МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет: **ННІ Каразінський банківський інститут**
Кафедра: **Інформаційних технологій та математичного моделювання**
Спеціальність: **122 Комп'ютерні науки**
Освітня програма: **Комп'ютерні науки та інформаційні технології в бізнесі**

Група: АК-21М **денна форма навчання**

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«ІННОВАЦІЙНІ МЕТОДИ МОТИВАЦІЇ В ОНЛАЙН-НАВЧАННІ: СТВОРЕННЯ TELEGRAM-БОТА ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ»

ЗА НАКАЗОМ № 4601-5/3045 ВІД 25 ВЕРЕСНЯ 2024 РОКУ

здобувача вищої освіти **Шопіна Валерія Валерійовича**

Робота допущена до захисту в ЕК

протокол кафедри ІТММ № 4 від 30.11.2024

Завідувач кафедри ІТММ
к. п. н., доцент
_____**Н. І. Стяглик**

Науковий керівник
к. ф.- м. н., доцент
_____**Л. Д. Філатова**

м. Харків 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Факультет навчально-науковий інститут "Каразінський банківський інститут"
Кафедра інформаційних технологій та математичного моделювання
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри

Н. І. Стяглик
"25 " вересня 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Шопіна Валерія Валерійовича
(прізвище, ім'я, по батькові студента)

1. Тема роботи «Інноваційні методи мотивації в онлайн-навчанні: створення Telegram-бота для вивчення англійської мови»

Керівник роботи к. ф.-м. н., доцент Л. Д. Філатова

затверджено наказом по університету від "25" вересня 2024 року № 4601-5/3045

2. Строк подання студентом роботи 20 листопада 2024 року

3. Перелік питань, які потрібно розробити:
У розділі 1: Проаналізувати мотиваційні підходи, гейміфікацію та роль чат-ботів у сучасній освіті.
У розділі 2: Розробити концептуальну модель Telegram-бота, описати гейміфікацію, фінансові стимули та соціальну взаємодію.
У розділі 3: Описати технічну архітектуру бота, основні функціональні можливості та реалізацію навчальних міні-ігор.
У розділі 4: Оцінити перспективи розвитку бота, визначити виклики впровадження та стратегії їх подолання.

4. План роботи

№ з/п	Назви етапів роботи
1	Вибір здобувачем теми кваліфікаційної магістерської роботи
2	Затвердження плану і завдання кваліфікаційної магістерської роботи
3	Здача кваліфікаційної магістерської роботи керівнику
4	Підпис кваліфікаційної магістерської роботи керівника
5	Підпис кваліфікаційної магістерської роботи у нормоконтролера
6	Допуск завідувачем кафедри до захисту кваліфікаційної магістерської роботи
7	Захист кваліфікаційної магістерської роботи

5. Дата видачі завдання 25 вересня 2024 року

Студент _____ **В. В. Шопін**
підпис ініціали, прізвище

Керівник роботи _____ **Л. Д. Філатова**
підпис ініціали, прізвище

РЕФЕРАТ
НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ «ІННОВАЦІЙНІ МЕТОДИ МОТИВАЦІЇ В ОНЛАЙН-НАВЧАННІ: СТВОРЕННЯ TELEGRAM-БОТА ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ»

Шопіна Валерія Валерійовича

Кваліфікаційна магістерська робота містить: 91 сторінок, 1 таблиці, 57 рисунків, список літератури з 37 найменувань.

Об'єктом дослідження є підтримка мотивації у процесі онлайн-навчання.

Предмет дослідження – Telegram-бот для підтримки мотивації у вивченні англійської мови.

Мета кваліфікаційної магістерської роботи полягає у розробці та дослідженні Telegram-бота, який використовує гейміфікацію, фінансові стимули та соціальну взаємодію для підтримки мотивації користувачів у довготривалому процесі вивчення англійської мови.

Завданнями кваліфікаційної магістерської роботи є:

у першому розділі проведено аналіз теоретичних аспектів мотивації в онлайн-навчанні, зокрема ролі гейміфікації, чат-ботів та специфіки українського ринку онлайн-освіти.

у другому розділі розглянуто основи проектування Telegram-бота LingOn, включаючи використання системи балів, міні-ігор, фінансових стимулів і соціальної взаємодії для створення мотиваційного середовища.

у третьому розділі детально описано технічну архітектуру бота, включаючи використання асинхронної бібліотеки Aiogram, структуру бази даних та основні функціональні можливості, такі як звітність, персоналізація профілю та інтеграція міні-ігор.

у четвертому розділі визначено перспективи розвитку бота, включаючи розширення функціоналу, інтеграцію нових технологій та розробку стратегій для подолання технічних і соціальних викликів.

Актуальність дослідження обумовлена необхідністю створення інноваційних інструментів, які сприяють підтримці мотивації у процесі вивчення мови, особливо у форматі онлайн. Telegram-бот LingOn відповідає сучасним потребам освітньої галузі, пропонуючи інтерактивне середовище, яке враховує особливості українського ринку онлайн-навчання.

За результатами дослідження було розроблено Telegram-бота, який забезпечує регулярне залучення користувачів, підтримує їхню зацікавленість та сприяє досягненню довгострокових цілей у вивченні англійської мови.

Практична новизна роботи полягає у впровадженні системи мотивації, яка поєднує елементи гейміфікації, фінансових стимулів і соціальної взаємодії. Бот LingOn реалізує підхід, орієнтований на користувача, забезпечуючи персоналізацію навчання та інтерактивність.

Одержані результати включають розробку Telegram-бота, який забезпечує регулярну участь користувачів, підтримує їхню зацікавленість та

сприяє досягненню довгострокових цілей у вивченні англійської мови.

КЛЮЧОВІ СЛОВА: МОТИВАЦІЯ, ГЕЙМІФІКАЦІЯ, TELEGRAM-БОТ, ОНЛАЙН-НАВЧАННЯ, АНГЛІЙСЬКА МОВА, ПЕРСОНАЛІЗАЦІЯ, МІНІ-ІГРИ, ФІНАНСОВІ СТИМУЛИ, СОЦІАЛЬНА ВЗАЄМОДІЯ.

ABSTRACT
AT QUALIFICATION MAGISTER WORK
«INNOVATIVE METHODS OF MOTIVATION IN ONLINE LEARNING: CREATING A TELEGRAM BOT FOR LEARNING ENGLISH»

Valerii Shopin

The master's qualification thesis includes: 91 pages, 1 table, 57 figures, and a reference list of 37 sources.

The object of the research is the maintenance of motivation in the process of online learning.

The subject of the research is a Telegram bot for supporting motivation in learning English.

The purpose of the master's qualification thesis is to develop and study a Telegram bot that uses gamification, financial incentives, and social interaction to support user motivation in the long-term process of learning English.

The objectives of the master's qualification thesis are:

In the first chapter, an analysis of the theoretical aspects of motivation in online learning is conducted, including the role of gamification, chatbots, and the specifics of the Ukrainian online education market.

The second chapter examines the principles of designing the Telegram bot LingOn, including the use of a points system, mini-games, financial incentives, and social interaction to create a motivational environment.

The third chapter describes in detail the technical architecture of the bot, including the use of the asynchronous Aiogram library, the database structure, and key functionalities such as reporting, profile personalization, and mini-game integration.

The fourth chapter identifies the prospects for the bot's development, including functional expansion, integration of new technologies, and strategies for overcoming technical and social challenges.

The relevance of the research is driven by the necessity to create innovative tools that support motivation in the language learning process, particularly in an online format. The Telegram bot LingOn meets the modern needs of the educational sector, offering an interactive environment tailored to the specifics of the Ukrainian online learning market.

The research results include the development of a Telegram bot that ensures regular user engagement, maintains their interest, and facilitates the achievement of long-term goals in learning English.

The practical novelty of the thesis lies in the implementation of a motivational system that combines elements of gamification, financial incentives, and social interaction. The LingOn bot adopts a user-centered approach, providing personalized learning and interactivity.

The obtained results include the development of a Telegram bot that ensures regular user participation, maintains their engagement, and promotes the achievement of long-term goals in learning English.

KEYWORDS: MOTIVATION, GAMIFICATION, TELEGRAM BOT,

ONLINE LEARNING, ENGLISH LANGUAGE, PERSONALIZATION, MINI-GAMES, FINANCIAL INCENTIVES, SOCIAL INTERACTION.

ЗМІСТ

ВТУП

У сучасному світі, де знання іноземних мов є ключовим фактором успіху, вивчення англійської мови стає не лише актуальним, а й життєво необхідним для інтеграції у глобальний соціум та професійний розвиток. Проте, досягнення успіху у вивченні мови вимагає не лише якісних навчальних матеріалів, але й високого рівня мотивації. Мотивація є важливим фактором, який сприяє тривалому навчанню, подоланню труднощів та досягненню поставлених цілей.

Розвиток інформаційних технологій відкрив нові можливості для вдосконалення навчальних методик, серед яких важливе місце займають чат-боти. Завдяки доступності платформ, таких як Telegram, використання ботів для освіти стає дедалі популярнішим. Вони забезпечують інтерактивність, персоналізацію та гнучкість навчання, адаптуючи підхід до потреб кожного користувача. Це дозволяє долати бар'єри традиційних методів навчання та мотивувати користувачів до регулярної активності.

Telegram-бот LingOn, який став основою цього дослідження, розроблений для підтримки мотивації у вивченні англійської мови. Основна ідея бота полягає у впровадженні гейміфікації, інтеграції соціальної взаємодії, а також фінансових стимулів для створення захоплюючого навчального середовища. Використання інноваційних елементів, таких як персонажі, досягнення, міні-ігри та системи звітності, дозволяє не лише забезпечити цікавість до навчального процесу, але й підтримувати регулярність занять.

Вибір Telegram як платформи для реалізації LingOn зумовлений його популярністю серед українських користувачів, а також широким функціоналом, який дозволяє створювати зручні та ефективні навчальні рішення.

Магістерська кваліфікаційна робота спрямована на дослідження теоретичних основ мотивації в навчанні та розробку Telegram-бота для

підтримки мотивації у вивченні англійської мови. У ході роботи проведено аналіз наукових підходів до гейміфікації, фінансових стимулів і соціальної взаємодії, а також розглянуто перспективи розвитку функціоналу бота. Дослідження сприятиме кращому розумінню факторів мотивації та їхнього впливу на ефективність онлайн-навчання.

РОЗДІЛ 1. АНАЛІЗ ТЕОРЕТИЧНИХ ПІДХОДІВ ДО МОТИВАЦІЇ В ОНЛАЙН-НАВЧАННІ

Мотивація є складним і багатогранним процесом, який спрямовує, підтримує та активує цілеспрямовану поведінку. Вона є рушійною силою, що підштовхує людину до дії, будь то для задоволення базових потреб чи для досягнення більш високих цілей. У контексті освіти мотивація відіграє центральну роль, сприяючи зусиллям, наполегливості та ставленню учня до навчання. Як стверджують Шунк, Пінтріх і Міс (2008), мотивація — це «процеси, які ініціюють і підтримують цілеспрямовану діяльність» [1]. Її вплив охоплює як початкове прийняття рішення про навчання, так і тривале залучення до цього процесу.

1.1. Основні компоненти мотивації

Мотивація в освіті охоплює низку психологічних процесів, що визначають постановку цілей, їх досягнення та адаптацію до викликів. Вона складається з трьох ключових компонентів:

Активація — рішення розпочати навчальну діяльність.

Інтенсивність — рівень зусиль, прикладених для виконання завдання.

Наполегливість — здатність продовжувати навчання, попри труднощі.

У процесі вивчення мов ці компоненти набувають особливого значення. Наприклад, активація може проявлятися як перше рішення вивчати нову мову, інтенсивність — як кількість часу, витраченого на практику, а наполегливість — як здатність долати труднощі протягом тривалого періоду.

У педагогічній психології мотивацію поділяють на два основні типи: внутрішню та зовнішню. Вони відрізняються за джерелом стимулів і впливу.

Внутрішня мотивація виникає з інтересу та задоволення від процесу навчання. Учні, мотивовані внутрішньо, насолоджуються самою діяльністю, що робить навчання більш приємним і тривалим. Теорія самодетермінації

(Десі та Райан) підкреслює, що внутрішня мотивація залежить від задоволення трьох ключових психологічних потреб:

Автономія — відчуття контролю над навчанням.

Компетентність — віра у власну здатність досягати успіху.

Спорідненість — зв’язок із іншими, який створює відчуття приналежності [2].

Наприклад, студент може вивчати англійську мову через інтерес до культури чи задоволення від досягнення нових висот у знаннях. Мовні платформи, такі як Telegram-боти, можуть стимулювати внутрішню мотивацію через персоналізацію завдань, інтерактивні ігри та функції, що заохочують соціальну взаємодію.

На відміну від внутрішньої, зовнішня мотивація базується на досягненні результатів, які не пов’язані із задоволенням від самого процесу. Наприклад, учні можуть бути мотивовані вивчати мову для отримання сертифіката, кар’єрного зростання чи соціального визнання. Зовнішні стимули, такі як оцінки, нагороди чи конкретні цілі, забезпечують учням структуру і конкретні напрямки дій.

Хоча зовнішню мотивацію часто критикують за нестійкість, вона ефективна у формальному навчальному середовищі. Наприклад, студент може докладати значних зусиль для успішного складання іспиту, навіть якщо навчання не приносить задоволення.

Теорія самодетермінації. ТСД пояснює, як задоволення автономії, компетентності й спорідненості сприяє формуванню внутрішньої мотивації. Вона стверджує, що учні, які відчувають контроль над своїм навчанням і бачать свій прогрес, більш схильні до активної участі й наполегливості. У вивченні мови автономія проявляється через можливість вибору способу навчання, компетентність — через досяжність цілей, а спорідненість — через інтерактивні спільноти.

Теорія очікування. Ця модель, розроблена Віктором Врумом, наголошує на раціональних очікуваннях учня щодо своїх зусиль. Вона

базується на трьох аспектах:

Очікування — віра, що докладені зусилля принесуть результат.

Інструментальність — переконання, що успіх приведе до бажаних наслідків.

Цінність (валентність) — значущість очікуваного результату для учня [3].

Наприклад, учень буде більш мотивованим вивчати англійську мову, якщо він вірить, що це відкриє йому нові кар'єрні можливості або дозволить подорожувати. Освітні боти можуть підсилювати ці елементи, демонструючи практичне застосування знань, наголошуючи на досягнутому прогресі й персоналізуючи навчальні цілі.

Синергія теорій. Об'єднання внутрішньої та зовнішньої мотивації разом із теоріями самодетермінації та очікування створює потужну основу для навчального середовища. Наприклад, Telegram-боти можуть підтримувати автономію через індивідуалізовані завдання, розвивати компетентність через чітке відображення прогресу та сприяти спорідненості через інтерактивні функції. Одночасно вони можуть забезпечувати зовнішні стимули, такі як рейтинги, бали чи нагороди, які підсилюють очікування успіху.

Мотивація є багатовимірним явищем, яке вимагає поєднання внутрішніх і зовнішніх стимулів. Теорії самодетермінації та очікування дають глибоке розуміння цих процесів і мають практичне застосування у створенні сучасних освітніх платформ. Їхнє інтегрування дозволяє створити навчальне середовище, що стимулює залученість, наполегливість і задоволення, особливо у таких тривалих процесах, як вивчення мов.

1.2. Гейміфікація як інструмент підвищення мотивації

Гейміфікація стала популярним підходом в освіті, оскільки вона підвищує зацікавленість, мотивацію та відданість учнів. Інтегруючи ігрові

елементи, такі як винагороди, зворотний зв’язок, виклики та соціальна взаємодія, у навчальний процес, гейміфікація робить його більш привабливим і цікавим. Хоча вона не перетворює навчання на гру, використання ігрових механізмів створює інтерактивний досвід, який сприяє залученню, наполегливості та покращенню результатів [4].

Основні принципи гейміфікації базуються на постановці цілей, зворотному зв’язку в режимі реального часу, винагородах, соціальній взаємодії та поступовому ускладненні завдань. Цілі дають учням чіткий напрямок і структуру навчання. У гейміфікації вони часто поділяються на короткострокові, наприклад, виконання щоденних завдань, і довгострокові, такі як досягнення певного рівня володіння мовою. Це забезпечує учням відчуття досягнення та прогресу, мотивуючи їх продовжувати навчання.

Миттєвий зворотний зв’язок є важливим для підтримання мотивації. На освітніх платформах він може реалізовуватися через індикатори прогресу, значки чи оцінки. Швидкий зворотний зв’язок допомагає учням краще розуміти свій прогрес і дає їм змогу контролювати навчання, забезпечуючи почуття досягнення.

Винагороди, такі як бали, сертифікати чи таблиці лідерів, слугують потужними стимулами, які визнають зусилля, прогрес і послідовність. Вони надають учням додаткову мотивацію залишатися залученими у процес, сприяючи формуванню звички до регулярного навчання.

Соціальна взаємодія також відіграє ключову роль, оскільки гейміфікація стимулює спільне навчання через групові завдання, форуми або дружні змагання. Взаємодія з іншими створює відчуття приналежності та підтримки, що значно підсилює мотивацію учнів.

Поступове ускладнення завдань допомагає тримати учнів у зоні оптимальної складності. Наприклад, мовні платформи можуть починати з базових тем і поступово переходити до складніших завдань у міру того, як учень демонструє прогрес. Такий підхід забезпечує постійне відчуття розвитку та досягнень.

Гейміфікація трансформує навчальний процес на захопливу подорож, яка підтримує інтерес і мотивацію учнів. При правильному впровадженні її елементи створюють інтерактивне та ефективне середовище, яке забезпечує не лише миттєве задоволення, але й довготривалий розвиток.

Численні освітні платформи використовують ці принципи гейміфікації для створення мотивуючого, захоплюючого навчального середовища. Структуруючи навчання навколо ігрових елементів, ці платформи досягли неабиякого успіху в підтримці постійної залученості учнів. Серед яскравих прикладів – Duolingo, Khan Academy і Quizlet, кожна з яких розробила унікальні гейміфіковані програми для підвищення мотивації та покращення результатів навчання.

Duolingo – це відомий додаток для вивчення мов, який активно використовує гейміфікацію. Від балів і стрічок до таблиць лідерів і досягнень, Duolingo включає в себе безліч ігрових елементів, які спонукають користувачів до постійної практики [5]. Платформа нараховує бали досвіду (XP) за проходження уроків, що сприяє підвищенню рівня та розблокуванню нового контенту. Функція «смуга», яка відстежує дні практики підряд, винагороджує користувачів за регулярну участь. Функція таблиці лідерів Duolingo дозволяє користувачам порівнювати свій прогрес з друзями або іншими учнями, додаючи елемент змагання [6]. Таке поєднання короткострокових досягнень і довгострокового прогресу виявилося дуже ефективним у підтримці мотивації користувачів повертатися до додатку щодня, демонструючи силу винагород, зворотного зв'язку та соціальної взаємодії.

Академія Хана пропонує інший підхід, використовуючи бейджі, бали та відстеження прогресу з різних предметів [7]. Виконуючи вправи і досягаючи проміжних результатів, учні заробляють бали, які впливають на їхній загальний профіль, тоді як бейджики присуджуються за конкретні досягнення, такі як оволодіння навичками або підтримання високого рівня активності. Система Академії Хана ґрунтується на навчанні майстерності, що

дозволяє учням розвиватися у власному темпі [8]. Це заохочує учнів ставити і досягати індивідуальних цілей, що відповідає принципу цілепокладання. Акцент Академії Хана на поступових викликах і навчанні майстерності підсилює ідею про те, що прогрес має бути самостійним, що робить платформу високоефективною для широкого кола учнів.

Quizlet, насамперед відомий своєю системою навчання на основі флеш-карт, включає гейміфікацію за допомогою таких режимів, як «Матч» і «Гравітація», які перетворюють навчання на ігровий досвід. Режим «Матч» пропонує учням швидко підібрати терміни до визначень, тоді як «Гравітація» вводить елемент часу, де користувачі повинні дати правильну відповідь до того, як термін «впаде» з екрану [9]. Quizlet нараховує бали за швидкість і точність, заохочуючи учнів змагатися з собою та іншими. Такий підхід робить вивчення лексики і термінів одночасно цікавим і складним, використовуючи принципи зворотного зв'язку, винагороди та поступового ускладнення завдань. Ігри Quizlet пропонують учням приємний спосіб закріпити знання, роблячи навіть повторювані завдання на запам'ятовування цікавими [10].

Ці платформи ілюструють, як принципи гейміфікації – при стратегічному застосуванні – можуть трансформувати навчальний процес. Зокрема, у вивченні мов, де шлях до майстерності є довгим, гейміфікація пропонує потужний спосіб підтримувати мотивацію та сприяти послідовній практиці.

1.3. Чат-боти в освіті

Чат-боти зробили революцію в онлайн-навчанні, забезпечуючи інтерактивну та персоналізовану підтримку [11]. Ці програми на основі штучного інтелекту імітують розмову, відповідають на запитання, надають зворотний зв'язок і спрямовують учнів у їхньому навчанні. Вони стали невід'ємною частиною сучасної освіти, особливо в онлайн-середовищі, де

забезпечують послідовну та масштабовану підтримку у різних дисциплінах, таких як мови, природничі науки чи математика.

Однією з головних переваг чат-ботів є їхня здатність підвищувати доступність і мотивацію учнів. Завдяки зворотному зв'язку в режимі реального часу чат-боти допомагають уникнути втрати інтересу до навчання, відстежують прогрес і адаптуються до індивідуального темпу [12]. Наприклад, у вивченні мов чат-бот може бути співрозмовником, пропонувати граматичні вправи, словникову практику або навіть симулювати реальні розмови. Це створює безпечне середовище для практики, яке сприяє вдосконаленню навичок [13].

Окрім підтримки мотивації, чат-боти сприяють самоспрямованому навчанню, допомагаючи студентам організовувати свій розклад і дотримуватися дисципліни. Вони можуть нагадувати про майбутні уроки, оцінювати виконані завдання та вітати з досягненнями, що спонукає до послідовності й зосередженості на цілях. Їхні адаптивні функції дозволяють аналізувати сильні та слабкі сторони учня, пропонуючи персоналізований контент для вирішення індивідуальних проблем [14].

Важливою перевагою чат-ботів є їхня масштабованість. Вони можуть підтримувати тисячі учнів одночасно, виконуючи рутинні завдання, як-от відповіді на типові запитання чи перевірка простих завдань. Це розвантажує викладачів і забезпечує цілодобову підтримку [15]. Послідовність відповідей чат-ботів також допомагає уникнути варіацій, які можуть виникати у викладачів-людей, що особливо важливо для мовного навчання.

Попри переваги, використання чат-ботів має певні виклики. Обмеження у розумінні складних мовних конструкцій, сленгу чи ідіом можуть призводити до нерелевантних відповідей. Брак емоційного інтелекту означає, що боти не можуть адаптуватися до емоційного стану учня, надавати емпатичну підтримку чи заспокоювати. Крім того, питання конфіденційності даних залишається актуальним, оскільки чат-боти потребують доступу до інформації користувачів для персоналізації навчання

[16].

Незважаючи на обмеження, чат-боти продовжують відігравати важливу роль в освіті, роблячи навчання доступним, цікавим і ефективним. Їхнє впровадження вимагає балансу між технічними можливостями та етичними стандартами, щоб забезпечити максимальну користь для учнів.

1.4. Специфіка українського ринку онлайн-навчання

Попит на знання англійської мови в Україні та широке використання цифрових платформ, зокрема Telegram, підкреслюють унікальні характеристики українського ринку онлайн-навчання. Зростаюча потреба у знаннях англійської мови збігається з глобальними бізнес-вимогами та можливостями особистого розвитку, а широке розповсюдження Telegram пропонує адаптивну платформу для доставки освітнього контенту. Розуміння цієї динаміки має вирішальне значення для розробки цілеспрямованих та ефективних освітніх інструментів в українському цифровому ландшафті, що розвивається.

В Україні володіння англійською мовою все частіше розглядається як життєво важливий актив на робочому місці, в освіті та особистому розвитку. Згідно з Індексом володіння англійською мовою EF за 2020 рік, Україна посіла 44 місце зі 100 країн, що свідчить про «середній рівень володіння» серед населення [17]. Така позиція відображає зростаючу потребу в удосконаленні мовних навичок як з особистих, так і з професійних причин. Попит охоплює різні демографічні групи: студенти, молоді спеціалісти та люди старшого віку активно вивчають англійську мову для покращення кар'єрних можливостей, подорожей та доступу до міжнародних ресурсів.

Володіння англійською мовою в Україні стало ключем до конкурентних можливостей працевлаштування, особливо в умовах посилення інтеграції країни із західними ринками. Опитування Українського інституту соціології, проведене у 2021 році, показало, що понад 61%

респондентів відчувають сильну потребу у володінні англійською мовою у своєму повсякденному та професійному житті [18]. Ця потреба особливо відчутна в таких секторах, як IT, фінанси та бізнес, де англійська мова необхідна для спілкування з міжнародними клієнтами та доступу до глобальних ресурсів. Зважаючи на те, що Україна є домівкою для зростаючої IT-індустрії та численних технологічних хабів, знання англійської мови часто є обов'язковою вимогою для ролей, які передбачають співпрацю з англомовними ринками. Ця динаміка спонукає багатьох професіоналів, особливо в таких містах, як Київ, шукати мовні курси та інструменти онлайн-навчання, щоб відповідати вимогам ринку праці.

Вивчення англійської мови не обмежується лише молодими фахівцями. Опитування, проведене Київським міжнародним інститутом соціології у 2022 році, показало, що за останні п'ять років рівень вивчення англійської мови серед людей віком від 31 року і старше зріс [19]. Дорослі та навіть люди старшого віку долучаються до програм вивчення мови, усвідомлюючи, що володіння нею не лише покращує перспективи працевлаштування, а й відкриває двері до культурних та інформаційних ресурсів. Ця тенденція підкреслює попит на вивчення англійської мови в Україні з боку різних поколінь, зумовлений прагненням до самовдосконалення та бажанням брати участь у глобалізованій економіці.

Цей попит ще більше підживлюється освітніми реформами в Україні, в яких акцент робиться на вивченні іноземних мов. Школи та університети включили англійську мову до основної навчальної програми, а приватні мовні заклади поширилися, щоб задовольнити попит на спеціалізовану підготовку. Крім того, платформи онлайн-навчання пропонують гнучкі та доступні способи для студентів і працюючих дорослих покращити свої навички англійської мови за межами традиційних класів. Такі платформи, як Duolingo, Coursera та Udemy, повідомляють про високий рівень реєстрації українських користувачів, особливо на курси англійської мови, що свідчить про значний інтерес до вивчення мови за допомогою цифрових інструментів

[20][21][22]. Багато українців віддають перевагу онлайн-ресурсам через їхню гнучкість, доступність і можливість навчатися у власному темпі, що відповідає щільному графіку працюючих професіоналів.

Зі зростанням доступності інтернету в Україні цифрові рішення для вивчення мов стали основним методом покращення рівня володіння англійською мовою. Опитування, проведене Міністерством цифрової трансформації у 2021 році, показало, що 83% українців мають доступ до інтернету, і майже половина з них використовує його в освітніх цілях [23]. Цю тенденцію лише посилила нещодавня пандемія, яка збільшила залежність від цифрових ресурсів та підкреслила необхідність самостійного навчання. Таким чином, ринок онлайн-вивчення англійської мови в Україні відображає поєднання економічних прагнень, змін в освітній політиці та зростаючої доступності цифрових технологій.

Telegram набув величезної популярності в Україні, ставши однією з найпоширеніших комунікаційних платформ в країні. Ця тенденція пояснюється кількома факторами, серед яких простота використання платформи, конфіденційність даних та здатність підтримувати обмін високоякісними медіаматеріалами. До 2023 року приблизно 72% українців повідомили, що використовують Telegram як основне джерело новин та соціальної комунікації, що є значним зростанням порівняно з попередніми роками. Канали, групи та функції ботів роблять платформу універсальною, задовольняючи не лише потреби у спілкуванні, а й у поширенні контенту та інтерактивному навчанні.

Швидке впровадження Telegram як новинної та інформаційної платформи в Україні розпочалося значною мірою через політичні чинники і з часом лише посилилося. Після російського вторгнення у 2022 році Telegram став для українців критично важливим інструментом доступу до інформації про національні події, оновлення уряду та місцеві новини в режимі реального часу. У кризових ситуаціях, таких як відключення електроенергії та перебої в роботі інших каналів зв'язку, Telegram залишався доступним і надійним, що

ще більше вкоренило його в повсякденному житті українців. Така залежність від Telegram для отримання новин та оновлень спільнот позиціонує платформу як надійний простір, де також можна ефективно поширювати освітній контент.

Унікальні можливості платформи, зокрема боти та групові функції, створюють ідеальне середовище для інтерактивного навчання. Боти Telegram можуть бути запрограмовані на надання персоналізованого навчального контенту, оцінювання та відстеження прогресу, що робить їх цінним інструментом для вивчення мов. Боти в Telegram пропонують цілий ряд функцій, від простої текстової взаємодії до мультимедійних вікторин, що дозволяє отримати персоналізований і цікавий досвід, який часто важко порівняти з традиційними умовами навчання в класі. Для тих, хто вивчає мову, Telegram-боти можуть запропонувати щоденні вправи, вправи на поповнення словникового запасу та практичні тести, які роблять навчання більш доступним та адаптованим до індивідуальних потреб.

Групи та канали в Telegram також сприяють залученню до вивчення мови, стимулюючи взаємодію у спільноті. В українському контексті групи в Telegram можуть створити середовище співпраці, де учні діляться порадами, прогресом і досвідом, пропонуючи відчуття підтримки та підзвітності. Наприклад, бот для вивчення мови в Telegram може дозволити користувачам приєднатися до мовних груп, де вони можуть практикувати розмовні навички з однолітками, брати участь у дискусіях або навіть брати участь у дружньому змаганні за допомогою вікторин чи словникових завдань. Цей соціальний елемент відповідає перевагам спільного навчання багатьох українців, які схильні цінувати взаємодію з однолітками як частину свого освітнього досвіду.

Статистика свідчить, що база користувачів Telegram в Україні є різноманітною, охоплює різні вікові групи та регіони. Дослідження, проведене компанією Statista у 2021 році, показало, що понад 60% українських користувачів Telegram мешкають у таких містах, як Київ та

Харків, де попит на цифрові освітні ресурси є найвищим [24]. Однак платформа також набула значної популярності в сільській місцевості, ставши цінним ресурсом для користувачів, які не мають доступу до традиційних навчальних центрів або фізичних мовних шкіл. Доступність Telegram для різних демографічних груп та регіонів України робить його ідеальною платформою для освітніх ініціатив, зокрема для вивчення англійської мови.

Враховуючи високий попит на знання англійської мови та широке розповсюдження Telegram, використання цієї платформи для вивчення мови в Україні є дуже стратегічним вибором. Telegram пропонує функції, які відповідають принципам гейміфікації та навчання на основі спільнот, що є ефективними стратегіями для підтримки мотивації та сприяння залученості. Створюючи освітні інструменти на базі Telegram, розробники можуть враховувати унікальні вподобання та потреби українських учнів, які отримують вигоду від гнучких, інтерактивних та соціально залучених форматів.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ ТА ПРОЕКТУВАННЯ TELEGRAM-БОТА ДЛЯ ПІДТРИМКИ МОТИВАЦІЇ

2.1. Постановка задачі та цілі дослідження

Проект LingOn має на меті вирішити поширену проблему у вивченні мов: підтримання мотивації та послідовності. LingOn – це Telegram-бот, спеціально розроблений для подолання бар'єрів, з якими стикаються багато студентів, але не шляхом вивчення ефективності бота, а шляхом активного впровадження механізмів, які надихають на щоденне залучення та послідовний прогрес. Вивчення мови, особливо для тих, хто навчається самостійно, часто пов'язане з проблемами, пов'язаними з мотивацією, що заважає людям приступити до регулярної практики та відстежувати свій прогрес. LingOn націлений на вирішення цієї проблеми шляхом створення структурованого, цікавого та гейміфікованого середовища на платформі, знайомій більшості українських користувачів, з мотиваційними інструментами, які забезпечують чіткі цілі та винагороду за кожне навчальне зусилля.

Вивчення мови, особливо за допомогою самостійних методів, пов'язане з унікальними викликами. Без структури класу учням часто важко підтримувати регулярний навчальний режим, що призводить до підходу «увімкнув-вимкнув», який гальмує прогрес. LingOn прагне вирішити цю проблему, вбудовуючи мотиваційні функції безпосередньо в платформу Telegram, надаючи учням підтримку, зворотній зв'язок і заохочення, необхідні для того, щоб вивчення англійської мови стало невід'ємною частиною їхнього повсякденного життя.

Рішення використовувати Telegram як основну платформу для LingOn ґрунтується на його широкому використанні та адаптивності. Telegram є одним з найпопулярніших інструментів комунікації в Україні, а його функціональні можливості роблять його ідеальним для створення

інтерактивного навчального процесу. Через бот-інтерфейс Telegram LingOn пропонує щоденні мовні вправи, відстеження прогресу та винагороди, що імітують заохочення, які можна знайти в структурованому навчальному середовищі. Такий підхід дозволяє учням легко займатися мовною практикою в додатку, яким вони вже користуються, усуваючи потребу в додатковому програмному забезпеченні. Ця зручність має вирішальне значення для підтримання постійної активності, оскільки дослідження показали, що учні з більшою ймовірністю дотримуються програм, які легко інтегруються в їхній щоденний розпорядок дня [25].

Мотиваційна система LingOn побудована на принципах гейміфікації – стратегії, яка, як було доведено, покращує залученість учнів, роблячи процес навчання більш схожим на гру. Елементи гейміфікації, такі як бали, рівні та рейтинги, не просто для розваги; вони створюють вимірювані цілі та винагороди, які мотивують учнів долати складні моменти. Ці функції дозволяють учням візуалізувати свій прогрес у конкретний спосіб, забезпечуючи короткострокові винагороди (наприклад, бали) поряд з довгостроковими цілями (наприклад, підвищення рейтингу або досягнення нових рівнів). У традиційному класі ці мотиваційні контрольні точки можуть бути отримані від зворотного зв'язку від викладача або одногрупників. У LingOn цю роль бере на себе сам бот, створюючи мотиваційну структуру, яка сприяє самопідзвітності та відзначає кожну віху.

Ще однією ключовою особливістю LingOn є його здатність сприяти формуванню почуття спільноти серед учнів. Соціальні можливості Telegram дозволяють створювати групи, де користувачі можуть ділитися досягненнями, брати участь у групових завданнях і пропонувати підтримку. Цей соціальний компонент вирішує поширену проблему самостійного навчання - ізоляцію. Дослідження показали, що учні, які відчувають себе частиною спільноти, з більшою ймовірністю залишаються відданими своїм цілям. LingOn використовує функції чату в Telegram, щоб створити цю соціальну динаміку, роблячи навчання більш спільним і, як наслідок, більш

приємним. Створюючи спільноту навколо вивчення мови, LingOn гарантує, що користувачі відчувають підтримку, допомагаючи їм долати падіння мотивації та святкувати свій прогрес з іншими, хто поділяє подібні цілі.

Потреба в такому інструменті, як LingOn, підкреслюється зростаючим попитом на знання англійської мови в Україні. Оскільки країна стає все більш інтегрованою у світову економіку, англійська мова стає необхідною для професійного розвитку та міжнародного спілкування. Багато українців визнають цінність знань англійської мови, але їм не вистачає ресурсів або часу для традиційних занять. LingOn пропонує рішення, пропонуючи вивчення мови у гнучкому форматі, використовуючи мотиваційні методи, щоб зробити процес корисним, а не обтяжливим. У цьому контексті бот LingOn слугує як практичним освітнім ресурсом, так і щоденним мотиватором, допомагаючи учням досягати послідовного прогресу, не відчуваючи себе перевантаженими.

Основною метою проекту LingOn є створення Telegram-бота, який слугуватиме надійним мотиваційним інструментом для тих, хто вивчає англійську мову, усуваючи поширені перешкоди у самостійному вивченні мови. Завдання дослідження, викладені тут, спрямовані на те, щоб спрямувати процес розробки, гарантуючи, що функціональні можливості бота будуть тісно пов'язані з потребами та проблемами учнів.

1. Визначити мотиваційні проблеми, з якими стикаються студенти, що вивчають англійську мову самостійно, і як Telegram-бот може вирішити ці проблеми.

Ця мета зосереджена на розумінні конкретних демотиваторів, які призводять до непослідовного вивчення мови, таких як відсутність зворотного зв'язку, відсутність взаємодії з однолітками та труднощі з відстеженням прогресу. Проаналізувавши ці фактори, проект може розробити цілеспрямовані рішення, які нададуть учням необхідну підтримку. Це включає щоденні підказки для роздумів над навчальною діяльністю, відстеження послідовності навчання та системи заохочення, які створюють

основу для формування звичок.

2. Розробити структуру бота, яка ефективно включає елементи гейміфікації для сприяння безперервній взаємодії.

Інтеграція гейміфікації є центральним елементом місії LingOn, яка полягає в тому, щоб перетворити вивчення мови на більш захоплюючий процес. Проект буде зосереджений на розробці інтуїтивно зрозумілої системи нарахування балів і рівнів, яка мотивуватиме учнів регулярно взаємодіяти з ботом. Елементи гейміфікації, такі як бали, ранги та щоденні проходження, заохочують користувачів розглядати навчання як прогресивну подорож, мотивуючи їх повертатися щодня. Крім того, бот буде пропонувати візуальні підказки для прогресу, сприяючи чіткому відчуттю досягнень.

3. Впровадити систему щоденних звітів та персоналізованого зворотного зв'язку, щоб підтримувати мотивацію учнів та відстежувати їхній прогрес.

Щоденні звіти є основою мотиваційної системи LingOn. Це завдання спрямоване на створення системи, яка підтримує постійну залученість користувачів, вимагаючи від них ділитися підсумками своєї щоденної навчальної діяльності, такими як витрачений час, основні напрямки та нова лексика. Замість індивідуального зворотного зв'язку, кожен звіт публікується в груповому чаті, забезпечуючи публічний доказ зусиль кожного користувача. Така наочність сприяє підвищенню почуття відповідальності та мотивації, оскільки користувачі можуть бачити прогрес один одного і створювати профіль, який відображає їхню відданість справі, зміцнюючи позитивну культуру навчання в спільноті.

4. Розробити підхід, орієнтований на спільноту, в рамках бота, щоб сприяти формуванню почуття приналежності та підзвітності.

Використовуючи функції груп і чату в Telegram, LingOn прагне створити спільноту учнів, що підтримує один одного. Ця мета передбачає розробку соціальних функцій, які дозволяють користувачам ділитися досягненнями, брати участь у спільних завданнях і заохочувати один одного.

Завдяки груповій діяльності та соціальному визнанню LingOn підвищує мотивацію учнів, розвиваючи почуття приналежності до спільноти. Така динаміка підзвітності заохочує послідовність, оскільки користувачі з більшою ймовірністю залишатимуться залученими, коли знатимуть, що їхні однодумці також працюють над досягненням подібних цілей.

5. Запровадити щомісячну підписку, щоб підвищити прихильність користувачів та підтримати мотивацію

Особливістю LingOn є його модель підписки, яка слугує фінансовим зобов'язанням до навчання. Проект передбачає щомісячну підписку, яка надає користувачам доступ до платформи. Цей крок заохочує до постійної участі, розвиваючи почуття відповідальності, подібно до абонементу в спортзалі. Маючи підписку, користувачі з більшою ймовірністю будуть цінувати свій прогрес і залишатимуться вмотивованими до регулярної участі, зміцнюючи свою прихильність до вивчення мови.

6. Передбачити потенційні технічні проблеми і розробити бота так, щоб забезпечити високу доступність і зручність для користувача.

Успіх LingOn залежить від зручного дизайну, який безперешкодно працює в інтерфейсі Telegram. Ця мета зосереджена на передбаченні потенційних викликів, таких як управління високим трафіком користувачів, забезпечення конфіденційності даних та оптимізація продуктивності на різних пристроях. Вирішуючи ці питання на випередження, LingOn прагне надати доступний, інтуїтивно зрозумілий інструмент, на який учні можуть покладатися щодня без технічних перешкод, які можуть стримувати використання.

2.2. Концептуальна основа розробки мотиваційного функціоналу

Мотиваційна система бота LingOn побудована на двох основних стратегіях: структурованій системі балів і рейтингу та системі щоденних звітів з винагородами. Разом ці компоненти створюють цікаве та

структуроване навчальне середовище, яке відповідає теоріям мотивації, принципам гейміфікації та унікальним потребам тих, хто вивчає мову. Завдяки функціям, які заохочують щоденну взаємодію, відстежують вимірюваний прогрес і визнають досягнення, LingOn вирішує проблему підтримки мотивації в середовищі, де студенти навчаються в індивідуальному темпі. Ця система використовує ключові мотиваційні фактори, такі як винагорода за послідовні зусилля, визнання прогресу та соціальне підкріплення, щоб сприяти позитивному та продуктивному досвіду для користувачів.

Ще одним важливим компонентом мотиваційного функціоналу LingOn є система щоденних звітів та винагород. Ця система заохочує користувачів відстежувати свою навчальну діяльність, відповідаючи щодня на кілька простих запитань, таких як «Скільки часу ви витратили на навчання сьогодні?» або «Яку нову лексику ви вивчили?». Ці щоденні звіти – не лише спосіб для користувачів відстежувати свій прогрес, але й інструмент для побудови послідовної рутини. Вбудовуючи ці підказки безпосередньо в бота, LingOn допомагає користувачам набути продуктивних звичок у навчанні та сформувати почуття відповідальності за свій прогрес.

Щоденні звіти виконують кілька функцій у системі мотивації. По-перше, це вправа на рефлексію, що дозволяє користувачам оцінювати свої щоденні досягнення і критично осмислювати свої навчальні звички. Дослідження в галузі педагогічної психології підкреслюють важливість рефлексії для закріплення знань; коли учні оцінюють, що вони вивчили і скільки зусиль доклали, вони стають більш свідомими щодо свого прогресу та сфер, які потребують вдосконалення. Таке самоусвідомлення сприяє підвищенню почуття відповідальності за процес навчання, що змушує користувачів відчувати себе більш відданими своїм цілям. У LingOn щоденні звіти виступають як контрольна точка, де користувачі підтверджують свої наміри і коригують свої стратегії, що сприяє стійкій мотивації.

По-друге, щоденні звіти прив'язані до винагород, які забезпечують

користувачам негайну винагороду за їхні зусилля. Кожен заповнений звіт приносить користувачеві бонусні бали і сприяє збільшенню кількості вдалих спроб, даючи йому невелику винагороду за послідовність. Якщо користувач заповнює звіт протягом семи днів поспіль, він може отримати додаткову винагороду у вигляді балів, що зміцнює звичку до регулярної рефлексії. Така негайна винагорода за кожен звіт підвищує мотивацію, пропонуючи користувачам відчутні результати їхніх зусиль, сприяючи створенню позитивного зворотного зв'язку, де кожен день участі збільшує ймовірність подальшого залучення.

Винагороди за щоденні звіти також входять до більш широкої системи гейміфікації бота, де накопичені звіти та стрічки трансформуються у вищі рейтинги та визнання. Такий багаторівневий підхід інтегрує концепцію «мікро-винагород» у ширшу систему прогресії, гарантуючи, що користувачі відчувають постійну винагороду за невеликі досягнення, працюючи над досягненням більших цілей. Поєднуючи щоденні винагороди з довгостроковими перевагами, LingOn підтримує як короткострокову мотивацію, так і довгострокову прихильність, роблячи вивчення мови стійкою звичкою, а не швидкоплинною гонитвою.

Нарешті, щоденні звіти мають соціальний вимір в рамках орієнтованого на спільноту дизайну LingOn. Користувачі можуть ділитися основними моментами своїх звітів у групових чатах або на дошці лідерів спільноти, створюючи платформу для взаємної підтримки та заохочення. Ділячись своїми щоденними досягненнями, користувачі не лише отримують визнання, але й сприяють створенню сприятливого навчального середовища, де інших мотивує колективний прогрес. Спостереження за діяльністю інших підсилює цінність щоденних зусиль, оскільки користувачі розуміють, що вони є частиною спільноти, яка працює над досягненням схожих цілей. Цей спільний досвід розвиває почуття товариськості та відповідальності, перетворюючи вивчення мови на колективну подорож, а не на ізольоване завдання.

На додаток до балів і нагород, система звітності LingOn включає нагадування, які допомагають користувачам не відставати від графіка. Наприклад, якщо користувач не завершив свій звіт до кінця дня, бот м'яко підштовхує його, щоб заохотити поміркувати над своїм навчанням. Ці нагадування діють як підказки, які не дають користувачам занедбати свої навчальні звички, посилюючи важливість щоденного залучення до навчання. Підштовхуючи користувачів до їхньої рутини, LingOn допомагає учням не відставати від курсу, навіть у дні, коли мотивація може бути низькою.

2.3. Гейміфікаційні елементи у LingOn

Гейміфікація слугує центральною стратегією проекту LingOn для підвищення мотивації, залучення та тривалості навчання для тих, хто вивчає англійську мову. Впроваджуючи ігрові елементи, LingOn прагне зробити процес навчання корисним і приємним.

Система балів у LingOn, відома як «Лінгони», є основою гейміфікованої структури бота. Лінгони функціонують як відчутна міра зусиль і досягнень користувача, винагороджуючи його за кожну взаємодію з ботом. Такі дії, як проходження словникових тестів, надсилання щоденних звітів та участь у міні-іграх, приносять користувачам лінгони. Це постійне позитивне підкріплення допомагає сформувати звичку до взаємодії, оскільки користувачі бачать, як їхні бали зростають з кожною сесією. Лінгони створюють негайне відчуття винагороди, мотивуючи користувачів повертатися частіше і підтримувати свій навчальний режим.

Ранги та рівні додають ще один рівень мотивації, сегментуючи прогрес користувачів на етапи, які відображають довгострокову прихильність і розвиток навичок. Ранги безпосередньо пов'язані з кількістю днів, протягом яких користувач був активним у боті, що свідчить про його відданість протягом тривалого часу. Наприклад, користувач, який постійно входить в систему, може перейти від рангу «Початківець» до «Початківця», і врешті-

решт досягти вищих рангів, таких як «Профі» або «Експерт». Таке просування дає учням чіткий візуальний індикатор їхніх постійних зусиль, що підкреслює важливість послідовності.

Нижче наведено таблицю (Табл. 2.1) з назвами рангів у LingOn та кількістю активних днів поспіль, необхідних для досягнення кожного рангу.

Таблиця 2.1 Система рангів

Назва рангу	Кількість днів до досягнення
Новатор	1 день
Кріпиш	30 днів
Трудяга	90 днів
Профі	180 днів
Експерт	365 днів
Гуру	500 днів
Легенда	730 днів
Титан	1000 днів

З іншого боку, рівні визначаються накопиченням Лінгонів, що дозволяє користувачам просуватися через набір заздалегідь визначених етапів, взаємодіючи з ботом. Кожен рівень вимагає певної кількості балів для розблокування, що заохочує користувачів виконувати різні дії для просування. Досягнення нового рівня дає користувачам відчуття досягнення, підтверджуючи їхній прогрес і заохочуючи продовжувати. Вищі рівні супроводжуються додатковими винагородами, такими як значки або доступ до ексклюзивного контенту, що дає учням відчутні переваги за їхній прогрес.

Поєднуючи бали, звання та рівні, LingOn задовольняє як короткострокові, так і довгострокові мотиваційні потреби. Лінгони надають негайну винагороду за кожну взаємодію, задовольняючи потребу учня в миттєвому задоволенні. Ранги та рівні, однак, заохочують постійну взаємодію, пропонуючи проміжні етапи, які визнають послідовні зусилля.

Такий багаторівневий підхід розвиває всебічне почуття досягнення, підтримуючи мотивацію учнів у короткостроковій перспективі, а також стимулюючи довгострокову відданість справі.

Ця структура ґрунтується на психологічних теоріях, таких як теорія цілепокладання, яка стверджує, що чіткі, досяжні цілі підвищують мотивацію. Ранги та рівні LingOn діють як прогресивні цілі, кожна з яких є більш складною, ніж попередня, заохочуючи користувачів залишатися залученими. Пропонуючи різні етапи, які відповідають різним аспектам мотивації, гейміфікована система LingOn допомагає учням виробити звичку до постійної мовної практики, що в кінцевому підсумку призводить до кращого запам'ятовування та оволодіння мовою.

Вплив гейміфікації на зацікавленість і тривалість навчання добре задокументований в освітніх установах, і дизайн LingOn використовує ці переваги. Включаючи бали, ранги та рівні, бот створює середовище, де користувачі мотивовані брати участь регулярно і протягом тривалого часу. Гейміфікація не тільки заохочує початковий інтерес, але й підтримує його, пропонуючи постійні винагороди та заохочення. У LingOn гейміфікована структура гарантує, що користувачі залишатимуться активно залученими, оскільки перспектива заробляти лінгони і просуватися по кар'єрних сходах утримує їх у грі.

Одним з ключових впливів гейміфікації на залученість є створення стану «потоку» – концепції, популяризованої психологом Міхалом Чіксентміхалі [26]. У контексті вивчення мови стан потоку виникає, коли користувачі повністю занурюються в навчальний процес, втрачаючи відлік часу і відчуваючи глибоке почуття залученості. Накопичення балів і просування по рівнях LingOn заохочує цей стан потоку, надаючи досяжні завдання, які відповідають поточному рівню навичок користувача. Коли учні бачать, що їхні Lingons зростають, і очікують на наступний ранг, вони стають більш цілеспрямованими та вмотивованими, що сприяє більш тривалій та продуктивній взаємодії з мовними матеріалами.

Крім того, гейміфікація допомагає пом'якшити поширену проблему втоми учнів, яка часто призводить до високого рівня відсіву в онлайн-навчанні. Пропонуючи невеликі, часті винагороди, LingOn підтримує залученість користувачів, не перевантажуючи їх. Перспектива досягти наступного рівня або заробити новий значок дає учням відчуття мети, роблячи кожну навчальну сесію більш значущою. Дослідження в галузі поведінкової психології показують, що мозок виділяє дофамін – нейромедіатор «винагороди» – щоразу, коли людина досягає мети, навіть незначної [27]. Цей викид дофаміну підкріплює навчальну поведінку, що робить користувачів більш схильними регулярно повертатися до бота, оскільки вони підсвідомо асоціюють мовну практику з позитивними відчуттями.

Соціальні аспекти гейміфікації, такі як таблиці лідерів і визнання спільноти, ще більше підвищують зацікавленість, сприяючи формуванню почуття спільноти. LingOn включає в себе таблиці лідерів, які відображають користувачів з найвищим рейтингом, що дозволяє користувачам бачити, де вони знаходяться в порівнянні зі своїми однолітками. Таке публічне визнання прогресу задовольняє людське прагнення до соціального схвалення та визнання. Учні, які бачать себе в списку лідерів, відчувають почуття гордості, що мотивує їх утримувати свою позицію або підніматися вище. Для тих, хто ще не досяг вершини, таблиця лідерів слугує джерелом натхнення, спонукаючи їх частіше займатися, щоб досягти вищих рейтингів.

Ще одним наслідком гейміфікації є її здатність подовжувати тривалість навчання, оскільки користувачі, які відчувають себе залученими, швидше за все, братимуть участь у навчанні більш послідовно і впродовж довших періодів часу. При традиційному вивченні мови студентам може бути важко підтримувати інтерес після кількох тижнів. Однак функції гейміфікації LingOn створюють стійкий мотиваційний цикл, який продовжує залучення. Структура рангів та рівнів заохочує користувачів ставити перед собою поступові цілі, що зменшує ризик вигорання. Наприклад, учень, який

досягнув рангу «Кріпиш» через 30 днів, може бути мотивований досягти рангу «Трудяга» через 60 днів. Такий підхід розділяє процес вивчення мови на керовані етапи, кожен з яких має свої винагороди та відчуття досягнення.

Крім того, система гейміфікації LingOn надає учням негайний зворотній зв'язок про їхній прогрес, що відіграє вирішальну роль у підтримці мотивації. Коли користувачі бачать, як після виконання завдання збільшується кількість їхніх Лінгонів, вони отримують візуальне підтвердження свого прогресу. Цей цикл зворотного зв'язку посилює зв'язок між зусиллями та винагородою, даючи зрозуміти, що кожна сесія сприяє досягненню їхніх загальних цілей. Постійно демонструючи користувачам, що їхні зусилля мають значення, структура гейміфікації LingOn підтримує постійну зацікавленість, допомагаючи учням подолати тенденцію припиняти вивчення мови через відсутність відчутного прогресу.

Нарешті, поєднання балів, рангів і рівнів впливає на залученість користувачів, створюючи «мислення зростання» у спільноті LingOn. За словами психолога Керол Двек, мислення зростання – віра в те, що свої здібності можна покращити, докладаючи зусиль – призводить до більшої стійкості та наполегливості. Гейміфіковані елементи LingOn сприяють розвитку мислення зростання, розглядаючи вивчення мови як подорож з різними етапами досягнень. Користувачі, які починають з рівня «Новатор» і просуваються до вищих рівнів, засвоюють переконання, що постійні зусилля ведуть до прогресу, що зміцнює їхню прихильність до навчання.

У боті LingOn Лекситрони (Рис. 2.1) слугують унікальними колекційними аватарами, призначеними для мотивації користувачів і позначення їхнього прогресу. Кожен Лекситрон уособлює унікального персонажа, що відображає різні риси та тематики, здатні надихнути та зацікавити студентів. Після семи днів послідовної навчальної активності користувачі отримують свій перший Лекситрон як винагороду, а також можливість розблокувати інших персонажів, вирішуючи конкретні завдання або купуючи їх за зароблені Лінгони у внутрішньому магазині бота.

Лекситрони виходять за рамки простої візуалізації. Вони діють як почесні значки та відображають шлях користувача та його відданість, даючи відчуття досягнення та ігровий елемент у вивченні мови. Кожен персонаж має власну особистість та історію, що додає ще більшого занурення в навчальний процес.

Рис. 2.1. Персонажі в боті LingOn

Лекситрони відображаються в щоденних звітах і профілях користувачів, надаючи учням видиме визнання їхніх зусиль. Коли користувачі взаємодіють з LingOn і збирають більше Лекситронів, вони розвивають свою унікальну ідентичність в рамках платформи, зміцнюючи свою прихильність до вивчення мови. Кожен лекситрон слугує нагадуванням про досягнуті цілі та заохочує до нових звершень.

Використовуючи лекситронів і як символи досягнень, і як особисті мотиватори, LingOn ефективно використовує психологічні переваги гейміфікації. Персонажі створюють середовище, де навчання не тільки структуроване, але й корисне, оскільки користувачі відчувають зв'язок зі своїм прогресом через персонажів, які відображають їхній шлях та особистість.

2.4. Структура та концепція міні-гри як частини навчального процесу

Міні-гра «Nip and Branklin's Adventure» в проекті LingOn представляє захопливий і комплексний підхід до вивчення мови, розроблений для того, щоб провести користувачів через інтерактивну подорож англійською мовою. В основі цієї міні-гри лежить книга, створена за допомогою штучного інтелекту, яка містить найуживанішу лексику для повсякденного спілкування, що охоплює рівні від A1 до B2. Поєднуючи вправи на читання, аудіювання, письмо та говоріння в розділах цієї книги, міні-гра пропонує всебічний навчальний досвід, який задовольняє різноманітні мовні навички у веселому, сюжетно-орієнтованому форматі.

Сюжетна лінія розповідає про двох персонажів, Ніпа та Бранкліна, які вирушають у пригоду, сповнену викликів, що вимагають творчого підходу та вирішення проблем. У міру просування користувачі супроводжують Ніпа та Бранкліна на шляху до великих винаходів, взаємодіючи з контентом, який не лише розширює їхній словниковий запас, а й розвиває навички читання, аудіювання та продуктивного мовлення. Цей унікальний підхід до навчання має на меті сприяти довгостроковій мотивації, поєднуючи елементи сторітелінгу та вивчення мови, що робить практику англійської менш схожою на рутину, а більш схожою на захоплюючу пригоду.

Ця міні-гра включає в себе структуроване, послідовне навчання, організовуючи кожен рівень навколо глави з книги, згенерованої штучним інтелектом. Кожен розділ розділений на два основні сегменти:

1. Читання та Розуміння на слух. Перша частина кожного розділу зосереджена на розумінні письмової та усної англійської мови. Користувачі читають уривки, що розповідають про пригоди Ніпа та Бранкліна, супроводжувані аудіокомпонентами для закріплення навичок аудіювання. Читаючи та слухаючи історію, користувачі зустрічаються з високочастотною лексикою у значущих контекстах, що сприяє природному запам'ятовуванню

слів. Структура розповіді також допомагає учням асоціювати лексику з конкретними діями, емоціями та обстановкою, посилюючи запам'ятовування через контекст.

2. Активна практика письма та говоріння. Після засвоєння матеріалу першої частини користувачі переходять до активного застосування вивченого. Цей сегмент заохочує учнів використовувати нещодавно вивчену лексику та фрази, практикуючи побудову речень, реагуючи на підказки та беручи участь у керованих діалогах, що імітують реальні життєві сценарії. Ця інтерактивна практика спонукає користувачів будувати речення, висловлювати ідеї та критично ставитися до вживання слів, розвиваючи мовні навички, необхідні для ефективної комунікації.

Структура навчання розроблена таким чином, щоб бути інтуїтивно зрозумілою та захоплюючою. Вбудовуючи лексику та граматичну практику в захоплюючу сюжетну лінію, міні-гра робить процес навчання більш релевантним і таким, що запам'ятовується. Поступове просування через розділи історії гарантує, що користувачі переглядають і закріплюють лексику та мовні структури, з якими вони вже зустрічалися раніше, що сприяє кращому запам'ятовуванню матеріалу.

Міні-гра в LingOn пропонує численні мотиваційні переваги, які відповідають принципам гейміфікації, сприяючи послідовному навчанню та підтримці зацікавленості користувачів. Цей унікальний підхід використовує сторітелінг, інтерактивні вправи та чітку систему прогресування, перетворюючи вивчення англійської мови на приємний та корисний досвід.

Однією з ключових переваг міні-гри є її наративна структура, яка об'єднує мовні вправи в захоплюючу сюжетну лінію. На відміну від традиційних методів, які можуть здаватися повторюваними, пригоди Ніпа та Бранкліна розгортаються з кожним розділом, заохочуючи користувачів продовжувати навчання, слідуючи за героями у їхній подорожі. Такий підхід до розповіді використовує природний потяг людини до наративів, створюючи емоційний зв'язок, який сприяє запам'ятовуванню лексики та

граматики. Користувачі стають зацікавленими в тому, що відбувається далі, перетворюючи навчання з повторюваного завдання на безперервну подорож відкриттів.

Міні-гра також підтримує цілісний розвиток навичок, об'єднуючи вправи на читання, аудіювання, письмо та говоріння в одну інтегровану структуру. Такий всеохоплюючий підхід гарантує, що користувачі розвинуть збалансований набір мовних навичок, уникаючи поширеної помилки зосередження виключно на пасивних навичках, таких як читання, або на активних, таких як говоріння. Кожен розділ поєднує ці елементи, надаючи користувачам повний, багатогранний мовний досвід. У міру просування користувачі помічають покращення в усіх аспектах англійської мови, що підвищує мотивацію, демонструючи відчутний прогрес у своїх здібностях.

Структура міні-гри ретельно розроблена, щоб полегшити постановку цілей і відстежувати прогрес. Кожен розділ або рівень являє собою невелику, досяжну віху, що дає користувачам відчуття досягнення, коли вони завершують кожен розділ. Таке просування від розділу до розділу зміцнює прихильність користувачів, забезпечуючи регулярні винагороди та позначаючи чіткі етапи. Крім того, оскільки рівень складності поступово зростає з кожним розділом, LingOn дозволяє користувачам ставити реалістичні цілі, що є критично важливим для довгострокової мотивації. Прогрес не лише візуально представлений у грі, але й відчувається через покращене розуміння та впевнене використання мови, що поглиблює відчуття досягнення в міру того, як користувачі просуваються вперед.

Зворотній зв'язок у реальному часі є центральним елементом гри, особливо в активних практичних частинах. Якщо користувачі роблять помилки у побудові речень або розумінні підказок, бот надає м'які виправлення та пояснення, перетворюючи кожну помилку на цінну можливість для навчання. Цей механізм миттєвого зворотного зв'язку підвищує впевненість, дозволяючи користувачам зосередитися на конкретних сферах вдосконалення, не відчуваючи розчарування. Таким

чином, LingOn створює сприятливе навчальне середовище, де користувачі відчувають себе комфортно, експериментуючи з мовою, не боячись осуду чи невдачі.

Дизайн міні-гри підтримує мотивацію користувачів, поступово вводячи нову лексику та мовні структури. Такий стабільний темп запобігає когнітивному перевантаженню та посилює відчуття постійного досягнення, оскільки користувачі набувають здатності розуміти та будувати речення зі зростаючою впевненістю. Це узгоджується з теорією самовизначення, яка підкреслює, що люди з більшою ймовірністю продовжуватимуть докладати зусилля, коли вони відчувають компетентність, автономію та спорідненість. Збалансовані рівні складності та прогресивні виклики в міні-грі гарантують, що учні відчувають свою спроможність і мотивацію до розвитку.

На відміну від навчального середовища з високим рівнем стресу, міні-гра створює середовище з низьким тиском, зосереджене на приємній практиці. Формат сюжетної лінії дозволяє користувачам вивчати англійську у власному темпі, зменшуючи стрес, який часто асоціюється з традиційним навчанням. Така структура сприяє розвитку мислення, заохочуючи користувачів розглядати кожну спробу як крок до вдосконалення, а не як сценарій успіху чи невдачі. Зменшуючи страх невдачі та зосереджуючи увагу на зростанні, міні-гра підтримує розвиток стійкості та наполегливості - важливих якостей для стійкого успіху у вивченні мови.

Завдяки захоплюючій розповіді, комплексним вправам на розвиток навичок, чітким індикаторам прогресу, зворотному зв'язку в реальному часі та атмосфері підтримки, міні-гра в LingOn забезпечує потужний мотиваційний досвід. Користувачі не лише бачать, як покращуються їхні навички англійської, але й насолоджуються навчальною подорожжю, яка є водночас складною та досяжною. Ця міні-гра втілює динамічний та захоплюючий підхід до вивчення мови, перетворюючи вивчення англійської на змістовну, захоплюючу пригоду.

2.5. Фінансові стимули та модель підписки

Фінансовий аспект мотивації, хоча традиційно не виділяється в освіті, набуває все більшої популярності в цифровому навчальному просторі. У дизайні LingOn фінансові стимули відіграють унікальну роль у мотивації користувачів до досягнення своїх цілей з вивчення мови. Модель підписки LingOn передбачає фінансові зобов'язання, які заохочують користувачів залишатися залученими, створюючи відчуття підзвітності та усвідомлення цінності процесу навчання.

Модель підписки в LingOn розроблена таким чином, щоб слугувати не просто платою за доступ; вона є одночасно і шлюзом до повного освітнього досвіду бота, і мотиваційним інструментом, який сприяє залученню користувачів та їхній підзвітності. Завдяки скромній щомісячній платі LingOn пропонує користувачам доступне, але значуще фінансове зобов'язання, яке змінює їхнє сприйняття вивчення мови та взаємодії з платформою. На відміну від сервісів, де преміум-функції доступні лише для платних користувачів, підписка на LingOn надає повний доступ до захоплюючого, структурованого та приємного процесу вивчення англійської мови. Такий підхід поєднує фінансові зобов'язання з навчальним прогресом, роблячи вивчення мови невід'ємною частиною повсякденного життя користувачів.

Справжня цінність цієї моделі полягає в тому, що вона виховує почуття відповідальності. Подібно до членства в спортзалі, де фінансові зобов'язання заохочують регулярне відвідування, підписка на LingOn підкріплює навчальні цілі користувача. Щомісячна плата діє як тонке, але постійне нагадування про інвестиції, які користувачі зробили у своє особисте зростання, що, в свою чергу, посилює їхню відданість процесу навчання. Таким чином, кожна взаємодія з LingOn стає свідченням відданості користувача, де кожен завершений урок або вправа будується на цінності, отриманій від його фінансових інвестицій.

Ключовою психологічною концепцією, що лежить в основі цієї мотивації, є «ефект безповоротних витрат» [28]. У поведінковій економіці цей ефект означає, що люди з більшою ймовірністю продовжуватимуть займатися діяльністю, в яку вони вже вклали гроші або зусилля. Для підписників LingOn щомісячний платіж посилює цей ефект, нагадуючи користувачам про їхні інвестиції та заохочуючи їх залишатися залученими до платформи, зменшуючи ймовірність відмови від неї. Пов'язуючи фінансову мотивацію з навчальним досвідом, LingOn створює збалансовану мотиваційну екосистему, яка зміцнює навчальні цілі користувачів і допомагає розвинути стійкі навчальні звички, необхідні для засвоєння мови.

На додаток до підвищення почуття відповідальності, фінансові інвестиції також підвищують сприйняття цінності сервісу [29]. Коли користувачі розуміють, що пропуски занять не лише переривають їхній мовний прогрес, але й призводять до фінансових витрат, вони з більшою ймовірністю будуть брати активну участь у навчанні. Ці скромні, але ефективні інвестиції підвищують сприйняття цінності послуги, заохочуючи користувачів отримувати максимальну користь від кожного заняття і продовжувати рухатися до своїх мовних цілей. Таке сприйняття виходить за рамки грошової вартості; воно відповідає прагненню користувачів до вільного володіння мовою та особистісного зростання, сприяючи довгостроковому залученню та зміцненню мислення, орієнтованого на максимальну віддачу від інвестицій у навчання.

Крім того, підписка на LingOn узгоджується з теорією самодетермінації (SDT), яка підкреслює автономію, компетентність і зв'язок як фундаментальні компоненти мотивації. Завдяки підписці користувачі отримують доступ до комплексного середовища для вивчення мови в індивідуальному темпі, що дає їм змогу взаємодіяти з контентом у спосіб, який відповідає їхнім унікальним потребам і темпу. Така автономія розвиває почуття відповідальності та підзвітності за свій прогрес, поглиблюючи зв'язок з платформою. Свобода навчатися у власний спосіб посилює

внутрішню мотивацію користувачів, а фінансові зобов'язання зміцнюють їхнє почуття відповідальності. Таким чином, модель підписки стає інструментом, що розширює можливості, який не лише заохочує до постійного використання, але й сприяє більш осмисленому та персоналізованому навчанню.

Очевидною перевагою моделі підписки є те, що вона сприяє формуванню у користувачів почуття спільноти та мети. Хоча LingOn постійно розвивається, щоб задовольнити мінливі потреби своїх учнів, користувачі розуміють, що їхня фінансова підтримка відіграє важливу роль у підтримці та вдосконаленні платформи на благо всіх. Знаючи, що їхня підписка сприяє постійному розвитку LingOn, вони можуть відчувати сильніший зв'язок з платформою, а також спільне почуття мети з іншими учнями. Це створює цикл мотивації, який посилює не тільки індивідуальну прихильність, але й досвід спільноти, оскільки користувачі підтримують як власне зростання, так і колективні цілі спільноти LingOn.

Психологічний вплив фінансових зобов'язань також відіграє певну роль у тому, як користувачі підходять до своєї навчальної поведінки. Багато підписників ставлять особисті цілі, щоб максимізувати цінність своєї підписки, використовуючи фінансові зобов'язання як інструмент саморегуляції. Наприклад, користувач може поставити собі за мету виконувати певну кількість словникових вправ на тиждень або підтримувати постійний режим навчання. Така самопідзвітність заохочує дисциплінована та цілеспрямовану поведінку, що має важливе значення для успішного вивчення мови. Поєднуючи фінансову мотивацію з освітньою структурою, LingOn підтримує користувачів у розвитку продуктивних навчальних звичок, які сприяють тривалому успіху.

Підписка також встановлює «психологічний контракт» між LingOn та її користувачами, культивуючи лояльність і довгострокову прихильність. Коли користувачі підписуються, вони укладають неявну угоду про регулярну взаємодію з ботом, що посилює їхню лояльність до платформи. Цей

психологічний контракт додає ще один рівень мотивації, заохочуючи користувачів залишатися з LingOn навіть у періоди низького ентузіазму або зовнішніх відволікань. Таким чином, фінансові зобов'язання посилюють залученість користувачів і підвищують ймовірність стійкого прогресу, що дозволяє користувачам з часом досягти значного поліпшення своїх мовних навичок.

Більше того, дохід від підписки безпосередньо підтримує стабільність LingOn, дозволяючи постійно додавати нові функції, оновлення та навчальний контент, що робить досвід користування свіжим і захоплюючим. Користувачі, які бачать регулярні вдосконалення і додану вартість, можуть відчути нове почуття мотивації, знаючи, що їхні інвестиції безпосередньо сприяють зростанню і вдосконаленню платформи. Цей цикл інвестицій та інновацій гарантує, що LingOn залишається динамічним інструментом, який адаптується до мінливих потреб своїх користувачів. Підтримуючи стійку фінансову модель, LingOn гарантує, що користувачі залишатимуться залученими в довгостроковій перспективі, оскільки платформа розвивається паралельно з їхнім навчанням.

Окрім індивідуальної мотивації, модель підписки позитивно впливає на ширшу спільноту LingOn, дозволяючи платформі надавати покращені послуги всім користувачам [30]. Постійні реінвестиції у функції бота та освітні пропозиції створюють середовище, де користувачі отримують колективну вигоду від прихильності своїх колег. Спостереження за прогресом платформи надихає користувачів залишатися залученими, знаючи, що їхня підтримка сприяє розвитку ресурсу, який росте і вдосконалюється для всіх. Ця спільна вигода не тільки зміцнює рішучість користувачів покращувати свої мовні навички, але й посилює почуття єдності та взаємного прогресу в спільноті LingOn.

2.6. Інтеграція соціальної взаємодії у мотиваційні механізми

Соціальна взаємодія є потужною силою у підтримці мотивації, особливо в умовах самостійного навчання, коли учні можуть відчувати себе ізольованими [31]. У LingOn інтеграція залучення спільноти та взаємодії з користувачами через чати Telegram відіграє життєво важливу роль у підвищенні мотивації та створенні сприятливої навчальної атмосфери. У цьому розділі досліджується роль спільноти у підтримці мотивації та вплив прямої взаємодії з користувачами через групові чати на платформі Telegram.

Добре структурована спільнота надає учням мережу однолітків, які поділяють схожі цілі та виклики, створюючи відчуття товариськості та взаємної підтримки, що посилює прихильність до навчання. У LingOn аспект спільноти пропонує більше, ніж просто соціальну активність; вона слугує джерелом мотивації, заохочення та підзвітності. Дослідження в галузі педагогічної психології показують, що студенти, які взаємодіють з однолітками під час навчання, з більшою ймовірністю будуть продовжувати навчання, оскільки вони отримують заохочення від спільного досвіду і відчувають себе менш ізольованими. Спільнота LingOn сприяє досягненню цих переваг, об'єднуючи студентів, які працюють над досягненням спільної мети: покращенням своїх навичок володіння англійською мовою.

Спільнота LingOn мотивує користувачів, розвиваючи почуття спільної мети. Вивчення мови часто може здаватися складним завданням, і учням важко помітити свій прогрес з плином часу. Завдяки участі у спільноті користувачі можуть спостерігати за зусиллями та досягненнями інших, що допомагає нормалізувати виклики, з якими вони стикаються, і зміцнює ідею про те, що наполегливість веде до вдосконалення. Наприклад, коли користувачі бачать, як інші повідомляють про послідовне щоденне навчання або досягнення певних результатів, вони відчувають мотивацію наслідувати їхній приклад, створюючи позитивний цикл взаємного заохочення. Це спільне відчуття подорожі допомагає учням залишатися відданими своїй справі, навіть коли прогрес здається повільним або складним.

Крім того, спільнота LingOn допомагає користувачам встановлювати

орієнтири для власного прогресу. Спостерігаючи за іншими, користувачі можуть краще оцінити, як виглядають послідовні зусилля, і прагнути до подібних рівнів залучення. Ця динаміка відома як «соціальне порівняння» - психологічний процес, коли люди оцінюють себе на основі поведінки інших. У LingOn соціальне порівняння працює позитивно, надаючи користувачам приклади відданості та послідовності, заохочуючи їх прагнути до подібних результатів. Коли користувачі діляться своїми досягненнями, такими як перехід на новий рівень або засвоєння складного списку слів, вони сприяють створенню мотивуючої атмосфери, яка святкує досягнення кожного учня.

Підтримка громади також відіграє важливу роль у подоланні труднощів і невдач. Вивчення мови рідко буває лінійним процесом; користувачі часто переживають плато або моменти розчарування. У такі моменти участь у спільноті може надати учням заохочення та пораду. Спільнота LingOn дозволяє користувачам ділитися своїми труднощами та отримувати настанови або мотивацію від інших, хто зіткнувся з подібними труднощами. Таке колективне вирішення проблем не тільки допомагає користувачам долати перешкоди, але й зміцнює цінність наполегливості. Таким чином, спільнота LingOn стає джерелом життєстійкості, де учні черпають сили від своїх однолітків, щоб подолати труднощі.

Крім того, спільнота надає платформу для здорової конкуренції, яка може бути значним мотиватором для багатьох користувачів. LingOn включає в себе таблиці лідерів і функції обміну прогресом, які дозволяють користувачам порівнювати свої досягнення, створюючи дружнє конкурентне середовище. Ця конкуренція може стимулювати користувачів збільшувати свої зусилля, оскільки вони мотивовані бажанням покращити своє становище в спільноті. Хоча конкуренція не є єдиним мотиваційним фактором, вона додає цікавий елемент, який приваблює користувачів, що прагнуть досягнень і визнання.

Телеграм-чати слугують центральною платформою для взаємодії користувачів у LingOn, забезпечуючи простір, де студенти можуть

спілкуватися, співпрацювати та взаємодіяти один з одним. Ця взаємодія має вирішальне значення для підтримки мотивації користувачів, оскільки вона створює жваве навчальне середовище, де користувачі відчувають себе частиною колективного досвіду. Telegram, як знайома і зручна платформа, ідеально підходить для такої взаємодії, оскільки дозволяє спілкуватися в режимі реального часу, обмінюватися мультимедійними матеріалами і мати легкий доступ як до групових, так і до приватних чатів.

Групові чати Telegram у LingOn дозволяють користувачам брати участь у спільному навчанні, обговорюючи свій досвід, обмінюючись порадами та ставлячи запитання. Наприклад, користувач, який має труднощі з певним граматичним правилом або словниковим запасом, може поставити запитання в чаті, отримавши відповіді та поради від колег, які, можливо, вже подолали подібні труднощі. Такий обмін знаннями сприяє формуванню почуття причетності до спільноти, де користувачі не лише отримують підтримку від інших, але й роблять свій внесок, допомагаючи своїм одноліткам. Така взаємна взаємодія посилює почуття приналежності та створює середовище, де кожен відчуває, що може запропонувати щось цінне.

Наявність взаємодії в реальному часі в чатах Telegram також сприяє рівню безпосередності та залученості, якого часто бракує асинхронному навчанню. Користувачі можуть долучатися до дискусій, відповідати на підказки або брати участь у реальних випробуваннях, що додає цікавості навчальному процесу. Ця безпосередність створює більш динамічне навчальне середовище, де користувачів заохочують регулярно реєструватися та брати активну участь. LingOn використовує цю динаміку, організовуючи заходи на основі чату, такі як словникові завдання або тематичні дискусії, які спонукають користувачів взаємодіяти з мовою в неформальній, спільній обстановці.

Окрім групових дискусій, Telegram дозволяє користувачам формувати менші підгрупи або спілкуватися в чатах один на один, що забезпечує більш сфокусовану та персоналізовану взаємодію. Така взаємодія в малих групах

дає можливість користувачам спілкуватися на основі спільних навчальних цілей, рівнів володіння мовою або конкретних інтересів, таких як розмовна практика чи граматика. Пропонуючи як групові, так і індивідуальні варіанти взаємодії, LingOn задовольняє різноманітні соціальні потреби своїх користувачів, дозволяючи учням знайти той вид взаємодії, який найкраще підтримує їхню мотивацію та стиль навчання.

Більше того, взаємодія користувачів через чати Telegram додає ще один рівень відповідальності, оскільки користувачі, які регулярно беруть участь у чатах, виробляють рутину участі. Спільна природа чату мотивує користувачів до постійної участі в ньому, оскільки вони відчувають почуття відповідальності перед своїми колегами. Ця відповідальність зміцнює звичку до щоденної практики, яка є важливою для збереження мови та прогресу в її вивченні. Наприклад, користувачі, які беруть участь у щотижневих сесіях обміну звітами, можуть відчувати більшу мотивацію до навчання, знаючи, що вони поділяться своїм прогресом зі спільнотою.

Взаємодія в чатах Telegram також сприяє культурному обміну, оскільки користувачі з різним досвідом об'єднуються, щоб вивчати англійську мову. Таке розмаїття збагачує навчальний процес, оскільки користувачі знайомляться з різними поглядами та культурними контекстами, що покращує їхнє розуміння мови на практиці та в культурному контексті. Вивчення мови - це не лише засвоєння лексики та граматики; воно також передбачає розуміння нюансів, ідіоматичних виразів та культурних особливостей. Взаємодіючи з різноманітною групою учнів, користувачі LingOn можуть розширити свої мовні навички у спосіб, що виходить за рамки формальних уроків, роблячи свій навчальний досвід більш цілісним і значущим.

Нарешті, чати в Telegram дозволяють адміністраторам і фасилітаторам LingOn безпосередньо взаємодіяти з користувачами, надаючи їм рекомендації, зворотний зв'язок і мотивацію. Присутність модераторів або мовних експертів підвищує довіру до чату і дає користувачам впевненість у

тому, що вони є частиною добре організованої та підтримуючої навчальної платформи. Ці фасилітатори також можуть ініціювати дискусії, ставити запитання та підкреслювати важливі досягнення спільноти, додатково мотивуючи користувачів, визнаючи їхній прогрес і внесок. Таке активне залучення фасилітаторів гарантує, що користувачі відчувають підтримку не лише з боку своїх колег, а й з боку самої платформи.

РОЗДІЛ 3. РОЗРОБКА ТЕХНІЧНОЇ ТА ФУНКЦІОНАЛЬНОЇ АРХІТЕКТУРИ LINGON

3.1. Архітектура бекенду

Бот LingOn використовує Aiogram, бібліотеку Python, що побудована на принципах асинхронного програмування та дозволяє боту обробляти кілька запитів користувачів одночасно. Це запобігає «зависанню» бота або затримці відповідей, навіть коли з ним взаємодіє багато користувачів одночасно. Для LingOn, де користувачі активно надсилають звіти, купують можливості та отримують доступ до навчальних матеріалів, асинхронна обробка є важливою для підтримки безперебійної роботи користувачів.

Aiogram пропонує простий та інтуїтивно зрозумілий API, який дозволяє легко кодувати різні функції бота. Він підтримує всі методи API Telegram і легко інтегрується з додатковими інструментами для розширення функціональності бота. Це спрощує реалізацію таких функцій, як надсилання повідомлень, робота з кнопками та меню, організація чатів тощо.

Також ця бібліотека дозволяє структурувати код на модулі або обробники, забезпечуючи гнучкість у додаванні та зміні функціональності. У LingOn кожна функція бота, така як реєстрація користувача, управління платежами або персоналізація профілю, інкапсульована в окремий обробник. Це призводить до організованої архітектури, яку легко підтримувати і масштабувати в міру додавання нових функцій у майбутньому.

Aiogram підтримує як веб-хуки, так і тривалі опитування для обробки подій в реальному часі в Telegram [32]. Це означає, що LingOn може швидко реагувати на всі дії користувача, від натискання кнопки до надсилання повідомлення. Швидка обробка подій підвищує ефективність бота, забезпечуючи миттєвий зворотний зв'язок, що має вирішальне значення для підтримки мотивації користувачів у навчальному середовищі.

Завдяки асинхронній та багатозадачній роботі, Aiogram не тільки

зберігає стабільність при великому обсязі запитів, але й оптимізує використання ресурсів [33]. Це особливо важливо для LingOn, який обслуговує велику кількість користувачів одночасно. Aiogram дозволяє боту бути менш ресурсоємним, зменшуючи витрати на сервер та обслуговування.

Aiogram легко інтегрується з іншими популярними бібліотеками Python, такими як asyncio для управління асинхронними завданнями та apscheduler для планування подій. У LingOn apscheduler використовується для автоматизації нагадувань, щоденних завдань та інших сповіщень користувача відповідно до його часового поясу [34]. Синергія між цими бібліотеками дозволяє боту працювати безперебійно та ефективно.

Бібліотека asyncio ще більше оптимізує взаємодію, керуючи завданнями одночасно, тоді як apscheduler обробляє заплановані завдання, такі як надсилання щоденних нагадувань або сповіщень користувачам. Планування налаштовується на часовий пояс України, що робить нагадування відповідними до місцевого часу.

Структура бота складається з різних модулів, кожен з яких виконує певні функції, наприклад:

- Реєстрація користувачів: Нові користувачі реєструються, а їхні дані ініціалізуються в базі даних.

- Звіти: Користувачі надсилають щоденні звіти, в яких детально описують свою навчальну діяльність.

- Персоналізація: Дозволяє користувачам обирати аватарки та здібності, щоб персоналізувати свій навчальний шлях.

- Заморожування прогресу: Дозволяє користувачам призупинити свій прогрес у випадках, коли вони не можуть брати активну участь, не втрачаючи своїх досягнень.

- Контроль адміністратора: Надає адміністраторам доступ до управління користувачами та налаштуваннями.

- Оплата: Керує платежами користувачів, поновленням підписки та промоакціями.

Кожна з цих функцій інкапсульована в окремі обробники, що робить код модульним, організованим і простим в обслуговуванні та розширенні.

Telegram-бот LingOn використовує структуровану реляційну базу даних (Рис. 3.1), яка відіграє центральну роль у роботі бота, зберігаючи дані користувачів, відстежуючи прогрес, керуючи покупками та реалізуючи різні функції бота. База даних побудована з використанням MySQL, обраної за її надійну продуктивність, масштабованість і широке застосування в галузі [35]. Здатність MySQL ефективно обробляти великі масиви даних та підтримка ACID гарантують цілісність та узгодженість даних, що є критично важливими для відстеження прогресу користувачів та управління фінансовими транзакціями [36]. Крім того, сумісність MySQL з бібліотеками Python спрощує інтеграцію з бекендом бота, забезпечуючи безперебійну взаємодію та оптимізоване виконання запитів [37]. Цей вибір технології баз даних посилює модульну функціональність і масштабованість системи, дозволяючи боту обслуговувати зростаючу базу користувачів, зберігаючи при цьому високу продуктивність і надійність.

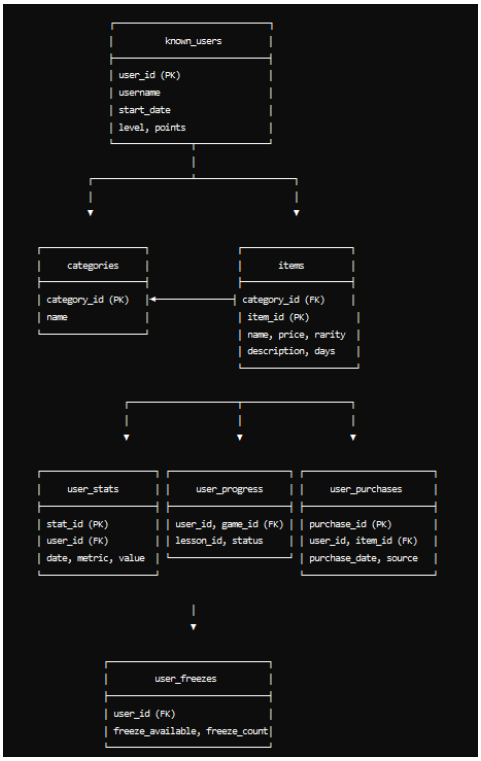

Рис. 3.1. Схема бази даних

Таблиця known_users є центральним елементом для зберігання інформації про користувачів. Кожен користувач має унікальний user_id (первинний ключ), а додаткові поля містять основні дані, такі як ім'я, дата початку використання бота, рівень та накопичені бали. Ця таблиця є базою для прив'язки всіх даних, пов'язаних із користувачем, по всій системі.

Таблиця categories відповідає за визначення різних типів предметів, доступних у боті (наприклад, аватари або здібності). Вона складається з полів для category_id і назви категорії, що допомагає систематизувати предмети та забезпечити структуру для зручного їх упорядкування.

Таблиця items містить детальну інформацію про кожен товар, який користувач може придбати або розблокувати. Кожен товар має category_id, що зв'язує його з відповідною категорією, а також додаткові атрибути, такі як item_id, назва, ціна, рідкість, опис та тривалість (якщо це релевантно). Ці дані визначають характеристики товару, його вартість та особливості.

Таблиця user_stats записує щоденні показники діяльності кожного користувача, такі як витрачений на навчання час чи певні маркери прогресу. Кожен запис містить унікальний stat_id, зовнішній ключ user_id, а також дату, тип показника та його значення. Це дозволяє ботові відстежувати індивідуальні патерни навчання користувачів, що є основою для персоналізованих рекомендацій і зворотного зв'язку.

Таблиця user_progress відображає прогрес користувачів у міні-іграх та уроках бота. Вона містить поля для user_id, game_id, lesson_id та статусу завершення, що дозволяє детально відслідковувати прогрес користувача в межах структурованих уроків та модулів контенту.

Таблиця user_purchases зберігає всі транзакції користувачів з товарами. Кожна покупка має purchase_id, а також user_id та item_id, що зв'язує покупку з користувачем і товаром. Додаткові поля включають дату покупки (date_purchase) та джерело (source), яке вказує, чи було товар придбано або розблоковано іншим способом.

Таблиця user_freezes забезпечує користувачам можливість тимчасово

«заморожувати» прогрес у боті. Вона містить поля freeze_available і freeze_count, які відстежують кількість доступних і використаних днів заморозки. Це дозволяє користувачам тимчасово призупиняти навчання без втрати прогресу чи нарахованих досягнень.

Відносини «один-до-багатьох» реалізуються у кількох місцях:

- У таблиці known_users один користувач може мати кілька записів у таблицях user_stats та user_purchases, що дозволяє зберігати детальну статистику активності та історію покупок кожного користувача.

- У таблиці categories кожна категорія (category_id) може бути пов'язана з кількома предметами у таблиці items, що забезпечує гнучкість у створенні різноманітного каталогу товарів.

Відносини «багато-до-багатьох» моделюються за допомогою відносин «один-до-багатьох»:

- Наприклад, кожен user_id у user_purchases може асоціюватися з кількома товарами, створюючи індивідуальний перелік покупок користувача.

Схема використовує первинні ключі для унікальної ідентифікації кожного запису та зовнішні ключі для підтримки реляційної цілісності, що запобігає появі некоректних або "осиротілих" записів.

3.2. Концептуальний дизайн бота та основні функціональні можливості

Реєстрація користувача. Процес реєстрації в LingOn починається з надсилання користувачем команди /start. Ця дія запускає послідовність інтерактивних запитань, спрямованих на те, щоб зрозуміти цілі користувача та готовність до участі в проєкті. Навігація здійснюється за допомогою кнопок, що забезпечує зручність користування та плавне проходження кожного етапу.

Під час реєстрації бот запитує користувача про його конкретні цілі з вивчення англійської мови та часовий проміжок, протягом якого він хоче їх досягти (Рис. 3.3). Ці дані є ключовими, оскільки вони формують основу для

персоналізованого відстеження прогресу, мотивуючи користувача дотримуватися встановлених дедлайнів.

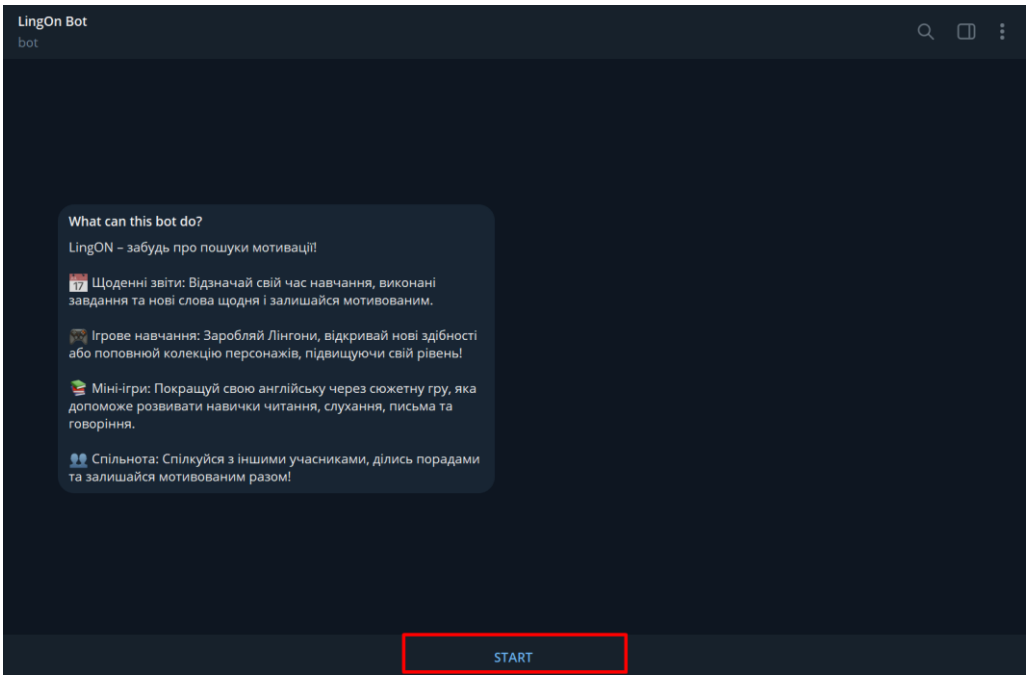

Рис. 3.2. Вітальне повідомлення

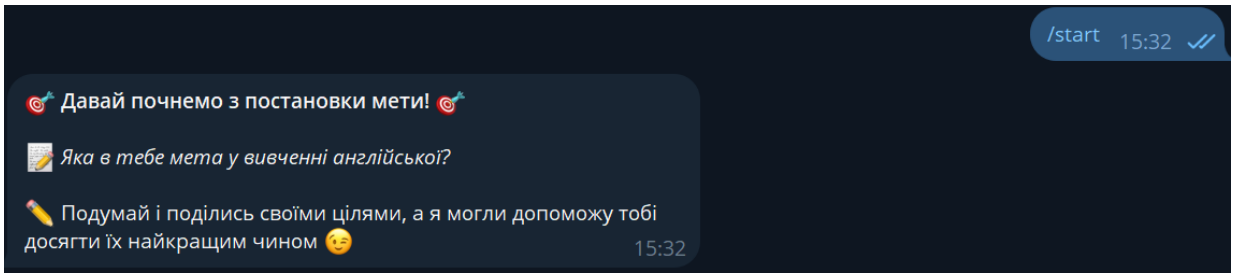

Рис. 3.3. Постановка мети

На цьому скріншоті показано підказку бота щодо навчальних цілей користувача, де він заохочується до глибокого осмислення своїх цілей. Зібрана тут інформація згодом використовується для того, щоб користувач не відставав від своєї навчальної траєкторії.

Після того, як користувач визначив свої цілі, бот надсилає запит (Рис. 3.4), щоб переконатися, що користувач уважно ознайомлений з правилами, що регулюють його участь у проєкті. Користувачам надається можливість визнати та підтвердити своє розуміння цих правил, що сприяє формуванню

чіткого почуття відповідальності та підзвітності.

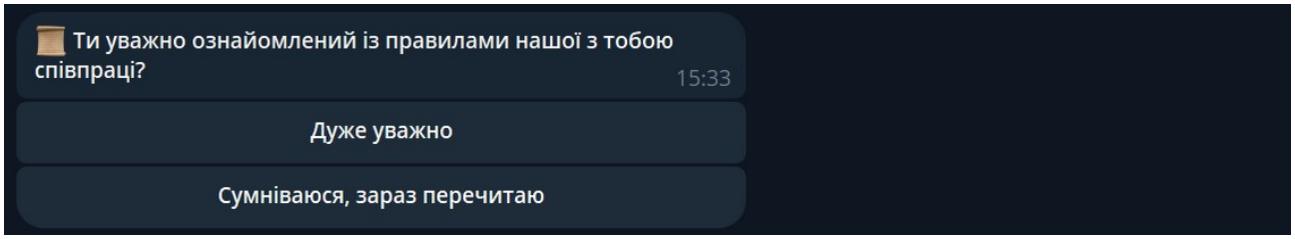

Рис. 3.4. Запит на ознайомлення з правилами

Цей крок допомагає зміцнити мислення, яке має вирішальне значення для довгострокового успіху в навчанні.

Після перегляду правил бот пропонує користувачеві підтвердити свою відданість досягненню поставлених цілей до визначеного терміну (Рис. 3.5). Цей етап функціонує як імітація «контракту», підвищуючи мотивацію користувача і роблячи його прихильність до навчання більш чіткою.

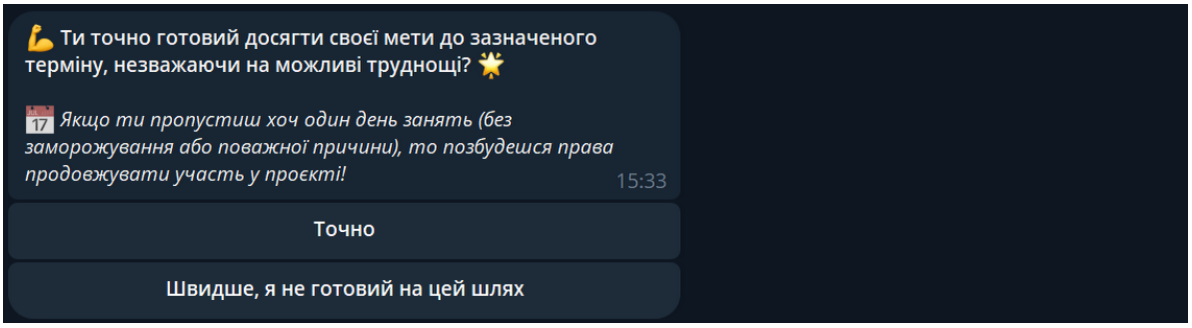

Рис. 3.5. Підтвердження готовності досягти поставленої мети

Тут користувачеві нагадують про важливість залишатися на шляху, незважаючи на потенційні виклики.

Останнім кроком реєстрації є придбання підписки, яка надає користувачеві доступ до повного функціоналу бота. На перший місяць пропонується знижка, щоб зробити рішення більш привабливим. Бот надає різні варіанти оплати, такі як криптовалюта або банківський переказ (Рис. 3.6), для зручної обробки транзакцій.

Інтуїтивний дизайн забезпечує користувачеві можливість завершити транзакцію швидко та без зайвих дій.

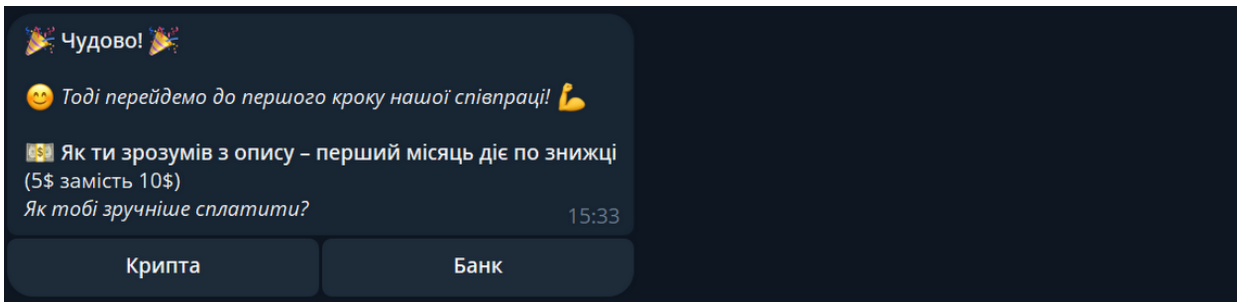

Рис. 3.6. Вибір способу оплати для підписки

Після завершення реєстрації бот публікує вітальне повідомлення в чаті спільноти разом з коротким описом навчальних цілей користувача. Це повідомлення має подвійну мету: воно мотивує користувача, роблячи його цілі видимими, і заохочує підтримку та залучення спільноти.

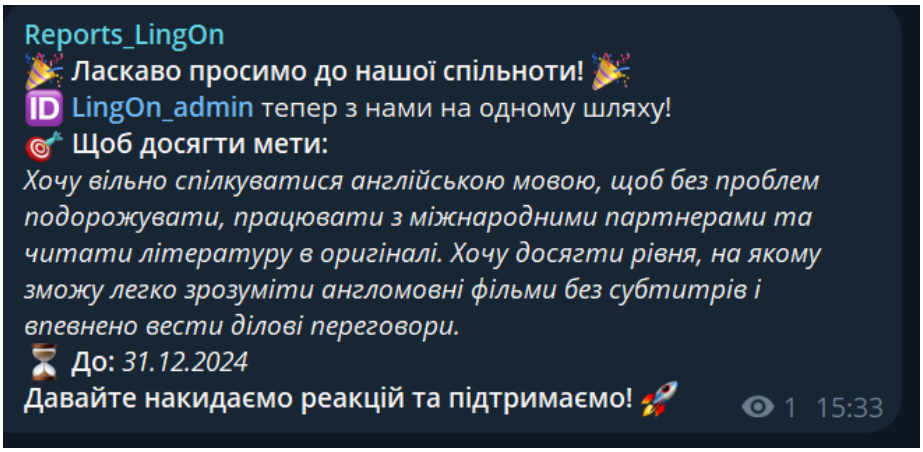

Рис. 3.7. Повідомлення про приєднання нового користувача

Публічно ділячись цілями кожного нового учасника, бот сприяє створенню сприятливого навчального середовища.

Процесс оплати. Після натискання кнопки «Оплата» в чаті з'являється детальне повідомлення з описом поточного статусу підписки (Рис. 3.8).

Це повідомлення містить:

- Статус підписки (наприклад, активна)
- Дата закінчення поточного періоду підписки
- Поточну ціну на місяць
- Активні знижки та термін їх дії

Натиснувши «Сплатити», бот ініціює етап оплати (Рис. 3.9).

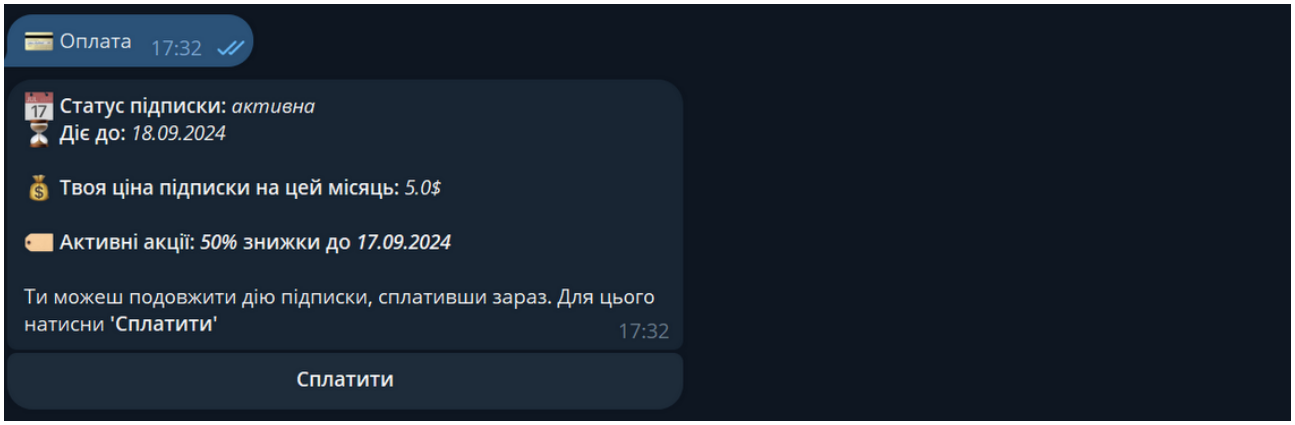

Рис. 3.8. Інтерфейс управління підпискою користувача

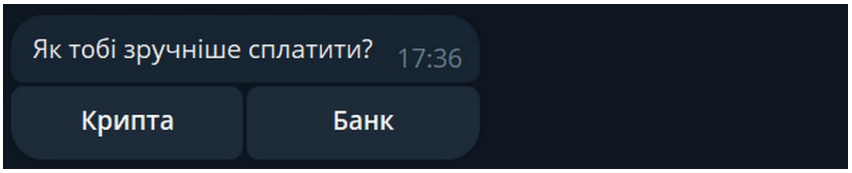

Рис. 3.9. Етап оплати підписки

Після вибору способу оплати бот показує конкретні інструкції в залежності від вибору. Наприклад, якщо ви виберете криптовалюту, він покаже суму для переказу та адресу гаманця (Рис. 3.10).

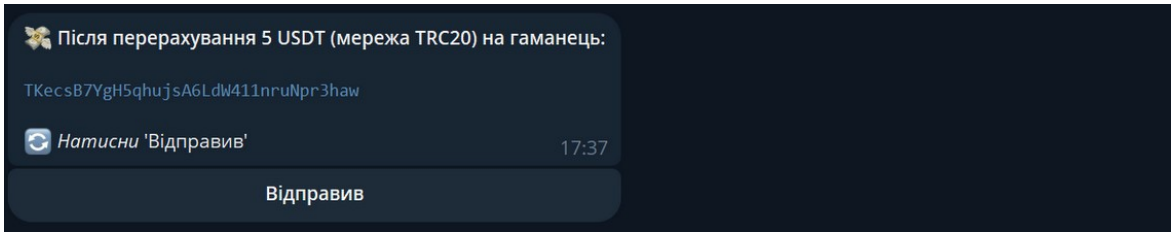

Рис. 3.10. Інструкція для оплати криптовалютою

Після завершення транзакції користувачі можуть натиснути кнопку «Відправлено», щоб повідомити бота про свій платіж.

Зазвичай підтвердження обробляється протягом години, але більшість платежів перевіряється протягом п'яти хвилин. Після підтвердження бот

оновлює статус підписки і показує повідомлення про підтвердження (Рис. 3.11).

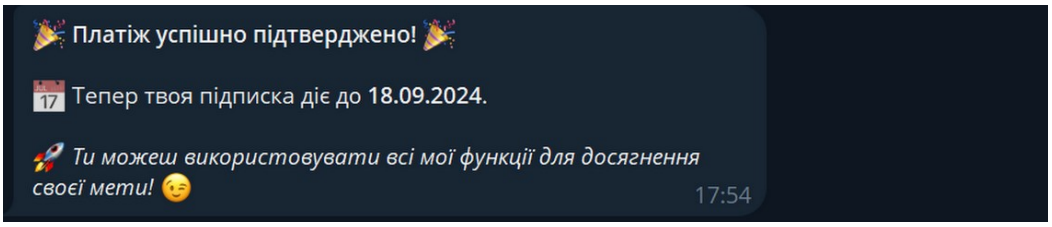

Рис. 3.11. Повідомлення про успішне підтвердження оплати

Якщо користувач обирає оплату банківським переказом, йому буде запропоновано зв'язатися з адміністратором для отримання подальших інструкцій (Рис. 3.12), що гарантує безпечну обробку конфіденційної платіжної інформації.

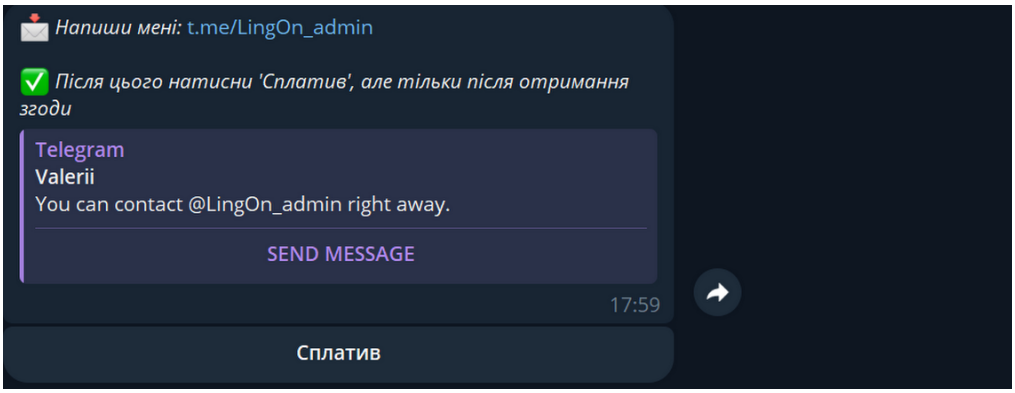

Рис. 3.12. Інструкція для оплати банківською картою

Надсилання звіту. Після завершення навчальної сесії користувач може відзвітувати про свій прогрес, натиснувши кнопку «Відправити звіт». Це ініціює серію запитань, щоб зафіксувати навчальну діяльність користувача та його самооцінку за день.

Спочатку бот запитує: «Скільки часу сьогодні пішло на вивчення?» Потім користувач вводить загальну кількість хвилин, присвячених вивченню англійської мови. Наприклад, якщо користувач займався 60 хвилин, він вводить «60» (Рис. 3.14).

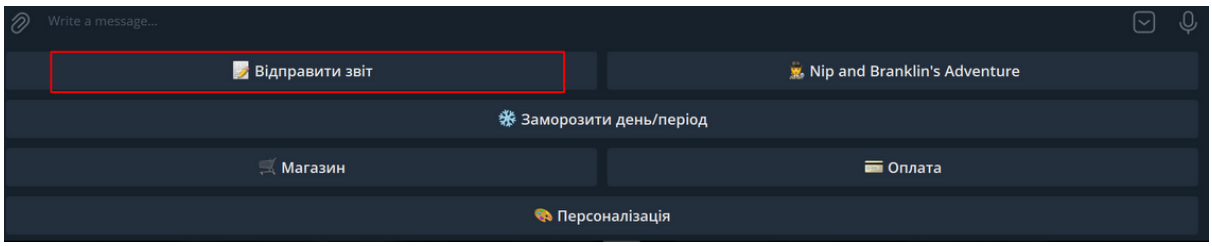

Рис. 3.13. Головне меню з кнопкою для надсилання звіту

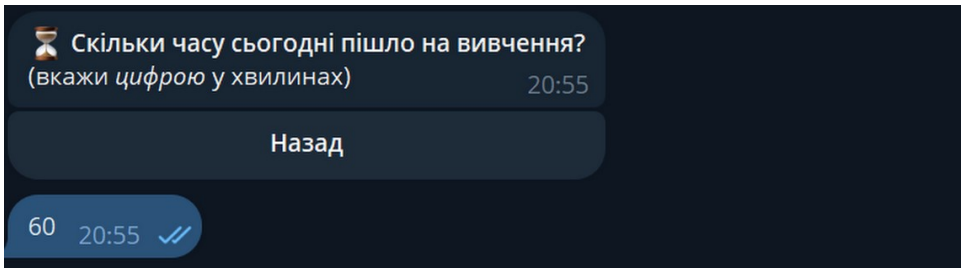

Рис. 3.14. Введення часу, витраченого на вивчення англійської мови

Далі бот пропонує користувачеві описати конкретні дії, виконані під час сесії (Рис. 3.15). Це може включати проходження рівня міні-гри, перегляд словникового запасу або практику граматики. Користувач надає короткий опис цих дій.

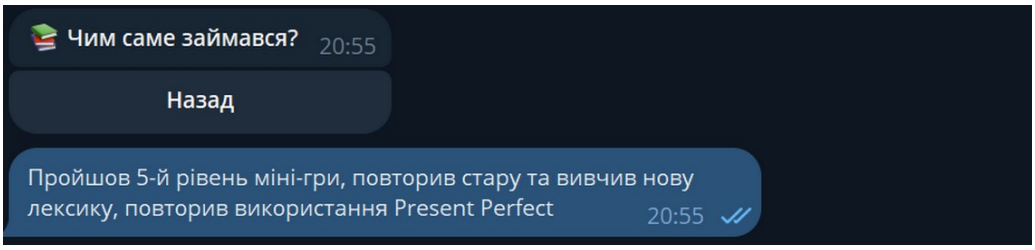

Рис. 3.15. Введення опису навчальної активності користувача

Бот запитує: «Які нові слова ви вивчили сьогодні?». Користувач перераховує новий словниковий запас, розділяючи слова комами. Цей запис допомагає відстежувати нові мовні елементи, вивчені під час сесії.

Після підбиття підсумків користувачеві пропонується оцінити свою продуктивність за шкалою від 1 до 5, натиснувши відповідну кнопку (Рис. 3.17). Ця самооцінка допомагає відстежувати щоденний прогрес і загальну ефективність.

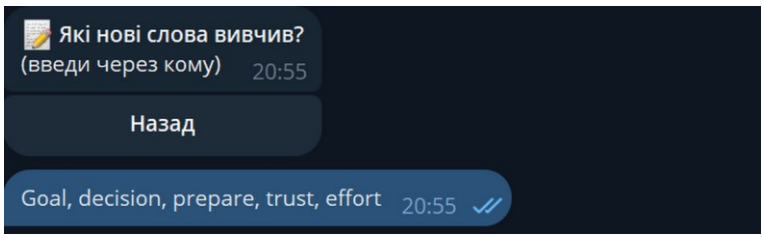

Рис. 3.16. Введення нових слів, вивчених під час навчальної сесії

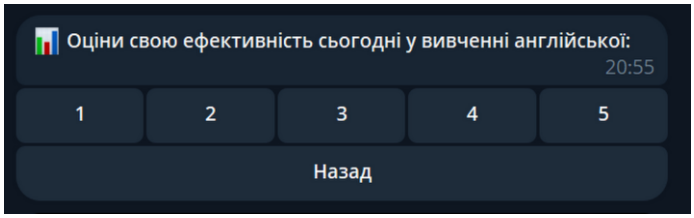

Рис. 3.17. Самооцінка ефективності навчального дня

Після заповнення всіх полів з'являється підсумок звіту, який дає користувачеві можливість або «✅ Підтвердити звіт», або відредагувати певні записи, такі як витрачений час, нові слова або рейтинг продуктивності, якщо це необхідно (Рис. 3.18).

Рис. 3.18. Підсумковий звіт

Після підтвердження звіту він публікується в чаті групи спільноти, а користувач отримує бали в якості винагороди, які сприяють підвищенню його рейтингу і мотивують до подальшої участі.

Такий процес звітування сприяє підзвітності та саморефлексії, дозволяючи користувачам ефективно відстежувати свій навчальний шлях, не припиняючи участі в проєкті.

Заморожування прогресу. Бот LingOn має практичну функцію для користувачів, яким може знадобитися тимчасово призупинити свій прогрес у навчанні. Ця функція, відома як «заморожування прогресу», дозволяє користувачам уникнути невдач у навчанні під час виконання особистих або професійних зобов'язань, які заважають їм виконувати щоденні вимоги до активності. Користувачі можуть заморозити свій прогрес на один день або обрати довший період, залежно від своїх потреб.

Користувач починає з вибору кнопки «Заморозити день/період» в головному меню бота (Рис. 3.19). Вибравши її, користувач ініціює процес тимчасового призупинення прогресу.

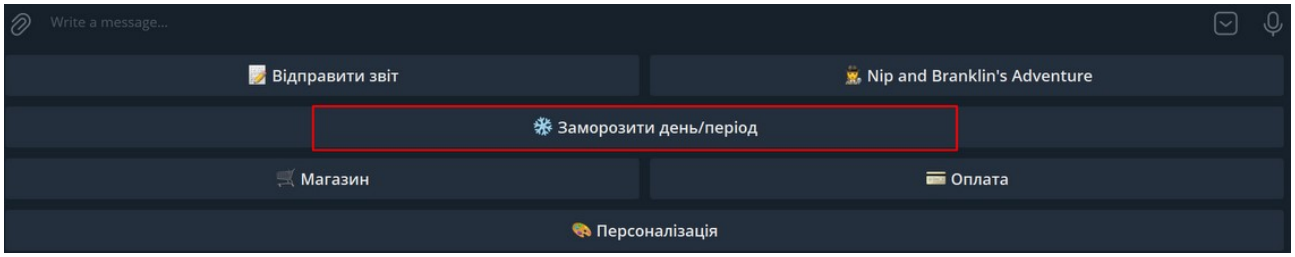

Рис. 3.19. Початок процесу заморожування

Після того, як користувач отримав доступ до опції заморожування, бот відображає повідомлення із зазначенням того, скільки днів користувач має у своєму розпорядженні для заморожування. Потім бот запитує, чи хоче користувач використати один день або довший період. Варіанти відповідей представлені у вигляді кнопок, що дозволяє користувачеві швидко зробити свій вибір (Рис. 3.20).

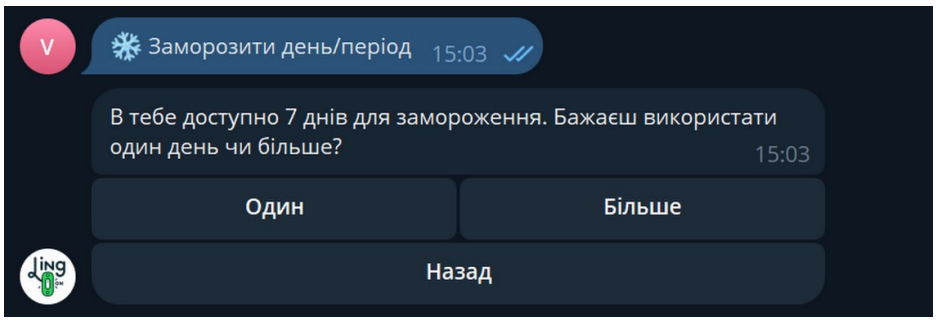

Рис. 3.20. Вибір кількості днів для заморожування прогресу

Після вибору заморозки на один день бот пропонує користувачеві вказати причину тимчасової паузи (Рис. 3.21). Цей крок передбачений для посилення підзвітності та забезпечення цілеспрямованої паузи користувача. Користувач може ввести свою причину, наприклад, «необхідність завершити робочий проект з терміновим дедлайном». Цей запис записується, щоб створити почуття відповідальності, нагадуючи користувачеві про його навчальні цілі навіть під час перерви.

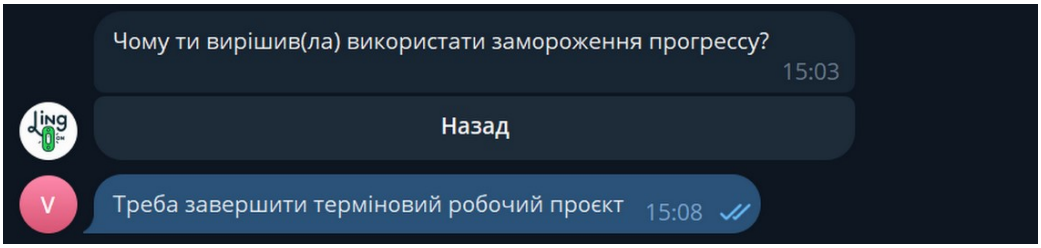

Рис. 3.21. Вказівка причини для заморожування прогресу

Після введення причини бот підтверджує, що запит користувача на заморожування був успішно оброблений. Він надає підсумкове повідомлення про те, що прогрес користувача заморожено на один день, і вказує дату, коли користувач повинен відновити звітування про свою діяльність (Рис. 3.22).

Дотримуючись цих кроків, користувач може швидко та ефективно призупинити свій прогрес на один день, не впливаючи на загальний план навчання.

Для користувачів, які планують більш тривалу паузу, бот LingOn пропонує гнучкість і можливість заморозити прогрес на кілька днів.

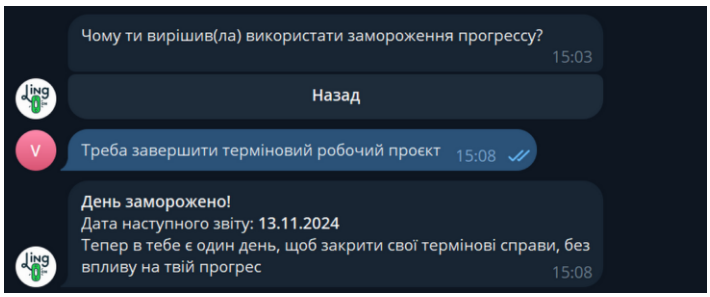

Рис. 3.22. Підтвердження заморожування прогресу

Подібно до одноденного заморожування, користувач починає з вибору кнопки «Заморозити день/період» у головному меню бота. Вибравши цю опцію, користувач обирає «Більше», щоб вказати, що йому потрібен довший період заморозки (Рис. 3.23).

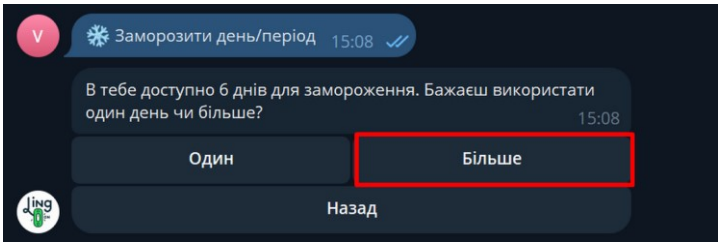

Рис. 3.23. Вибір заморожування на тривалий період

Бот запитує, чи хоче користувач використати кілька днів з наявного балансу, чи запросити певний період (Рис. 3.24). Тут користувач може вибрати «Із доступних», щоб використати певну кількість днів, або «Період», якщо йому потрібна індивідуальна тривалість, яка вимагає схвалення адміністратора.

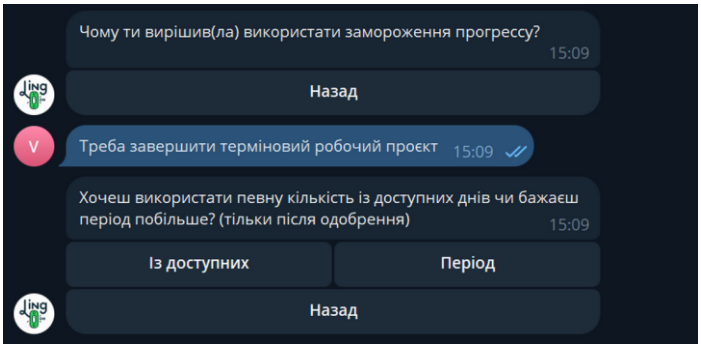

Рис. 3.24. Вибір кількості днів або періоду заморожування

Якщо користувач обирає «Із доступних», бот просить його вказати кількість днів, які він бажає заморозити. Користувач може ввести число в межах доступного балансу (Рис. 3.25). Такий підхід забезпечує гнучкість, гарантуючи при цьому, що користувач не перевищить виділену йому кількість днів заморозки.

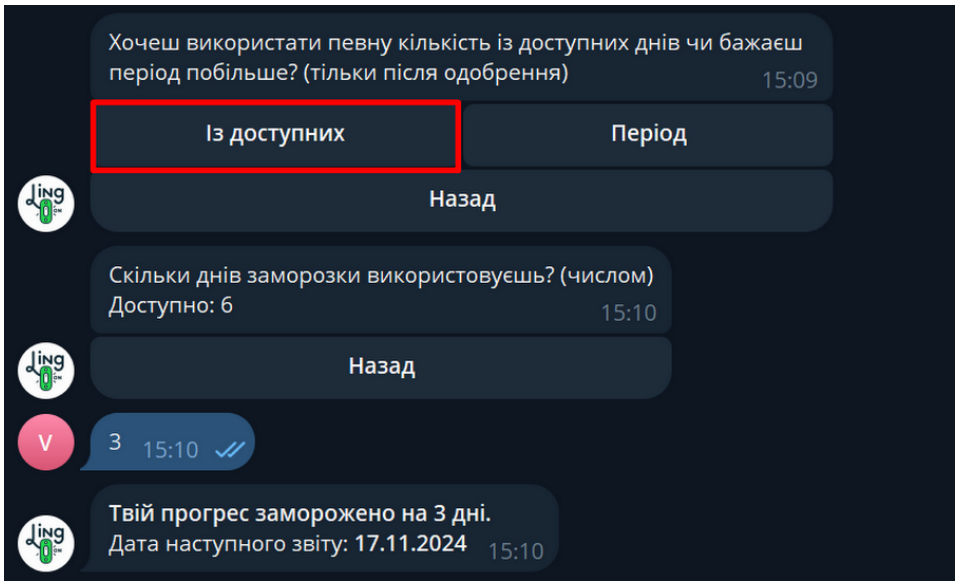

Рис. 3.25. Заморожування прогресу на обрану кількість днів

Після того, як користувач введе бажану кількість днів, бот підтвердить запит на заморозку. Він відображає повідомлення із зазначенням тривалості заморозки та нової дати, коли прогрес користувача буде відновлено. Цей крок гарантує, що користувач буде повністю проінформований про свій новий розклад, допомагаючи йому залишатися організованим і зосередженим на своєму навчальному плані.

Якщо користувач обирає «Період» для довшої паузи, бот пропонує йому зв'язатися з адміністратором проєкту для затвердження. Бот надає посилання для зв'язку, пояснюючи, що користувач повинен чітко вказати причину свого запиту на продовження паузи (Рис. 3.26). Цей додатковий крок гарантує, що тривала відсутність є виправданою, зберігаючи цілісність навчального проєкту та сприяючи підзвітності серед учасників.

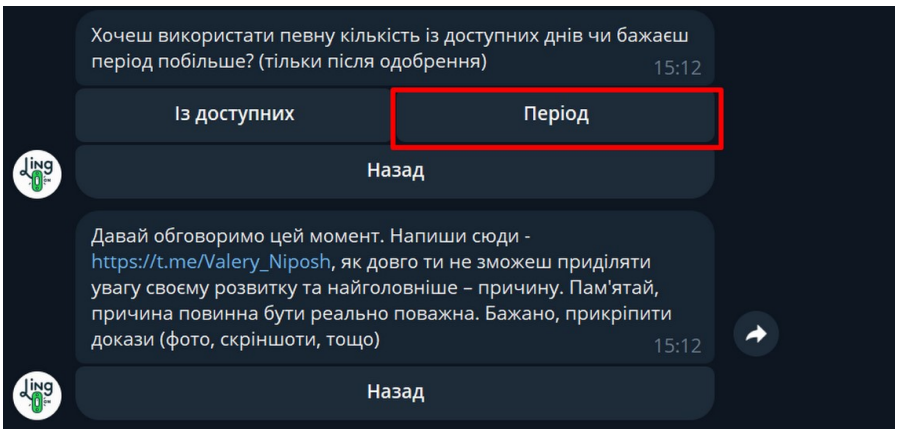

Рис. 3.26. Запит на продовження періоду заморозки

Користувач може вибрати «Період», якщо він знає, що буде зайнятий тривалими справами, наприклад, інтенсивним робочим проектом або особистою поїздкою. У таких випадках користувач повинен надати адміністратору детальну інформацію, можливо, включаючи докази, щоб підтвердити свою потребу в більш тривалій перерві.

Персоналізація профілю користувача. У LingOn персоналізація відіграє ключову роль у залученні користувачів та покращенні їхнього досвіду. Користувачі можуть налаштувати свій профіль, активуючи персонажів і вибираючи спеціальні здібності, які можна придбати у внутрішньому магазині за Лінгони відповідно до їхніх мовних цілей. Такі налаштування не лише дозволяють користувачам виразити свою індивідуальність, а й забезпечують функціональність, адаптовану до їхніх навчальних потреб.

Після вибору пункту «Персоналізація» в головному меню користувач бачить кнопки «Мої персонажі» та «Мої здібності» (Рис. 3.27). Це меню слугує відправною точкою для всіх дій з кастомізації в LingOn.

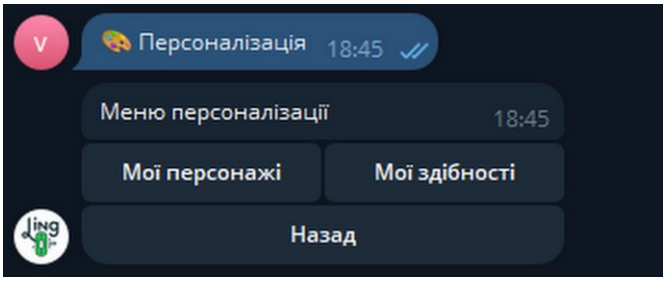

Рис. 3.27. Меню персоналізації

Вибравши «Мої персонажі», користувачеві буде запропоновано обрати персонажа зі списку (Рис. 3.28).

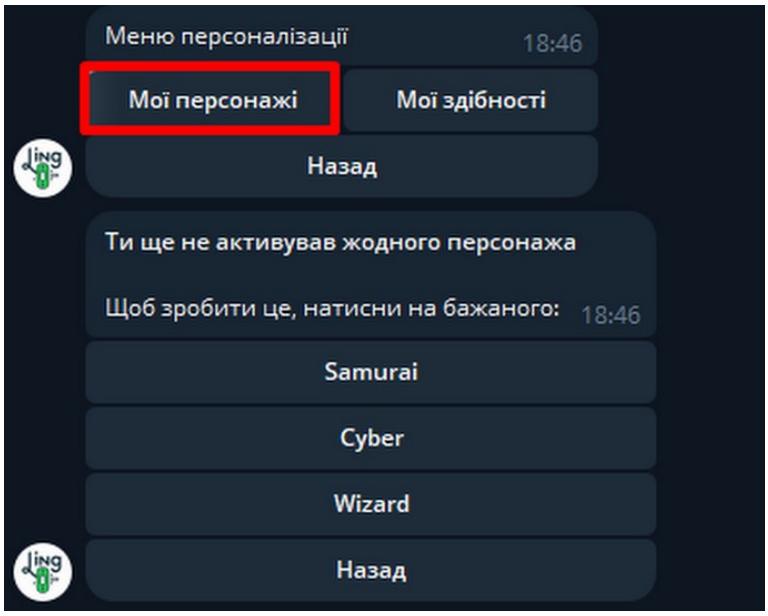

Рис. 3.28. Вибір персонажа для активації

Наприклад, користувач обирає персонажа «Cyber», після чого з'являється повідомлення з підтвердженням того, чи дійсно він хоче активувати саме цього персонажа (Рис. 3.29).

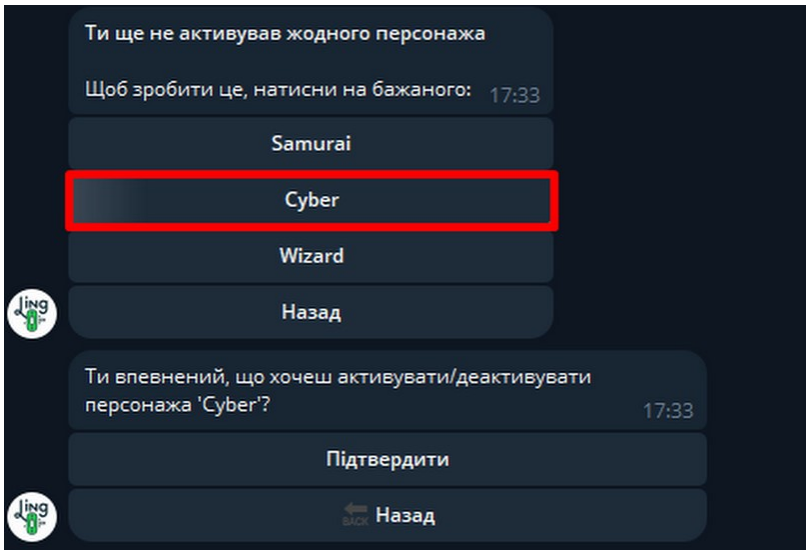

Рис. 3.29. Підтвердження активації персонажа

Бот запитує підтвердження, відображаючи дві кнопки: «Підтвердити» і «Назад». Натиснувши «Підтвердити», користувач завершує процес активації (Рис. 3.30).

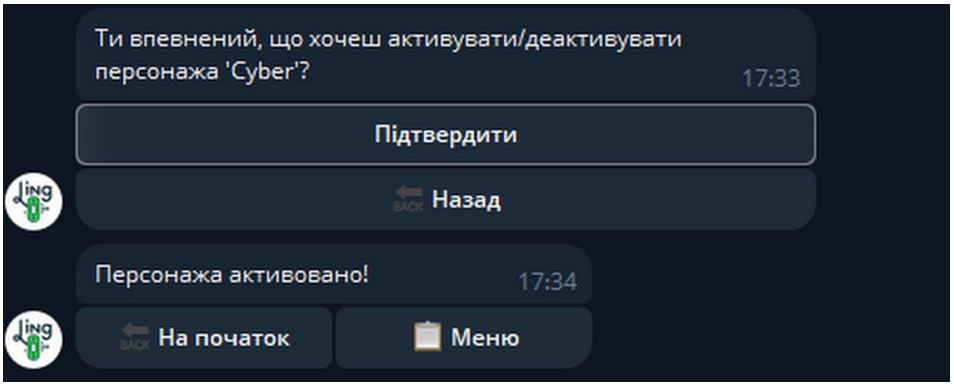

Рис. 3.30. Завершення активації персонажа

Після успішної активації бот відповідає повідомленням, яке підтверджує, що персонаж «Cyber» тепер активний. Тепер цей персонаж буде з'являтися у щоденних звітах користувача, персоналізованого акценту до його досягнень.

Повернувшись до меню «Персоналізація», користувач може обрати «Мої здібності». Тут йому надається можливість переглянути список доступних здібностей, щоб покращити свій навчальний процес (Рис. 3.31).

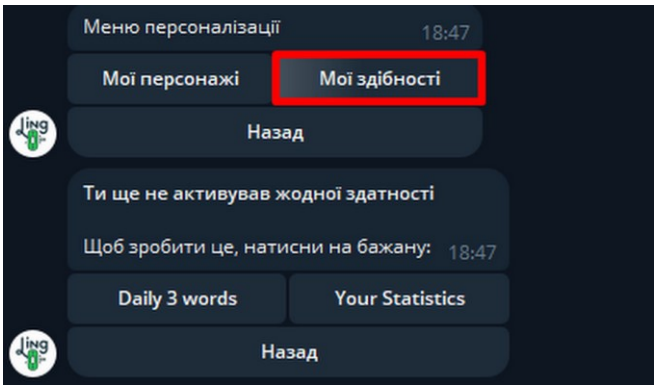

Рис. 3.31. Вибір пункту «Мої здібності»

Кожна функція надає унікальні переваги і призначена для підтримки конкретних навчальних цілей.

Якщо користувач обирає «Щоденні 3 слова», відображається детальний опис цієї функції (Рис. 3.32). Ця функція допомагає користувачеві розширювати свій словниковий запас, вводячи три нових слова щодня з визначеннями, прикладами речень та синонімами/антонімами.

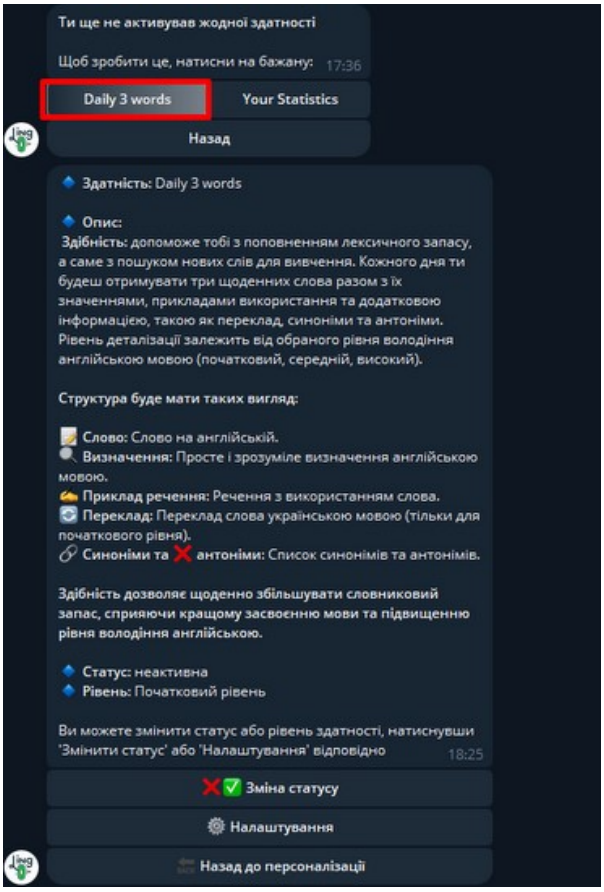

Рис. 3.32. Опис функції «Щоденні 3 слова»

Користувач може налаштувати параметри здібності, вибравши «Налаштування». Це дозволяє обрати рівень складності для щоденного словникового запасу відповідно до рівня володіння мовою, від базового до просунутого (Рис. 3.33).

Бот пропонує користувачеві три рівні на вибір:

- Початковий (A1-A2): Базові визначення з перекладом.

- Середній (B1-B2): Визначення без перекладу.

- Просунутий (C1-C2): Складні визначення без перекладу, призначені для студентів з високим рівнем володіння мовою.

Якщо, наприклад, буде обрано «Середній», то бот підтверджує вибір, оновлюючи налаштування (Рис. 3.34).

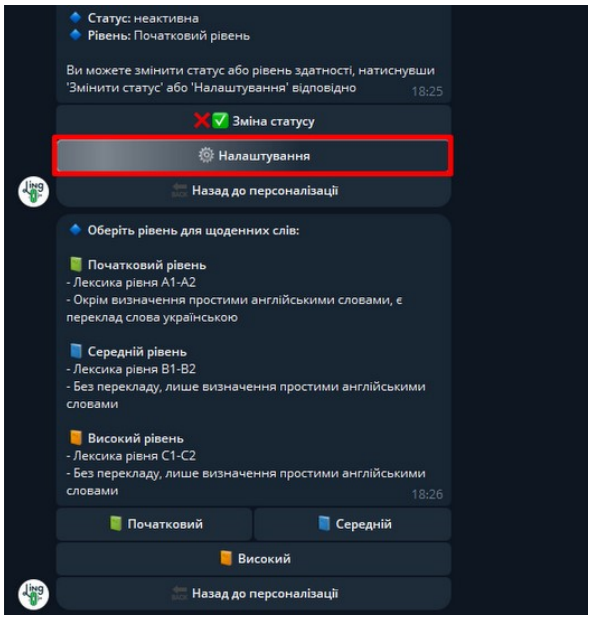

Рис. 3.33. Меню налаштувань здібності «Щоденні 3 слова»

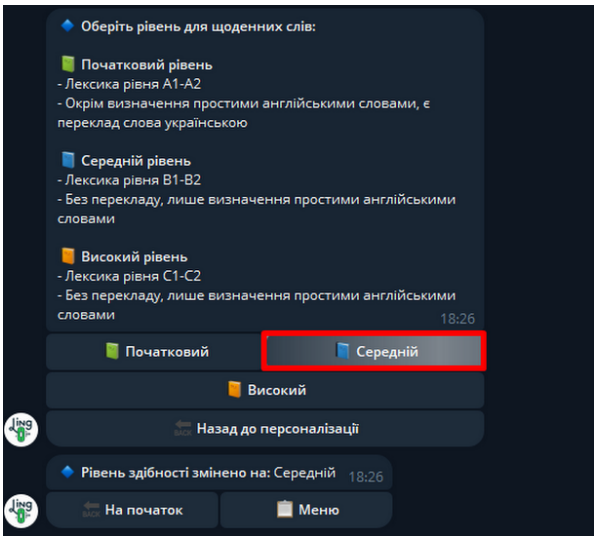

Рис. 3.34. Вибір рівня складності здібності «Щоденні 3 слова»

Завдяки цим можливостям налаштування LingOn пропонує користувачам індивідуальний підхід, який підлаштовується під їхні навчальні вподобання та робить процес вивчення більш захопливим. Можливість обирати персонажів і налаштовувати їхні здібності не лише підтримує цілі користувачів, а й робить навчання цікавим та інтерактивним.

Внутрішній магазин. Внутрішній магазин бота пропонує користувачам спрощений спосіб персоналізувати свій навчальний досвід, обираючи персонажів (Лекситронів), набуваючи здібностей і використовуючи опції заморожування прогресу.

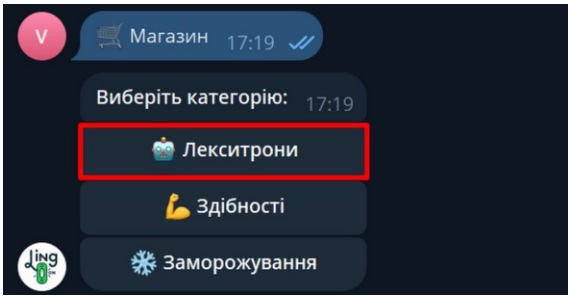

Рис. 3.35. Вибір категорії «Лекситрони»

Після натискання на категорію «Лекситрони» (Рис. 3.35), користувачі знайомляться з низкою аватарів персонажів, які можна придбати та активувати (Рис. 3.36). Кожен персонаж, або Лекситрон, являє собою унікальну аватарку, яку користувачі можуть додати до свого профілю, додаючи нотку персоналізації до своєї навчальної подорожі. Лекситрони мають різні імена та стилі, кожен з яких має власну індивідуальність та візуальну привабливість.

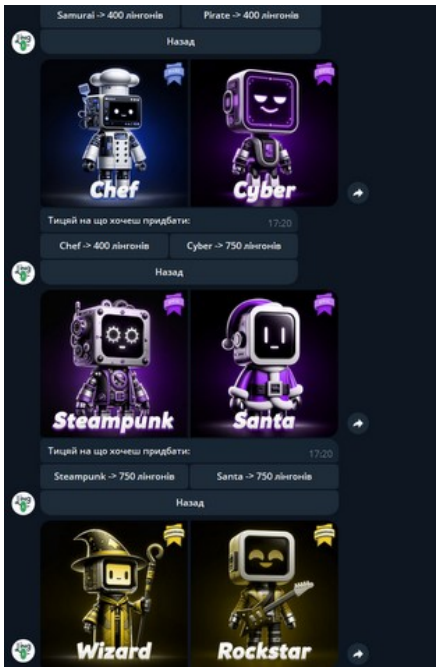

Рис. 3.36. Перелік доступних Лекситронів

Коли користувач обирає певного Лекситрона, відкривається детальний перегляд, що містить таку інформацію, як ім'я персонажа, опис, ціна та рівень рідкісності (наприклад, легендарний) (Рис. 3.37). Ця інформація дозволяє користувачам приймати обґрунтовані рішення на основі своїх уподобань та наявних Лінгонах (валюта проєкта).

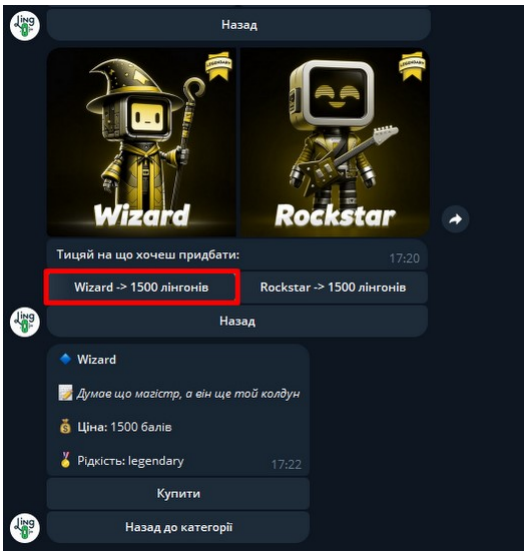

Рис. 3.37. Деталі обраного Лекситрона

Після того, як користувач вирішив придбати Лекситрона, він може перейти до покупки, натиснувши кнопку «Купити», після чого з'явиться повідомлення з підтвердженням, щоб переконатися, що він готовий завершити транзакцію (Рис. 3.38).

Рис. 3.38. Підтвердження покупки Лекситрона

Після підтвердження покупки з'являється повідомлення про успіх, яке вказує на те, що новий персонаж тепер доступний у розділі «Персоналізація» (Рис. 3.39). Цей безперешкодний процес купівлі не лише дозволяє користувачам візуально налаштувати свій профіль, але й сприяє формуванню почуття причетності та залученості, оскільки вони можуть бачити обраного ними Лекситрона у щоденних звітах, додаючи персоналізований елемент до своїх досягнень.

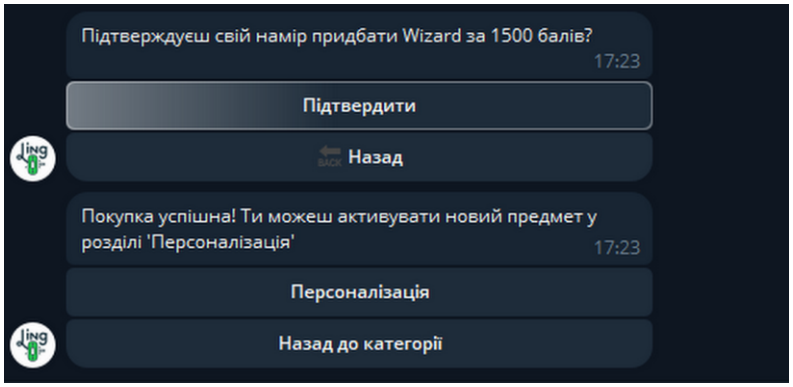

Рис. 3.39. Повідомлення про успішну покупку Лекситрона

Розділ «Здібності» магазину надає користувачам функціональні інструменти, які ще більше збагачують їхній навчальний досвід. Тут користувачі можуть вибирати з різних здібностей, кожна з яких призначена для того, щоб додати цінності їхньому навчальному процесу (Рис. 3.40).

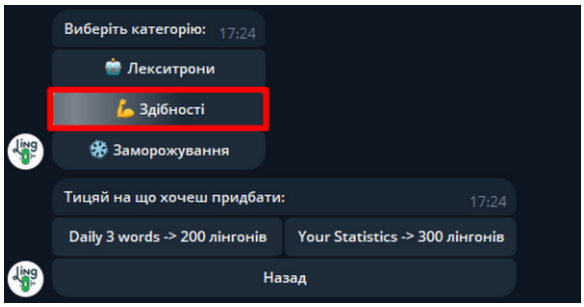

Рис. 3.40. Розділ «Здібності» у магазині

Коли функцію було обрано, з’являється детальний опис, який окреслює

її функціональність. Користувачі можуть придбати здібність, натиснувши кнопку «Купити», яка ініціює процес підтвердження покупки (Рис. 3.41).

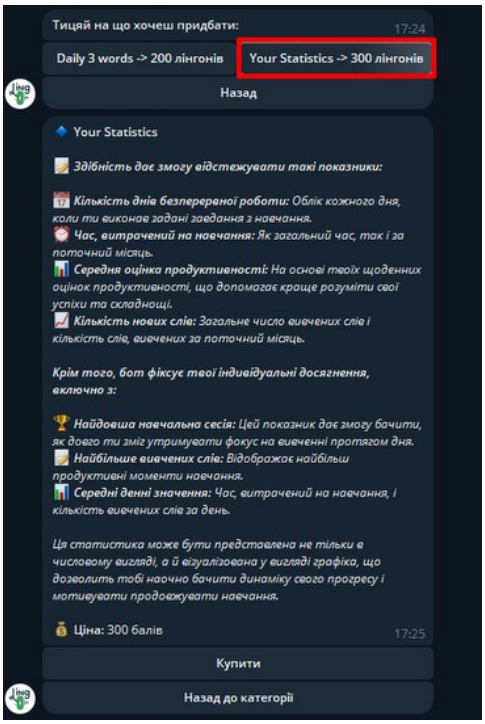

Рис. 3.41. Приклад опису здібності

Категорія «Заморожування» пропонує купівлю днів замороження прогресу. Це дозволить користувачам «заморозити» свій прогрес на певну кількість днів, наприклад, 1, 3, 5 або 10 днів, залежно від обраної опції. Кожна тривалість заморожування відображається з відповідною вартістю в Лінгонах, що допомагає користувачам обрати варіант, який найкраще відповідає їхнім поточним потребам.

Коли користувач обирає варіант кількості днів заморозки, він отримує детальну інформацію про переваги цієї функції. Наприклад, вибравши 5-денну опцію заморозки, користувач отримає 5 днів заморозки, які він може активувати та буде мати можливість не подавати звіти або взаємодіяти з ботом протягом цього періоду, не ризикуючи втратити прогрес або бути видаленим з проекту (Рис. 3.43). Ця функція особливо корисна для підтримки послідовності в досягненнях користувача, забезпечуючи при цьому гнучкість на час, коли він не може брати активну участь у проекті.

Рис. 3.42. Опції заморожування прогресу

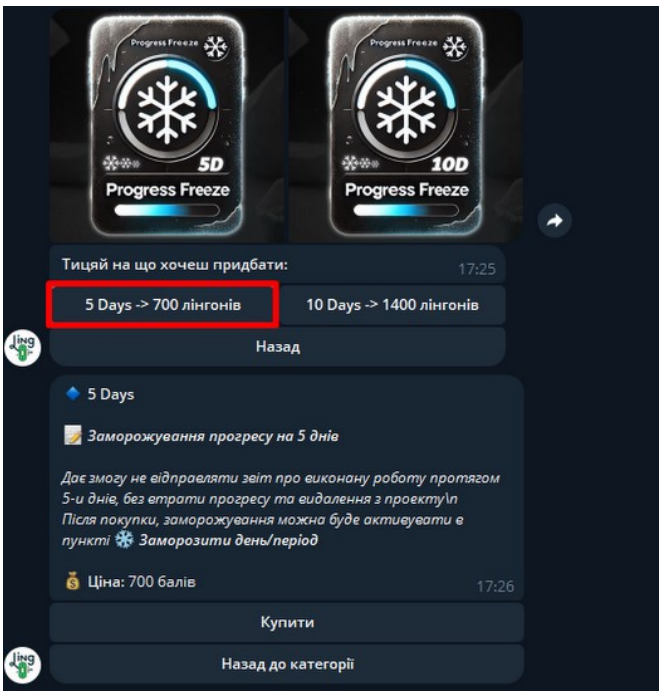

Рис. 3.43. Деталі опції заморожування

Внутрішній магазин LingOn створений для того, щоб зробити навчання англійської більш індивідуальним та цікавим. Такий підхід сприяє регулярній

взаємодії з ботом, даючи користувачам гнучкість та необхідні інструменти для підтримки мотивації та прогресу в навчанні.

Міні-гра «Nip and Franklin's Adventure». Після натискання відповідної кнопки у головному меню, користувач залучається до інтерактивних уроків, що сприяють розвитку чотирьох основних навичок володіння мовою.

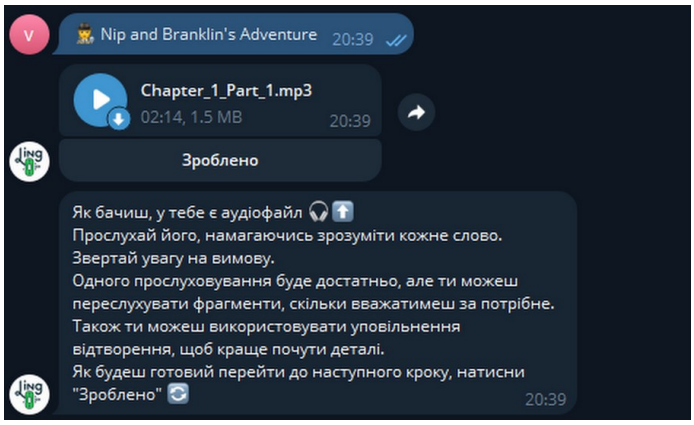

Рис. 3.44. Завдання на прослуховування аудіофайлу

Перша частина уроку присвячена саме вдосконаленню навичок аудіювання та читання та починається з того, що користувач отримує аудіофайл із озвученням фрагмента глави книги (Рис. 3.44). Користувачеві пропонується уважно прослухати його, звертаючи особливу увагу на кожне слово та вимову. Бот припускає, що одного прослуховування може бути достатньо, але користувачам рекомендується повторно прослуховувати окремі частини для кращого засвоєння матеріалу. Повідомлення також містить рекомендації щодо використання повільної швидкості відтворення, щоб краще зрозуміти деталі. Закінчивши прослуховування, користувач натискає кнопку «Зроблено», щоб перейти до наступного кроку.

Після завершення вправи з аудіювання бот надає текст аудіоуривка для покращення розуміння (Рис. 3.45). Користувачам пропонується виконати вправу у два етапи:

- По-перше, прослухати аудіо ще раз, слідуючи за текстом, зосереджуючись на правильній вимові.

- По-друге, прочитати текст вголос без аудіо, практикуючи інтонацію та виразність.

Закінчивши, треба натиснути кнопку «Зроблено», щоб перейти далі.

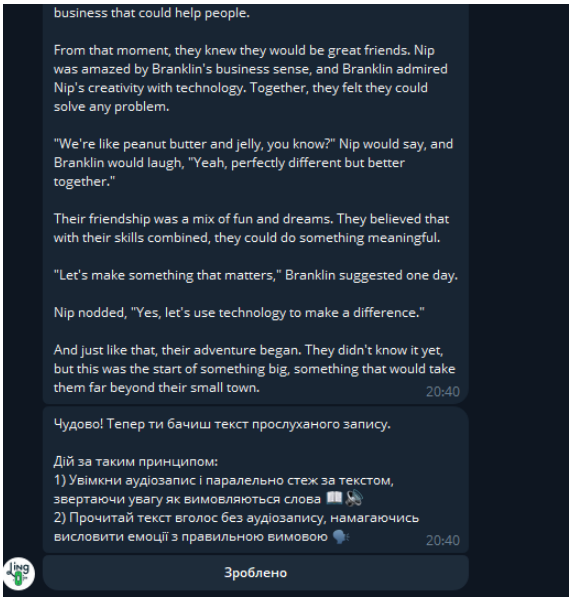

Рис. 3.45. Завдання на читання тексту

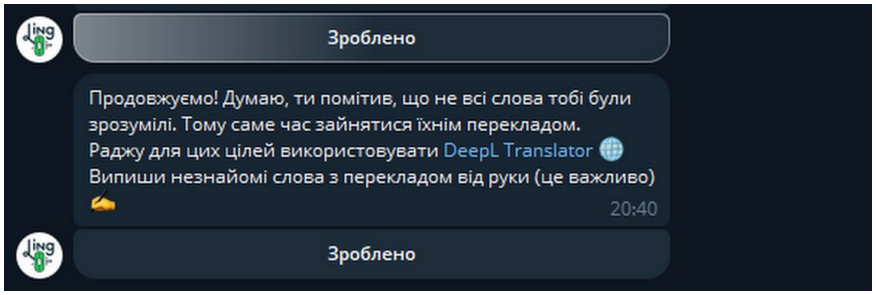

Рис. 3.46. Завдання на переклад та запис незнайомих слів

Розуміючи, що користувачі можуть зіткнутися з незнайомою лексикою, бот пропонує їм перекласти незрозумілі слова (Рис. 3.46). Він рекомендує використовувати перекладач DeepL і заохочує записувати незнайомі слова від руки, підкреслюючи важливість активної участі для кращого запам'ятовування. Цей крок допомагає користувачам розширити словниковий запас і поглибити розуміння.

Бот пропонує прослухати аудіо ще раз, цього разу не дивлячись на текст, щоб покращити розуміння без візуальної підтримки (Рис. 3.47).

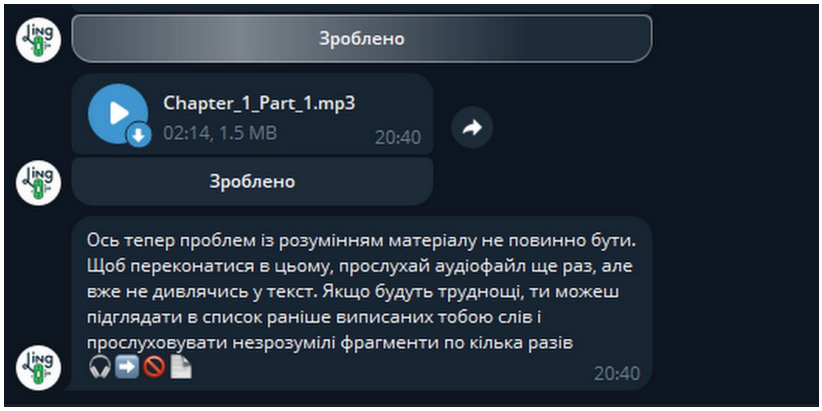

Рис. 3.47. Завдання на повторне прослуховування без тексту

За потреби користувач може повернутися до своїх нотаток щодо невідомих слів і зосередитися на повторному прослуховуванні будь-яких незрозумілих частин. Ця вправа призначена для зміцнення слухових навичок та забезпечення розуміння без допомоги тексту.

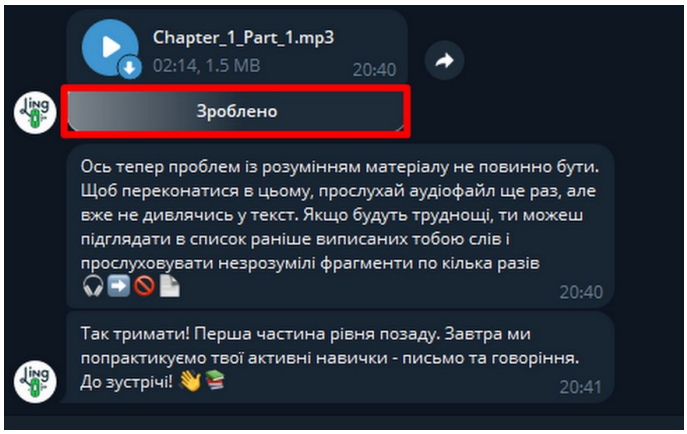

Рис. 3.48. Завершення першої частини рівня

Після виконання всіх кроків бот вітає користувача із завершенням першої частини рівня.

Ця серія завдань у міні-грі підтримує збалансований підхід до опанування пасивних навичок англійської мови, поєднуючи вправи на аудіювання, читання та поповнення словникового запасу для сприяння всебічному розвитку навичок. Кожен крок підкріплений інтерактивними підказками, що гарантує, що користувачі залишатимуться залученими та

зосередженими на своїх навчальних цілях.

Друга частина уроку в міні-грі «Nip and Branklin's Adventure» спрямована на розвиток навичок усного та писемного мовлення.

Бот починає з того, що пропонує користувачеві переглянути текст попередньої частини уроку, щоб переконатися, що він пам'ятає значення всіх ключових слів (Рис. 3.49). Це заохочує користувача встановити міцніший зв'язок зі словниковим запасом і загальним контекстом.

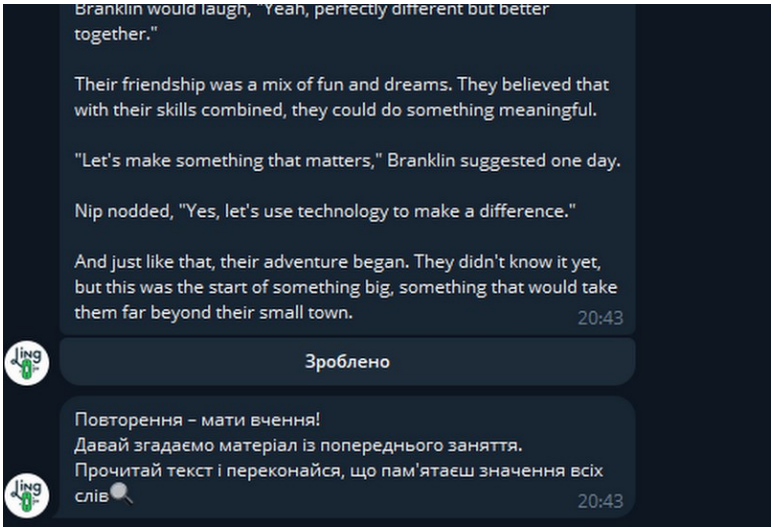

Рис. 3.49. Початок другої частини уроку

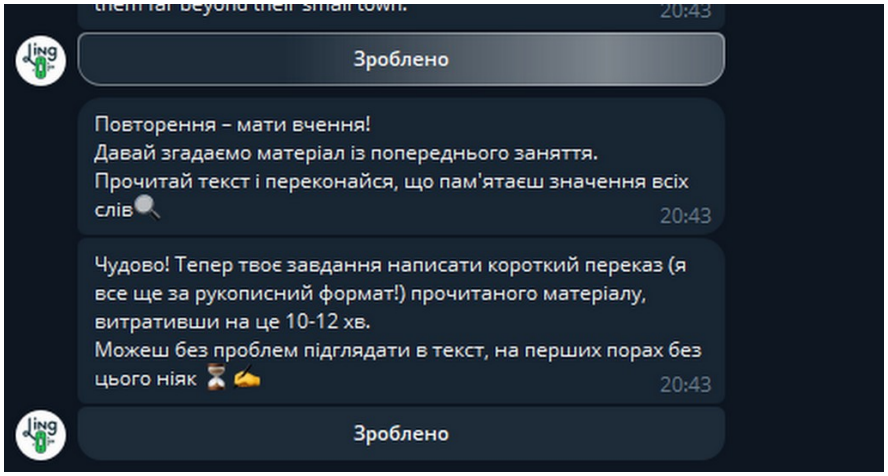

Рис. 3.50. Завдання написати короткий переказ тексту

Далі користувачеві пропонується написати від руки короткий переказ тексту (Рис. 3.50). Ця вправа слугує продуктивним методом для закріплення

розуміння, гарантуючи, що користувач зможе пригадати та систематизувати основні моменти матеріалу.

Після того, як користувач напише переказ, бот рекомендує скористатися сервісом для перекладу, щоб перевірити свою роботу (Рис. 3.51). Це заохочує до самостійної перевірки та виправлення помилок, допомагаючи закріпити нову лексику та вирази.

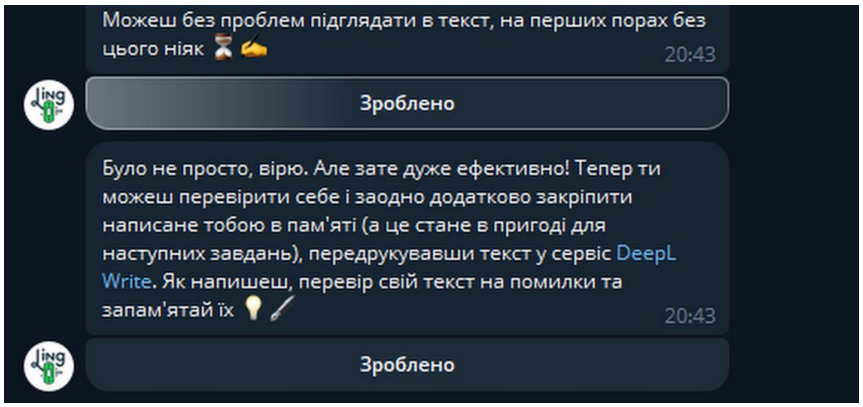

Рис. 3.51. Перевірка написаного тексту

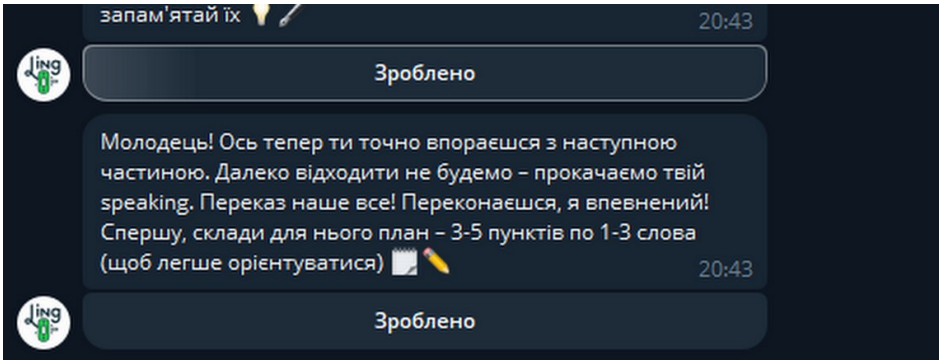

Рис. 3.52. Підготовка до усного переказу

Далі бот пропонує підготуватися до усного переказу тексту, створивши простий план з 3-5 ключових моментів (Рис. 3.52). Цей план діє як ментальна підказка для підтримки розмовної практики, допомагаючи зосередитися на структурованому переказі.

Час перейти до самого переказу, використовуючи план як орієнтир (Рис. 3.53). Ця вправа розвиває впевненість користувача в усному мовленні, заохочуючи його висловлюватися чітко та природно.

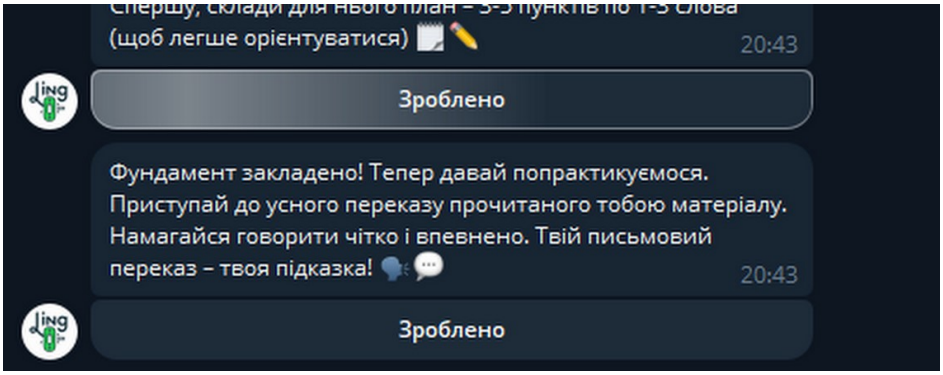

Рис. 3.53. Практика усного переказу

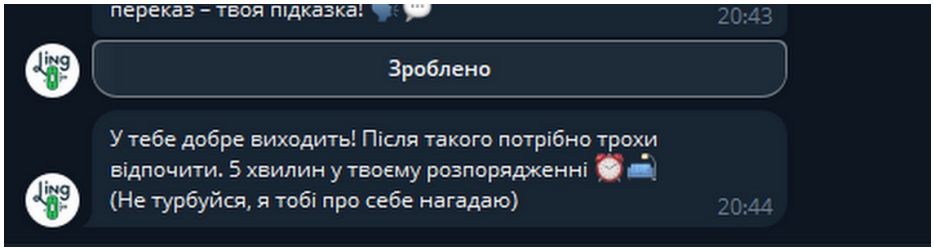

Рис. 3.54. П'ятихвилинна перерва

Розуміючи інтенсивність практики, бот пропонує зробити невелику перерву, щоб користувач міг перезарядитися (Рис. 3.54). Він сам відлічить 5 хвилин і повідомить, коли час перерви завершиться. Такий підхід балансує між продуктивністю та самопочуттям, гарантуючи, що користувач залишається зосередженим і мотивованим.

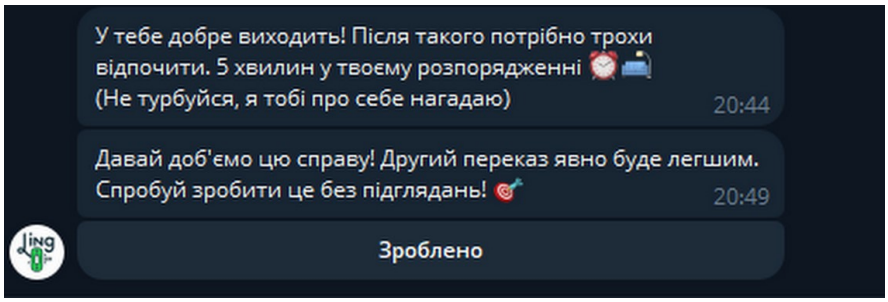

Рис. 3.55. Повторний переказ

Після перерви бот дає завдання ще раз переказати текст, але вже не дивлячись у свої записи (Рис. 3.55).

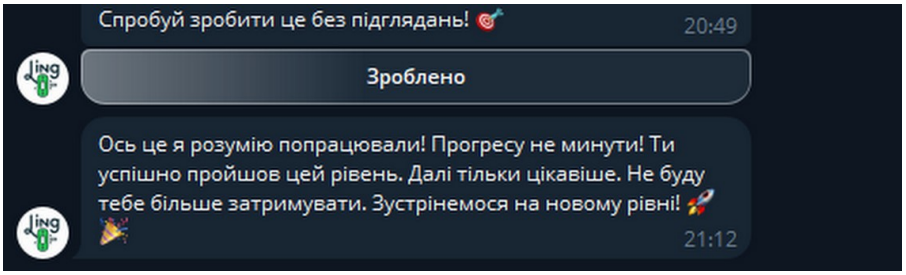

Рис. 3.56. Завершення другої частини рівня

Після успішного виконання цих завдань бот вітає користувача із завершенням рівня, запевняючи, що він робить відчутний прогрес (Рис. 3.56).

Завдяки таким структурованим завданням друга частина уроку дозволяє користувачеві перейти від розуміння до активного використання матеріалу, продовжуючи розвивати свої навички говоріння та письма.

РОЗДІЛ 4. ПЕРСПЕКТИВИ РОЗВИТКУ ТА ВИКЛИКИ ВПРОВАДЖЕННЯ TELEGRAM-БОТА LINGON

4.1. Можливості розширення функціоналу та інтеграція нових технологій

LingOn був розроблений як динамічний інструмент для вивчення мови, і існує значний простір для збагачення його освітнього впливу та залучення користувачів завдяки новим функціям. Подальший розвиток може бути зосереджений на поглибленні інтерактивності, розширенні персоналізованого досвіду та використанні інноваційних освітніх підходів. Розглянемо кілька ключових напрямків для потенційного розширення.

Однією з цікавих можливостей є міні-гра, яка фокусується на вивченні англійської мови за допомогою популярних пісень. Ця вправа може передбачати прослуховування пісень, а потім виконання таких вправ, як доповнення пропущених слів, розпізнавання ключової лексики або переклад пісенних фраз. Пов'язуючи слова з музикою, ритмом та емоціями, користувачі можуть краще запам'ятовувати словниковий запас, а також покращувати навички аудіювання та розуміння в приємному, реальному контексті.

Ще однією можливою функцією є «Навчання за допомогою опису картинок», коли користувачі переглядають зображення та описують його зміст англійською мовою. Ця міні-гра може розширити словниковий запас, творче мислення та структурування речень, спонукаючи користувачів формулювати те, що вони бачать. Функція може включати словникові підказки, які допоможуть користувачам скласти детальний опис, пропонуючи завдання, придатне для учнів з різним рівнем володіння мовою.

LingOn також може запропонувати користувачам можливість поекспериментувати з популярними методиками вивчення мови, заснованими на методах з онлайн-ресурсів, наприклад з YouTube

відеороликів. Ці підходи не були б офіційними методиками, а скоріше кураторськими практиками, які користувачі могли б спробувати і переглянути. Збір відгуків користувачів про ці методи може допомогти вдосконалити пропозиції бота, гарантуючи, що користувачі отримають користь від найефективніших методів навчання.

Можливість для користувачів розробляти та відстежувати свої персоналізовані навчальні стратегії може забезпечити більш глибокий рівень кастомізації. Користувачі могли б створювати план навчання з конкретними цілями і кроками, який бот періодично перевіряв би, спонукаючи їх замислитися над своїм прогресом. Така персоналізована методологія могла б допомогти користувачам зосередитися на своїх унікальних цілях, надаючи при цьому цінні дані про те, які стратегії працюють найкраще. Бот міг би аналізувати ці методи, про які вони самі звітують, і потенційно розробляти нові рекомендації для всіх користувачів на основі колективних інсайтів.

Виклики дуже ефективні для підтримки мотивації, особливо якщо вони оформлені у вигляді міні-ігор з різноманітними цілями та проміжними етапами. Наприклад, завдання можуть передбачати відстеження виконання певної кількості мовних вправ, таких як написання статей або вивчення нових слів. Ці завдання можна змінювати щотижня, пропонуючи нові теми та цілі, які можуть переслідувати користувачі, супроводжуючи їх метриками для відстеження прогресу. Завдяки частому оновленню завдань бот буде постійно пропонувати нові та цікаві завдання, що сприятимуть постійному прогресу в навчанні.

У майбутньому бот може підтримувати створення спільнот за допомогою «розмовних клубів». Вони дозволять користувачам підключатися до розмов в режимі реального часу або асинхронно, заохочуючи розвиток навичок говоріння та слухання. Користувачі могли б обмінюватися аудіоповідомленнями, приєднуватися до запланованих голосових чатів або обмінюватися навчальними матеріалами, створюючи сприятливе середовище спільноти. Дана опція дозволить користувачам практикувати розмовні

навички один з одним, отримувати зворотній зв'язок від однолітків і залишатися вмотивованими, беручи участь у колективній подорожі з вивчення мови.

Інтегруючи ці вдосконалені міні-ігри, стратегії навчання, що налаштовуються, та елементи, керовані спільнотою, LingOn може перетворитися на ще більш комплексний інструмент. Ці потенційні розширення пропонують різноманітні способи підтримки залучення користувачів, адаптації до різних стилів навчання та створення багатого, мотивуючого середовища для вивчення мови.

4.2. Виклики та стратегії подолання технічних і соціальних ризиків

Хоча потенціал LingOn для підтримки вивчення мов є значним, важливо визнати виклики, які можуть виникнути в міру розвитку бота. Ці виклики охоплюють як технічні, так і соціальні аспекти, оскільки бот повинен підтримувати функціональність і актуальність, залучаючи при цьому різноманітну базу користувачів.

Масштабованість і продуктивність.

Зі збільшенням кількості користувачів LingOn зростатиме попит на серверні ресурси. Це зростання може призвести до повільного часу відгуку, що вплине на якість обслуговування користувачів, особливо в періоди пікового навантаження. Щоб вирішити цю проблему, важливо зосередитися на оптимізації продуктивності бекенда:

- Використання асинхронного програмування

LingOn використовує aiogram, яка дозволяє обробляти декілька запитів користувачів одночасно без затримок. Майбутні оновлення можуть ще більше вдосконалити цю функцію, гарантуючи, що кожна дія виконуватиметься безперебійно навіть при великому користувацькому трафіку.

- Балансування навантаження та оптимізація бази даних

Розподіл навантаження на сервер і впровадження ефективної індексації бази даних запобігає виникненню «вузьких місць» у роботі з даними. Такі методи, як кешування частих запитів і використання хмарних ресурсів, можуть гарантувати, що бот залишатиметься оперативним навіть під час пікового попиту.

Захист даних користувачів має першорядне значення для будь-якої освітньої платформи, особливо тієї, що зберігає особисту інформацію та прогрес користувача. Будь-який витік даних може підірвати довіру користувачів і зашкодити репутації LingOn. Наступні стратегії мають вирішальне значення для захисту даних:

- Шифрування конфіденційних даних

Впроваджуючи шифрування всіх конфіденційних даних, включаючи платіжну інформацію та особисті дані користувачів, LingOn може забезпечити безпеку цієї інформації на кожному етапі зберігання та передачі.

- Регулярні перевірки та оновлення безпеки

Проведення періодичних перевірок безпеки та оновлення кодової бази та сервера може допомогти виявити вразливості на ранній стадії. Застосування суворого контролю доступу для адміністраторів і розробників також зменшить ризик.

Підтримка гнучкості для майбутньої інтеграції План розвитку LingOn включає різні потенційні розширення, такі як додаткові міні-ігри та розширені функції штучного інтелекту. Однак реалізація цих нових функцій без порушення існуючої функціональності може бути складним завданням. Збереження гнучкості:

- Модульна структура коду

Архітектура бота є модульною, що дозволяє додавати нові функції як незалежні компоненти. Це мінімізує втручання в існуючу функціональність, забезпечуючи плавну інтеграцію будь-якої нової розробки.

- Безперервне тестування

Використання автоматизованого тестування та середовища пісочниці

для тестування нових функцій дозволяє розробникам виявляти будь-які проблеми сумісності до того, як функції будуть запущені. Такий підхід допомагає запобігти перебоям у роботі активних користувачів.

З огляду на численні доступні навчальні платформи, встановлення довіри з користувачами та створення репутації LingOn як надійного освітнього інструменту є життєво важливим.

- Прозорість у політиці користувачів

Чітке інформування про політику використання даних, умови підписки та права користувачів сприяє зміцненню довіри. Регулярне інформування користувачів про зміни в цих політиках також демонструє прихильність до їхньої конфіденційності.

- Послідовна, якісна підтримка

Забезпечення своєчасної та ефективної підтримки користувачів на запити, технічні проблеми або відгуки допомагає створити позитивну репутацію та впевненість користувачів у прихильності LingOn до підтримки користувачів.

LingOn обслуговує користувачів з різними цілями і темпами навчання, що вимагає індивідуального підходу для підтримки залучення широкої аудиторії.

- Пропозиції щодо контенту, орієнтовані на користувачів

Регулярне опитування користувачів про їхні уподобання щодо нових тем або функцій може забезпечити відповідність бота інтересам користувачів.

- Персоналізовані сповіщення та заохочення

Цільові нагадування, засновані на прогресі та цілях кожного користувача, допоможуть підтримувати їхню залученість і мотивацію до подальшої роботи.

Для платформи, орієнтованої на вивчення мови, важливим є створення спільноти, в якій користувачі відчувають підтримку та безпеку.

- Чіткі правила та стандарти спільноти

Встановлення та дотримання правил шанобливого та конструктивного спілкування в будь-якому спільному просторі допоможе підтримувати позитивну навчальну атмосферу.

- Можливості для співпраці користувачів

Створення просторів, де користувачі можуть співпрацювати або наставляти інших, може підвищити залученість і зменшити ізоляцію, сприяючи створенню більш згуртованої спільноти.

Передбачаючи і вирішуючи ці проблеми, LingOn може створити гнучке, стійке і орієнтоване на користувача середовище, яке не тільки приваблює нових користувачів, але й утримує існуючих, забезпечуючи своє місце як надійного освітнього інструменту в ландшафті вивчення мов.

ВИСНОВКИ

На основі виконаної роботи можна зробити наступні висновки: онлайн-освіта потребує ефективних мотиваційних інструментів для забезпечення тривалої участі користувачів. Використання елементів гейміфікації, фінансових стимулів та соціальної взаємодії суттєво підвищує зацікавленість у навчанні. Telegram-бот LingOn дозволяє інтегрувати гейміфікаційні механізми та інструменти соціальної взаємодії в освітній процес, підвищуючи регулярність і якість навчання завдяки інтерактивним функціям та персоналізації.

Впровадження гейміфікаційних елементів, таких як персонажі (Лекситрони), рівні, очки (Лінгони), та міні-ігри, сприяє підвищенню залученості користувачів і створенню конкурентного середовища, яке мотивує до досягнення нових результатів. Міні-гра «Nip and Franklin's Adventure» інтегрує аудіювання, читання, письмо та говоріння в єдиний навчальний процес, надаючи користувачам можливість вдосконалювати свої навички у цікавій формі.

Модель підписки вартістю $10/місяць не тільки забезпечує доступ до розширених функцій, але й слугує мотиваційним фактором, змушуючи користувачів дотримуватись навчального графіка, щоб отримати максимальну користь від витрачених коштів. Telegram-бот LingOn базується на сучасних технологіях, що забезпечують швидку та надійну роботу, включаючи бекенд для обробки запитів, базу даних для збереження прогресу користувачів і інтеграцію із зовнішніми сервісами.

Спільнота користувачів, створена в Telegram, сприяє залученню до навчання, забезпечуючи взаємну підтримку та змагання між учасниками, що позитивно впливає на мотивацію. Основними труднощами є технічні ризики, такі як збої в роботі сервера, та соціальні виклики, зокрема необхідність постійного залучення нових користувачів. Для їх подолання запропоновано використання резервних систем та активного маркетингу в соціальних

мережах.

Перспективи подальшого розвитку Telegram-бота LingOn передбачають впровадження сучасних технологій, таких як генеративний штучний інтелект, покращення інтерактивних елементів та розширення навчального контенту. Подальші дослідження можуть зосередитися на оцінці довгострокової ефективності мотиваційних механізмів та аналізі впливу соціальної взаємодії на навчальний прогрес. Ці заходи сприятимуть підвищенню якості навчання та забезпеченню стійкості проекту. У результаті виконана робота закладає ґрунт для подальшого розвитку онлайн-освіти з використанням інноваційних рішень та підходів, що базуються на сучасних технологіях і практиках.

СПИСОК ЛІТЕРАТУРИ

1. Schunk, D. H., Pintrich, P. R., & Meece, J. L. (2008). *Motivation in Education: Theory, Research, and Applications.* Pearson.

2. Deci, E. L., & Ryan, R. M. (2000). The "What" and "Why" of Goal Pursuits: Human Needs and the Self-Determination of Behavior. *Psychological Inquiry,* 11(4), 227-268. DOI: 10.1207/S15327965PLI1104_01. URL: https://selfdeterminationtheory.org/SDT/documents/2000_DeciRyan_PIWhatWhy.pdf

3. Vroom, V. H. (1964). *Work and Motivation.* Wiley.

4. Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From Game Design Elements to Gamefulness: Defining "Gamification". *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments,* 9-15. URL: https://doi.org/10.1145/2181037.2181040

5. Duolingo. How to keep yourself motivated all year long. URL: https://blog.duolingo.com/how-to-keep-yourself-motivated-all-year-long/

6. Duolingo. How do Duolingo Leaderboards work? URL: https://blog.duolingo.com/duolingo-leagues-leaderboards/

7. Khan Academy. What are energy points, badges, and avatars. URL: https://support.khanacademy.org/hc/en-us/articles/202487710-What-are-energy-points-badges-and-avatars

8. Khan Academy. Introducing mastery learning to students. URL: https://www.khanacademy.org/khan-for-educators/k4e-us-demo/xb78db74671c953a7:getting-to-know-khan/xb78db74671c953a7:getting-to-know-khan-academy/a/introducing-mastery-learning-to-students

9. Quizlet. Study Modes. URL: https://quizlet.com/gb/features/study-modes

10. Quizlet Blog. How to make studying fun for every class you're taking. URL: https://quizlet.com/blog/how-to-make-studying-fun-for-every-class-youre-taking

11. Wang, Y., & Liao, C. (2020). The Impact of AI Chatbots on Online

Learning: A Study of Personalization and Engagement. URL: https://doi.org/10.1016/j.compedu.2020.103923

12. Adamopoulou, E., & Moussiades, L. (2020). An Overview of Chatbot Technology in Education. *AI and Ethics,* 1(2), 10-21.

13. Hill, J., Randolph Ford, W., & Farreras, I. G. (2015). Real Conversations with Artificial Intelligence: A Comparison between Human–Human Online Conversations and Human–Chatbot Conversations. *Computers in Human Behavior,* 49, 245–250. URL: https://doi.org/10.1016/j.chb.2015.02.026

14. Holmes, W., Bialik, M., & Fadel, C. (2019). Artificial Intelligence in Education: Promises and Implications for Teaching and Learning.

15. Latham, A. (2021). The Role of AI Chatbots in Scaling Online Education.

16. European Agency for Safety and Health at Work. (2024). Artificial intelligence and education – a teacher-centred approach to safety and health. URL: https://osha.europa.eu/sites/default/files/documents/AI-and-education_EN.pdf

17. EF English Proficiency Index 2020. URL: https://www.ef.com/wwen/epi/

18. Інститут соціології НАН України. Звіт за результатами соціологічного дослідження 2021 року. URL: https://www.i-soc.com.ua/

19. Київський міжнародний інститут соціології. URL: https://kiis.com.ua/

20. Duolingo. Language Learning Trends in Ukraine. URL: https://blog.duolingo.com/

21. Coursera. Education in Eastern Europe: Trends and Growth. URL: https://www.coursera.org/

22. Udemy. Online Learning in Ukraine. URL: https://about.udemy.com/

23. Міністерство цифрової трансформації України. URL: https://thedigital.gov.ua/

24. Statista. Regional Telegram User Statistics in Ukraine. URL: https://www.statista.com/

25. Kapp, K. M. (2012). "The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education." Pfeiffer. URL: https://www.wiley.com/en-us/The+Gamification+of+Learning+and+Instruction-p-9781118096345

26. Csikszentmihalyi, M. (1990). "Flow: The Psychology of Optimal Experience." Harper and Row. URL: https://www.flowtheory.com/

27. Schultz, W. (2015). "Neuronal Reward and Decision Signals: From Theories to Data." *Physiological Reviews*. URL: https://journals.physiology.org/

28. Arkes, H. R., & Blumer, C. (1985). "The Psychology of Sunk Cost." *Organizational Behavior and Human Decision Processes.* URL: https://doi.org/10.1016/0749-5978(85)90049-4

29. Thaler, R. H. (1985). "Mental Accounting and Consumer Choice." *Marketing Science.* URL: https://doi.org/10.1287/mksc.4.3.199

30. McMillan, D. W., & Chavis, D. M. (1986). "Sense of Community: A Definition and Theory." *Journal of Community Psychology.* URL: https://doi.org/10.1002/1520-6629(198601)

31. Vygotsky, L. S. (1978). "Mind in Society: The Development of Higher Psychological Processes." Harvard University Press. URL: https://www.hup.harvard.edu/catalog.php?isbn=9780674576292

32. Aiogram Documentation. "Asynchronous Telegram Bot API Framework for Python." URL: https://docs.aiogram.dev/en/latest/

33. Asyncio Documentation. "Asyncio: Asynchronous I/O for Python." URL: https://docs.python.org/3/library/asyncio.html

34. APScheduler Documentation. "Advanced Python Scheduler: Task Scheduling for Python." URL: https://apscheduler.readthedocs.io/en/latest/

35. MySQL Documentation. "MySQL: The World's Most Popular Open Source Database." URL: https://dev.mysql.com/doc/

36. Oracle MySQL Technical Overview. "MySQL and ACID Compliance: Ensuring Data Integrity." URL: https://www.oracle.com/mysql/

37. MySQL Connector for Python. "MySQL Connector: Simplifying Python

Database Integration." URL: https://dev.mysql.com/doc/connector-python/en/

ДОДАТКИ

ДОДАТОК А

ВМІСТ ФАЙЛУ MAIN.PY. ГОЛОВНИЙ ФАЙЛ

```
import keyboard
import asyncio
from aiogram import types, executor
from create_bot import dp, bot
import shared_resources
from datetime import datetime, timedelta
from handlers import new_user
from handlers import sent_report
from handlers import freeze_progress
from handlers import statistic
from handlers import admin_control
from handlers import payment
from handlers import nip_and_branklin
from handlers import shop
from handlers import personalization
from handlers import channel_admin
from mysql.connector import connect, Error
from shared_resources import db_config
from apscheduler.schedulers.asyncio import AsyncIOScheduler
from pytz import timezone
import logging
import os
from aiogram.types import InputFile

# Initialize the scheduler with the desired timezone
scheduler = AsyncIOScheduler(timezone=timezone('Europe/Kiev'))

# Configure logging at the very beginning
logging.basicConfig(
    level=logging.DEBUG,  # Set to DEBUG to capture all logs
    format='%(asctime)s - %(name)s - %(levelname)s -
%(message)s',
)

def initialize_database():
    try:
        # Connect to the MySQL database
        conn = connect(**db_config)
        cursor = conn.cursor()

        # Create the table if it doesn't exist for known users
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS known_users (
                user_id BIGINT PRIMARY KEY,
                username VARCHAR(255) DEFAULT NULL,
                first_name VARCHAR(255) DEFAULT NULL,
                start_date TEXT,
                deadline TEXT,
```

```
                last_report_date TEXT,
                days_of_usage INTEGER,
                last_payment TEXT,
                next_payment TEXT,
                last_notify DATE,
                user_rank VARCHAR(50) DEFAULT 'Новатор',
                points INTEGER DEFAULT 0,
                level INTEGER DEFAULT 1,
                promo_code TEXT,
                promo_end_date DATE,
                total_paid DECIMAL(10, 2) DEFAULT 0.00
            )
        ''')
        print("Table 'known_users' checked/created.")

        # Create the categories table (as it's a dependency for
items)
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS categories (
                category_id INTEGER AUTO_INCREMENT PRIMARY KEY,
                name VARCHAR(255) NOT NULL
            )
        ''')
        print("Table 'categories' checked/created.")

        # Create the items table (as it is required for other
tables)
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS items (
                item_id INTEGER AUTO_INCREMENT PRIMARY KEY,
                category_id INTEGER,
                name VARCHAR(255) NOT NULL,
                description TEXT,
                rarity VARCHAR(50) DEFAULT '',
                price INTEGER NOT NULL,
                image_url TEXT,
                is_visible BOOLEAN DEFAULT TRUE,
                days INTEGER DEFAULT 0,
                FOREIGN KEY (category_id) REFERENCES
categories(category_id)
            )
        ''')
        print("Table 'items' checked/created.")

        # Create the table for user statistics if it doesn't
exist
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS user_stats (
                stat_id INTEGER AUTO_INCREMENT PRIMARY KEY,
                user_id BIGINT,
                date DATE,
                metric TEXT,
                value REAL,
```

```
                FOREIGN KEY (user_id) REFERENCES
known_users(user_id)
            )
        ''')
        print("Table 'user_stats' checked/created.")

        # The games table must be created before the materials
table
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS games (
                game_id INTEGER AUTO_INCREMENT PRIMARY KEY,
                game_name TEXT,
                description TEXT
            )
        ''')
        print("Table 'games' checked/created.")

        # Now create the materials table
        cursor.execute('''
           CREATE TABLE IF NOT EXISTS materials (
                material_id INTEGER AUTO_INCREMENT PRIMARY KEY,
                game_id INTEGER,
                lesson_id INTEGER,
                text_content TEXT,
                audio_url VARCHAR(255),
                FOREIGN KEY (game_id) REFERENCES games(game_id)
            )
        ''')
        print("Table 'materials' checked/created.")

        # Create the table for user progress if it doesn't exist
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS user_progress (
                user_id BIGINT,
                game_id INTEGER,
                lesson_id INTEGER,
                part_id INTEGER,
                completion_status INTEGER DEFAULT 0,
                PRIMARY KEY (user_id, game_id),
                FOREIGN KEY (user_id) REFERENCES
known_users(user_id),
                FOREIGN KEY (game_id) REFERENCES games(game_id)
            )
        ''')
        print("Table 'user_progress' checked/created.")

        # Create the user_abilities table
        cursor.execute('''
             CREATE TABLE IF NOT EXISTS user_abilities (
                user_id BIGINT,
                item_id INTEGER,
                is_active BOOLEAN DEFAULT FALSE,
                level VARCHAR(50),
```

```
                sent_words TEXT,
                PRIMARY KEY (user_id, item_id),
                FOREIGN KEY (user_id) REFERENCES
known_users(user_id),
                FOREIGN KEY (item_id) REFERENCES items(item_id)
            )
        ''')
        print("Table 'user_abilities' checked/created.")

        # Create the user_characters table
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS user_characters (
                user_id BIGINT,
                item_id INTEGER,
                is_active BOOLEAN DEFAULT FALSE,
                PRIMARY KEY (user_id, item_id),
                FOREIGN KEY (user_id) REFERENCES
known_users(user_id),
                FOREIGN KEY (item_id) REFERENCES items(item_id)
            )
        ''')
        print("Table 'user_characters' checked/created.")

        # Create the purchases table
        cursor.execute('''
              CREATE TABLE IF NOT EXISTS user_purchases (
                  purchase_id INTEGER AUTO_INCREMENT PRIMARY
KEY,
                  user_id BIGINT,
                  item_id INTEGER,
                  purchase_date TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,
                  source VARCHAR(50) DEFAULT 'bought',
                  FOREIGN KEY (user_id) REFERENCES
known_users(user_id),
                  FOREIGN KEY (item_id) REFERENCES
items(item_id)
                  )
              ''')
        print("Table 'user_purchases' checked/created.")

        # Create the words table
        cursor.execute('''
                  CREATE TABLE IF NOT EXISTS words (
                      word_id INTEGER AUTO_INCREMENT PRIMARY
KEY,
                      level VARCHAR(50),
                      word VARCHAR(255) NOT NULL,
                      definition TEXT,
                      example_sentence TEXT,
                      translation VARCHAR(255),
                      synonyms TEXT,
                      antonyms TEXT
```

```
                )
            ''')
        print("Table 'words' checked/created.")

        # Create the user_freezes table if it doesn't exist
        cursor.execute('''
                CREATE TABLE IF NOT EXISTS user_freezes (
                        user_id BIGINT PRIMARY KEY,
                        freeze_available INTEGER DEFAULT 0,
                        freeze_count INTEGER DEFAULT 0,
                        FOREIGN KEY (user_id) REFERENCES
known_users(user_id)
                    )
                ''')
        print("Table 'user_freezes' checked/created.")

        # Create the table for polls data
        cursor.execute('''
                CREATE TABLE IF NOT EXISTS polls  (
                        post_number INT PRIMARY KEY,
                        keyword TEXT,
                        correct_answer INTEGER,
                        needs_evaluation BOOLEAN DEFAULT 1
            )
        ''')
        print("Table 'polls_data' checked/created.")

        # Create the poll_id_mapping table to map poll_id to
post_number
        cursor.execute('''
                CREATE TABLE IF NOT EXISTS poll_id_mapping (
                        poll_id BIGINT PRIMARY KEY,
                        post_number INT,
                        FOREIGN KEY (post_number) REFERENCES
polls(post_number)
            )
        ''')
        print("Table 'poll_id_mapping' checked/created.")

        # Create the poll_answers table
        cursor.execute('''
                    CREATE TABLE IF NOT EXISTS user_answers (
                        user_id INTEGER,
                        post_number INT,
                        answer INTEGER,
                        PRIMARY KEY (user_id, post_number)
                       )
                ''')
        print("Table 'poll_answers' checked/created.")

        # Create the 'polls' table if it doesn't exist
        cursor.execute('''
                    CREATE TABLE IF NOT EXISTS polls (
```

```
                        post_number INT PRIMARY KEY,
                        question TEXT NOT NULL,
                        options TEXT NOT NULL,
                        correct_answer INT NOT NULL,
                        post_time DATETIME NOT NULL,
                        image_path TEXT,
                        needs_evaluation BOOLEAN DEFAULT TRUE
                    )
                ''')
        logging.info("Table 'polls' checked/created.")

        # Create the 'user_answers' table if it doesn't exist
        cursor.execute('''
                    CREATE TABLE IF NOT EXISTS user_answers (
                        user_id BIGINT NOT NULL,
                        post_number INT NOT NULL,
                        answer INT NOT NULL,
                        PRIMARY KEY (user_id, post_number)
                    )
                ''')
        logging.info("Table 'user_answers' checked/created.")

        # Commit the changes
        conn.commit()
    except Error as e:
        print(f"An error occurred: {e}")
    finally:
        # Ensure the connection is closed even if an error
occurs
        if conn.is_connected():
            conn.close()

initialize_database()

def update_days_of_usage(user_id):
    try:
        with connect(**db_config) as conn:
            with conn.cursor() as cursor:
                cursor.execute('SELECT start_date FROM
known_users WHERE user_id = %s', (user_id,))
                result = cursor.fetchone()

                if result and result[0]:
                    start_date_str = result[0]
                    start_date =
datetime.strptime(start_date_str, '%Y-%m-%d').date()
                    today = datetime.now().date()

                    days_of_usage = (today - start_date).days

                    cursor.execute('UPDATE known_users SET
days_of_usage = %s WHERE user_id = %s', (days_of_usage,
user_id))
```

```
                    conn.commit()
                    print(f"Updated days of usage for user
{user_id}: {days_of_usage} days")
                else:
                    print(f"Start date not found for user
{user_id}")
    except Error as e:
        print(f"An error occurred: {e}")


@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    user_id = message.from_user.id
    username = message.from_user.username
    first_name = message.from_user.first_name
    conn = None
    try:
        # Connect to the MySQL database
        conn = connect(**db_config)
        cursor = conn.cursor()
        # Check if the user is already known
        cursor.execute('SELECT 1 FROM known_users WHERE user_id
= %s', (user_id,))
        result = cursor.fetchone()
        print(f"Database check result for user {user_id}:
{result}")
        user_known = result is not None

        if user_known:
            print(f"User ID {user_id} is known...")
            # User is known, proceed with checking report and
payment
            if await check_report_and_payment(user_id):
                update_days_of_usage(user_id)
                await bot.send_message(user_id, "Вітаю знову!
Виберіть опцію з меню або введіть команду.",

reply_markup=keyboard.get_main_menu_keyboard(user_id))
        else:
            print(f"User ID {user_id} is not known...")
            # Insert the new user into the database
            cursor.execute(
                'INSERT INTO known_users (user_id, username,
first_name) VALUES (%s, %s, %s)',
                (user_id, username if username else None,
first_name if first_name else None)
            )
            conn.commit()

            if user_id not in new_user.user_states:
                new_user.user_states[user_id] = 'goal'
                await bot.send_message(user_id,
                    "🎯 <b>Давай почнемо з постановки мети!</b>
```

```
🎯\n\n"
                    "📝 <i>Яка в тебе мета у вивченні англійської?</i>\n\n"
                    "✏️ Подумай і поділись своїми цілями, а я могли допоможу тобі досягти їх найкращим чином 😉",
parse_mode='HTML')
                new_user.register_handlers_new_user(dp)
    except Error as db_err:
        await message.answer(f"A database error occurred: {db_err}")
    except Exception as e:
        await message.answer(f"An unexpected error occurred: {e}")


@dp.message_handler(lambda message: message.text.lower() == "📝 відправити звіт" or
sent_report.user_states.get(message.from_user.id, [-1])[-1] == "StartNewReport")
async def send_report(message: types.Message):
    user_id = message.from_user.id
    print(f"Current user history: {sent_report.user_states.get(user_id, 'No history')}")
    if await check_report_and_payment(user_id):
        update_days_of_usage(user_id)
        sent_report.user_states[user_id].append('TimeSpent')
        await bot.send_message(user_id,"⏳ <b>Скільки часу сьогодні пішло на вивчення?</b>\n(вкажи <i>цифрою</i> у хвилинах)",reply_markup=keyboard.get_back(user_id, 'sentreport'),parse_mode='HTML')


@dp.message_handler(lambda message: message.text.lower() == "🦸 nip and branklin's adventure")
async def nip_and_branklin_wrapper(message: types.Message):
    user_id = message.from_user.id
    if await check_report_and_payment(user_id):
        update_days_of_usage(user_id)
        await nip_and_branklin.start_adventure_1(message)

@dp.message_handler(lambda message: message.text.lower() == "❄️ заморозити день/період")
async def freeze_day_wrapper(message: types.Message):
    user_id = message.from_user.id
    # freeze_progress.user_states[user_id] = 'Freeze_day_period'
    if await check_report_and_payment(user_id):
        update_days_of_usage(user_id)
        await freeze_progress.freeze_day_handler(message, user_id)

@dp.callback_query_handler(lambda c: c.data ==
```

```
"freeze_day_wrapper")
async def handle_freeze_day_wrapper(callback_query:
types.CallbackQuery):
    user_id = callback_query.from_user.id
    if await check_report_and_payment(user_id):
        update_days_of_usage(user_id)
        await
freeze_progress.freeze_day_handler(callback_query.message,
user_id)

@dp.message_handler(lambda message: message.text.lower() == "📊
статистика")
async def show_statistics_wrapper(message: types.Message):
    user_id = message.from_user.id
    if await check_report_and_payment(user_id):
        update_days_of_usage(user_id)
        await statistic.show_statistics(message)

@dp.message_handler(lambda message: message.text.lower() == "🛒
магазин")
async def show_menu(message: types.Message):
    user_id = message.from_user.id
    if await check_report_and_payment(user_id):
        update_days_of_usage(user_id)
        await shop.show_shop(message)

@dp.message_handler(lambda message: message.text.lower() == "🎨
персоналізація")
async def show_menu(message: types.Message):
    user_id = message.from_user.id
    if await check_report_and_payment(user_id):
        update_days_of_usage(user_id)
        await personalization.show_personalization_menu(message)


@dp.message_handler(lambda message: message.text.lower() == "💳
оплата")
async def payment_info_wrapper (message: types.Message):
    await payment.payment_info(message)

async def check_report_and_payment(user_id):
    try:
        # Connect to the MySQL database
        with connect(**db_config) as conn:
            with conn.cursor() as cursor:
                # Step 1: Check if the user exists in the
database
                cursor.execute('SELECT last_report_date FROM
known_users WHERE user_id = %s', (user_id,))
                result = cursor.fetchone()

                if result is None:
```

```
                    # User not found in the database, prompt
them to register
                    await bot.send_message(user_id, "Тебе ще
немає в базі. Введи команду /start, щоб зареєструватися")
                    return False

                # Step 2: Check the last report status
                last_report_date_str = result[0]
                if last_report_date_str:
                    last_report_date =
datetime.strptime(last_report_date_str, '%Y-%m-%d').date()  #
MySQL date format
                    yesterday = (datetime.now() -
timedelta(days=1)).date()
                    if last_report_date < yesterday:
                        # Last report was not sent yesterday or
today
                        await bot.send_message(user_id, f"Ти не
надіслав звіт вчора... Треба поговорити
{shared_resources.telegram_contact_link}")
                        return False
                # If there's no last report date, or it's today
or yesterday, proceed

                # Step 3: Check payment and subscription status
                payment_message_text =
payment.check_payment_status(user_id)
                subscription_status =
payment.check_subscription_expiry(user_id)

                if "Оплата успішно пройшла!" not in
payment_message_text and subscription_status != "активна":
                    await bot.send_message(user_id,
                        "❌ <b>Твоя підписка неактивна або ще
перевіряється.</b>\n\n"
                        "🔄 <i>Будь ласка, онови підписку, щоб
продовжити користування ботом.</i>\n\n"
                        "👇 Натисни на кнопку нижче для оновлення
підписки 👇",

reply_markup=keyboard.pay_button(user_id),
                        parse_mode='HTML')
                    return False

                # If everything is fine
                return True

    except Error as e:
        print(f"Database error occurred: {e}")
        return False
    except Exception as e:
        print(f"Failed to check payment and subscription status:
```

```
{e}")
        return False

async def send_morning_message_and_update():
    process_id = os.getpid()
    try:
        conn = connect(**db_config)
        cursor = conn.cursor(dictionary=True)

        # Fetch all user_ids and their days_of_usage
        cursor.execute("SELECT user_id, days_of_usage FROM
known_users")
        users = cursor.fetchall()

        for user in users:
            user_id = user['user_id']
            days_of_usage = user['days_of_usage']
            print(f"Sending reminder to user {user_id}")

            # Send daily reminder message
            await bot.send_message(user_id, "🔔 Добрий ранок! Не
забудь сьогодні позайматися англійською! 📚", parse_mode='HTML')
            print(f"Reminder sent to user {user_id}")

            # Update user stats (rank and level)
            new_rank, rank_changed, new_level, level_changed =
await update_user_stats(user_id)

            # Assign character if conditions met
            await assign_character(user_id, days_of_usage,
cursor, conn)

            # Notify user if rank or level has changed
            if rank_changed:
                await bot.send_message(user_id, f"🎉 Вітаємо! Ви
досягли нового рангу: {new_rank}!", parse_mode='HTML')
            if level_changed:
                await bot.send_message(user_id, f"🎉 Вітаємо! Ви
досягли нового рівня: {new_level}!", parse_mode='HTML')

        cursor.close()
        conn.close()

    except Exception as e:
        print(f"Error during send_morning_message_and_update
from process ID: {process_id}: {e}")


async def assign_character(user_id, days_of_usage, cursor,
conn):
    if days_of_usage == 7:
        cursor.execute("SELECT item_id, image_url FROM items
```

```
WHERE name = 'Lexitron'")
        character = cursor.fetchone()
        if character:
            item_id = character['item_id']
            image_url = character['image_url']

            # Check if the character is already assigned to the
user
            cursor.execute("SELECT 1 FROM user_characters WHERE
user_id = %s AND item_id = %s", (user_id, item_id))
            if not cursor.fetchone():
                # Assign the character to the user
                cursor.execute("INSERT INTO user_characters
(user_id, item_id, is_active) VALUES (%s, %s, TRUE)", (user_id,
item_id))
                conn.commit()

                # Send a message with the character's picture
                local_image_path = os.path.join(os.getcwd(),
image_url.replace("\\", "/"))  # Ensure correct path format for
local files
                if os.path.isfile(local_image_path):
                    await bot.send_photo(user_id,
photo=InputFile(local_image_path), caption="🎉 Мої
вітання!\nСьогодні твій 7-й день шляху до поставленної цілі, а
це значить, що ти заслуговуєш на свого першого Лексітрона!\nТак
тримати!\nТепер твої звіти стануть помітнішими, а ти матимеш
додадковий стимул йти далі, за більш крутими Лексітронами 😉")
                else:
                    print(f"Image file not found:
{local_image_path}")

async def send_reminders(reminder_text):
    process_id = os.getpid()  # Get current process ID
    print(f"Executing send_reminders from process ID:
{process_id}")
    try:
        with connect(**db_config) as conn:
            with conn.cursor(dictionary=True) as cursor:
                cursor.execute("SELECT user_id FROM
known_users")  # Assuming 'known_users' is the table name
                users = cursor.fetchall()
                print(f"Found {len(users)} users to send
reminders to from process ID: {process_id}")

                for user in users:
                    user_id = user['user_id']
                    await bot.send_message(user_id,
reminder_text)
                    print(f"Sent reminder to user ID:
{user_id}")
    except Exception as e:
        print(f"Error during send_reminders from process ID:
```

```
{process_id}: {e}")

def schedule_jobs():
    # Morning task at 7:45 AM Kiev time
    scheduler.add_job(send_morning_message_and_update, 'cron',
hour=7, minute=45, timezone=timezone('Europe/Kiev'))

    # Additional evening reminders
    scheduler.add_job(send_reminders, 'cron', hour=21, minute=0,
args=["Не забудь надіслати звіт сьогодні!"],
timezone=timezone('Europe/Kiev'))
    scheduler.add_job(send_reminders, 'cron', hour=23, minute=0,
args=["Ти шо? Тут 1 лише година залишилась, щоб надіслати звіт.
Поквапся!"], timezone=timezone('Europe/Kiev'))
    scheduler.add_job(send_reminders, 'cron', hour=23,
minute=45, args=["15 хвилин на звіт..."],
timezone=timezone('Europe/Kiev'))

    # Congratulate users who have reached their deadline
    scheduler.add_job(check_and_congratulate_deadlines, 'cron',
hour=6, minute=00, timezone=timezone('Europe/Kiev'))

    # Evaluate poll answers at 22:00 Kiev time
    scheduler.add_job(channel_admin.evaluate_answers, 'cron',
hour=22, minute=00, timezone=timezone('Europe/Kiev'))

async def check_and_congratulate_deadlines():
    today = datetime.now().date()
    print(f"Today's date: {today}")  # Debug: Print today's date

    try:
        with connect(**db_config) as conn, conn.cursor() as
cursor:
            cursor.execute("""
                SELECT user_id
                FROM known_users
                WHERE STR_TO_DATE(deadline, '%Y-%m-%d') = %s""",
(today,))
            users = cursor.fetchall()
            print(f"Users found with today's deadline: {users}")
# Debug: Print the users found

            for user in users:
                user_id, first_name = user
                message = (
                    f"🎉 <b>Вітаємо!</b> 🎉\n\n"
                    "Ти досягнув своєї мети! Ми дуже пишаємося
твоїм досягненням.\n"
                    "Продовжуй у тому ж дусі, не зупиняйся на
досягнутому! 🚀")
                await bot.send_message(user_id, message,
parse_mode='HTML')
```

```
    except Error as e:
        print(f"Database error occurred while checking
deadlines: {e}")


async def update_user_stats(user_id):
    conn = connect(**db_config)
    cursor = conn.cursor(dictionary=True)

    # Update user rank
    cursor.execute("SELECT days_of_usage, user_rank FROM
known_users WHERE user_id = %s", (user_id,))
    result = cursor.fetchone()
    days_of_usage = result['days_of_usage'] if result and
result['days_of_usage'] is not None else 0
    current_rank = result['user_rank'] if result else None
    new_rank = get_user_rank(days_of_usage)

    rank_changed = False
    if new_rank != current_rank:
        cursor.execute("UPDATE known_users SET user_rank = %s
WHERE user_id = %s", (new_rank, user_id))
        conn.commit()
        rank_changed = True

    # Update user level
    cursor.execute("SELECT points, level FROM known_users WHERE
user_id = %s", (user_id,))
    result = cursor.fetchone()
    points = result['points'] if result and result['points'] is
not None else 0
    current_level = result['level'] if result else None
    new_level = get_user_level(points)

    level_changed = False
    if new_level != current_level:
        cursor.execute("UPDATE known_users SET level = %s WHERE
user_id = %s", (new_level, user_id))
        conn.commit()
        level_changed = True

    cursor.close()
    conn.close()

    return new_rank, rank_changed, new_level, level_changed

def get_user_rank(days_of_usage):
    for rank, days in shared_resources.RANKS:
        if days_of_usage >= days:
            return rank
    return "Новатор"

def get_user_level(points):
```

```
    for level, threshold in reversed(shared_resources.LEVELS):
        if points >= threshold:
            return level
    return 1

def add_points_to_user(user_id, points_to_add):
    conn = connect(**db_config)
    cursor = conn.cursor(dictionary=True)

    # Update the user's points
    cursor.execute("UPDATE known_users SET points = points + %s
WHERE user_id = %s", (points_to_add, user_id))
    conn.commit()

    # Check and update user level based on new points
    cursor.execute("SELECT points FROM known_users WHERE user_id
= %s", (user_id,))
    result = cursor.fetchone()
    if result:
        points = result['points']
        new_level = get_user_level(points)
        cursor.execute("UPDATE known_users SET level = %s WHERE
user_id = %s", (new_level, user_id))
        conn.commit()

    cursor.close()
    conn.close()

@dp.callback_query_handler(lambda c: c.data == "main_menu")
async def show_main_menu(callback_query: types.CallbackQuery):
    user_id = callback_query.from_user.id
    await bot.send_message(user_id, "Головне меню",
reply_markup=keyboard.get_main_menu_keyboard(user_id))

dp.register_message_handler(admin_control.update_payment,
commands=['update_payment'])


if __name__ == '__main__':
    channel_admin.set_scheduler(scheduler)

    schedule_jobs()

    channel_admin.schedule_polls()

    scheduler.start()
    executor.start_polling(dp, skip_updates=True)
```

ДОДАТОК Б

ВМІСТ ФАЙЛУ SENT_REPORT.PY. ВІДПРАВКА ЗВІТУ

```
import keyboard
from aiogram import types, Dispatcher
from create_bot import dp, bot
import shared_resources
from handlers import statistic
from datetime import datetime, timedelta
import main
from collections import defaultdict
from mysql.connector import connect, Error
from shared_resources import db_config
import os
from aiogram.types import InputFile
import requests
from io import BytesIO

user_states = defaultdict(list)
user_report = {}

@dp.callback_query_handler(lambda c:
c.data.startswith('back_sentreport_'))
async def handle_back_sentreport(callback_query:
types.CallbackQuery):
    parts = callback_query.data.split('_')
    user_id = int(parts[2])
    print(f"User {user_id} requested to go back. Current states:
{user_states[user_id]}")
    await bot.answer_callback_query(callback_query.id)

    # Revert to the previous state
    if user_states[user_id]:
        user_states[user_id].pop()  # Remove the current state
    if user_states[user_id]:
        current_state = user_states[user_id][-1]  # Get the new
current state
    else:
        current_state = None  # Reset to default if no previous
states

    # Handle the previous state
    if current_state == 'TimeSpent':
        await bot.send_message(user_id, "Скільки часу сьогодні
пішло на вивчення? (цифрою в хвилинах)",
reply_markup=keyboard.get_back(user_id, 'sentreport'))
    elif current_state == 'ActivitiesDone':
        await bot.send_message(user_id, "Чим саме займався?",
reply_markup=keyboard.get_back(user_id, 'sentreport'))
    elif current_state == 'NewWords':
        await bot.send_message(user_id, "Які нові слова вивчив?
```

```
(введи через кому)", reply_markup=keyboard.get_back(user_id,
'sentreport'))
    else:
        await bot.send_message(user_id, "Ти в головному меню",
reply_markup=keyboard.get_main_menu_keyboard(user_id))
        user_states[user_id].append('Report_finished')
        print(f"User {user_id} state after handling back:
{user_states[user_id]}")

@dp.callback_query_handler(lambda c:
c.data.startswith('report_yesterday_'))
async def handle_report_yesterday(callback_query:
types.CallbackQuery):
    user = callback_query.from_user
    user_id = user.id

    if await main.check_report_and_payment(user_id):
        # Ensure user report data structure is initialized
        if user_id not in user_report:
            user_report[user_id] = {}

        # Set the report date to yesterday explicitly in the
user report data
        yesterday_date = (datetime.now() -
timedelta(days=1)).strftime('%d.%m.%Y')
        user_report[user_id]['report_date'] = yesterday_date

        # Update the state to begin report filling
        user_states[user_id].append('TimeSpent')
        prompt = "⌛ <b>Скільки часу вчора пішло на
вивчення?</b>\n(вкажи <i>цифрою</i> у хвилинах)"
        await bot.send_message(user_id, prompt,
reply_markup=keyboard.get_back(user_id,
'sentreport'),parse_mode='HTML')

        # Answer the callback query to provide instant feedback
        await bot.answer_callback_query(callback_query.id)


# Handler for collecting time spent
@dp.message_handler(lambda message:
user_states.get(message.from_user.id, [-1])[-1] == 'TimeSpent')
async def time_spent_handler(message: types.Message):
    user_id = message.from_user.id
    text = message.text.strip()

    if text.isdigit() and int(text) >= 15:  # Check if input is
numeric and meets the minimum criteria
        if user_id not in user_report:
            user_report[user_id] = {}
        user_report[user_id]['time_spent'] = int(text)
        user_states[user_id].append('ActivitiesDone')
        print(f"After appending, user {user_id} states:
```

```
{user_states[user_id]}")
        await bot.send_message(user_id,"📚 <b>Чим саме
займався?</b>",reply_markup=keyboard.get_back(user_id,
'sentreport'),parse_mode='HTML')
    else:
        print(f"After appending, user {user_id} states:
{user_states[user_id]}")
        await bot.send_message(user_id,
            "❗<b>Будь ласка, введи кількість хвилин
цифрами</b>\n"
            "і не забувай про критерій - <i>не менше 15 хвилин
занять</i>.",parse_mode='HTML')


# Handler for collecting activities done
@dp.message_handler(lambda message:
user_states.get(message.from_user.id, [-1])[-1] ==
'ActivitiesDone')
async def activities_done_handler(message: types.Message):
    user_id = message.from_user.id
    if user_id not in user_report:
        user_report[user_id] = {}
    user_report[user_id]['activities_done'] = message.text
    user_states[user_id].append('NewWords')
    print(f"After appending, user {user_id} states:
{user_states[user_id]}")
    await bot.send_message(user_id,"📝 <b>Які нові слова
вивчив?</b>\n(введи через
кому)",reply_markup=keyboard.get_back(user_id,
'sentreport'),parse_mode='HTML')

# Handler for collecting new words
@dp.message_handler(lambda message:
user_states.get(message.from_user.id, [-1])[-1] == 'NewWords')
async def new_words_handler(message: types.Message):
    user_id = message.from_user.id
    words = [word.strip() for word in message.text.split(',') if
word.strip()]

    if len(words) >= 3:
        if user_id not in user_report:
            user_report[user_id] = {}
        user_report[user_id]['new_words'] = words
        user_states[user_id].append('PerformanceRating')
        print(f"After appending, user {user_id} states:
{user_states[user_id]}")
        await bot.send_message(
            user_id,
            "📊 <b>Оціни свою ефективність сьогодні у вивченні
англійської:</b>",reply_markup=keyboard.get_rating_keyboard(user
_id, 'sentreport'),parse_mode='HTML')
    else:
```

```
        print(f"After appending, user {user_id} states:
{user_states[user_id]}")
        await bot.send_message(user_id,"⚠ <b>Ти маєш вивчити
щонайменше 3 нових слова сьогодні.</b>\n""Введи ці слова
<i>(через кому)</i>:",reply_markup=keyboard.get_back(user_id,
'sentreport'),parse_mode='HTML')

@dp.callback_query_handler(lambda c:
c.data.startswith('rating:'))
async def process_rating(callback_query: types.CallbackQuery):
    user = callback_query.from_user
    rating = int(callback_query.data.split(':')[1])  # Extract
the rating from the callback data

    if 1 <= rating <= 5:
        if user.id not in user_report:
            user_report[user.id] = {}
        user_report[user.id]['performance_rating'] = rating
        user_states[user.id].append('ReportCompleted')
        print(f"After appending, user {user.id} states:
{user_states[user.id]}")
        await bot.answer_callback_query(callback_query.id)
        await prepare_report_submission(user)
        print(f"{user_report}")
    else:
        print(f"After appending, user {user.id} states:
{user_states[user.id]}")
        await bot.send_message(user.id,"🔢 <b>Оцінка має бути
числом від 1 до
5.</b>",reply_markup=keyboard.get_rating_keyboard(user.id,
'sentreport'),parse_mode='HTML')


async def prepare_report_submission(user):
    user_id = user.id
    if user_id not in user_report:
        await bot.send_message(user_id, "Не знайдено даних звіту
для відправки",
reply_markup=keyboard.get_main_menu_keyboard(user_id))
        return

    await display_report_summary(user, user_report[user_id])

def create_report_message(user, user_report):
    conn = connect(**db_config)
    cursor = conn.cursor(dictionary=True)

    # Fetch user details along with the active character's
item_id
    cursor.execute('''
        SELECT
            ku.user_rank,
            ku.days_of_usage,
```

```
            ku.level,
            uc.item_id AS current_character_id
        FROM
            known_users ku
        LEFT JOIN
            user_characters uc
        ON
            ku.user_id = uc.user_id AND uc.is_active = TRUE
        WHERE
            ku.user_id = %s
    ''', (user.id,))
    result = cursor.fetchone()

    if result:
        user_rank = result['user_rank']
        days_of_usage = result['days_of_usage']
        level = result['level']
        current_character_id = result['current_character_id']
    else:
        user_rank = "Новатор"
        days_of_usage = 0
        level = 1
        current_character_id = None

    day_addition = statistic.get_day_addition(days_of_usage)

    report_date = user_report.get('report_date',
datetime.now().strftime('%d.%m.%Y'))
    if user.username:
        user_hashtag = f"<a
href='tg://user?id={user.id}'>#{user.username}</a>"
        user_mention = f"<a
href='tg://user?id={user.id}'>{user.username}</a>"
    else:
        user_hashtag = f"<a
href='tg://user?id={user.id}'>#{user.first_name}</a>"
        user_mention = f"<a
href='tg://user?id={user.id}'>{user.first_name}</a>"

    words_count = len(user_report.get('new_words', []))
    words_label = statistic.get_word_addition(words_count)
    performance_rating =
int(user_report.get('performance_rating', '0'))
    activities_done = user_report.get('activities_done', 'Нічого
не зазначено')

    # Fetch active character's image URL if available
    image_url = None
    if current_character_id:
        cursor.execute("SELECT image_url FROM items WHERE
item_id = %s", (current_character_id,))
        character = cursor.fetchone()
        if character:
```

```
            # Assume the image_url is a relative path
            image_url =
os.path.join(os.path.dirname(os.path.abspath(__file__)), '..',
character['image_url'])

    # Fetch counts of items by rarity
    cursor.execute('''
        SELECT
            SUM(CASE WHEN i.rarity = 'common' THEN 1 ELSE 0 END)
AS common_count,
            SUM(CASE WHEN i.rarity = 'rare' THEN 1 ELSE 0 END)
AS rare_count,
            SUM(CASE WHEN i.rarity = 'epic' THEN 1 ELSE 0 END)
AS epic_count,
            SUM(CASE WHEN i.rarity = 'legendary' THEN 1 ELSE 0
END) AS legendary_count
        FROM
            user_purchases up
        JOIN
            items i ON up.item_id = i.item_id
        WHERE
            up.user_id = %s
    ''', (user.id,))
    item_counts = cursor.fetchone()

    cursor.close()
    conn.close()

    common_count = item_counts['common_count'] or 0
    rare_count = item_counts['rare_count'] or 0
    epic_count = item_counts['epic_count'] or 0
    legendary_count = item_counts['legendary_count'] or 0

    report_message = (
        f"{user_hashtag}\n"
        f"Ранг: <b>{user_rank}</b> ({days_of_usage}
{day_addition})\n"
        f"Рівень: <b>{level}</b>\n"
        f"Кількість предметів за рідкістю:\n"
        f"⚪ <b>{common_count}</b> 🔵 <b>{rare_count}</b> 🟣
<b>{epic_count}</b> 🟡 <b>{legendary_count}</b>\n"
        f"-----------------------------------------------------
------------\n\n"
        f"<b>📅 Звіт за {report_date}</b>\n"
        f"              ----------------------              \n"
        f"Сьогодні {user_mention} займався(лася) наступним:\n"
        f"<i>{activities_done}</i>\n"
        f"              ----------------------              \n"
        f"⏳ На все про все витративши
<b>{user_report.get('time_spent', '0')} хвилин</b>\n"
        f"              ----------------------              \n"
        f"📚 При тому було вивчено <b>{words_count}
```

```
{words_label}</b>\n"
        f"                    ----------                    \n"
        f"🏆 За виконану роботу {user_mention} оцінив(ла) себе
на <b>{performance_rating}/5</b>\n"
        f"-------------------------------------------------------
------------\n"
    )

    return report_message, image_url


async def display_report_summary(user, user_report):
    report_message, image_url = create_report_message(user,
user_report)

    if image_url:
        # Check if the image_url is a local path
        image_path = os.path.abspath(image_url)
        if os.path.isfile(image_path):
            await bot.send_photo(user.id,
photo=InputFile(image_path), caption=report_message,

reply_markup=keyboard.get_report_editing_keyboard(),
parse_mode='HTML')
        else:
            await bot.send_message(user.id, "Image file not
found.")
    else:
        await bot.send_message(user.id, report_message,
reply_markup=keyboard.get_report_editing_keyboard(),
                               parse_mode='HTML')

@dp.callback_query_handler(lambda c: c.data.startswith('edit_'))
async def edit_report_section(callback_query:
types.CallbackQuery):
    user_id = callback_query.from_user.id
    action = callback_query.data.split('_')[1]
    user_states[user_id] = action  # Set the state to the
current action

    prompt_map = {
        "time": "⏳ <b>Введи новий витрачений час</b> <i>(у
хвилинах)</i>:",
        "words": "📝 <b>Введи нові вивчені слова</b> <i>(через
кому)</i>:",
        "performance": "🔢 <b>Введи нову оцінку</b> <i>(за
шкалою від 1 до 5)</i>:"
    }
    prompt = prompt_map.get(action, "⚠️ <b>Невірна дія</b>")

    await bot.send_message(user_id, prompt, parse_mode='HTML')
```

```
@dp.message_handler(lambda message:
user_states.get(message.from_user.id, '') in ['time', 'words',
'performance'])
async def process_edit_input(message: types.Message):
    user = message.from_user
    user_id = user.id
    action = user_states.get(user_id)
    data = message.text.strip()

    try:
        if action == "time":
            user_report[user_id]['time_spent'] = int(data)  #
Convert and validate time spent as integer
        elif action == "words":
            words = [word.strip() for word in data.split(',') if
word.strip()]
            if len(words) < 3:
                await message.reply("You must learn at least 3
new words. Please enter the words (comma-separated):")
                return  # Keep the state active for re-entry
            user_report[user_id]['new_words'] = words
        elif action == "performance":
            rating = int(data)
            if 1 <= rating <= 5:
                user_report[user_id]['performance_rating'] =
rating
            else:
                raise ValueError("Performance rating must be
between 1 and 5")
    except ValueError as ve:
        await message.reply(f"Invalid input: {ve}")
        return

    # Clear the state
    user_states[user_id] = []

    # Call the summary function with the user object, not the ID
    await display_report_summary(user, user_report[user_id])


@dp.callback_query_handler(lambda c: c.data == 'confirm_report')
async def finalize_report(callback_query: types.CallbackQuery):
    user = callback_query.from_user
    if user.id in user_report:
        await send_report_to_sheet(user)  # Pass the user object
        main.add_points_to_user(user.id, 5)
        del user_report[user.id]  # Clean up after sending
    else:
        await callback_query.message.answer("Report data is
missing.")


async def send_report_to_sheet(user):
```

```
    user_id = user.id
    if user_id in user_report:
        conn = None
        try:
            time_spent =
int(user_report[user_id].get('time_spent', 0))
            new_words_count =
len(user_report[user_id].get('new_words', []))
            performance_rating =
int(user_report[user_id].get('performance_rating', 0))

            # Connect to the MySQL database
            conn = connect(**db_config)
            cursor = conn.cursor()

            # Insert user stats into the database
            today_str = datetime.now().strftime('%Y-%m-%d')
            queries = [
                ('INSERT INTO user_stats (user_id, date, metric,
value) VALUES (%s, %s, %s, %s)',
                 (user_id, today_str, 'time_spent',
time_spent)),
                ('INSERT INTO user_stats (user_id, date, metric,
value) VALUES (%s, %s, %s, %s)',
                 (user_id, today_str, 'new_words_count',
new_words_count)),
                ('INSERT INTO user_stats (user_id, date, metric,
value) VALUES (%s, %s, %s, %s)',
                 (user_id, today_str, 'performance_rating',
performance_rating))
            ]

            for query, params in queries:
                cursor.execute(query, params)

            conn.commit()

            # Send report to channel
            await send_report_to_channel(user,
user_report[user_id])
            await bot.send_message(user_id, "Звіт вже у групі
😉\n А ти тримай свої заслужені 5 Лінгонів 💥💪",
reply_markup=keyboard.get_main_menu_keyboard(user_id))
            user_report[user_id] = {"id": user_id}

        except Error as db_err:
            await bot.send_message(user_id, f"Виникла помилка
бази даних: {db_err}")
        except Exception as e:
            await bot.send_message(user_id, f"Виникла помилка:
{e}")
        finally:
            if conn and conn.is_connected():
```

```
                conn.close()
    else:
        await bot.send_message(user_id, "Не знайдено даних звіту
для відправки",
reply_markup=keyboard.get_main_menu_keyboard(user_id))

async def send_report_to_channel(user, user_report):
    REPORT_CHANNEL_ID = shared_resources.REPORT_CHANNEL_ID
    report_message, character_image_url =
create_report_message(user, user_report)

    try:
        # Connect to the MySQL database and update the last
report date
        with connect(**db_config) as conn:
            cursor = conn.cursor(dictionary=True)  # Ensure the
cursor returns dictionaries

            # Update the last_report_date in the database only
if the report is for today
            today = datetime.now().strftime('%Y-%m-%d')
            if user_report.get('report_date') == today:
                cursor.execute('UPDATE known_users SET
last_report_date = %s WHERE user_id = %s',
                               (today, user.id))
                conn.commit()

        # Function to resolve the full path to the image
        def resolve_image_path(image_path):
            image_absolute_path =
os.path.join(os.path.dirname(os.path.abspath(__file__)), '..',
image_path)
            if os.path.isfile(image_absolute_path):
                return image_absolute_path
            raise ValueError("Image could not be loaded or does
not exist.")

        # Send the report message with the character image (if
available) to the designated channel
        if character_image_url:
            image_path = resolve_image_path(character_image_url)
            await bot.send_photo(REPORT_CHANNEL_ID,
photo=InputFile(image_path), caption=report_message,
parse_mode='HTML')
        else:
            await bot.send_message(REPORT_CHANNEL_ID,
report_message, parse_mode='HTML')

        if user.id not in user_states or user_states[user.id] is
None:
            user_states[user.id] = []
        user_states[user.id].append("-")
```

```
    except Error as db_err:
        print(f"Database error occurred: {db_err}")
    except Exception as e:
        print(f"Failed to send report to channel or update the
sheet with the report date: {e}")

async def send_reminders(message):
    today = datetime.now().strftime('%Y-%m-%d')
    query = """
    SELECT user_id FROM known_users
    WHERE last_report_date IS NULL OR last_report_date < %s;
    """
    user_ids = []  # List to hold user_ids for debugging
    with connect(**db_config) as conn:
        with conn.cursor() as cursor:
            cursor.execute(query, (today,))
            users_needing_reminders = cursor.fetchall()
            # Unpacking tuples to retrieve user_ids
            user_ids = [user[0] for user in
users_needing_reminders]

    # Print all user_ids retrieved for debugging
    print(f"User IDs needing reminders: {user_ids}")

    # Iterate through user_ids to send reminders
    for user_id in user_ids:
        try:
            await bot.send_message(user_id, message)
        except Exception as e:
            print(f"Failed to send reminder to user {user_id}:
{e}")


def register_handlers_sent_report(dp: Dispatcher):
    dp.register_message_handler(time_spent_handler,
                                lambda message:
user_states.get(message.from_user.id, [-1])[-1] == "TimeSpent")
```