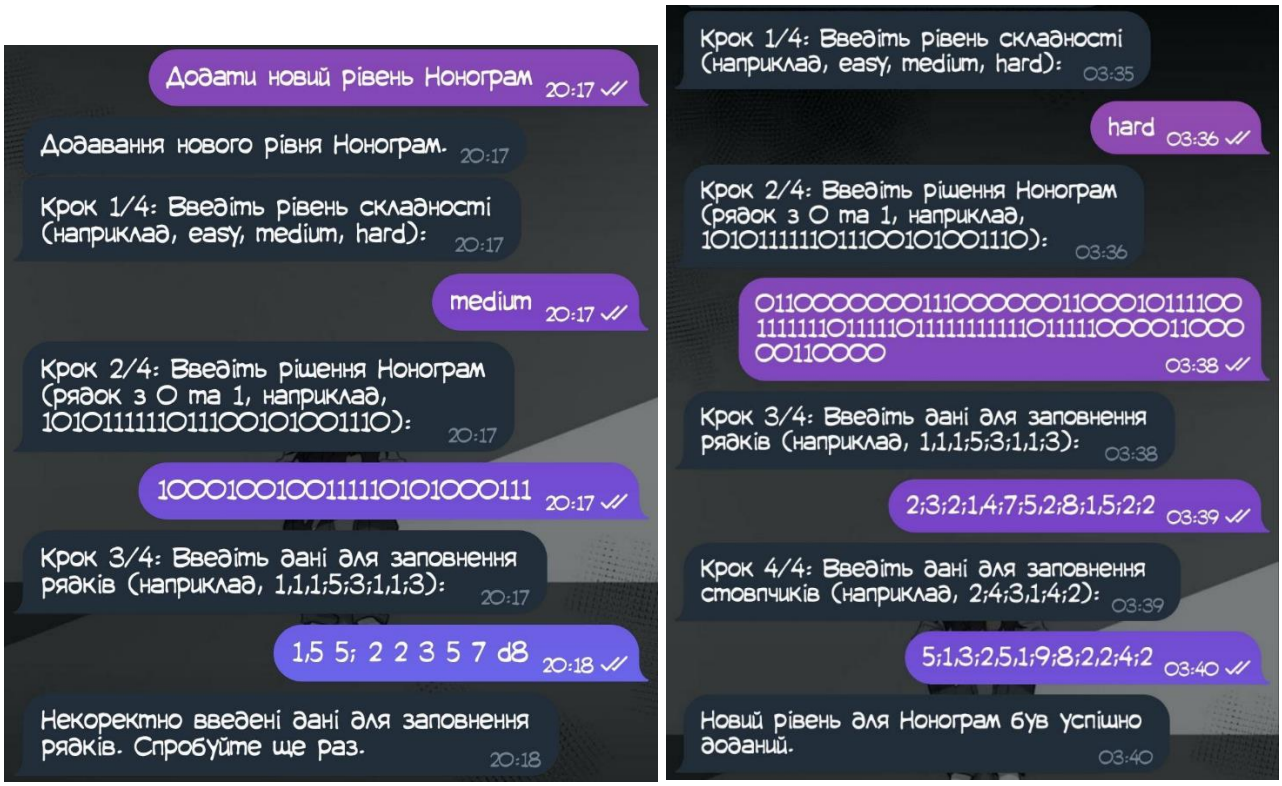


Рисунок 3.27 – Процес додавання нового рівня Нонограм

Процес додавання нового рівня для гри Судоку схожий до Нонограм. Але використовується інший формат даних для позначення вигляду початкового та заповненого полів для рівня. Заповнене поле (рішення Судоку) повинне складатися лише з чисел від 1 до 9 і бути довжиною 81 символ. Початковий стан поля (вигляд поля при запуску рівня) – повинен відповідати довжині рішення, і складатися з чисел від 0 до 9, де 0 – це пуста комірка, яку користувач має заповнити.

Якщо всі дані введені коректно – то новий запис буде доданий в базу даних, інакше бот надішле повідомлення про помилку введених значень і те, що рівень доданим не буде.

Це показано на рисунку 3.28.

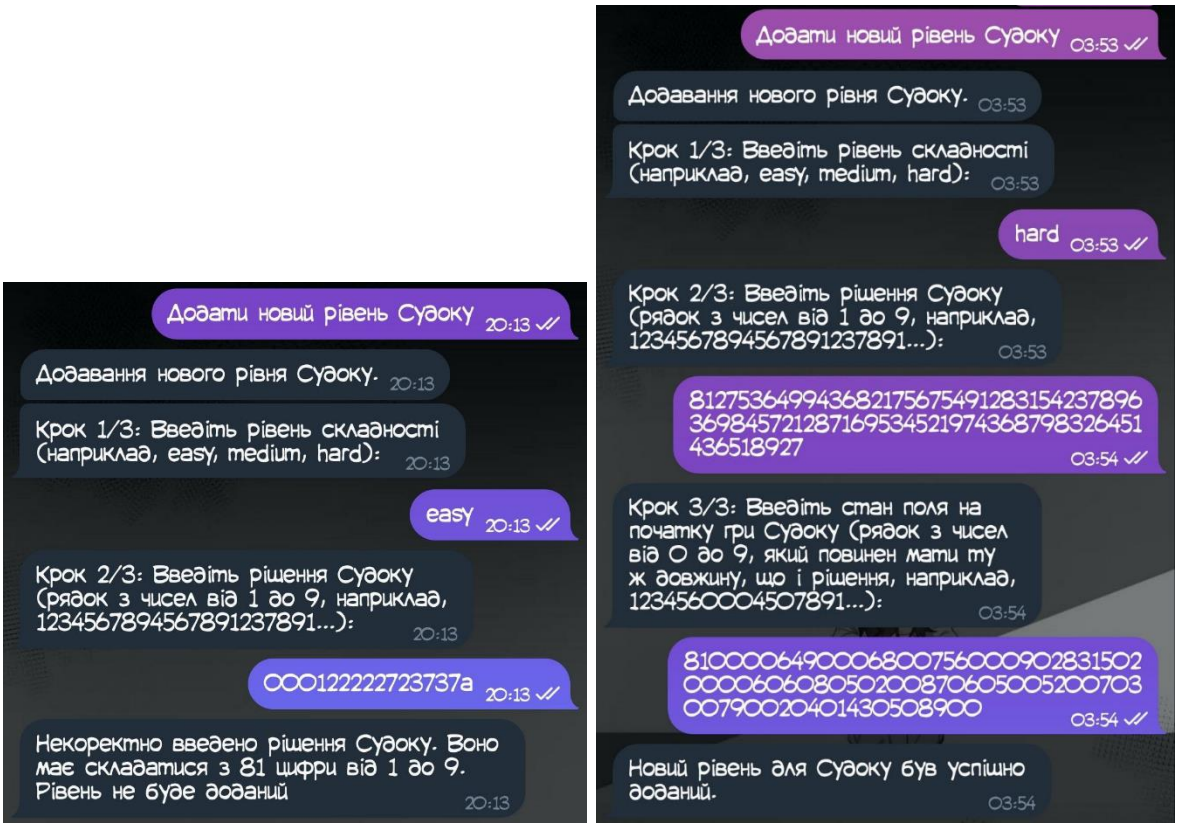


Рисунок 3.28 – Процес додавання нового рівня Судоку

Команда перегляду списку всіх користувачів – виводить маркованим списком телеграм-нікнейми всіх користувачів телеграм-бота з бази даних.

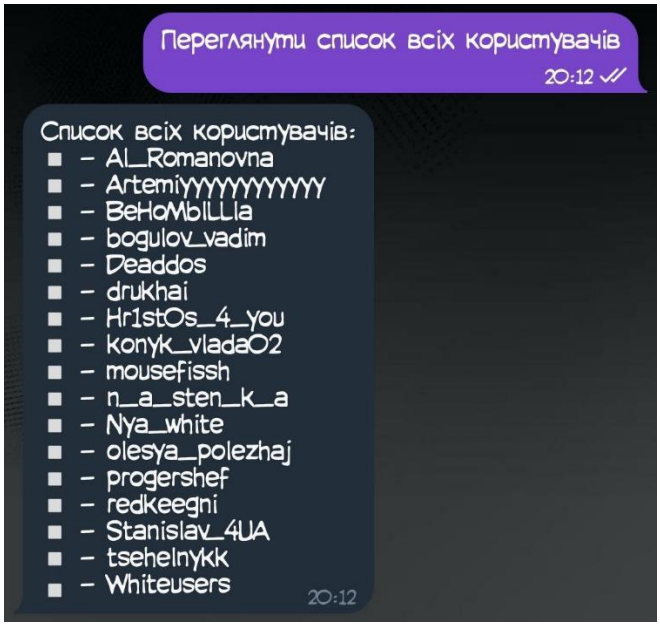


Рисунок 3.29 – Перегляд списку всіх користувачів бота з бази даних

Процес видалення користувача теж відбувається у формі діалогу з ботом. Бот просить надіслати нікнейм користувача. Далі він аналізує чи зареєстрований такий користувач у базі даних: якщо так – то видаляє спочатку пов’язаний з ним запис у таблиці user_progress (якщо такий запис взагалі існує), а потім видаляє і запис про самого користувача з таблиці users. Надалі бот надсилає в чат повідомлення, про успішне видалення заданого користувача. Якщо користувача з таким нікнеймом не зареєстровано, то бот сповіщає про це.

Процес видалення і список користувачів після успішної операції зображено на рисунку 3.30.

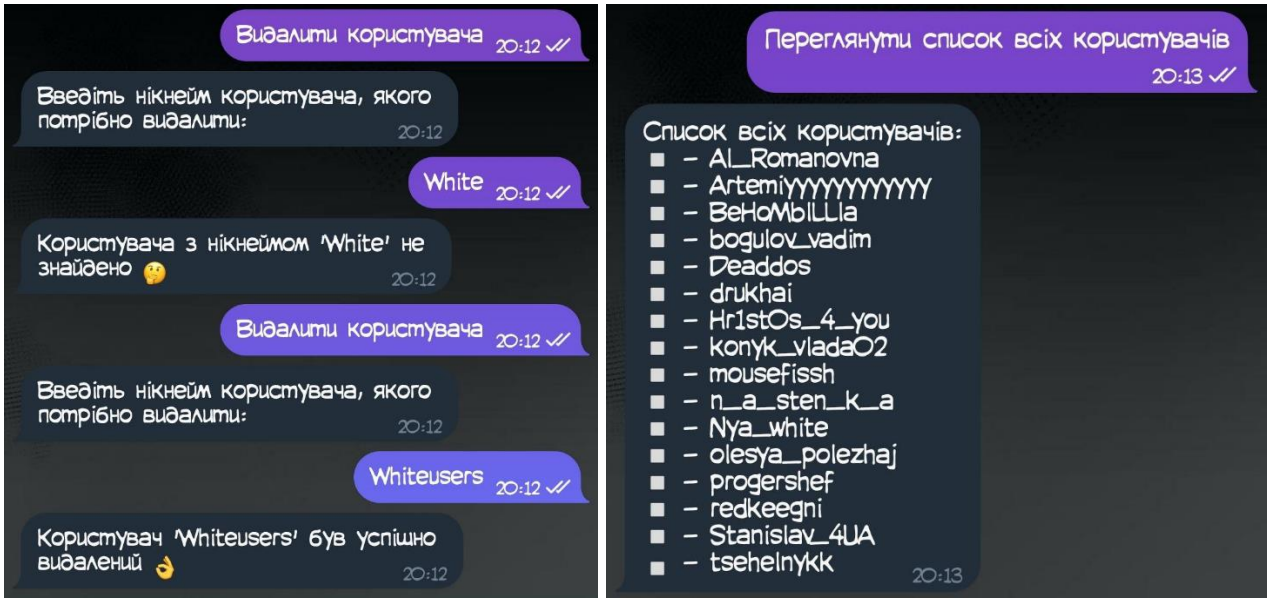


Рисунок 3.30 – Видалення користувача і перегляд списку після видалення

Остання кнопка – «Вийти з режима адміністратора» – змінює інтерфейс адміністративної панелі на інтерфейс користувача, а саме меню при початку роботи з ботом.

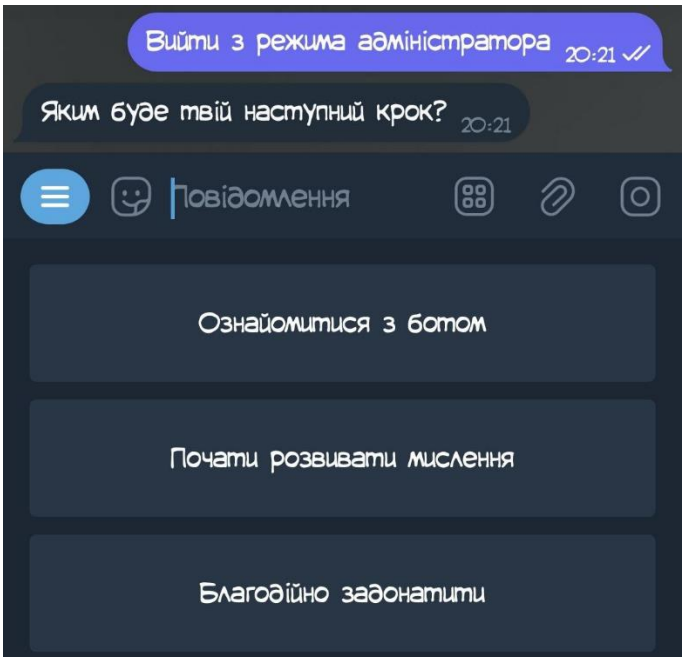


Рисунок 3.31 – Вихід з режиму адміністратора

4 ТЕСТУВАННЯ ТЕЛЕГРАМ-БОТУ

4.1 Вибір способу тестування боту

Тестування телеграм-боту є критичним етапом у процесі його розробки та впровадження. Бот повинен надійно функціонувати у різноманітних сценаріях, адекватно реагувати на команди користувачів, забезпечувати коректність обробки даних. Тому ретельне тестування боту перед його запуском у продуктивне середовище є необхідним для забезпечення високої якості та надійності кінцевого продукту.

Мануальне тестування - це процес перевірки програмного забезпечення, при якому тестувальник вручну виконує тестові випадки без використання автоматизованих інструментів.

Мета мануального тестування - виявлення дефектів, помилок і недоліків у функціональності, інтерфейсі або продуктивності програмного продукту.

Переваги мануального тестування:

− гнучкість і адаптивність: тестувальники можуть швидко адаптуватися до змін у вимогах або нових сценаріїв тестування, що робить мануальне тестування особливо корисним на ранніх етапах розробки;

− реальна оцінка користувацького досвіду: мануальне тестування дозволяє оцінити зручність і інтуїтивність користувацького інтерфейсу з погляду кінцевого користувача;

− виявлення візуальних і юзабіліті-помилок: тестувальник може помітити проблеми з дизайном, верстkою та зручністю використання, які автоматизовані тестові інструменти можуть пропустити.

Одже було вирішено обрати мануальне тестування для телеграм-боту через його гнучкість, можливість оцінки користувацького досвіду та ефективність для невеликих проектів. Воно дозволило швидко ідентифікувати та виправити помилки, забезпечивши високу якість кінцевого продукту.

4.2 Проведення тестування з використанням тест-кейсів

Вирішено створити тест-кейси – чітко визначені сценарії тестування, які описують конкретні кроки, очікувані результати та критерії успішності. Тест-кейси дозволяють систематично перевірити всі аспекти роботи бота, виявити можливі дефекти та переконатися у відповідності бота до вимог користувачів.

Нижче наведено таблицю з тест-кейсами для створеного телеграм-бота. У таблиці описані різні сценарії використання бота, включаючи взаємодію з користувачами, обробку команд та відповідність функціональних вимог.

Таблиця 4.1 – Тест-кейси

Тест-кейс	Опис	Кроки	Очікуваний результат	Результат
TC-01	Перевірка команди /start та з'єднання з базою даних	1. Ввести команду /start у чаті з ботом	Бот має відповісти привітальним повідомленням. Має з’явитися стартове кнопкове меню	Пройдено
TC-02	Перевірка обробки некоректної команди чи повідомлення	1. Ввести не передбачену команду /comand 2. Написати не передбачене повідомлення «hi»	Бот має відповісти повідомленням про помилку або надати інструкції щодо правильного використання команд	Пройдено
TC-03	Перевірка виведення інформації про бот	1. Ввести команду /info 2. Натиснути на кнопку «Ознайомитися з ботом»	Бот має надіслати одне і те ж повідомлення з описом про себе в обох випадках	Пройдено

Продовження табл. 4.1

Тест-кейс	Опис	Кроки	Очікуваний результат	Результат
ТС-04	Перевірка виведення інформації для донату	1. Ввести команду /donate 2. Натиснути на кнопку «Благодійно задонатити»	Бот має надіслати одне і те ж повідомлення з подякою та реквізитами для благодійного донату	Пройдено
ТС-05	Перевірка коректної взаємодії між меню бота	1. Перейти до меню засобів розвитку мислення через кнопку «Почати розвивати мислення» 2. Повернутися завдяки кнопці «В попереднє меню»	Замість кнопок стартового меню – мають з’явитися кнопки модулів розвитку мислення і навпаки	Пройдено
ТС-06	Перевірка роботи модуля надсилання цікавих фактів	1. Ввести команду /facts 2. Натиснути на кнопку «Дізнатися цікавий факт»	Бот повинен надіслати повідомлення з цікавим фактом	Пройдено
ТС-07	Перевірка роботи модуля вивчення англійських слів	1. Ввести команду /english 2. Натиснути на кнопку «Вивчати англійську»	Бот повинен надіслати повідомлення з англійським словом, його перекладом та картинку	Пройдено
ТС-08	Перевірка роботи модуля швидкої математики	1. Ввести команду /math 2. Натиснути на кнопку «Швидка математика»	Бот повинен надсилати повідомленнями математичні питання та змінити кнопкове меню. Нова кнопка повертає в попереднє меню. В кінці тесту бот повинен надіслати повідомлення з результатом та помилками	Пройдено

Продовження табл. 4.1

Тест-кейс	Опис	Кроки	Очікуваний результат	Результат
ТС-09	Перевірка роботи модуля Судоку	1. Ввести команду /sudoky 2. Натиснути на кнопку «Грати Судоку»	Бот повинен надсилати текстові повідомлення і змінювати кнопкове меню. Завантажувати різні рівні гри, перевіряти умову доступу рівнів. Завантажувати поле для гри у вигляді смайликів. Обробляти відповідь користувача і звіряти її з правильною: якщо так надсилати оновлене поле для гри. Надсилати різні повідомлення для закінчення гри в залежності від співвідношення завершений рівень\доступний рівень	Пройдено
ТС-10	Перевірка роботи модуля Нонограм	1. Ввести команду /nonogram 2. Натиснути на кнопку «Грати Нонограм»	Бот повинен надсилати текстові повідомлення і змінювати кнопкове меню. Завантажувати різні рівні гри, перевіряти умову доступу рівнів. Завантажувати поле для гри у вигляді картинки та кнопок. Комірки мають змінювати колір після натиснення на кнопки. Надсилати різні повідомлення для закінчення гри в залежності від співвідношення завершений рівень\доступний рівень	Пройдено

Продовження табл. 4.1

Тест-кейс	Опис	Кроки	Очікуваний результат	Результат
ТС-11	Перевірити закритість доступу меню адміністративної панелі	1. Ввести команду /admin з стороннього акаунту 2. Ввести команду /admin з акаунту розробника (зазначеного в БД як admin)	Бот перевіряє статус користувача, який надіслав повідомлення: для стороннього користувача надсилає повідомлення про закритість доступу, для адміністратора – заклик керувати та замінить кнопкове меню на меню адміністративної панелі	Пройдено
ТС-12	Перевірити закритість доступу функціоналу з меню адміністративної панелі	1. Ввести текстове повідомлення з функціоналу адміністративної панелі(наприклад «Видалити користувача» з стороннього акаунту та розробника	Для стороннього акаунту – бот надішле повідомлення, що відповідь на це повідомлення не передбачена. Для акаунту розробника – надішле повідомлення з проханням ввести нікнейм користувача для видалення.	Пройдено
ТС-13	Перевірити роботу адміністративної панелі	1. Послідовно запускати з меню адіміністративної панелі – запропонований функціонал	Бот повинен надсилати повідомлення в залежності від обраної функції, перевіряти коректність введених даних перед внесенням змін у базу даних. При некоректності – надсилати про це повідомлення та завершувати роботу запущеної функції. Повертатися в стартове меню після натиснення «Вийти з режима адміністратора»	Пройдено

ВИСНОВКИ

У ході виконання проекту було проведене дослідження необхідності розвитку мислення у суспільстві, виявлена проблема низького рівня розвитку мислення та визначені шляхи її подолання. Зокрема вирішено розробити багатофункціональний телеграм-бот, призначений для розвитку мислення користувачів месенджеру.

Проведений аналіз аналогів засобів для розвитку мислення дозволив виділити їх переваги та недоліки і, спираючись на це, сформувати вимоги до функціоналу телеграм-боту.

Надалі були чітко сформовані мета проекту та список задач для її досягнення. Також, визначено які засоби реалізації буде оптимальніше використати.

Важливий етап – моделювання телеграм-боту, під час якого був побудований ряд діаграм, щоб показати роботу системи телеграм-боту розвитку логічного мислення з різних сторін. Зокрема, діаграма IDEF0 демонструє основні етапи розробки бота, декомпозиція функціональної моделі 1-го та 2-го рівнів деталізує їх, діаграма варіантів використання показує різні сценарії взаємодії користувача, адміністратора та бази даних з ботом, а логічна модель бази даних відображає структуру і зв'язки між даними, що використовуються ботом.

Проведена розробка телеграм-бота включала створення самого бота, його меню через команди та кнопки, а також модулі функціоналу: цікавих фактів, англійських слів, швидкої математики, Судоку та Нонограм. Додатково була створена адміністративна панель, налаштовано підключення і робота з базою даних.

Завершальний етап проекту – тестування бота. Було проведено мануальне тестування з використанням тест-кейсів, що дозволило переконатися у правильності та ефективності роботи всіх функціональних модулів телеграм-бота.

Результатом проекту стало створення телеграм-бота, що виконує функції, спрямовані на розвиток логічного мислення користувачів. Проведені етапи

проекту, від дослідження та аналізу аналогів до моделювання, розробки та тестування бота, дозволили створити якісний продукт, що відповідає визначеним вимогам та цілям. Таким чином, проект досяг своїх цілей, надавши користувачам зручний інструмент для покращення своїх здібностей логічного мислення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Логічне мислення це | Український психологічний ХАБ | ПСИХОЛОГ. ПСИХОЛОГ. URL: https://www.psykholoh.com/post/логічне-мислення-це (дата звернення: 08.04.2024).

2. Why Critical Thinking Is Important (& How to Improve It). Be Brain Fit. URL: https://bebrainfit.com/critical-thinking/ (date of access: 08.04.2024).

3. 30 Interesting Telegram Statistics you need to check (2024) - Skillademia. Skillademia. URL: https://www.skillademia.com/statistics/telegram-statistics/ (date of access: 09.04.2024).

4. How Many People Use Telegram in 2024?. Backlinko. URL: https://backlinko.com/telegram-users (date of access: 09.04.2024).

5. Telegram Statistics In 2024 (Usage By Country & Financials). DemandSage. URL: https://www.demandsage.com/telegram-statistics/ (date of access: 10.04.2024).

6. Internews. Дослідження «Ставлення населення до медіа та споживання різних типів медіа у 2021 році». *detector.media*. URL: https://detector.media/infospace/article/193921/2021-11-18-doslidzhennya-stavlennya-naselennya-do-media-ta-spozhyvannya-riznykh-typiv-media-u-2021-rotsi/ (дата звернення: 10.04.2024).

7. Капраль О. Р., Велика М. Б. РОЛЬ ЧАТ-БОТІВ В ЕПОХ У ДІДЖИТАЛІЗАЦІЇ. Вісник Херсонського національного технічного університету. 2023. № 3(82). С. 53–58. URL: https://doi.org/10.35546/kntu2078-4481.2022.3.7 (дата звернення: 11.04.2024).

8. Кейси використання чат-ботів у бізнесі. Інтернет-видання Коло. Останні новини Полтави та Полтавщини. URL: https://kolo.news/category/biznes/36815 (дата звернення: 12.04.2024).

9. Крупа А. ТЕХНОЛОГІЯ ЧАТ-БОТ ЯК ЧИННИК КОМП'ЮТЕРНО-ПОСЕРЕДНИЦЬКОЇ КОМУНІКАЦІЇ ЦИФРОВОГО СУСПІЛЬСТВА. Humanities

Studies. 2022. № 12(89). С. 130–141. URL: https://doi.org/10.26661/hst-2022-12-89-15 (дата звернення: 12.04.2024).

10. Що таке чат-бот: секрети використання та основні переваги. Блог HelpCrunch. URL: https://helpcrunch.com/blog/uk/shcho-take-chat-bot/ (дата звернення: 13.04.2024).

11. Discover 6 Innovative Ways to Use Chatbots for Marketing. Learn Hub | G2. URL: https://learn.g2.com/chatbots-for-marketing (date of access: 13.04.2024).

12. Приклади використання чат-ботів у бізнесі URL: https://gerabot.com/article/prikladi_vikoristannya_chatbotiv_u_biznesi (дата звернення: 14.04.2024).

13. Sulistyanto I., Prellany N. The Effectiveness of Using Bot Telegram in Teaching Reading Narrative Text at the Tenth Grade of SMAN 1 Grogol Kediri. JARTIKA Jurnal Riset Teknologi dan Inovasi Pendidikan. 2020. Vol. 3, no. 2. P. 195–200. URL: https://doi.org/10.36765/jartika.v3i2.67 (date of access: 15.04.2024).

14. Як розвинути свій мозок і стати генієм: 4 ефективних способи. *Ukr.Media*. URL: https://ukr.media/psihologiya/386890/ (дата звернення: 16.04.2024).

15. Best Telegram Bot Examples to Get Inspired by in 2024. Email and Internet Marketing Blog. URL: https://sendpulse.com/blog/telegram-bot-examples (date of access: 17.04.2024).

16. Лекція 6. Нотація IDEF0. *Головна | Elib LNTU*. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/Кондіус%202%20готовва/page9.html (дата звернення: 26.04.2024).

17. Books P. W. Fortran Crash Course + Hacking + Android Crash Course + Python Crash Course. CreateSpace Independent Publishing Platform, 2017. 128 p.

18. Conceptual Modeling - ER 2006 / ed. by D. W. Embley, A. Olivé, S. Ram. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006. URL: https://doi.org/10.1007/11901181 (date of access: 30.04.2024).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку

«Телеграм-бот для розвитку логічного мислення»

2024

А.1 Призначення й мета телеграм-бота для розвитку логічного мислення

А.1.1 Призначення телеграм-боту

Телеграм-бот призначений для швидкого та зручного доступу користувачів месенджера Телеграм до різносторонніх засобів розвитку мислення.

А.1.2 Мета створення телеграм-боту

Мета даного проекту – розробка багатофункціонального телеграм-бота з привабливим оформленням та зручним інтерфейсом, призначеного для розвитку мислення користувачів месенджеру під час періодів неформального відпочинку, який буде вирізнятися своєю доступністю та різнонаправленністю у питанні способів розвитку та підтримки розумових здібностей.

А.1.3 Цільова аудиторія

Цільовою аудиторією даного проекту є Замовник та україномовне населення різних вікових категорій, які користуються додатком Телеграм та зацікавлені у розвитку та підтримці своїх розумових здібностей та мислення загалом.

А.2 Вимоги до проекту

А.2.1 Вимоги до структури й функціонування

Телеграм-бот для розвитку логічного мислення повинен бути реалізований за допомогою мови програмування Python на платформі Телеграм та відповідати визначеним вимогам:

- привабливе оформлення;
- зручний та зрозумілий інтерфейс;
- наявність навчальної частини по роботі з ботом;
- наявність повністю функціонуючих засобів розвитку мислення різного спрямування: виведення цікавих фактів, математичні тести, вивчення англійських слів, гра Нонограм та гра Судоку;
- наявність системи рівнів для зацікавлення роботи з ботом;
- наявність системи добровільного донату для розвитку проекту.

Кінцевий продукт даного проекту має бути представленим телеграм-ботом, який містить якісне інформаційне та графічне наповнення з повністю правильно реалізованим функціоналом.

А.2.2 Вимоги до персоналу

Адміністратор боту не повинен мати особливих технічних навичок для роботи з телеграм-ботом і його підтримки. Єдиними вимогами є наявність навичок користування персональним комп’ютером та месенджером Телеграм.

А.2.3 Вимоги до збереження інформації

Уся інформація про запити надходить спочатку на сервер Розробника, а далі над ними виконується обробка та шифрування сервером Телеграм. Кінцевою дією – є зворотній зв’язок між ботом та користувачем.

Уся інформація про користувача та його успіхи розвитку повинна зберігатися у базі даних реалізованій засобами системи управління базами даних MySQL.

А.2.2 Вимоги до розмежування доступу

Розроблюваний телеграм-бот має бути загальнодоступним у мережі Інтернет. Права доступу до інформації розмежовані за групами користувачів: адміністратор бота та користувач. Адміністратор має необмежений доступ до даних з правами перегляду, додавання, редагування та видалення. Доступ до адміністративної панелі надається окремим користувачам, які зазначені у базі даних, як адміністратори.

Користувач може користуватися всім функціоналом бота, але не впливати на існуючі дані. Він може почати роботу з ботом, переглянути навчальну інформацію по роботі з ним, спробувати різні засоби розвитку мислення, які представлені у боті: ігри Судоку та Нонограм, перегляд цікавих фактів, проходження математичних тестів, вивчення нових англійських слів. Завдяки системі рівнів, доступ до всіх рівнів відразу закритий – але поступово відкривається з проходженням доступних.

А.3 Структура телеграм-боту

А.3.1 Загальна інформація про структуру телеграм-боту

До структури телеграм-боту входять клієнтський інтерфейс, бот-сервер, web-сервер, а також усі модулі, які відповідають за функціонал роботи засобів розвитку мислення. Детальніше:

- клієнтський інтерфейс: Телеграм-клієнт – користувачі взаємодіють з ботом через інтерфейс месенджера Телеграм. Вони можуть надсилати текстові повідомлення та натискати на кнопки;
- бот-сервер: API Телеграм – бот взаємодіє з API Телеграм для отримання та відправки повідомлень та зображень;
- web-сервер: бот може мати вбудований web-сервер, який взаємодіє з API Телеграм та обробляє отримані повідомлення та дані;
- модуль надсилання цікавих фактів: при виборі користувачем пункту «Дізнатися цікавий факт» надсилає текстове повідомлення з рандомним цікавим фактом із наявних у базі даних;
- модуль вивчення англійських слів: при виборі користувачем пункту «Вивчати англійську» надсилає текстове повідомлення або зображення з словами англійської мови рівня А1 та їх перекладом українською, з наявних у базі даних;
- модуль математики: при виборі користувачем пункту «Швидка математика» запускає тест з простими математичними обрахунками у режимі телеграм-повідомлень «Питання-відповідь»;
- модуль гри Судоку: при виборі користувачем пункту «Грати Судоку» питає в користувача який саме рівень з доступних він бажає вирішити, і після уточнення – запускає потрібний рівень гри;
- модуль гри Нонограм: при виборі користувачем пункту «Грати Нонограм» питає в користувача який саме рівень з доступних він бажає вирішити, і після уточнення – запускає потрібний рівень гри.

А.3.2 Навігаційне меню

Для зручної навігації повинно бути створене меню роботи з ботом через текстові вказівки або натискання відповідних кнопок, яке забезпечить доступ до всього функціоналу бота.

А.3.3 Управління контентом

Управління контентом має здійснюватися за допомогою адміністративної панелі. Усе необхідне інформаційне наповнення має міститися у базі даних. Графічні матеріали та інформацію для наповнення знаходить Розробник.

А.3.4 Дизайн телеграм-боту

Дизайн телеграм-боту має бути виконаний в сучасному, мінімалістичному та акуратному стилі. Здебільшого він буде виглядати як базовий чат месенджера, але стилістика текстових повідомлень від бота – має відрізнятися ввічливим ставленням та з застосуванням різного роду смайликів, в залежності від ситуації. Це створюватиме ефект оригінальності, живності та допоможе виділити основну інформацію в тексті, або візуально відокремити блоки інформації від бота.

Також буде створено дві варіації меню: через вибір текстової команди та через натискання кнопок. Текстове меню повинно мати весь перелік можливих команд, доступних користувачеві. Додані кнопки для застосування запуску команд з меню, повинні мати зручне і логічне розташування.

Шаблон майбутнього програмного продукту зображено на рисунку А.1.

Рисунок А.1 – Схема вигляду чату з ботом через текстове меню або кнопки команд

А.3.5 Система навігації (карта телеграм-боту)

Карта телеграм-боту зображена на рисунку А.2.

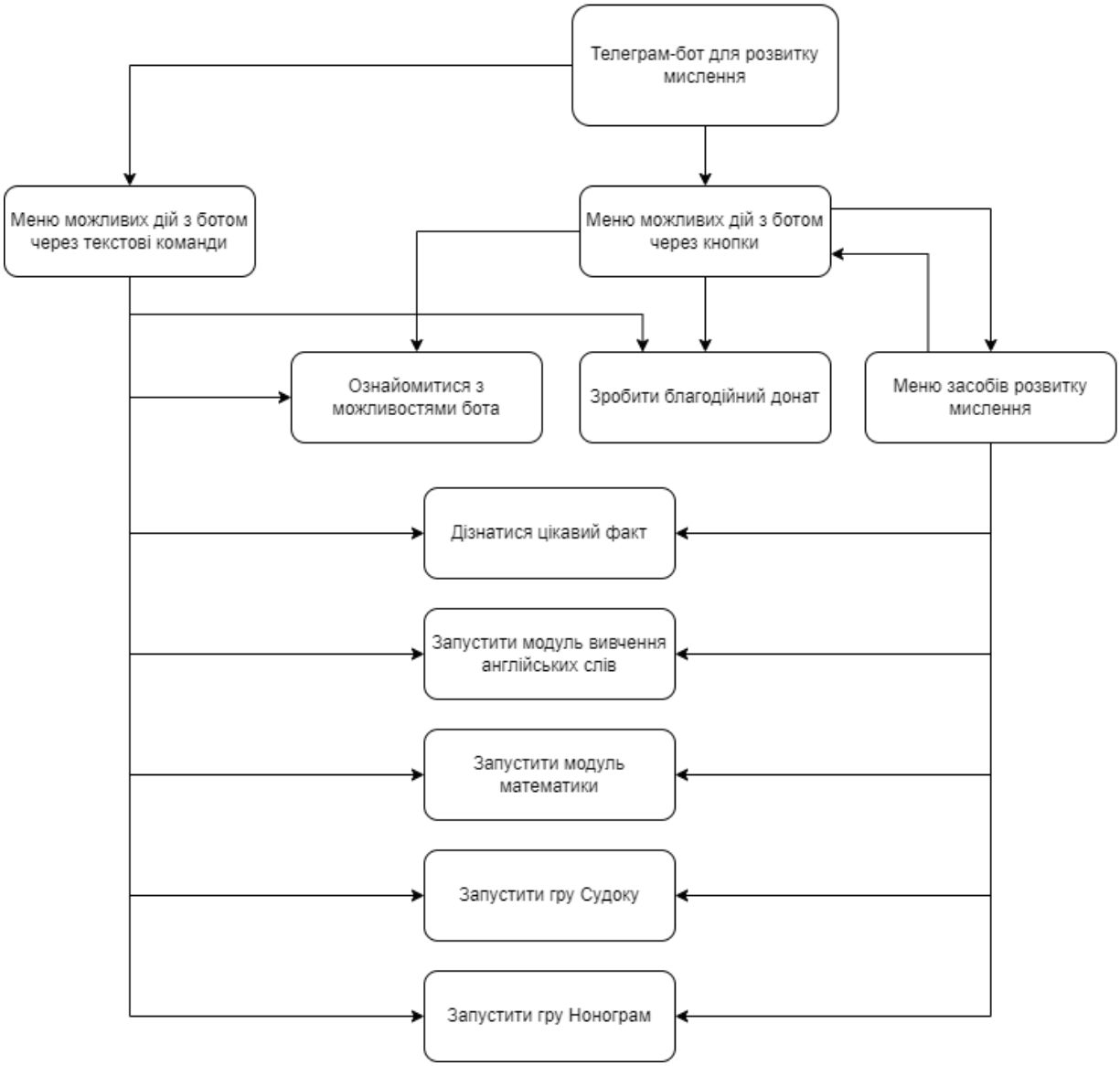


Рисунок А.2 – Система навігації для користувача

А.4 Вимоги до видів забезпечення

А.4.1 Вимоги до лінгвістичного забезпечення

Весь текст, який відповідає бот – має бути українською мовою, окрім випадку роботи – модулю вивчення англійської мови або необхідності використання термінології оригінальною мовою у модулі цікавих фактів.

А.4.2 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи телеграм-боту повинне бути стабільне інтернет-підключення та будь який спосіб підключення до месенджеру Телеграм: через web-браузер(Chrome, Opera, Firefox тощо…) або встановлений додаток.

А.5 Вимоги функціонування системи

А.5.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Можливість ознайомитися з функціоналом та інформацією про бота	Користувач
UN-02	Можливість благодійно задонатити на розвиток проекту	Користувач
UN-03	Можливість запуску модулю вивчення англійських слів	Користувач
UN-04	Можливість запуску модулю надсилання цікавих фактів	Користувач
UN-05	Можливість запуску модулю математики	Користувач
UN-06	Можливість запуску модулю гри Судоку	Користувач
UN-07	Можливість запуску модулю гри Нонограм	Користувач
UN-08	Редагування інформації про бота	Адміністратор
UN-09	Керування списком користувачів бота	Адміністратор
UN-10	Додавання нових рівнів ігор чи тестів	Адміністратор

А.5.2 Функціональні вимоги

Проаналізувавши потреби користувачів було визначено наступні функціональні вимоги:

− обмеження доступного функціоналу в залежності від категорії користувача;

- наявність блоку інформації про бота та його функціонал;
- наявність системи рівнів;
- наявність двох видів систем роботи з ботом: через текстові команди та через спеціальні кнопки;
- наявність адміністративної панелі та можливість доповнення бази даних з неї;
- можливість здійснення благодійного донату через спеціальний пункт меню;
- можливість запуску через меню засобів розвитку мислення: Судоку, Нонограм, дізнання цікавого факту, математичних тестів, вивчення англійських слів.

А.6 Склад і зміст робіт зі створення телеграм-боту для розвитку логічного мислення

Детальний опис етапів створення телеграм-боту наведено у таблиці А.2.

Таблиця А.2 – Етапи створення телеграм-боту

№	Склад і зміст робіт	Строк розробки
1	Розробка шаблону телеграм-боту	5 днів
2	Розробка меню бота через текстові команди	7 днів
3	Розробка меню бота через кнопки	7 днів
4	Розробка модулю надсилання цікавих фактів	12 днів
5	Розробка модулю вивчення англійських слів	12 днів
6	Розробка модулю математики	12 днів
7	Розробка модулю Судоку	12 днів
8	Розробка модулю Нонограм	12 днів
9	Розробка адміністративної панелі	10 днів
10	Alpha-тестування	5 днів
11	Beta-тестування	8 днів
12	Розміщення на хостингу	2 дні
13	Перевірка працездатності	2 дні
14	Написання супровідної документації	7 днів
15	Реліз телеграм-боту	1 день
	Загальна тривалість робіт	114 днів

А.7 Вимоги до складу й змісту робіт із введення телеграм-бота в експлуатацію

При відповідності створюваного боту вимогам технічного завдання (ТЗ), телеграм-бот має бути затверджено й розміщено на хостингу.

ДОДАТОК Б – ПЛАНУВАННЯ ПРОЕКТУ

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПЛАНУВАННЯ

розробки

«Телеграм-боту для розвитку логічного мислення»

2024

ДОДАТОК Б

Планування робіт

Деталізація мети проєкту методом SMART.

Для успішності та конкурентоспроможності проєкту треба на концептуальному етапі правильно визначити його мету за допомогою SMART-методу . Для виконавця даного проекту формат постановки SMART-мети такий:

«Розробити багатофункціональний телеграм-бот зі зручним інтерфейсом, який поєднає різні засоби розвитку мислення, такі як можливість дізнавання нових цікавих фактів, вивчення англійських слів, вирішення простих математичних тестів, а також головоломки Судоку та Нонограм, що сприятиме зацікавленню більшої кількості людей у корисному проведенні неформального відпочинку – а саме використання засобів розвитку мозку, до кінця 4 курсу - 01 червня 2024 року».

Результати деталізації мети даного проекту представлено в таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

	Розробити багатофункціональний телеграм-бот, який поєднає різні засоби розвитку мислення та зручний інтерфейс
	Розроблений телеграм-бот з функціями гри Судоку, гри Нонограм, можливості дізнатися нові цікаві факти, вивчити нові англійські слова, проходження простих математичних тестів
	Мета досяжна, є затверджене технічне завдання від замовника, а існуючі аналоги мають лиш долю функціоналу
	Зацікавити більшу кількість людей у корисному проведенні неформального відпочинку – а саме використання засобів розвитку мозку
	Є конкретний термін – до кінця 4 курсу (01 червня 2024 р.)

Планування змісту робіт.

Планування змісту робіт включає в себе використання WBS (ієрархічна структура робіт) – це графічне представлення елементів проєкту, що організовані ієрархічно в єдину структуру з продуктом проєкту. Структура декомпозиції робіт спрямована на ефективне виконання завдань поетапно і виступає ключовою складовою командної роботи, враховуючи продукти, дані та послуги. WBS служить каркасом для оцінки термінів, контролю та графіків робіт.

На першому рівні WBS розташований продукт проєкту. Дії та заходи для досягнення мети проєкту фіксуються на другому рівні декомпозиції. Декомпозиція робіт триває до тих пір, поки вони не стають елементарними (простими).

Елементарні роботи – це дії, що мають конкретний, однозначний результат, присвоєні конкретній особі, з обчислюваною витратою праці і тривалістю виконання. На рисунку Б.1 подано WBS для розробки телеграм-боту для розвитку мислення.

Планування структури виконавців.

Наступний крок після декомпозиції процесів – це планування структури виконавців, що включає розробку організаційної структури виконавців (OBS). OBS є графічною структурою, що відображає учасників або відповідальних осіб, задіяних у реалізації проєкту.

Відповідальні особи – це співробітники, які керують організацією і виконанням елементарних робіт, визначених в WBS. Кожну елементарну роботу можна розглядати як окремий проєкт. На рисунку Б.2 представлено організаційну структуру планування проєкту, а таблиця Б.2 містить список виконавців, які діють у проєкті.

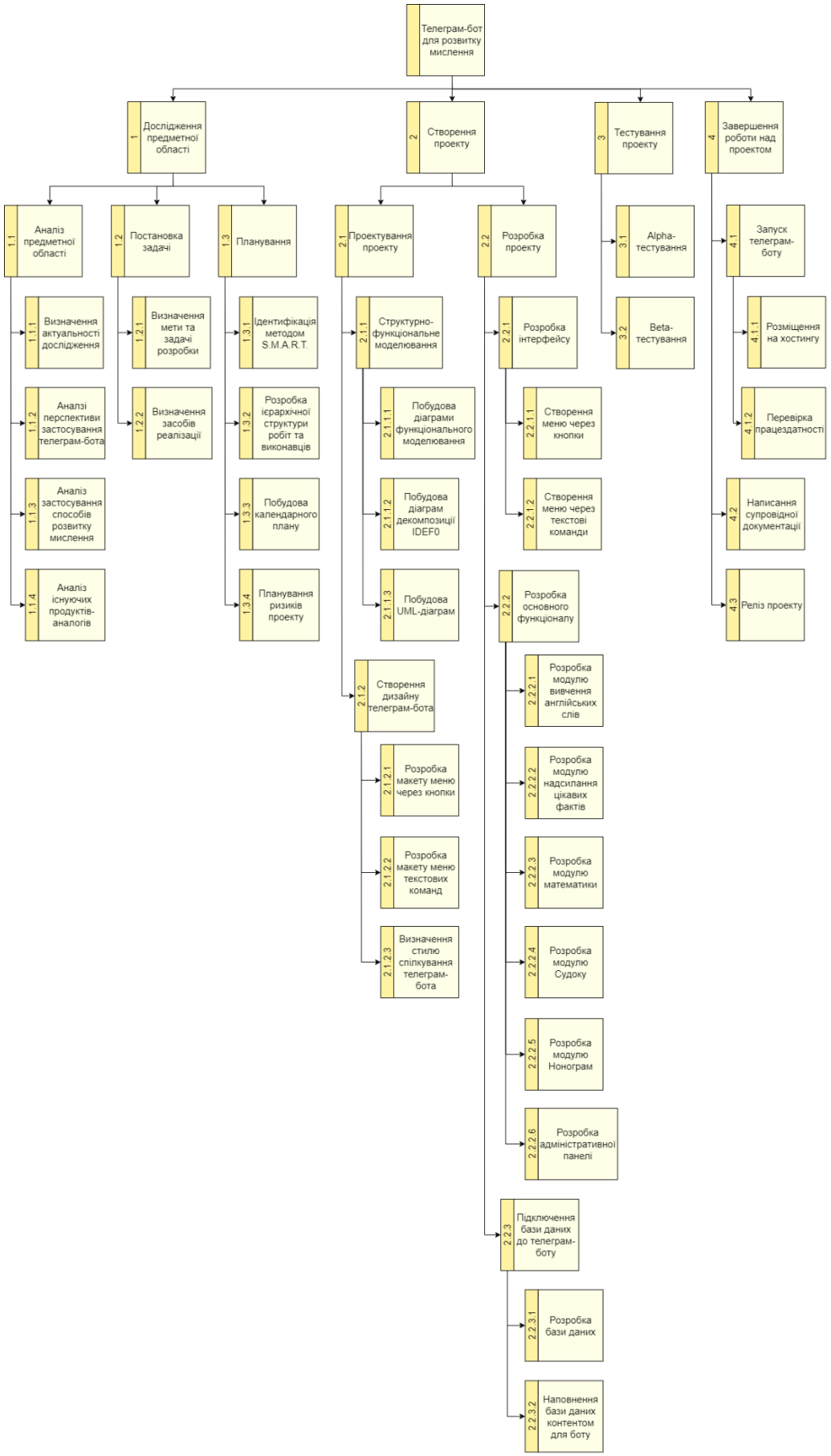


Рисунок 0.1 – WBS-структура робіт проєкту

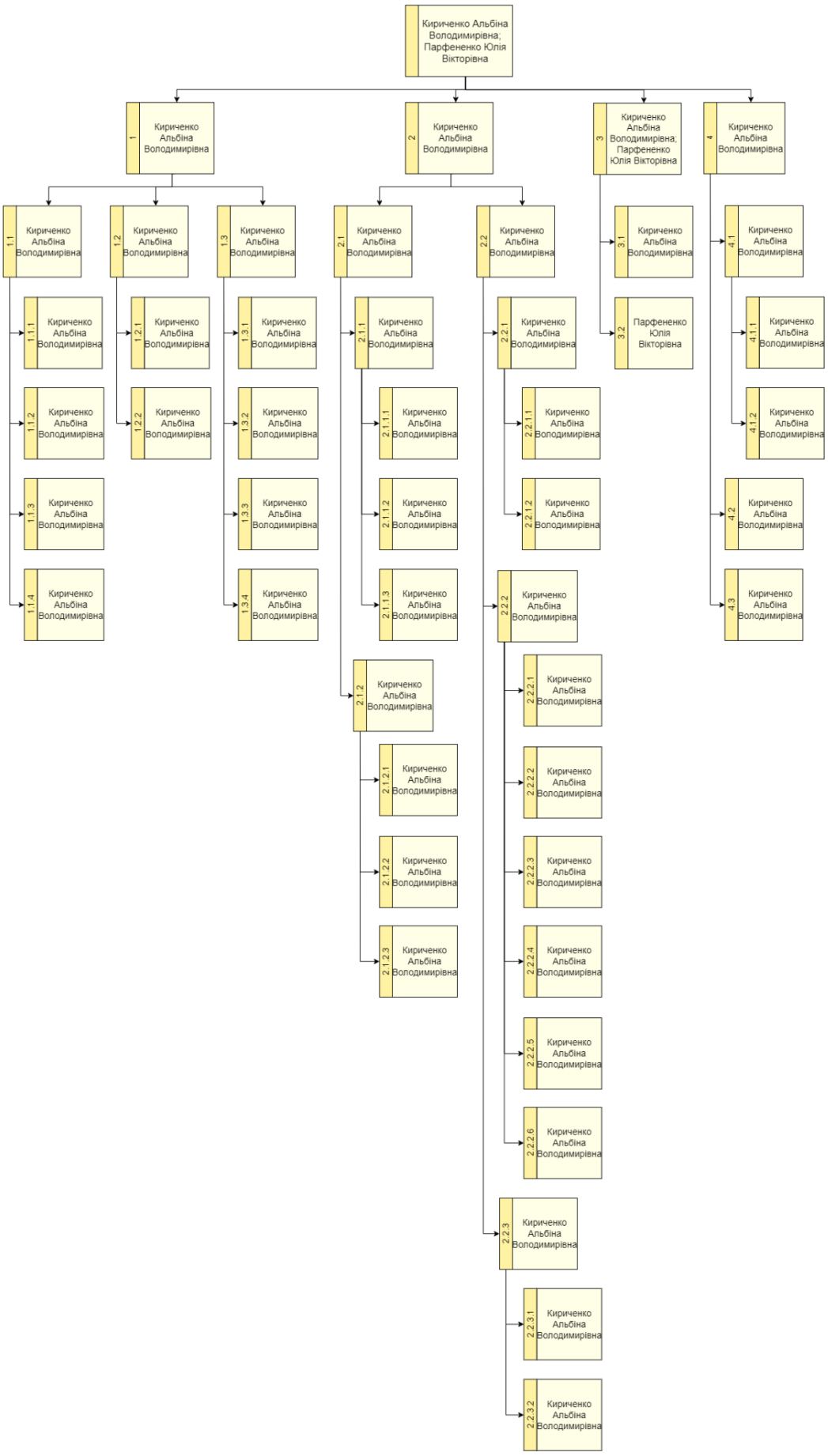


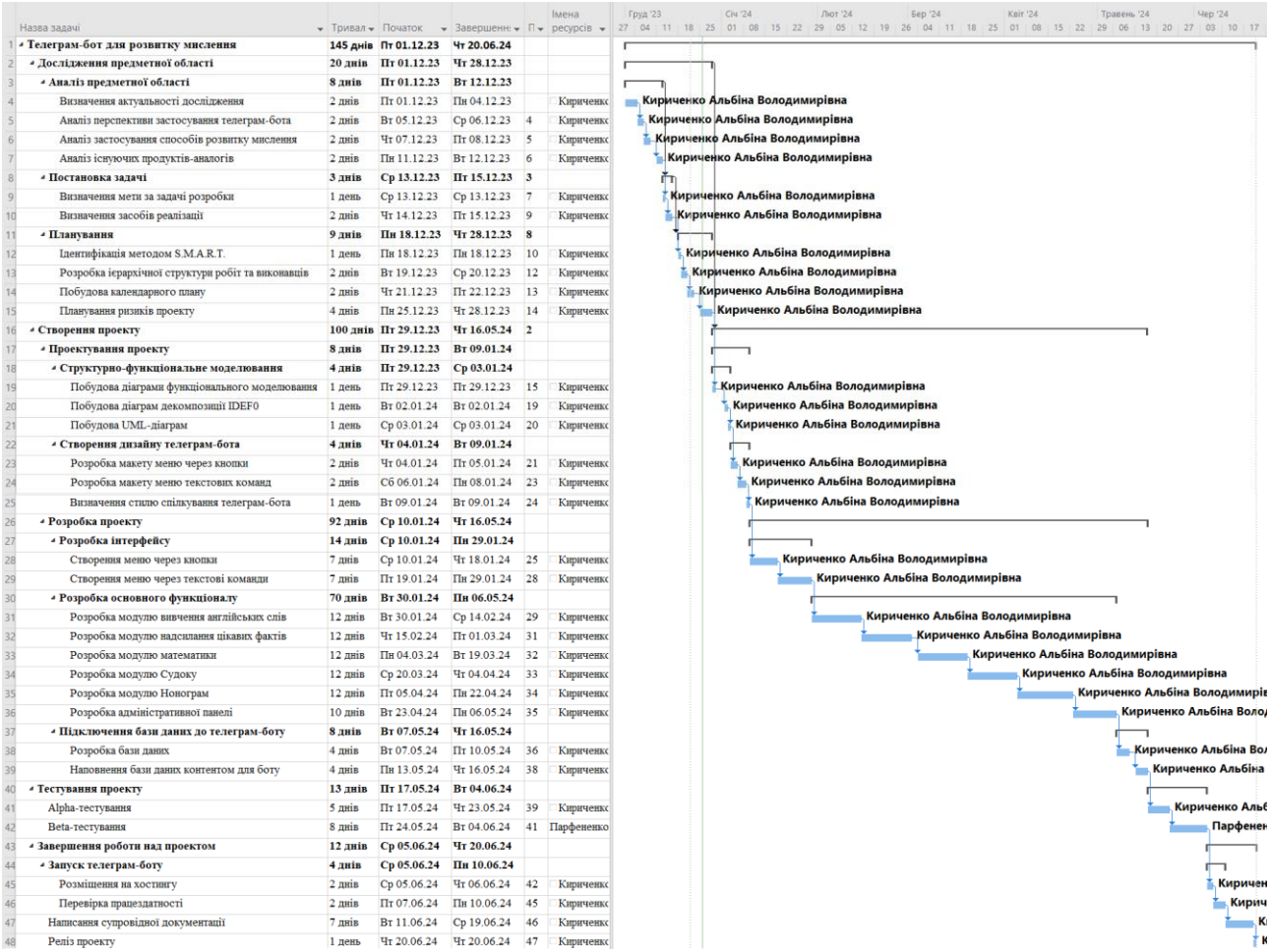
Рисунок 0.2 – OBS-структура робіт проєкту

Таблиця Б.2 – Виконавці проєкту

Роль	ПІБ	Проєктна роль
Розробник	Кириченко А.В.	Виконання front-end та back-end розробки
Проєктувальник	Кириченко А.В.	Виконання проєктування бази даних та розроблення структури телеграм-боту
Дизайнер	Кириченко А.В.	Створення дизайну телеграм-боту та його макету.
Alpha-Тестувальник	Кириченко А.В.	Тестування функціоналу бота на рівні
Beta-Тестувальник	Парфененко Ю.В.	Тестування функціоналу бота на рівні
Менеджер проєкту	Кириченко А.В.	Слідкування за виконанням термінів, розподілі ресурсів та завдань між учасниками. Виконання збору та аналізу даних.
Замовник проєкту	Парфененко Ю.В.	Формування завдання на розробку проєкту.

Діаграма Ганта.

Організація календарного графіка (діаграми Ганта) є важливим етапом у процесі планування проєкту, що представляє графічний перегляд розкладу виконання робіт з відображенням конкретних дат. Цей інструмент дозволяє отримати чітке уявлення про тривалість процесів при умові обмежених ресурсів та врахування вихідних та святкових днів. Календарний графік проєкту подано на рисунку Б.3.



	Назва задачі	Тривал	Початок	Завершенн	П	Імена ресурсів
1	**Телеграм-бот для розвитку мислення**	**145 днів**	**Пт 01.12.23**	**Чт 20.06.24**		
2	**Дослідження предметної області**	**20 днів**	**Пт 01.12.23**	**Чт 28.12.23**		
3	**Аналіз предметної області**	**8 днів**	**Пт 01.12.23**	**Вт 12.12.23**		
4	Визначення актуальності дослідження	2 днів	Пт 01.12.23	Пн 04.12.23		Кириченкс
5	Аналіз перспективи застосування телеграм-бота	2 днів	Вт 05.12.23	Ср 06.12.23	4	Кириченкс
6	Аналіз застосування способів розвитку мислення	2 днів	Чт 07.12.23	Пт 08.12.23	5	Кириченкс
7	Аналіз існуючих продуктів-аналогів	2 днів	Пн 11.12.23	Вт 12.12.23	6	Кириченкс
8	**Постановка задачі**	**3 днів**	**Ср 13.12.23**	**Пт 15.12.23**	**3**	
9	Визначення мети за задачі розробки	1 день	Ср 13.12.23	Ср 13.12.23	7	Кириченкс
10	Визначення засобів реалізації	2 днів	Чт 14.12.23	Пт 15.12.23	9	Кириченкс
11	**Планування**	**9 днів**	**Пн 18.12.23**	**Чт 28.12.23**	**8**	
12	Ідентифікація методом S.M.A.R.T.	1 день	Пн 18.12.23	Пн 18.12.23	10	Кириченкс
13	Розробка ієрархічної структури робіт та виконавців	2 днів	Вт 19.12.23	Ср 20.12.23	12	Кириченкс
14	Побудова календарного плану	2 днів	Чт 21.12.23	Пт 22.12.23	13	Кириченкс
15	Планування ризиків проекту	4 днів	Пн 25.12.23	Чт 28.12.23	14	Кириченкс
16	**Створення проекту**	**100 днів**	**Пт 29.12.23**	**Чт 16.05.24**	**2**	
17	**Проектування проекту**	**8 днів**	**Пт 29.12.23**	**Вт 09.01.24**		
18	**Структурно-функціональне моделювання**	**4 днів**	**Пт 29.12.23**	**Ср 03.01.24**		
19	Побудова діаграми функціонального моделювання	1 день	Пт 29.12.23	Пт 29.12.23	15	Кириченкс
20	Побудова діаграм декомпозиції IDEF0	1 день	Вт 02.01.24	Вт 02.01.24	19	Кириченкс
21	Побудова UML-діаграм	1 день	Ср 03.01.24	Ср 03.01.24	20	Кириченкс
22	**Створення дизайну телеграм-бота**	**4 днів**	**Чт 04.01.24**	**Вт 09.01.24**		
23	Розробка макету меню через кнопки	2 днів	Чт 04.01.24	Пт 05.01.24	21	Кириченкс
24	Розробка макету меню текстових команд	2 днів	Сб 06.01.24	Пн 08.01.24	23	Кириченкс
25	Визначення стилю спілкування телеграм-бота	1 день	Вт 09.01.24	Вт 09.01.24	24	Кириченкс
26	**Розробка проекту**	**92 днів**	**Ср 10.01.24**	**Чт 16.05.24**		
27	**Розробка інтерфейсу**	**14 днів**	**Ср 10.01.24**	**Пн 29.01.24**		
28	Створення меню через кнопки	7 днів	Ср 10.01.24	Чт 18.01.24	25	Кириченкс
29	Створення меню через текстові команди	7 днів	Пт 19.01.24	Пн 29.01.24	28	Кириченкс
30	**Розробка основного функціоналу**	**70 днів**	**Вт 30.01.24**	**Пн 06.05.24**		
31	Розробка модулю вивчення англійських слів	12 днів	Вт 30.01.24	Ср 14.02.24	29	Кириченкс
32	Розробка модулю надсилання цікавих фактів	12 днів	Чт 15.02.24	Пт 01.03.24	31	Кириченкс
33	Розробка модулю математики	12 днів	Пн 04.03.24	Вт 19.03.24	32	Кириченкс
34	Розробка модулю Судоку	12 днів	Ср 20.03.24	Чт 04.04.24	33	Кириченкс
35	Розробка модулю Нонограм	12 днів	Пт 05.04.24	Пн 22.04.24	34	Кириченкс
36	Розробка адміністративної панелі	10 днів	Вт 23.04.24	Пн 06.05.24	35	Кириченкс
37	**Підключення бази даних до телеграм-боту**	**8 днів**	**Вт 07.05.24**	**Чт 16.05.24**		
38	Розробка бази даних	4 днів	Вт 07.05.24	Пт 10.05.24	36	Кириченкс
39	Наповнення бази даних контентом для боту	4 днів	Пн 13.05.24	Чт 16.05.24	38	Кириченкс
40	**Тестування проекту**	**13 днів**	**Пт 17.05.24**	**Вт 04.06.24**		
41	Alpha-тестування	5 днів	Пт 17.05.24	Чт 23.05.24	39	Кириченкс
42	Beta-тестування	8 днів	Пт 24.05.24	Вт 04.06.24	41	Парфененко
43	**Завершення роботи над проектом**	**12 днів**	**Ср 05.06.24**	**Чт 20.06.24**		
44	**Запуск телеграм-боту**	**4 днів**	**Ср 05.06.24**	**Пн 10.06.24**		
45	Розміщення на хостингу	2 днів	Ср 05.06.24	Чт 06.06.24	42	Кириченкс
46	Перевірка працездатності	2 днів	Пт 07.06.24	Пн 10.06.24	45	Кириченкс
47	Написання супровідної документації	7 днів	Вт 11.06.24	Ср 19.06.24	46	Кириченкс
48	Реліз проекту	1 день	Чт 20.06.24	Чт 20.06.24	47	Кириченкс

Рисунок 0.3 – Календарний графік проєкту

Аналіз ризиків.

При виконанні якісної оцінки ризиків необхідно визначити ті з них, які потребують негайного усунення. Відповідно до важливості кожного ризику, застосовується відповідна стратегія реагування. Далі проводиться кількісне оцінювання ризиків. Ці дві оцінки можуть здійснюватися паралельно або окремо, залежно від рівня забезпечення проєкту. У таблиці Б.3 наведено перелік ризиків даного проєкту, а результати оцінки ризиків подано у таблиці Б.4. Таблиця Б.5 містить шкалу для класифікації ризиків за ступенем впливу на проєкт та ймовірністю їх виникнення.

Таблиця Б.3 – Ризики проєкту

№ ризику	Назва (опис) ризику
	Сезонна хвороба працівника (ГРВІ)
	Відсутність доступу до інтернет-з’єднання
	Вимкнення світла\блекаути
	Поломка техніки(обладнання) працівника (ноутбука)
	Неоптимальний розподіл часу
	Проблеми з сервером (відмова\ перебої в роботі)
	Зміни в вимогах до проекту
	Обмеженість інструментарію месенджеру
	Стихійне лихо
	Випуск «сирого» продукту (недогледіла проблеми при тестуванні)

Таблиця Б.4 – Результати визначення ймовірності, впливу та рангу ризиків

№ ризику	Назва (опис) ризику	Ймовірність (0.1 – 0.9)	Вплив (0.05 – 0.8)	Ранг
	Сезонна хвороба працівника			
	Відсутність доступу до інтернет-з’єднання			
	Вимкнення світла			
	Поломка техніки			
	Неоптимальний розподіл часу			
	Проблеми з сервером			
	Зміни в вимогах до проекту			
	Обмеженість інструментарію месенджеру			
	Стихійне лихо			
	Випуск «сирого» продукту			

Таблиця Б.5 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
	Низька	Низький	Прийнятні
	Середня	Середній	Виправдані
	Висока	Високий	Недопустимі

Для зменшення негативного впливу ризиків на проєкт необхідно провести процес планування реагування на них, включаючи оцінку можливих наслідків та розробку відповідних заходів. Аналіз здійснюється на основі показників, що визначені у таблиці Б.4. У результаті планування заходів реагування на ризики проєкту була створена матриця ймовірності виникнення та впливу ризиків (табл. Б.6). Зеленим кольором позначено прийнятні ризики, жовтим – виправдані, а червоним – неприпустимі.

Таблиця Б.6 – Матриця ймовірності та впливу

Ймовірність ризику	**Вплив загрози (ризику)**				
	Дуже малий	**Малий**	**Середній**	**Великий**	**Дуже великий**

Розподіл ризиків проєкту за рівнем подано в таблиці Б.7 відповідно до значень індексу.

Таблиця Б.7 – Шкала оцінювання за рівнем ризику

№	**Назва**	**Межі**	**Ризики, які входять (номера)**
	Прийнятні	0,005≤R≤0,05	
	Виправдані	0,05<R≤0,14	
	Недопустимі	0,14<R≤0,72	

Детальний опис ризиків та відповідних стратегій реагування наведено в таблиці Б.8.

Таблиця Б.8 – Ризики проєкту та стратегії реагування

ID ризику	Статус ризику	Опис ризику	Ймовір-ність виник-нення	Вплив ризику	Ранг ризику	Тип стратегії реагу-вання	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
	Виправ-даний	Сезонна хвороба працівника				Змен-шення	Підтримувати імунітет, не контактувати з хворими людьми.	Купити ліки для швидкого реагування на простуду. Бути готовою працювати в суботу щоб наздогнати календарний план.
	Виправ-даний	Відсутність доступу до інтернет-з'єднання				Змен-шення	Підключення на постійній основі як WI-FI, так і мобільного інтернету від різних операторів зв'язку.	Скласти перелік робіт, які будуть виконуватися без підключення до інтернету, до поки проблема не вирішиться.
	Виправ-даний	Вимкнення світла				Змен-шення	Визначити місце для розробки проекту – де ймовірність вимк-нення менша.	Знаходження\створення місця з постійним живленням (генератором)
	Виправ-даний	Поломка техніки				Змен-шення	Перед початком роботи над проектом – провести тех-перегляд.	Придбати запасне обладнання в центр роботи над проектом.
	Прий-нятний	Неоптима-льний розподіл часу				Змен-шення	Створення календарного плану з урахуванням свят і інших затримуючих факторів.	Бути готовою працювати в вихідні щоб наздогнати календарний план і встигнути в дедлайни.
	Виправ-даний	Проблеми з сервером				Прий-няття	Дослідити тенденцію проблем серверу Телеграм, та швидкість їх вирішення, щоб переконатися чи варто все таки використовувати цей сервер.	Якщо сервер месенджера Телеграм «ляже», то робота з ботом теж буде недоступною, як і робота з месенджером.
	Прий-нятний	Зміни в вимогах до проекту				Змен-шення	Затвердити всі вимоги по проекту – з дипломним керівником, до початку стадії розробки.	Намагатися вносити зміни до вимог, щоб нові вимоги були максимально схожі до попередніх.

Продовження таблиці Б.8

ID ризику	Статус ризику	Опис ризику	Ймовір-ність виник-нення	Вплив ризику	Ранг ризику	Тип стратегії реагу-вання	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
	Прий-нятний	Обмеже-ність інструмен-талу месенджеру				Змен-шення	Вивчити весь доступний функціонал та можливості месенджеру у питанні створення телеграм бота.	Знайти альтернативні варіанти додавання функціоналу бота.
	Прий-нятний	Стихійне лихо				Змен-шення	Вибрати основне місце розроблення проекту, де тенденція і ймовірність виникнення подібних явищ найменша	Передислокуватися в місце для розробки проекту, де наслідки явища мають найменший вплив
	Прий-нятний	Випуск «сирого» продукту				Змен-шення	Провести додаткове тестування сайту після основного, а також проводити невеликі тести в процесі роботи.	Скласти план «швидкого реагування» для виправлення помилок в найшвидші терміни.

ДОДАТОК В – ЛІСТИНГ ПРОГРАМНОГО КОДУ

Файл _bot.py:

```
import telebot
# Вказуємо токен бота
bot = telebot.TeleBot('7109174421:AAGtGv3NzcIRBd0vmYbsy6lbycnVVr_UfYI')
```

Файл connect_db.py:

```
import mysql.connector

def connect_to_database():
    try:
        conn = mysql.connector.connect(
            host='localhost',
            user='root',
            password='28072003',
            database='prokachaisya_bot'
        )
        return conn
    except mysql.connector.Error as e:
        print("Помилка підключення до бази даних:", e)
```

Файл thinking.py:

```
from telebot import types
from repository.connect_db import connect_to_database
from bot._bot import bot

conn = connect_to_database()

def start_game_menu_sudoky(chat_id):
    markup = types.ReplyKeyboardMarkup()  # Визначаємо змінну markup тут
    btn1 = types.KeyboardButton('Продовжити останню гру Судоку')
    btn2 = types.KeyboardButton('Вибрати рівень Судоку')
    btn3 = types.KeyboardButton('В попереднє меню')
    markup.row(btn1)
    markup.row(btn2)
    markup.row(btn3)
    bot.send_message(chat_id, f'Оберіть гру Судоку: ☺', reply_markup=markup)

def game_menu_sudoky(chat_id):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton('Зберегти прогрес і вийти')
    markup.add(btn1)
    bot.send_message(chat_id, "Ти завжди можеш продовжити пізніше ☺",
reply_markup=markup)

def start_game_menu_nonogram(chat_id):
    markup = types.ReplyKeyboardMarkup()
    btn1 = types.KeyboardButton('Продовжити останню гру Нонограм')
    btn2 = types.KeyboardButton('Вибрати рівень Нонограм')
    btn3 = types.KeyboardButton('В попереднє меню')
```

```
    markup.row(btn1)
    markup.row(btn2)
    markup.row(btn3)
    bot.send_message(chat_id, f'Оберіть гру Нонограм: ☺', reply_markup=markup)

def game_menu_nonogram(chat_id):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton('В попереднє меню')
    markup.add(btn1)
    bot.send_message(chat_id, "Ти завжди можеш продовжити пізніше ☺",
reply_markup=markup)

def game_menu_math(chat_id):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton('Закінчити і вийти')
    markup.add(btn1)
    bot.send_message(chat_id, "Правила математичного тесту прості 📝:\n"
                              "📎 Бот надсилає питання - і очікує відповідь\n"
                              "Надішліть результат обрахунків\n"
                              "📌 При необхідності - округляйте до *сотих*\n"
                              "📌 Знак між цілою і дробовою частиною - *крапка* (.)\n"
                              "📌 В питаннях з варіантами - пишіть *№* варіанту\n"
                              "Пам'ятайте, що ви не обмежені в часі □\n"
                              "📎 Приємного розвитку математичного мислення!",
reply_markup=markup, parse_mode='Markdown')

def introduction(chat_id):
    bot.send_message(chat_id,
```

```
            'Я телеграм-бот для *розвитку твого мислення*! У мені об\'єднані різні засоби для цього:\n'
            '□ Головоломки Судоку та Нонограм\n'
            '🏴 Можливість дізнатися цікавий факт\n'
            '□ Трохи математичних вправ\n'
            '🕮 Можливість вивчення англійських слів!\n'
            'Вибирай, що тобі до душі, та починай! ☺', parse_mode='Markdown')


def donate(chat_id):
    bot.send_message(chat_id,
                     f'Дякую, що хочеш підтримати проект 💝\nБлагодійний донат можеш здійснити на карту: 4441114425123799')


def start_thinking(chat_id):
    markup = types.ReplyKeyboardMarkup()
    btn1 = types.KeyboardButton('Дізнатися цікавий факт')
    btn2 = types.KeyboardButton('Вивчати англійську')
    btn3 = types.KeyboardButton('Швидка математика')
    btn4 = types.KeyboardButton('Грати Судоку')
    btn5 = types.KeyboardButton('Грати Нонограм')
    btn6 = types.KeyboardButton('В попереднє меню')
    markup.row(btn1, btn4)
    markup.row(btn2, btn5)
    markup.row(btn3, btn6)
    bot.send_message(chat_id, 'Вибери свій засіб для розвитку ✨', reply_markup=markup)


def interesting_facts(chat_id, cursor):
    cursor.execute("SELECT fact FROM facts ORDER BY RAND() LIMIT 1")
```

```
    fact = cursor.fetchone()
    bot.send_message(chat_id, f"Факт: *{fact}*", parse_mode='Markdown')


def english_word(chat_id, cursor):
    cursor.execute("SELECT word, translate, photo FROM english ORDER BY RAND() LIMIT 1")
    word, translate, photo_url = cursor.fetchone()
    bot.send_message(chat_id, f"Нове слово: *{word}*\nПереклад: {translate}", parse_mode='Markdown')
    bot.send_photo(chat_id, photo_url)
```

Файл math.py:

```
import random
import math
import ast
from bot._bot import bot
from handlers.thinking import game_menu_math, start_thinking


def fast_math(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, "Привіт! Готові до швидкої математики? □")
    game_menu_math(chat_id)
    bot.send_message(chat_id, "Тут буде 10 питань різної складності. Готові? Почнемо!")
    # для підрахунку балів і відстеження помилок
    score = 0
    mistakes = []
    # Рекурсивно викликаємо функцію для постановки наступного питання
```

```
    ask_question(message, 1, score, mistakes)
def ask_question(message, question_number, score, mistakes):
    chat_id = message.chat.id
    if question_number <= 10:
        # Генеруємо випадкове питання
        question_type, question, correct_answer, options_text, points = generate_question(question_number)
        # Відправляємо питання користувачу
        bot.send_message(chat_id, f"Питання {question_number}: {question}")
        if options_text:
            bot.send_message(chat_id, options_text)
        # Очікуємо відповідь користувача
        bot.register_next_step_handler(message, process_answer, question, correct_answer, points, question_number, score, mistakes)
    else:
        # Виводимо результати тесту
        percentage_score = (score / 100) * 100
        bot.send_message(chat_id, f"Ваш результат: {percentage_score}%")
        if mistakes:
            bot.send_message(chat_id, "Ви допустили деякі помилки. Зверніть увагу на такі аспекти:")
            for mistake in mistakes:
                bot.send_message(chat_id, mistake)


def process_answer(message, question, correct_answer, points, question_number, score, mistakes):
    if message.text == "Закінчити і вийти":
        start_thinking(message.chat.id)
        return
```

```
    user_answer = message.text.strip()
    try:
        user_answer = safe_eval(user_answer)
    except:
        pass
    if user_answer == correct_answer:
        score += points
    else:
        mistakes.append(
            f"🎾 Питання {question_number}: правильна відповідь - {correct_answer}, ваша відповідь - {user_answer}")
    # Задаємо наступне питання
    ask_question(message, question_number + 1, score, mistakes)


# Функція для генерації питань
def generate_question(question_number):
    question_generators = [
        generate_simple_calculation,
        generate_simple_calculation,
        generate_complex_calculation,
        generate_factorial,
        generate_trigonometry,
        generate_logarithm_formula_question,
        generate_logarithm_formula_question,
        generate_multiplication_formula_question,
        generate_derivative_question,
        generate_derivative_question,
    ]
```

```
    question_generator = question_generators[question_number - 1]
    question, correct_answer, points, options_text = question_generator()
    return "math", question, correct_answer, options_text, points
# Функції для генерації різних типів математичних тестів
def generate_simple_calculation():
    num1 = random.randint(1, 20)
    num2 = random.randint(1, 20)
    operator = random.choice(["+", "-", "*", "/"])
    question = f"{num1} {operator} {num2}"
    if operator == '+':
        answer = num1 + num2
    elif operator == '-':
        answer = num1 - num2
    elif operator == '*':
        answer = num1 * num2
    elif operator == '/':
        raw_answer = num1 / num2
        if raw_answer.is_integer():
            answer = int(raw_answer)
        elif round(raw_answer, 1) == raw_answer:
            answer = round(raw_answer, 1)
        else:
            answer = round(raw_answer, 2)
    points = 5
    return question, answer, points, ""

def generate_complex_calculation():
    num = random.randint(1, 20)
```

```
    question = f"Квадратний корінь з {num ** 2}"
    answer = num
    points = 5
    return question, answer, points, ""

def generate_logarithm_formula_question():
    log_questions = [
        {
            "question": "Назвіть формулу основної логарифмічної тотожності",
            "answer": "a^(loga(b)) = b",
        },
        {
            "question": "Назвіть формулу логарифму одиниці",
            "answer": "loga(1) = 0",
        },
        {
            "question": "Назвіть формулу логарифму добутку",
            "answer": "loga(x · y) = loga(x) + loga(y)",
        },
        {
            "question": "Назвіть формулу логарифму частки",
            "answer": "loga(x / y) = loga(x) - loga(y)",
        },
        {
            "question": "Назвіть формулу логарифму степеня числа",
            "answer": "loga(x^y) = y*loga(x)",
        },
        {
```

```
            "question": "Назвіть формулу логарифму числа, рівного основі",
            "answer": "loga(a) = 1",
        },
    ]
    selected_question = random.choice(log_questions)
    question = selected_question["question"]
    correct_answer = selected_question["answer"]
    # Генеруємо ще неправильні два випадкові варіанти відповідей
    options = [correct_answer]
    while len(options) < 3:
        random_option = random.choice(log_questions)["answer"]
        if random_option not in options:
            options.append(random_option)
    # Перемішуємо варіанти відповідей
    random.shuffle(options)
    correct_option_index = options.index(correct_answer) + 1
    options_text = "\n".join([f"{i + 1}. {option}" for i, option in enumerate(options)])
    points = 10
    return question, correct_option_index, points, options_text

def generate_factorial():
    num = random.randint(1, 10)
    question = f"{num}!"
    answer = math.factorial(num)
    points = 10
    return question, answer, points, ""

def generate_trigonometry():
```

```
    angle = random.randint(0, 24) * 15
    radians = math.radians(angle)
    question_type = random.choice(["sin", "cos", "tan", "cot"])
    if question_type == "sin":
        question = f"sin({angle}°)"
        answer = round(math.sin(radians), 2)
    elif question_type == "cos":
        question = f"cos({angle}°)"
        answer = round(math.cos(radians), 2)
    elif question_type == "tan":
        question = f"tan({angle}°)"
        answer = round(math.tan(radians), 2)
    elif question_type == "cot":
        if math.tan(radians) == 0:
            question = f"cot({angle}°)"
            answer = None
        else:
            question = f"cot({angle}°)"
            answer = round(1 / math.tan(radians), 2)
    points = 10
    return question, answer, points, ""
def safe_eval(expression):
    try:
        return ast.literal_eval(expression)
    except (SyntaxError, ValueError):
        return None

# Для генерації питань на скорочене множення і похідні функції
```

```
def generate_multiplication_formula_question():
    multi_questions = [
        {
            "question": "Назвіть формулу квадрату суми (a+b)^2",
            "answer": "a^2+2ab+b^2",
        },
        {
            "question": "Назвіть формулу квадрату різниці (a-b)^2",
            "answer": "a^2-2ab+b^2",
        },
        {
            "question": "Назвіть формулу різниці квадратів a^2-b^2",
            "answer": "(a-b)(a+b)",
        },
        {
            "question": "Назвіть формулу суми кубів a^3+b^3",
            "answer": "(a+b)(a^2-ab+b^2)",
        },
        {
            "question": "Назвіть формулу різниці кубів a^3-2b^3",
            "answer": "(a-b)(a^2+ab+b^2)",
        },
    ]

    selected_question = random.choice(multi_questions)
    question = selected_question["question"]
    correct_answer = selected_question["answer"]
    # Генеруємо два випадкові варіанти відповідей
```

```
    options = [correct_answer]
    while len(options) < 3:
        random_option = random.choice(multi_questions)["answer"]
        if random_option not in options:
            options.append(random_option)
    # Перемішуємо варіанти відповідей
    random.shuffle(options)
    correct_option_index = options.index(correct_answer) + 1
    options_text = "\n".join([f"{i + 1}. {option}" for i, option in enumerate(options)])
    points = 15
    return question, correct_option_index, points, options_text
def generate_derivative_question():
    pohidna_questions = [
        {
            "question": "Знайдіть похідну від x^2",
            "answer": "2x",
        },
        {
            "question": "Знайдіть похідну від sin(x)",
            "answer": "cos(x)",
        },
        {
            "question": "Знайдіть похідну від cos(x)",
            "answer": "-sin(x)",
        },
        {
            "question": "Знайдіть похідну від tan(x)",
            "answer": "sec^2(x)",
```

```
    },
    {
        "question": "Знайдіть похідну від e^x",
        "answer": "e^x",
    },
    {
        "question": "Знайдіть похідну від x",
        "answer": "1",
    },
    {
        "question": "Знайдіть похідну від ln(x)",
        "answer": "1/x",
    },
]
selected_question = random.choice(pohidna_questions)
question = selected_question["question"]
correct_answer = selected_question["answer"]
# Генеруємо два випадкові варіанти відповідей
options = [correct_answer]
while len(options) < 3:
    random_option = random.choice(pohidna_questions)["answer"]
    if random_option not in options:
        options.append(random_option)
# Перемішуємо варіанти відповідей
random.shuffle(options)
correct_option_index = options.index(correct_answer) + 1
options_text = "\n".join([f"{i + 1}. {option}" for i, option in enumerate(options)])
points = 10
```

```
    return question, correct_option_index, points, options_text
```

Файл sudoky.py:

```
import mysql.connector
from repository.connect_db import connect_to_database
from bot._bot import bot
from handlers.thinking import game_menu_sudoky, start_game_menu_sudoky

conn = connect_to_database()

def continue_game(message):
    user_nickname = message.from_user.username
    progress_info = load_progress(user_nickname)
    if progress_info:
        level_id = progress_info['sudoku_level_id']
        bot.send_message(message.chat.id, "Ваша збережена гра:")
        start_sudoku(message, level_id)
    else:
        bot.send_message(message.chat.id, "У вас немає збереженої гри □")

def new_game(message):
    user_nickname = message.from_user.username
    max_sudoku_level = get_max_sudoku_level(user_nickname)
    bot.send_message(message.chat.id, f"Вам доступні всі рівні до {max_sudoku_level}. Новий рівень відкривається відразу при проходженні останнього. "
                                      f"Розвивайте своє логічне мислення разом з Судоку!")
    bot.send_message(message.chat.id, "Введіть номер рівня, який ви хочете розпочати:")
```

```
    bot.register_next_step_handler(message, handle_new_game_input, max_sudoku_level)


def handle_new_game_input(message, max_sudoku_level):
    try:
        level_id = int(message.text)
        if level_id > max_sudoku_level:
            bot.send_message(message.chat.id, f"Вибраний рівень недоступний ☹. Вам доступні рівні до {max_sudoku_level}. \nЗапустимо останній доступний рівень ☺")
            level_id = max_sudoku_level
        start_sudoku(message, level_id)
    except ValueError:
        bot.send_message(message.chat.id, "Невірний формат вводу 📝 Напишіть число ^^")


def get_max_sudoku_level(user_nickname):
    try:
        cursor = conn.cursor()
        cursor.execute("SELECT max_sudoky FROM users WHERE user_nickname = %s", (user_nickname,))
        max_level = cursor.fetchone()
        return max_level[0] if max_level else None
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)
        return None


def get_sudoku_level(level_id):
    try:
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT * FROM sudoku_levels WHERE id = %s", (level_id,))
```

```
        level = cursor.fetchone()
        return level
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)
        return None

def format_sudoku_board(board):
    emoji_digits = {
        '1': '1️⃣', '2': '2️⃣', '3': '3️⃣',
        '4': '4️⃣', '5': '5️⃣', '6': '6️⃣',
        '7': '7️⃣', '8': '8️⃣', '9': '9️⃣',
        '0': '  '
    }
    formatted_board = ''
    for i in range(0, 81, 9):
        row = board[i:i+9]
        formatted_row = ''.join([emoji_digits[digit] for digit in row])
        formatted_board += formatted_row + '\n\n'
    return formatted_board

def start_sudoku(message, level_id):
    level = get_sudoku_level(level_id)
    if level:
        user_nickname = message.from_user.username
        progress_info = load_progress(user_nickname)
        progress_state = progress_info['progress_state_sudoku'] if progress_info and
progress_info['sudoku_level_id'] == level_id else None
        if progress_state:
```

```
            formatted_board = format_sudoku_board(progress_state)
        else:
            formatted_board = format_sudoku_board(level['start_state'])
            save_progress(user_nickname, level_id, level['start_state'])
        bot.send_message(message.chat.id, f"Рівень : {level_id}\n\n{formatted_board}")
        bot.send_message(message.chat.id, "Введіть ваші відповіді у форматі: <рядок> <стовпчик> <значення>")
        game_menu_sudoky(message.chat.id)
        bot.register_next_step_handler(message, handle_sudoku_input, level_id, level['solution'])
    else:
        bot.send_message(message.chat.id, "Помилка завантаження рівня Судоку □")


def handle_sudoku_input(message, level_id, solution):
    try:
        user_input = message.text.strip()


        if user_input.lower() == "зберегти прогрес і вийти":
            user_nickname = message.from_user.username
            progress_state = load_progress(user_nickname)['progress_state_sudoku']
            save_progress(user_nickname, level_id, progress_state)
            bot.send_message(message.chat.id, "Прогрес збережено. Ви повернулися до попереднього меню.")
            start_game_menu_sudoky(message.chat.id)
            return


        user_input = user_input.split()
        if len(user_input) != 3:
```

```
            bot.send_message(message.chat.id, "Неправильний формат. Спробуйте знову: <рядок> <стовпчик> <значення>")
            bot.register_next_step_handler(message, handle_sudoku_input, level_id, solution)
            return

        row, col, value = map(int, user_input)
        index = (row - 1) * 9 + (col - 1)

        if solution[index] == str(value):
            progress_state = load_progress(message.from_user.username)['progress_state_sudoku']
            progress_state = progress_state[:index] + str(value) + progress_state[index + 1:]
            save_progress(message.from_user.username, level_id, progress_state)
            bot.send_message(message.chat.id, f"Вірно! 🎉 Значення {value} на позиції ({row},{col})")

            if '0' not in progress_state:
                bot.send_message(message.chat.id, "Вітаємо! Ви успішно завершили рівень! 🎉")
                bot.send_photo(message.chat.id, "https://decorize.com.ua/image/catalog/%D0%9D%D0%9E%D0%92%D0%86%20%D0%A4%D0%9E%D0%A2%D0%9E%20%D0%91%D0%86%D0%9B%D0%90%D0%9D/%D0%92%D1%96%D1%82%D0%B0%D1%8E.jpg")
                start_game_menu_sudoky(message.chat.id)
                update_max_sudoku_level(message.from_user.username, level_id, message.chat.id)
            else:
                start_sudoku(message, level_id)
        else:
            bot.send_message(message.chat.id, f"Неправильно □. Спробуйте інше значення: \n<рядок> <стовпчик> <значення>")
```

```
            bot.register_next_step_handler(message, handle_sudoku_input, level_id, solution)
    except Exception as e:
        bot.send_message(message.chat.id, "Сталася помилка □ Спробуйте знову")
        bot.register_next_step_handler(message, handle_sudoku_input, level_id, solution)


def save_progress(user_nickname, level_id, progress_state):
    try:
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO user_progress (user_nickname, sudoku_level_id,
progress_state_sudoku) VALUES (%s, %s, %s) "
            "ON DUPLICATE KEY UPDATE progress_state_sudoku =
VALUES(progress_state_sudoku), sudoku_level_id = VALUES(sudoku_level_id)",
            (user_nickname, level_id, progress_state)
        )
        conn.commit()
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)


def load_progress(user_nickname):
    try:
        cursor = conn.cursor(dictionary=True)
        cursor.execute(
            "SELECT progress_state_sudoku, sudoku_level_id FROM user_progress WHERE
user_nickname = %s",
            (user_nickname,)
        )
        progress = cursor.fetchone()
        return progress if progress else None
```

```
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)
        return None


def update_max_sudoku_level(user_nickname, level_id, chat_id):
    try:
        cursor = conn.cursor()
        #кількість записів у таблиці sudoku_levels
        cursor.execute("SELECT COUNT(*) FROM sudoku_levels")
        count_levels = cursor.fetchone()[0]
        #max_sudoky для даного користувача
        cursor.execute("SELECT max_sudoky FROM users WHERE user_nickname = %s", (user_nickname,))
        max_sudoky = cursor.fetchone()[0]
        # Перевіряємо, чи користувач розв'язав останній доступний рівень
        if level_id == max_sudoky:
            if count_levels == max_sudoky:
                # Виводимо повідомлення, якщо кількість рівнів співпадає з максимально доступним
                bot.send_message(chat_id, "Ого!□ Ти зміг вирішити всі наявні голововломки Судоку! 🔥 Молодець!\n"
                                          "Ми постараємося додати нових як можна скоріше! 📈")
            else:
                # Оновлюємо значення max_sudoky і виводимо повідомлення про відкриття нового рівня
                cursor.execute(
                    "UPDATE users SET max_sudoky = max_sudoky + 1 WHERE user_nickname = %s", (user_nickname,)
                )
                conn.commit()
```

```
            bot.send_message(chat_id, "Новий рівень вже відкритий для тебе - вперед розвивати мислення! 🔥")
        else:
            bot.send_message(chat_id, "Тренування ніколи не бувають зайвими! 🔥 Ти молодець!\n"
                                      "Але для відкриття нового рівня - розв'яжи останній тобі доступний □")
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)
```

Файл nonogram.py:

```
import mysql.connector
from repository.connect_db import connect_to_database
from bot._bot import bot
from handlers.thinking import game_menu_nonogram, start_game_menu_nonogram
from PIL import Image, ImageDraw, ImageFont
from telebot import types
import io
import ast

conn = connect_to_database()

def create_nonogram_image(board, row_conditions, col_conditions):

    size = len(board)
    cell_size = 30  # розмір кожної комірки
    margin = 50  # поле для умов
    img_size = size * cell_size + margin
```

```
    img = Image.new('RGB', (img_size, img_size), color='white')
    draw = ImageDraw.Draw(img)
    font = ImageFont.load_default()

    for y in range(size):
        for x in range(size):
            fill = 'black' if board[y][x] == 1 else 'white'
            draw.rectangle(
                [margin + x * cell_size, margin + y * cell_size, margin + (x + 1) * cell_size,
                 margin + (y + 1) * cell_size],
                fill=fill,
                outline='gray'
            )

    # малюємо числа для рядків
    for i, condition in enumerate(row_conditions):
        draw.text((5, margin + i * cell_size + 10), condition, fill='black', font=font)
    # малюємо числа для стовпців
    for i, condition in enumerate(col_conditions):
        draw.text((margin + i * cell_size + 10, 5), condition, fill='black', font=font)

    return img


def send_nonogram(chat_id, board, row_conditions, col_conditions):
    img = create_nonogram_image(board, row_conditions, col_conditions)
    bio = io.BytesIO()
    bio.name = 'nonogram.png'
```

```
    img.save(bio, 'PNG')
    bio.seek(0)
    markup = types.InlineKeyboardMarkup()
    for y in range(len(board)):
        row = []
        for x in range(len(board[y])):
            callback_data = f"{y},{x}"
            row.append(types.InlineKeyboardButton(' ', callback_data=callback_data))
        markup.row(*row)
    bot.send_photo(chat_id, bio, reply_markup=markup)

@bot.callback_query_handler(func=lambda call: True)
def handle_nonogram_input(call):
    y, x = map(int, call.data.split(','))
    user_nickname = call.from_user.username
    progress_info = load_progress(user_nickname)
    if not progress_info:
        bot.send_message(call.message.chat.id, "Сталася помилка завантаження гри.")
        return
    # Парсимо умови зафарбування
    parsed_row_conditions = parse_conditions(progress_info['row_conditions'])
    parsed_col_conditions = parse_conditions(progress_info['col_conditions'])
    board = progress_info['progress_state_nonogram']
    board[y][x] = 1 - board[y][x]  # змінюємо колір комірки
    save_progress(user_nickname, board, progress_info['nonogram_level_id'])
    bot.edit_message_media(
        media=types.InputMediaPhoto(
            create_nonogram_image(board, parsed_row_conditions, parsed_col_conditions)),
```

```
        chat_id=call.message.chat.id,
        message_id=call.message.message_id,
        reply_markup=call.message.reply_markup
    )
    if check_solution(board, progress_info['nonogram_level_id']):
        bot.send_message(call.message.chat.id, "Вітаємо! Ви успішно завершили рівень! 🎉")
        update_max_nonogram_level(user_nickname, progress_info['nonogram_level_id'], call.message.chat.id)


def save_progress(user_nickname, board, level_id):
    try:
        # Перевіряємо, чи існує рівень з таким id
        if not get_nonogram_solution(level_id):
            print(f"Рівень з id {level_id} не існує.")
            return
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO user_progress (user_nickname, nonogram_level_id, progress_state_nonogram) VALUES (%s, %s, %s) "
            "ON DUPLICATE KEY UPDATE progress_state_nonogram = VALUES(progress_state_nonogram), nonogram_level_id = VALUES(nonogram_level_id)",
            (user_nickname, level_id, str(board))
        )
        conn.commit()
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)


def load_progress(user_nickname):
```

```
    try:
        cursor = conn.cursor(dictionary=True)
        # Перевірка наявності запису для користувача
        cursor.execute(
            "SELECT progress_state_nonogram, nonogram_level_id FROM user_progress
WHERE user_nickname = %s",
            (user_nickname,)
        )
        progress = cursor.fetchone()
        # Якщо запису немає, створюємо новий запис з початковими значеннями
        if not progress:
            cursor.execute(
                "INSERT INTO user_progress (user_nickname, nonogram_level_id,
progress_state_nonogram) VALUES (%s, %s, %s)",
                (user_nickname, 1, str([[0] * 5 for _ in range(5)]))
            )
            conn.commit()
            cursor.execute(
                "SELECT progress_state_nonogram, nonogram_level_id FROM user_progress
WHERE user_nickname = %s",
                (user_nickname,)
            )
            progress = cursor.fetchone()
        else:
            # Якщо запис є, перевіряємо на NULL значення
            if progress['progress_state_nonogram'] is None or progress['nonogram_level_id'] is
None:
                cursor.execute(
                    "UPDATE user_progress SET nonogram_level_id = %s,
progress_state_nonogram = %s WHERE user_nickname = %s",
```

```
                (1, str([[0] * 5 for _ in range(5)]), user_nickname)
            )
            conn.commit()
            cursor.execute(
                "SELECT progress_state_nonogram, nonogram_level_id FROM
user_progress WHERE user_nickname = %s",
                (user_nickname,)
            )
            progress = cursor.fetchone()

        if progress and isinstance(progress['progress_state_nonogram'], str):
            progress['progress_state_nonogram'] =
ast.literal_eval(progress['progress_state_nonogram'])

        # умови зафарбування поля з таблиці nonogram_levels
        nonogram_level_id = progress['nonogram_level_id']
        row_conditions, col_conditions = get_nonogram_conditions(nonogram_level_id)
        progress['row_conditions'] = row_conditions
        progress['col_conditions'] = col_conditions

        return progress
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)
        return None

def lets_game_nonogram(message):
    user_nickname = message.from_user.username
    max_nonogram_level = get_max_nonogram_level(user_nickname)
```

```
    bot.send_message(message.chat.id, f"Вам доступні всі рівні до {max_nonogram_level}. Новий рівень відкривається відразу при проходженні останнього. "
                                      f"Розвивайте своє логічне мислення разом з Нонограм!")
    bot.send_message(message.chat.id, "Введіть номер рівня, який ви хочете розпочати:")
    bot.register_next_step_handler(message, handle_new_game_nonogram_input, max_nonogram_level)


def handle_new_game_nonogram_input(message, max_nonogram_level):
    try:
        level_id = int(message.text)
        if level_id > max_nonogram_level:
            bot.send_message(message.chat.id, f"Вибраний рівень недоступний ☹. Вам доступні рівні до {max_nonogram_level}. "
                                              f"\nЗапустимо останній доступний рівень ☺")
            level_id = max_nonogram_level
        start_nonogram(message, level_id)
    except ValueError:
        if message.text.isdigit():
            bot.send_message(message.chat.id, "Введіть число в межах доступних рівнів.")
        else:
            bot.send_message(message.chat.id, "Невірний формат вводу 📝 Напишіть число ^^")


def continue_game_nonogram(message):
    user_nickname = message.from_user.username
    progress_info = load_progress(user_nickname)
    if progress_info:
```

```
        if progress_info['nonogram_level_id'] is None or
progress_info['progress_state_nonogram'] is None:
            bot.send_message(message.chat.id, "Ще немає збереженої гри, тож запустимо
1-й рівень Ноноґрам")
            start_nonogram(message, 1)
        else:
            level_id = progress_info['nonogram_level_id']
            bot.send_message(message.chat.id, "Ваша збережена гра:")
            start_nonogram(message, level_id)
    else:
        bot.send_message(message.chat.id, "У вас немає збереженої гри □")
        start_nonogram(message, 1)


def start_nonogram(message, level_id):
    game_menu_nonogram(message.chat.id)
    # Отримуємо умови зафарбування рядків та стовпців для цього рівня
    row_conditions, col_conditions = get_nonogram_conditions(level_id)
    # Парсимо умови зафарбування
    parsed_row_conditions = parse_conditions(row_conditions)
    parsed_col_conditions = parse_conditions(col_conditions)
    # Перевіряємо, чи є збережений прогрес гри для цього рівня
    nickname = message.from_user.username
    progress_info = load_progress(nickname)
    if progress_info and progress_info['nonogram_level_id'] == level_id:
        # Якщо є збережений прогрес, використовуємо його для відображення поля гри
        board = progress_info['progress_state_nonogram']
    else:
        # Якщо збереженого прогресу немає, створюємо нове поле гри
        board_size = max(len(parsed_row_conditions), len(parsed_col_conditions))
```

```
        board = [[0] * board_size for _ in range(board_size)]
        # Зберігаємо новий прогрес гри
        save_progress(message.from_user.username, board, level_id)
    # Відправляємо гру користувачеві
    send_nonogram(message.chat.id, board, parsed_row_conditions,
parsed_col_conditions)



def get_nonogram_solution(level_id):
    try:
        cursor = conn.cursor(dictionary=True)
        cursor.execute(
            "SELECT solution, row_conditions, col_conditions FROM nonogram_levels
WHERE id = %s",
            (level_id,)
        )
        level = cursor.fetchone()
        if level:
            level['row_conditions'] = level['row_conditions'].split(',')
            level['col_conditions'] = level['col_conditions'].split(',')
        return level
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)
        return None

def check_solution(board, level_id):
    level = get_nonogram_solution(level_id)
    if not level:
        return False
```

```
    solution = level['solution']
    # Перетворюємо рішення з рядка на двовимірний масив
    size = len(board)
    if len(solution) != size * size:
        print("Рішення не відповідає розміру дошки.")
        return False
    solution_board = [[int(solution[i * size + j]) for j in range(size)] for i in range(size)]

    return board == solution_board

def parse_conditions(conditions):
    parsed_conditions = []
    for condition in conditions.split(';'):
        parsed_condition = [int(c) for c in condition.split(',')]
        # з'єднуємо числа в рядок
        parsed_conditions.append(';'.join(map(str, parsed_condition)))
    return parsed_conditions

def get_nonogram_conditions(level_id):
    try:
        cursor = conn.cursor(dictionary=True)
        cursor.execute(
            "SELECT row_conditions, col_conditions FROM nonogram_levels WHERE id =
%s",
            (level_id,)
        )
        conditions = cursor.fetchone()
```

```
        return conditions['row_conditions'], conditions['col_conditions'] if conditions else (None, None)
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)
        return None, None


def get_max_nonogram_level(user_nickname):
    try:
        cursor = conn.cursor()
        cursor.execute("SELECT max_nonogram FROM users WHERE user_nickname = %s", (user_nickname,))
        max_level = cursor.fetchone()
        return max_level[0] if max_level else None
    except mysql.connector.Error as e:
        print("Помилка з'єднання з базою даних:", e)
        return None


def update_max_nonogram_level(user_nickname, level_id, chat_id):
    try:
        cursor = conn.cursor()
        # Отримуємо кількість записів у таблиці nonogram_levels
        cursor.execute("SELECT COUNT(*) FROM nonogram_levels")
        count_levels = cursor.fetchone()[0]
        # Отримуємо значення max_nonogram для даного користувача
        cursor.execute("SELECT max_nonogram FROM users WHERE user_nickname = %s", (user_nickname,))
        max_nonogram = cursor.fetchone()[0]
        # Перевіряємо, чи користувач розв'язав останній доступний рівень
        if level_id == max_nonogram:
```

```
        if count_levels == max_nonogram:
            # Виводимо повідомлення, якщо кількість рівнів співпадає з max_nonogram
            bot.send_message(chat_id, "Ого! 🔥 Вітаємо! Ви успішно пройшли всі доступні рівні Нонограм! 🎉 \nТак тримати! Ми докладемо зусиль, щоб якнайскоріше додати нових ❤")
        else:
            # Оновлюємо значення max_nonogram і виводимо повідомлення про відкриття нового рівня
            cursor.execute(
                "UPDATE users SET max_nonogram = max_nonogram + 1 WHERE user_nickname = %s", (user_nickname,)
            )
            conn.commit()
            bot.send_message(chat_id, "Ви відкрили новий рівень Нонограм! 🎉 Приємного геймплею з Нонограм! ❤")
    else:
        bot.send_message(chat_id, "Тренування завжди корисне 🔥! Але для відкриття нового рівня потрібно вирішити останній доступний 😉")
except mysql.connector.Error as e:
    print("Помилка з'єднання з базою даних:", e)
```

Файл admin_panell.py:

```
import mysql
import re
from telebot import types
from repository.connect_db import connect_to_database

conn = connect_to_database()
```

```
def view_all_users(chat_id, bot):
    try:
        if conn:
            cursor = conn.cursor()
            cursor.execute("SELECT user_nickname FROM users")
            users = cursor.fetchall()
            if users:
                user_list = '\n'.join([f'▫ - {user[0]}' for user in users])
                bot.send_message(chat_id, f"Список всіх користувачів:\n{user_list}")
            else:
                bot.send_message(chat_id, "Ще немає користувачів у базі даних")
    except mysql.connector.Error as e:
        print("Помилка підключення до бази даних: ", e)


# Функція для запиту нікнейму для видалення
def delete_users(message, bot):
    chat_id = message.chat.id
    bot.send_message(chat_id, "Введіть нікнейм користувача, якого потрібно
видалити:")
    bot.register_next_step_handler(message, process_delete_user, bot)
# Функція для обробки введеного нікнейму та видалення користувача з бази даних
def process_delete_user(message, bot):
    chat_id = message.chat.id
    user_nickname = message.text
    try:
        cursor = conn.cursor()
```

```
        # Спочатку видаляємо записи з user_progress
        cursor.execute("DELETE FROM user_progress WHERE user_nickname = %s", (user_nickname,))
        # Потім видаляємо запис з таблиці users
        cursor.execute("DELETE FROM users WHERE user_nickname = %s", (user_nickname,))
        conn.commit()
        if cursor.rowcount > 0:
            bot.send_message(chat_id, f"Користувач '{user_nickname}' був успішно видалений 🤌")
        else:
            bot.send_message(chat_id, f"Користувача з нікнеймом '{user_nickname}' не знайдено □")
    except mysql.connector.Error as e:
        print("Помилка підключення до бази даних: ", e)


def menu(chat_id, bot):
    markup = types.ReplyKeyboardMarkup()
    btn1 = types.KeyboardButton('Переглянути список всіх користувачів')
    btn2 = types.KeyboardButton('Видалити користувача')
    btn3 = types.KeyboardButton('Додати новий рівень Судоку')
    btn4 = types.KeyboardButton('Додати новий рівень Нонограм')
    btn5 = types.KeyboardButton('Додати нові цікаві факти')
    btn6 = types.KeyboardButton('Додати нові англійські слова')
    btn7 = types.KeyboardButton('Вийти з режима адміністратора')

    markup.row(btn1, btn2)
    markup.row(btn3, btn4)
    markup.row(btn5, btn6)
```

```
    markup.row(btn7)
    bot.send_message(chat_id, f'✅ Ви адміністратор! Керуйте! 😎', reply_markup=markup)


#додавання нового рівня Нонограм
def add_nonogram(message, bot):
    chat_id = message.chat.id
    bot.send_message(chat_id, "Додавання нового рівня Нонограм.")
    bot.send_message(chat_id, "Крок 1/4: Введіть рівень складності (наприклад, easy, medium, hard):")
    bot.register_next_step_handler(message, process_difficulty, bot)
def process_difficulty(message, bot):
    chat_id = message.chat.id
    difficulty = message.text.strip().lower()
    # Перевірка чи введено коректний рівень складності
    if difficulty not in ['easy', 'medium', 'hard']:
        bot.send_message(chat_id, "Некоректно введено рівень складності. Рівень не буде доданий")
        return
    bot.send_message(chat_id, "Крок 2/4: Введіть рішення Нонограм (рядок з 0 та 1, наприклад, 101011111101110010100111 0):")
    bot.register_next_step_handler(message, process_solution, difficulty, bot)
def process_solution(message, difficulty, bot):
    chat_id = message.chat.id
    solution = message.text.strip()
    # Перевірка чи введено коректне рішення для Нонограм
    if not all(char in ['0', '1'] for char in solution):
        bot.send_message(chat_id, "Некоректно введено рішення Нонограм. Введіть рядок з 0 та 1.")
        return
```

```
    bot.send_message(chat_id, "Крок 3/4: Введіть дані для заповнення рядків (наприклад, 1,1,1;5;3;1,1;3):")
    bot.register_next_step_handler(message, process_row_conditions, difficulty, solution, bot)
def process_row_conditions(message, difficulty, solution, bot):
    chat_id = message.chat.id
    row_conditions = message.text.strip()
    # Перевіряєо чи адмін ввів дані у правильному форматі
    # через кому якщо для одного рядка, значення для наступного рядка через крапку з комою
    if not re.match(r'^[\d,;]+$', row_conditions):
        bot.send_message(chat_id, "Некоректно введені дані для заповнення рядків.")
        return
    # Перевірка, що останній символ - цифра (без крапки з комою)
    if not row_conditions[-1].isdigit():
        bot.send_message(chat_id, "Останнім символом має бути цифра.")
        return
    bot.send_message(chat_id, "Крок 4/4: Введіть дані для заповнення стовпчиків (наприклад, 2;4;3,1;4;2):")
    bot.register_next_step_handler(message, process_col_conditions, difficulty, solution, row_conditions, bot)
def process_col_conditions(message, difficulty, solution, row_conditions, bot):
    chat_id = message.chat.id
    col_conditions = message.text.strip()
    # Перевірка формату як і для рядків
    if not re.match(r'^[\d,;]+$', col_conditions):
        bot.send_message(chat_id, "Некоректно введені дані для заповнення стовпчиків. Рівень не буде доданий.")
        return
    if not col_conditions[-1].isdigit():
```

```
        bot.send_message(chat_id, "Останнім символом має бути цифра. Рівень не буде доданий.")
        return
    # Додавання нового рівня Нонограм у базу даних
    try:
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO nonogram_levels (difficulty, solution, row_conditions, col_conditions) VALUES (%s, %s, %s, %s)",
            (difficulty, solution, row_conditions, col_conditions))
        conn.commit()
        bot.send_message(chat_id, "Новий рівень для Нонограм був успішно доданий.")
    except Exception as e:
        print("Помилка: ", e)
        bot.send_message(chat_id, "Сталася помилка під час додавання нового рівня для Нонограм. Рівень не буде доданий.")


#додавання рівня судоку - подібно до нонограм
def add_sudoku(message, bot):
    chat_id = message.chat.id
    bot.send_message(chat_id, "Додавання нового рівня Судоку.")
    bot.send_message(chat_id, "Крок 1/3: Введіть рівень складності (наприклад, easy, medium, hard):")
    bot.register_next_step_handler(message, process_sudoku_difficulty, bot)
def process_sudoku_difficulty(message, bot):
    chat_id = message.chat.id
    difficulty = message.text.strip().lower()
    # Перевірка чи введено коректний рівень складності
    if difficulty not in ['easy', 'medium', 'hard']:
```

```
        bot.send_message(chat_id, "Некоректно введено рівень складності. Рівень не буде доданий")
        return
    bot.send_message(chat_id, "Крок 2/3: Введіть рішення Судоку (рядок з чисел від 1 до 9, наприклад, 123456789456789123789...):")
    bot.register_next_step_handler(message, process_sudoku_solution, difficulty, bot)
def process_sudoku_solution(message, difficulty, bot):
    chat_id = message.chat.id
    solution = message.text.strip()
    # Перевірка формату даних для рішення Судоку: 81 символ і це цифри від 1 до 9
    if not re.match(r'^[1-9]{81}$', solution):
        bot.send_message(chat_id, "Некоректно введено рішення Судоку. Воно має складатися з 81 цифри від 1 до 9. Рівень не буде доданий")
        return
    bot.send_message(chat_id, "Крок 3/3: Введіть стан поля на початку гри Судоку (рядок з чисел від 0 до 9, який повинен мати ту ж довжину, що і рішення, наприклад, 1234560004507891...):")
    bot.register_next_step_handler(message, process_sudoku_start_state, difficulty, solution, bot)
def process_sudoku_start_state(message, difficulty, solution, bot):
    chat_id = message.chat.id
    start_state = message.text.strip()
    # Перевірка чи введено коректний початковий стану поля для гри Судоку: 81 символ від 0 до 9
    if not re.match(r'^[0-9]{81}$', start_state):
        bot.send_message(chat_id, "Некоректно введено дані про початок гри Судоку. Він має складатися з 81 цифри від 0 до 9. Рівень не буде доданий")
        return
    # Перевірка чи довжина початкового стану співпадає з довжиною рішення
    if len(start_state) != len(solution):
```

```
        bot.send_message(chat_id, "Початковий стан має бути тієї ж довжини, що і рішення. Рівень не буде доданий")
        return
    # Додавання нового рівня Судоку у базу даних
    try:
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO sudoku_levels (difficulty, solution, start_state) VALUES (%s, %s, %s)",
            (difficulty, solution, start_state))
        conn.commit()
        bot.send_message(chat_id, "Новий рівень для Судоку був успішно доданий.")
    except Exception as e:
        print("Помилка: ", e)
        bot.send_message(chat_id, "Сталася помилка під час додавання нового рівня для Судоку. Рівень не буде доданий.")


#додавання англійських слів
def add_english(message, bot):
    chat_id = message.chat.id
    bot.send_message(chat_id, "Додавання нового слова до бази даних.")
    bot.send_message(chat_id, "Крок 1/3: Введіть слово англійською (наприклад, pillow):")
    bot.register_next_step_handler(message, process_word, bot)
def process_word(message, bot):
    chat_id = message.chat.id
    word = message.text.strip()
    # Перевірка чи це дійсно слово: перевірка на те, що використані лише букви
    if not word.isalpha():
```

```
        bot.send_message(chat_id, "Некоректно введено слово. Слово повинно складатися лише з букв. Спробуйте ще раз.")
        return
    bot.send_message(chat_id, "Крок 2/3: Введіть переклад слова (наприклад, подушка):")
    bot.register_next_step_handler(message, process_translation, word, bot)
def process_translation(message, word, bot):
    chat_id = message.chat.id
    translation = message.text.strip()
    # та ж перевірка що і для англ слова
    if not translation.isalpha():
        bot.send_message(chat_id, "Некоректно введено переклад. Переклад повинен складатися лише з букв. Спробуйте ще раз.")
        return
    bot.send_message(chat_id, "Крок 3/3: Введіть URL зображення (наприклад, https://example.com/image.jpg):")
    bot.register_next_step_handler(message, process_photo, word, translation, bot)
def process_photo(message, word, translation, bot):
    chat_id = message.chat.id
    photo_url = message.text.strip()
    # Перевірка чи введено коректний URL зображення: спочатку і в кінці тексту
    if not (photo_url.startswith('http://') or photo_url.startswith('https://')) or not photo_url.endswith(('.jpg', '.jpeg', '.png', '.gif')):
        bot.send_message(chat_id, "Некоректно введено URL зображення. Спробуйте ще раз.")
        return
    # Додавання нового слова до бази даних
    try:
        cursor = conn.cursor()
        cursor.execute(
```

```
            "INSERT INTO english (word, translate, photo) VALUES (%s, %s, %s)",
            (word, translation, photo_url)
        )
        conn.commit()
        bot.send_message(chat_id, "Нове слово було успішно додане до бази даних.")
    except Exception as e:
        print("Помилка: ", e)
        bot.send_message(chat_id, "Сталася помилка під час додавання нового слова до бази даних.")


#додавання цікавого факту
def add_facts(message, bot):
    chat_id = message.chat.id
    bot.send_message(chat_id, "Додавання нового факту.")
    bot.send_message(chat_id, "Введіть факт:")
    bot.register_next_step_handler(message, process_fact, bot)
def process_fact(message, bot):
    chat_id = message.chat.id
    fact = message.text.strip()
    # Додавання нового факту в базу даних
    try:
        cursor = conn.cursor()
        cursor.execute("INSERT INTO facts (fact) VALUES (%s)", (fact,))
        conn.commit()
        bot.send_message(chat_id, "Новий факт був успішно доданий.")
    except Exception as e:
        print("Помилка: ", e)
        bot.send_message(chat_id, "Сталася помилка під час додавання нового факту.")
```

Файл main.py:

```
import mysql
from handlers.thinking import introduction, donate, start_thinking, interesting_facts, english_word, start_game_menu_sudoky
from repository.connect_db import connect_to_database
from handlers.admin_panell import menu, view_all_users, delete_users, add_nonogram, add_sudoku, add_english, add_facts
from handlers.sudoky import new_game, continue_game
from handlers.math import fast_math
from handlers.nonogram import start_game_menu_nonogram, lets_game_nonogram, continue_game_nonogram
from telebot import types
from bot._bot import bot

conn = connect_to_database()

# Функція для додавання нового користувача у таблицю users
def add_user_to_database(user_nickname):
    try:
        if conn:
            cursor = conn.cursor()
            # Перевірка, чи користувач вже існує у таблиці
            cursor.execute("SELECT * FROM users WHERE user_nickname = %s", (user_nickname,))
            if not cursor.fetchone():
                # Якщо користувач не існує, додати його до таблиці
                cursor.execute("INSERT INTO users (user_nickname) VALUES (%s)", (user_nickname,))
                conn.commit()
```

```
    except mysql.connector.Error as e:
        print("Помилка додавання користувача до бази даних:", e)


# Описуємо функцію, яка буде відповідати на команду '/start'
@bot.message_handler(commands=['start'])
def handle_start_first(message):
    user_name = message.from_user.first_name
    # Додаємо користувача до таблиці users
    add_user_to_database(message.from_user.username)
    bot.send_message(message.chat.id, f'Привіт, {user_name}! Розпочнемо розвиток твого мислення^^')
    handle_start(message)

@bot.message_handler(commands=['Start','Старт','старт','noadmin'])
def handle_start(message):
    markup = types.ReplyKeyboardMarkup()  # Визначаємо змінну markup тут
    btn1 = types.KeyboardButton('Ознайомитися з ботом')
    btn2 = types.KeyboardButton('Почати розвивати мислення')
    btn3 = types.KeyboardButton('Благодійно задонатити')
    markup.add(btn1)
    markup.row(btn2)
    markup.row(btn3)
    bot.send_message(message.chat.id, f'Яким буде твій наступний крок?', reply_markup=markup)

#адміністративна панель для користувачів зі статусом admin
@bot.message_handler(commands=['admin'])
def handle_admin_panel(message):
```

```
    if status(message) == "admin":
        menu(message.chat.id, bot)
    else:
        # Якщо користувач не адміністратор - вивести повідомлення про відмову
        bot.send_message(message.chat.id, f'❌ Вибач, але тобі доступ закритий □')


#функція перевірк статуса користувача
def status(message):
    try:
        if conn:
            cursor = conn.cursor()
            # Перевірка, чи користувач має статус 'admin'
            cursor.execute("SELECT * FROM users WHERE user_nickname = %s AND
status = 'admin'", (message.from_user.username,))
            if cursor.fetchone():
                return "admin"
            else:
                return "user"
    except mysql.connector.Error as e:
        bot.send_message(message.chat.id,"Помилка з'єднання з базою даних", e)


@bot.message_handler(commands=['info'])
def handle_info(message):
    message.text = 'Ознайомитися з ботом'
    handle_message(message)
@bot.message_handler(commands=['donate'])
def handle_donate(message):
    message.text = 'Благодійно задонатити'
```

```
    handle_message(message)
@bot.message_handler(commands=['facts'])
def handle_facts(message):
    message.text = 'Дізнатися цікавий факт'
    handle_message(message)
@bot.message_handler(commands=['math'])
def handle_math(message):
    message.text = 'Швидка математика'
    handle_message(message)
@bot.message_handler(commands=['english'])
def handle_english(message):
    message.text = 'Вивчати англійську'
    handle_message(message)
@bot.message_handler(commands=['sudoky'])
def handle_sudoky(message):
    message.text = 'Грати Судоку'
    handle_message(message)
@bot.message_handler(commands=['nonogram'])
def handle_nonogram(message):
    message.text = 'Грати Ноноґрам'
    handle_message(message)

@bot.message_handler(func=lambda message: True)
def handle_message(message):
    chat_id = message.chat.id
    if message.text == 'Ознайомитися з ботом':
        introduction(chat_id)
    elif message.text == 'Благодійно задонатити':
```

```
    donate(chat_id)
elif message.text == 'Почати розвивати мислення':
   start_thinking(chat_id)
elif message.text == 'Дізнатися цікавий факт':
    if conn:
        try:
            interesting_facts(chat_id, conn.cursor())
        except mysql.connector.Error as e:
            bot.send_message(chat_id, e)
elif message.text == 'Вивчати англійську':
    if conn:
        try:
            english_word(chat_id, conn.cursor())
        except mysql.connector.Error as e:
            bot.send_message(chat_id, e)
elif message.text == 'Швидка математика':
    fast_math(message)
elif message.text == 'Грати Судоку':
    start_game_menu_sudoky(message.chat.id)
elif message.text == 'Вибрати рівень Судоку':
    new_game(message)
elif message.text == 'Продовжити останню гру Судоку':
    continue_game(message)
elif message.text == 'Грати Нонограм':
    start_game_menu_nonogram(message.chat.id)
elif message.text == 'Вибрати рівень Нонограм':
    lets_game_nonogram(message)
elif message.text == 'Продовжити останню гру Нонограм':
```

```
        continue_game_nonogram(message)
    elif message.text == 'В попереднє меню':
        handle_start(message)  # Викликаємо функцію для відображення попереднього меню
    elif message.text == 'Зберегти прогрес і вийти':
        handle_start(message)
    elif message.text == 'Закінчити і вийти':
        handle_start(message)
    else:
        # Функція спрацює лиш коли її запустив адміністратор
        if message.text == 'Переглянути список всіх користувачів':
            if status(message) == "admin":
                view_all_users(message.chat.id, bot)
        elif message.text == 'Видалити користувача':
            if status(message) == "admin":
                delete_users(message, bot)
        elif message.text == 'Додати новий рівень Ноноград':
            if status(message) == "admin":
                add_nonogram(message, bot)
        elif message.text == 'Додати новий рівень Судоку':
            if status(message) == "admin":
                add_sudoku(message, bot)
        elif message.text == 'Додати нові англійські слова':
            if status(message) == "admin":
                add_english(message, bot)
        elif message.text == 'Додати нові цікаві факти':
            if status(message) == "admin":
                add_facts(message, bot)
        elif message.text == 'Вийти з режима адміністратора':
```

```
        if status(message) == "admin":
            message.text = 'В попереднє меню'
            handle_start(message)
    else:
        bot.send_message(message.chat.id, f'Вибач, я можу відповідати лише на передбачені повідомлення ☹\nСкористайся меню для роботи зі мною ☺')


# Запускаємо бота
bot.polling(none_stop=True)
```