Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
ННІ "Комп'ютерних наук та штучного інтелекту"
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
В.о. завідувача кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. М. М. Хруслов
«___» _____ 2025 р

Пояснювальна записка
до кваліфікаційної роботи
бакалавра

на тему: "**АВТОМАТИЗОВАНИЙ КРОСПЛАТФОРМНИЙ ЗАСТОСУНОК ДЛЯ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ З ІНСТРУМЕНТАМИ АДМІНІСТРУВАННЯ**"

Спеціальність 151 –Автоматизація та комп'ютерно-інтегровані технології
Галузь знань 15 – Автоматизація та приладобудування
Освітня програма «Автоматизація та комп'ютерно-інтегровані технології»

Захищено на засіданні
Атестаційної комісії № 46
протокол № __ від __.06.2025 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ ЧУГАЙ А.М.

Виконав:
студент групи КУ–41
ЖУКОВ Олег Ігорович

Керівник: к.ф.-м.н., доцент, в.о. зав. кафедри комп'ютерних систем та робототехніки
ХРУСЛОВ Максим Михайлович

Рецензент: к.т.н., доцент ЗВО кафедри штучного інтелекту та програмного забезпечення
МАЦИЙ Ольга Борисівна

Харків – 2025

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається з наступних елементів: вступу, 3 розділів, висновків, списку використаних джерел і 3 додатків. Загальний обсяг роботи складає 83 сторінки, із яких 50 сторінок основної частини з 23 рисунками, 1 таблицею, 17 найменуваннями списку використаних джерел та трьома додатками.

Об'єкт дослідження – процес розробки автоматизованого кросплатформного застосунку зі вбудованими інструментами адміністрування.

Предмет дослідження – концептуальні моделі, технологічні рішення та архітектурні підходи до розробки кросплатформних додатків, а також механізми адміністрування користувачів і навчального контенту.

Мета роботи – розробка автоматизованого кросплатформного застосунку для вивчення іноземної мови з інтеграцією в Telegram та вбудованими інструментами адміністрування навчального контенту та перегляду результатів тестування користувачів.

Проблема, яку вирішує кваліфікаційна робота – відсутність доступного гнучкого інтегрованого рішення, що поєднує сучасні методики викладання іноземних мов та зручні інструменти адміністрування навчального контенту й користувачів на різних платформах, яке може бути використане як готовий продукт для бізнес-задач у сфері освіти.

Область застосування – дистанційна–онлайн освіта, корпоративне навчання, курси, мобільні та веб-орієнтовані платформи для вивчення іноземних мов та інших освітніх напрямків.

Ключові слова: кросплатформний застосунок, вивчення іноземних мов, тестування знань, мобільний додаток, веб-додаток, перегляд інформації про користувачів.

ABSTRACT

The explanatory note to the bachelor's qualification thesis consists of an introduction, 3 chapters, conclusions, a list of references, and 3 appendices. The total volume of the thesis is 83 pages, including 50 pages of the main part, comprising 23 figures, 1 tables, 17 items in the list of references, and three appendices.

The object of the study – an automated cross-platform application with built-in administrative tools.

The subject of the research – conceptual models, technological solutions and architectural approaches to developing cross-platform applications, as well as mechanisms for administering users and educational content.

The purpose of the work – development of an automated cross-platform application for learning a foreign language with integration with Telegram and built-in tools for administering educational content and viewing user testing results.

Problem solved by qualification thesis – the lack of an accessible, flexible, and integrated solution combining modern methods of teaching foreign languages and convenient tools for educational content administration and viewing user information and testing results across multiple platforms, which can be utilized as a ready-made product for educational business tasks.

Field of application – distance online education, corporate training, educational courses, mobile and web-based platforms for foreign language learning and other educational fields.

Keywords: cross-platform application, foreign language learning, knowledge testing, content administration, viewing user information, mobile application, web application.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface – набір визначень та протоколів для створення і взаємодії програмного забезпечення між собою.

UX – User Experience – загальне враження, що отримує користувач від взаємодії із застосунком або веб-сервісом.

DB – Database – організоване сховище даних, призначене для ефективного зберігання, пошуку та обробки інформації.

MongoDB – документо-орієнтована нереляційна база даних, яка використовує JSON-подібні документи для зберігання даних, широко застосовується в сучасних веб-застосунках.

HTTP – HyperText Transfer Protocol – протокол передачі даних, який використовується для обміну інформацією в Інтернеті між клієнтом та сервером.

HTTPS – HyperText Transfer Protocol Secure – захищена версія HTTP, яка забезпечує конфіденційність та цілісність даних шляхом використання шифрування.

REST – Representational State Transfer – архітектурний стиль, що визначає набір обмежень і правил, яких потрібно дотримуватись для створення гнучких веб-сервісів та API.

JSON – JavaScript Object Notation – текстовий формат обміну даними, який широко використовується для передачі інформації між клієнтом та сервером завдяки простоті й компактності.

SDK – Software Development Kit – набір інструментів, бібліотек та документації, що використовується розробниками для створення програмного забезпечення на певній платформі.

CRUD – Create, Read, Update, Delete – основні операції, що використовуються при роботі з даними у програмному забезпеченні (створення, читання, оновлення, видалення записів).

JS – JavaScript – динамічна мова програмування, яка широко використовується для створення інтерактивних веб-сторінок і веб-застосунків.

IDE – Integrated Development Environment – програмне середовище, яке забезпечує зручний набір інструментів (редактор, компілятор, відлагоджувач тощо) для розробки програмного забезпечення.

Bot API – Telegram Bot Application Programming Interface – офіційний набір інструментів від Telegram, призначений для створення інтерактивних ботів, які можуть автоматизувати завдання, взаємодіючи з користувачами.

SPA – односторінковий додаток (веб-додаток, який завантажує одну HTML-сторінку та динамічно оновлюється під час взаємодії користувача з додатком).

Хуки – функції в React, які дозволяють розробникам використовувати функції стану та життєвого циклу у функціональних компонентах.

Бекенд – серверна частина системи, яка обробляє запити, керує базами даних та обробляє бізнес-логіку.

Фронтенд – клієнтська частина системи, яка безпосередньо взаємодіє з користувачем, включаючи інтерфейс користувача та користувацький досвід.

ВСТУП

Актуальність роботи. У сучасних умовах стрімкого розвитку інформаційних технологій особливої популярності набувають освітні продукти, орієнтовані на дистанційну та змішану форми навчання. Це особливо помітно у сфері вивчення іноземних мов, де важливу роль відіграє інтерактивність та доступність освітнього контенту на різних платформах. Проте, більшість існуючих рішень не забезпечують достатнього рівня інтеграції між сучасними методиками навчання та зручністю адміністрування навчального контенту й результатів тестування користувачів, що знижує їх ефективність та гнучкість використання у різних освітніх сценаріях.

Таким чином, створення автоматизованого кросплатформного застосунку, інтегрованого з популярною платформою Telegram, яка останніми роками набуває значної популярності серед користувачів, зокрема в Україні, де на цій платформі активно створюються та розвиваються найпопулярніші новинні канали, що поєднує сучасні методи викладання іноземних мов, тестування знань і зручні адміністративні інструменти перегляду користувачів та результатів їхнього тестування, є актуальною та важливою задачею.

Мета роботи. Розробка автоматизованого кросплатформного застосунку для вивчення іноземної мови з інтеграцією в Telegram та вбудованими інструментами адміністрування навчального контенту та перегляду результатів тестування користувачів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– провести аналіз існуючих підходів та технологічних рішень у сфері розробки крос-платформних освітніх застосунків; – розробити концептуальну модель застосунку та визначити архітектурні рішення для його реалізації з урахуванням інтеграції в Telegram; – розробити та реалізувати інтерфейсну частину мобільного та веб-застосунків, інтегрованих з Telegram; – реалізувати серверну

частину з базою даних та забезпечити взаємодію клієнт-сервер з підтримкою інтеграції в Telegram; – інтегрувати в застосунок систему тестування знань з можливістю перегляду результатів користувачів; – провести тестування розробленого застосунку.

Об'єкт дослідження – автоматизований кросплатформний застосунок для вивчення іноземної мови, інтегрований з платформою Telegram, зі вбудованими адміністративними інструментами.

Предмет дослідження – моделі, технології та архітектурні рішення для створення автоматизованих крос-платформних застосунків навчального призначення з інтеграцією в Telegram та інструментами адміністрування контенту й результатів тестування.

Методи дослідження – аналіз існуючих рішень, системний аналіз, методи програмної інженерії, проєктування програмного забезпечення, методи тестування програмних продуктів.

Практичне значення роботи полягає у створенні готового програмного продукту, що може бути використаний для вирішення актуальних бізнес-задач у сфері дистанційної та змішаної освіти, корпоративного навчання та індивідуальних курсів вивчення іноземних мов через зручну та популярну платформу Telegram.

РОЗДІЛ 1
АНАЛІТИЧНИЙ РОЗДІЛ

1.1. Аналіз предметної області

В епоху цифрових технологій попит на зручні та ефективні методи вивчення іноземних мов стрімко зростає. Це зростання підживлюється глобалізацією, міжнародною співпрацею, віддаленою роботою та транскордонною мобільністю. Люди все частіше шукають гнучкі навчальні рішення, доступні будь-коли, будь-де та на різних платформах. Як результат, освітні технології (EdTech) стали одним із секторів індустрії програмного забезпечення, що найшвидше розвиваються, а програми для вивчення мов займають значну нішу [1].

Традиційна мовна освіта зазвичай передбачає відвідування мовних шкіл, найм репетиторів або реєстрацію на онлайн-курси через веб-сайти, які вимагають реєстрації користувачів та окремих установок. Ці підходи можуть бути незручними та негнучкими, особливо для користувачів, які віддають перевагу швидкому доступу через платформи, які вони вже використовують щодня. Сучасні учні вимагають інтерактивних, доступних та адаптивних систем, які можуть працювати в рамках їхніх щоденних цифрових процедур.

Основною вимогою в галузі мовної освіти є адаптивність до індивідуального прогресу та можливість пропонувати різні форми взаємодії: аудіювання, читання, письмо та тестування. Крім того, викладачам або системним адміністраторам часто потрібні інструменти для перегляду статистики та оцінки ефективності доставки контенту шляхом моніторингу результатів тестів або журналів активності. На жаль, більшість систем вивчення мов не пропонують вбудованих функцій для адміністративного контролю, натомість вимагаючи сторонніх систем або додаткових інформаційних панелей для візуалізації даних.

Telegram став надзвичайно актуальною та практичною платформою для інтеграції навчальних систем. Маючи понад 500 мільйонів активних користувачів у всьому світі та особливо високі показники впровадження в таких регіонах, як Україна, Східна Європа та Центральна Азія, він пропонує звичне та надійне середовище для користувачів. Можливості веб-додатків Telegram та Bot API дозволяють безперешкодно вбудовувати інтерактивні додатки в інтерфейс месенджера, дозволяючи розробникам надавати освітні послуги безпосередньо у вікні чату, минаючи традиційні процеси встановлення додатків. Це значно знижує бар'єр входу для кінцевих користувачів [2].

Запропонована система працює в цьому сучасному контексті. Вона має на меті поєднати захопливий освітній досвід для студентів з практичними інструментами для адміністраторів для перегляду інформації про користувачів та результатів тестування. Інтеграція з Telegram гарантує, що користувачі можуть отримати доступ до навчального середовища одним дотиком, а інструменти бекенду забезпечують належне управління контентом та активністю користувачів.

Підсумовуючи, сфера діяльності цього проєкту лежить на перетині мобільної та веб-розробки, управління освітнім контентом та інтеграції чат-ботів. Мета полягає в тому, щоб реагувати на потреби цифрових учнів та викладачів, що розвиваються, пропонуючи кросплатформне, доступне та адміністративно життєздатне рішення для вивчення мов.

1.2. Аналіз існуючих систем та рішень

Протягом останнього десятиліття з'явилися різні цифрові платформи для підтримки вивчення мов, кожна з яких має свій власний підхід, педагогічну модель та набір функцій. Деякі з найбільш визнаних та використовуваних систем включають Duolingo, Babbel, Memrise, Rosetta Stone та Busuu. Ці програми зарекомендували себе, пропонуючи інтерактивний навчальний досвід, зручні інтерфейси, елементи гейміфікації та структуровані мовні курси [3].

Duolingo — одна з найпопулярніших платформ для вивчення мов, відома своєю ігровою структурою та щоденними завданнями. Вона пропонує вправи на словниковий запас, граматику, вимову та переклад. Однак їй бракує налаштування для конкретних навчальних закладів та обмежений доступ до індивідуального прогресу користувача, окрім основних показників.

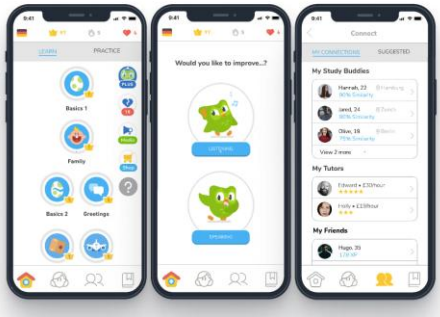

Рисунок 1.1 – Головні меню застосунку Duolingo.

Babbel більше зосереджується на розмовних навичках та граматиці, зі структурованими уроками, які імітують традиційний підхід до роботи в класі. Хоча Babbel ефективний для окремих учнів, його нелегко адаптувати для інтеграції в сторонні системи або користувацькі освітні контексти.

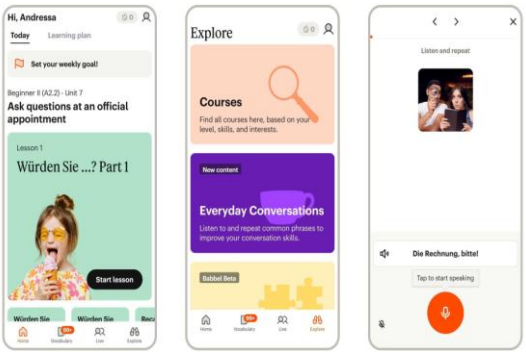

Рисунок 1.2 – Головні меню застосунку Babbel.

Memrise включає інтервальне повторення та мультимедійні елементи, такі як відео носіїв мови. Хоча він і захопливий, йому бракує адміністративних інструментів для вчителів або творців контенту, щоб детально відстежувати прогрес учнів.

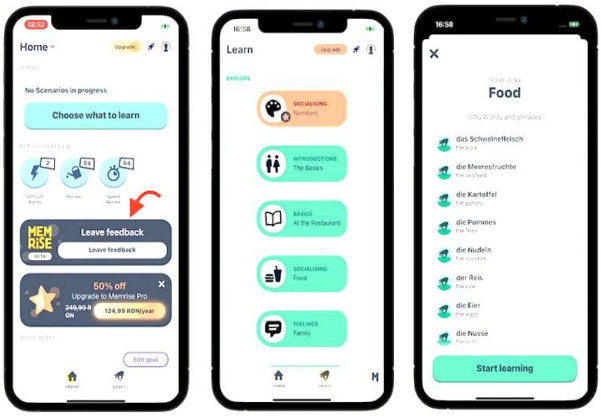

Рисунок 1.3 – Головні меню застосунку Memrise.

Rosetta Stone — це давно відома платформа, що пропонує захопливий мовний досвід. Вона відома своїми методами візуальних асоціацій, але вимагає платної підписки та є менш гнучкою з точки зору інтеграції або розгортання на альтернативних платформах, таких як месенджери.

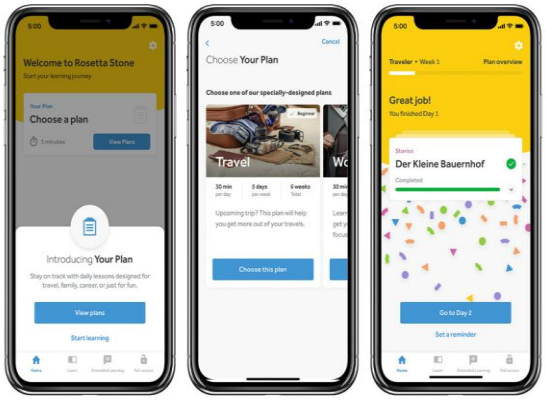

Рисунок 1.4 – Головні меню застосунку Rosetta Stone.

Busuu пропонує навчання на основі спільноти та зворотний зв'язок на основі штучного інтелекту. Хоча вона включає певне відстеження прогресу та взаємодію з однолітками, адміністративна сторона залишається обмеженою та в основному зосереджена на самостійному навчанні.

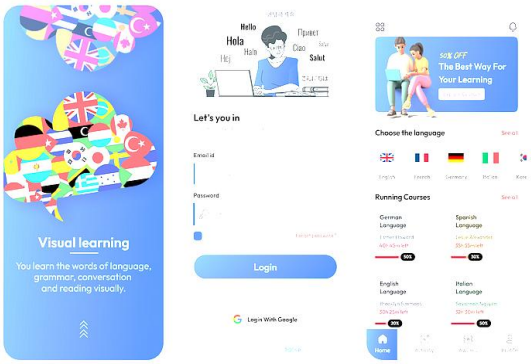

Рисунок 1.5 – Головні меню застосунку Busuu.

У всіх цих системах є кілька спільних обмежень:

1. Залежність від окремих програм, які необхідно завантажити та встановити;
2. Відсутність вбудованої інтеграції з платформами обміну повідомленнями, такими як Telegram;
3. Обмежені або відсутні адміністративні інструменти для перегляду детальних результатів тестування або індивідуальної успішності;
4. Закриті екосистеми, що перешкоджають легкому налаштуванню або розширенню для навчальних закладів;

Натомість, рішення, запропоноване в цій роботі, зосереджене на простоті доступу, інтеграції з веб-додатками Telegram та включенні адміністративних функцій. Воно розроблене таким чином, щоб бути легким, настроюваним та миттєво доступним через звичне середовище спілкування, що робить його придатним як для учнів, так і для адміністраторів контенту.

Таблиця 1.1

Таблиця порівняння основний характеристик платформ:

Платформа	Гейміфікація	Інтеграція з Telegram	Налаштовуваний контент	Потрібне встановлення
Duolingo	Так	Ні	Так	Так
Babbel	Частково	Ні	Так	Так
Memrise	Так	Ні	Частково	Так
Rosetta Stone	Ні	Ні	Ні	Так
Busuu	Так	Ні	Частково	Так
Запропонована система	Частково	Так	Так	Ні

Аналіз існуючих систем виявляє прогалину в інструментах, які поєднують доступність, адміністративні можливості та інтеграцію з повсякденними платформами, такими як Telegram. Цей пробіл створює можливість для розробки більш універсального та контекстно-залежного рішення для вивчення мови, адаптованого до сучасних освітніх та технологічних вимог.

1.3. Вибір технологій та обґрунтування засобів розробки

Розробка кросплатформного освітнього застосунку вимагає ретельного вибору технологій, що забезпечують швидкість реагування, зручність обслуговування та легкість інтеграції. Мета полягає в тому, щоб створити легку систему, до якої легко отримати доступ через месенджер Telegram, зберігаючи при цьому масштабованість бекенду та гнучкість фронтенду.

Для клієнтської частини (фронтенд) використовуються такі технології:

1. React – бібліотека JavaScript для створення сучасних компонентних інтерфейсів користувача. Вона дозволяє створювати динамічні та інтерактивні користувацькі інтерфейси.
2. Vite – сучасний інструмент для збірки фронтенду та сервер розробки, який значно підвищує швидкість збірки та підтримує гарячу заміну модулів.
3. Telegram Web App API – дозволяє вбудовувати застосунок безпосередньо в месенджер Telegram, забезпечуючи безперебійний користувацький досвід без встановлення додаткового програмного забезпечення [4].

Для серверної частини (бекенд):

1. Node.js – середовище виконання, яке дозволяє використовувати JavaScript на сервері, сприяючи швидким, асинхронним операціям.
2. Express.js – мінімальний та гнучкий фреймворк веб-застосунків, що надає надійний набір функцій для розробки API.
3. Telegraf.js – фреймворк для створення ботів Telegram, корисний для фонової взаємодії та адміністративних сповіщень [5].

Для бази даних та управління даними:

1. MongoDB – документоорієнтована NoSQL база даних, що пропонує гнучку схему проектування, придатну для зберігання профілів користувачів, тестових даних та метаданих контенту.
2. Mongoose – бібліотека моделювання об'єктних даних (ODM) для MongoDB та Node.js, яка спрощує перевірку даних, запити та забезпечення бізнес-логіки [6].

Додаткові інструменти та сервіси:

1. Postman – використовується для тестування кінцевих точок RESTful API під час розробки.
2. Git + GitHub – для контролю версій та командної співпраці [7].

Ці технології були обрані на основі їхнього відкритого коду, сильної підтримки спільноти розробників, простоти інтеграції та сумісності з

кросплатформним розгортанням. Разом вони забезпечують надійну основу для впровадження адаптивної, безпечної та масштабованої освітньої платформи, яка безперебійно працює як у Telegram, так і в стандартному середовищі браузера.

Висновки за розділом 1

У цьому аналітичному розділі розглянуто сферу сучасного цифрового вивчення мов, висвітлюючи поточні проблеми, з якими стикаються учні та викладачі у сфері доступу, налаштування та ефективного адміністрування контенту. У ньому було визначено зростаючу популярність таких платформ, як Telegram, та підкреслено їх потенціал як легкого, доступного середовища для розміщення освітніх послуг.

Огляд існуючих систем виявив як їхні сильні сторони, так і обмеження. Хоча популярні платформи пропонують захопливі інтерфейси та ефективні моделі навчання, їм зазвичай бракує вбудованих адміністративних інструментів та безшовної інтеграції з комунікаційними платформами. Цей аналіз підкреслює необхідність нової системи, яка б усувала ці прогалини.

Для вирішення цієї проблеми запропонована система була концептуалізована з акцентом на простоту, розширюваність та інтеграцію. Вибір технологій з відкритим кодом, таких як React, Node.js, MongoDB та Telegram Web App API, забезпечує реалізацію гнучкого та масштабованого рішення. Обрані інструменти забезпечують динамічну взаємодію клієнт-сервер, ефективне управління даними та інтуїтивно зрозумілий досвід як для учнів, так і для адміністраторів.

На завершення, проведений аналіз підтверджує доцільність та актуальність розробки інтегрованого в Telegram, кросплатформного освітнього додатку, який покращує вивчення мов, водночас надаючи надійні інструменти для адміністративного контролю.

РОЗДІЛ 2
ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Постановка задачі та функціональні вимоги

Метою цього проєкту є розробка кросплатформного освітнього застосунку для вивчення іноземних мов з інтегрованими інструментами для моніторингу прогресу користувачів та доставки контенту через Telegram. Система розроблена для підтримки як учнів, так і адміністраторів, забезпечуючи безперебійну комунікацію та оцінювання в режимі реального часу.

Основні цілі системи включають:

1. Спрощення доступу до навчальних матеріалів з мов через інтеграцію з веб-додатком Telegram
2. Надання інтерактивних мовних тестів, що оцінюють знання та прогрес
3. Надання адміністративного доступу до інформації про користувачів та результатів тестів
4. Підтримка як настільних, так і мобільних платформ без необхідності встановлення
5. Забезпечення масштабованої архітектури для майбутніх розширень

Основні функціональні вимоги:

1. Реєстрація та ідентифікація користувачів – ідентифікація користувачів через контекст Telegram або резервний доступ до браузера
2. Керування мовними тестами – створення, відображення та оцінка тестів, пов’язаних з мовою
3. Відстеження прогресу – відстеження результатів окремих користувачів з часом
4. Адміністративний доступ – дозвіл призначеним користувачам переглядати результати тестів та списки користувачів

5. Сповіщення (необов’язково) – надання нагадувань або оновлень користувачам через бота Telegram
6. Кросплатформна сумісність – забезпечення повної працездатності в мобільних та настільних додатках Telegram, а також у стандартних веб-браузерах

Цей набір вимог визначає обсяг системи та служить основою для її архітектурного та функціонального дизайну, який буде описано в наступних розділах.

2.2. Постановка задачі та функціональні вимоги

Концептуальна модель застосунку визначає його основні компоненти та зв'язки між ними. Запропонована система базується на модульній архітектурі, яка включає клієнтський інтерфейс (Telegram Web App), бекенд-сервіси (API та логіка бота) та підключення до публічного або приватного Telegram-каналу. Ця тісна інтеграція дозволяє функціонувати як освітнім, так і адміністративним робочим процесам в єдиному, оптимізованому середовищі.

В основі системи лежать такі концептуальні сутності:

1. Користувач – особа, ідентифікована Telegram, яка взаємодіє з застосунком. Додаткова реєстрація не потрібна; контекст Telegram дозволяє негайно персоналізувати та обробляти сеанси.
2. Адміністратор – спеціальний тип користувача, який має підвищені права доступу, такі як можливість переглядати результати тестування та публікувати матеріали в Telegram-каналі.
3. Тест – структурований набір питань, пов'язаних з вивченням мови, призначений для оцінки словникового запасу, граматики та розуміння прочитаного.
4. Результат тесту – результат завершеного тесту, пов'язаний з певним користувачем, включаючи бали та час завершення.

5. Навчальний матеріал – контент (текст, зображення або посилання), підготовлений та надісланий через бота користувачам або безпосередньо опублікований у Telegram-каналі.
6. Telegram-канал – інтегрований канал, який виступає як засобом мовлення, так і адміністративним каналом. Його можна використовувати для публікації мовних порад, планування оновлень або доставки оголошень про тести.

Концептуальна модель системи зосереджена на трьох основних взаємодіях:

1. Користувач – Бот/Веб-додаток: користувачі виконують тести та отримують відгуки через інтерфейс веб-додатку Telegram, який обслуговується в чатах Telegram.
2. Адміністратор – Бот: адміністратори використовують той самий інтерфейс (або спеціальні команди) для перегляду результатів тестів та надсилання сповіщень або повідомлень у Telegram-канал.
3. Система – Telegram-канал: бекенд-система підтримує автоматичну публікацію змістовних повідомлень (текст, зображення, кнопки, попередній перегляд) у Telegram-каналі або за запитом адміністратора.

Ця модель забезпечує:

1. Кросплатформний доступ без залежності від магазинів додатків
2. Зворотній зв'язок та збір даних у режимі реального часу
3. Інтегровану доставку освітнього контенту через екосистему обміну повідомленнями
4. Спрощене адміністративне управління

Додатковою концептуальною особливістю системи є її здатність динамічно адаптувати доставку контенту на основі активності каналу або статистики використання ботів. Це відкриває можливості для планування контенту, рекомендацій на основі ефективності або механік гейміфікації в майбутніх версіях.

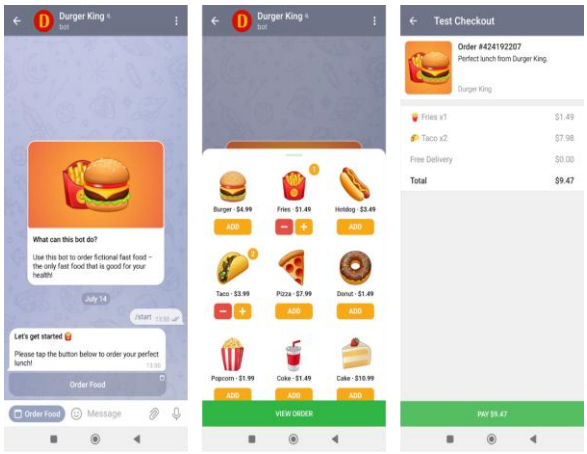

Рисунок 2.1 – Приклад WebApp застосунку, для покупки/замовлення їжі у закладі.

Така архітектура дозволяє системі масштабуватися горизонтально, розширюючи типи контенту, підтримувати багатомовні набори даних та, за бажанням, інтегрувати інструменти аналітики для відстеження залученості користувачів. Telegram у цьому контексті служить не лише точкою входу, але й власним рівнем доставки як для освіти, так і для комунікації.

Концептуальний дизайн забезпечує логічну основу для етапу впровадження та спрямовує рішення, пов'язані з дизайном API, схемою бази даних та шаблонами взаємодії з інтерфейсом [8, 9].

2.3. Архітектура системи

Архітектура системи розроблена для підтримки модульності, масштабованості та безшовної інтеграції з Telegram. Вона відповідає типовій клієнт-серверній моделі, доповненій можливостями Bot API та веб-додатку Telegram. Система розділена на кілька рівнів, які працюють разом для забезпечення запланованої функціональності.

1. Клієнтський рівень

Цей рівень включає інтерфейс веб-додатку Telegram, який використовується як звичайними користувачами, так і адміністраторами. Він розроблений за допомогою React та TypeScript і працює безпосередньо в Telegram

або стандартному браузері. Інтерфейс взаємодіє з серверною частиною через HTTPS-запити та спирається на контекст ініціалізації веб-додатку Telegram для автентифікації користувачів.

2. Рівень бота

Бот, реалізований на Telegraf.js, виконує роль посередника між Telegram та сервером: обробляє команди start, надсилає сповіщення, пересилає адміністративні запити та може працювати самостійно або разом із веб-додатком для фонової логіки та планового розповсюдження контенту.

3. Рівень API та бізнес-логіки

Рівень API, створений на Node.js і Express.js, обробляє тестування, керує користувачами та правами доступу, а також забезпечує зв’язок з базою MongoDB. Усі запити захищені та адаптовані до ролі користувача.

4. Рівень бази даних

MongoDB використовується для зберігання даних про користувачів, тести, результати та публікації. Структурування схем здійснено через Mongoose, що забезпечує типізацію та надійність.

5. Інфраструктура Telegram

Telegram інтегрується на всіх рівнях: WebApp використовується для авторизації та проходження тестів, бот — для сповіщень та керування, а канал — для публікації навчальних матеріалів. Це забезпечує безперервний, зручний і захищений освітній процес.

Модель розгортання: бекенд і бот розгортаються в хмарному середовищі, сумісному з Node.js (наприклад, Render, Railway або Heroku), тоді як фронтенд може обслуговуватися статично з того ж або окремого хоста. MongoDB Atlas або власний екземпляр керує базою даних.

Ця архітектура забезпечує:

1. Швидкий доступ для користувачів без встановлення
2. Централізоване керування контентом та аналітикою

3. Масштабоване розгортання через безвідмовний бекенд та документоорієнтовану базу даних
4. Низький поріг входу для впровадження через Telegram

2.4. Моделі взаємодії компонентів

Додаток використовує базу даних MongoDB із чітко визначеними схемами Mongoose для керування основними об'єктами: користувачами, тестами, результатами тестів та розподілом контенту. Кожна схема розроблена для гнучкості та масштабованості з чітким розподілом завдань.

Схема користувача

Ця схема використовує унікальний ідентифікатор користувача з Telegram для відстеження його активності в системі. Вона зберігає ім'я користувача та інформацію про те, чи заблокував він бота. Також система фіксує час реєстрації та останнього оновлення даних. Це дозволяє легко пов'язати тестові результати з конкретним користувачем, а також використовувати прапорець блокування для фільтрації користувачів при розсилці повідомлень. Ідентифікація користувачів через Telegram допомагає ефективно здійснювати управління контентом та взаємодію.

Схема тестування

Схема тестування описує основні метадані, пов'язані з кожним тестом, такими як його назва, опис, категорія та тип. Також зберігається інформація про візуальне оформлення тесту (наприклад, колір підсвітки або зображення обкладинки). Кожен тест містить посилання на питання, що є окремими об'єктами в системі, що дозволяє зберігати їх в базі даних та динамічно додавати нові. Цей підхід дає змогу гнучко керувати змістом тестів, адаптуючи систему під різні освітні сценарії.

Схема проходження тесту

Схема проходження тесту зберігає результати кожного користувача, пов'язані з конкретним тестом. Вона містить інформацію про тест, який

користувач пройшов, його відповіді, а також отриманий бал у відсотках. Ці дані використовуються для моніторингу успішності учасників тестування та аналізу їхнього прогресу. Масив відповідей дозволяє глибше аналізувати вибори користувачів та відслідковувати помилки, що зробили учасники.

Схема розповсюдження

Ця схема відповідає за управління контентом, що поширюється серед користувачів або на канали Telegram. Вона дозволяє адміністраторам надсилати текстові повідомлення, фото або відео з можливістю додавання інтерактивних кнопок для взаємодії з користувачем. Інформація про кількість надісланих повідомлень та невдалі спроби доставки допомагає відстежувати ефективність розповсюдження контенту. Це забезпечує високий рівень охоплення користувачів та допомагає зручніше керувати інформаційними кампаніями.

Ці схеми формують основу рівня бази даних системи. Їхня модульна структура дозволяє:

1. Легке додавання нових функцій (наприклад, тегування, категоризацію, рівні складності тестів)
2. Ефективне запитування статистики та показників панелі інструментів
3. Інтеграцію з API Telegram Bot для динамічної генерації контенту та циклів зворотного зв'язку

Ця структура відображає масштабовану та зрозумілу логіку серверної частини, яка підтримує навчання в режимі реального часу, управління користувачами та трансляцію контенту в екосистемі Telegram.

Висновки за розділом 2

У цьому розділі було надано вичерпний огляд дизайну системи, від початкового визначення проблеми до детальної технічної архітектури. Основним завданням було створити навчальний веб-застосунок, який є простим у

використанні, інтегрується безпосередньо з Telegram та містить адміністративні інструменти для моніторингу прогресу студентів.

Концептуальна модель зосереджувалася на модульних ролях (користувач, адміністратор), функціональних блоках (виконання тестів, зберігання результатів) та їх взаємодії через інфраструктуру Telegram, що дозволяло проводити як тестування в режимі реального часу, так і доставку освітнього контенту через спільний канал зв'язку.

Архітектура системи була структурована на п'яти рівнях — клієнт, бот, API, база даних та інтеграція Telegram — щоб забезпечити модульність, зручність обслуговування та зручність використання в реальному світі. Веб-застосунок Telegram та API ботів відіграють центральну роль як у взаємодії з фронтендом, так і в автоматизації бекенду.

Моделі даних були точно визначені за допомогою схем Mongoose, фіксуючи користувачів, метадані тестів, результати та події розповсюдження контенту. Така структура забезпечує легке маніпулювання даними, точне відстеження продуктивності та гнучку доставку контенту.

Поєднуючи React, Node.js, MongoDB та платформу Telegram, система відповідає цілям доступності, розширюваності та залучення користувачів. Вона підготовлена до подальшої масштабованості як в освітній сфері, так і в технічній інфраструктурі.

Цей етап проектування закладає основу для етапу впровадження, забезпечуючи плавний перехід від концепції до повнофункціонального застосунку.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ

3.1. Середовище та інструменти розробки

Впровадження освітньої платформи на базі Telegram спиралося на надійний та ретельно підібраний стек розробки, який забезпечував безперебійний користувацький досвід, повну інтеграцію з Telegram та адміністративні можливості. Вся система була розділена на три функціональні рівні: фронтенд (клієнтський інтерфейс), бекенд (API-сервер та база даних) та інфраструктура Telegram (Bot API та WebApp).

Клієнтський інтерфейс був повністю розроблений з використанням React, бібліотеки JavaScript, відомої своєю гнучкістю та продуктивністю. React дозволяв модульну компонентну розробку, що виявилося важливим для обробки кількох різних подань, таких як:

1. Сторінка навчання (перелік доступних тестів із зображеннями та кольорами)
2. Розділ Ігри (містить інтерактивні типи тестів, такі як загадки або зіставлення фраз)
3. Подання профілю (для відображення інформації про користувача та обміну посиланнями)
4. Екрани результатів (включаючи зворотний зв'язок у режимі реального часу щодо вибраних відповідей та підсумків)

Хуки React, такі як useState, useEffect та useMemo, широко використовувалися для керування логікою компонентів, побічними ефектами та оптимізацією продуктивності. Компоненти були розроблені для повторного використання та адаптивності в різних клієнтах Telegram. Стилізація здійснювалася за допомогою простого CSS та CSS-модулів, що дозволяло використовувати стилі з обмеженою областю видимості для кожного компонента

без використання зовнішніх фреймворків інтерфейсу користувача. Це дало розробникам повну свободу в розробці застосунку відповідно до естетики бренду та обмежень UX Telegram WebApp [10].

Фронтенд також реалізує Telegram WebApp SDK, який забезпечує безпечний доступ до даних користувачів, ініціалізує контекст бота, підтримує темизацію (наприклад, адаптацію темного/світлого режиму) та забезпечує єдиний вхід. Цей SDK був вирішальним для правильного зчитування initData, перевірки користувачів та керування взаємодією.

CORS був одним із найважливіших аспектів зв'язку клієнт-сервер. Оскільки застосунок обслуговується в WebView на основі iframe Telegram, він забезпечує дотримання суворих політик безпеки між джерелами. Бекенд був налаштований на додавання до білого списку певних джерел та включення облікових даних і заголовків, необхідних Telegram.

Бекенд був реалізований за допомогою Node.js та Express.js, що дозволило швидко створювати кінцеві точки RESTful API та ланцюжки проміжного програмного забезпечення. Обов'язки бекенду включають [11]:

1. CRUD-операції для тестів, питань та відповідей
2. Підключення користувачів до результатів тестування
3. Надання тестових даних фронтенду
4. Отримання та зберігання завершених результатів
5. Реагування на команди Telegram Bot через вебхуки

Використовувалося проміжне програмне забезпечення CORS з екосистеми Express з детальною конфігурацією для підтримки кількох доменів та забезпечення безпечних токенів доступу. Проміжне програмне забезпечення також використовувалося для перевірки запитів, форматування помилок, ведення журналу та обробки асинхронної логіки з синтаксисом async/await. Змінні середовища керувалися за допомогою dotenv, розділяючи токени, URI бази даних, прапорці розгортання та налаштування, специфічні для хоста.

Для постійного зберігання даних програма використовує MongoDB, розміщену в хмарі (MongoDB Atlas) [12]. Структура даних включає колекції для:

1. користувачів – ідентифікованих за ідентифікаторами користувачів Telegram
2. тестів – метаданих та посилань на документи з питаннями
3. тестів_пасінгів (проходжень тестів) – результатів, специфічних для користувача, та відстеження відповідей
4. розповсюджень – трансляцій повідомлень та кампаній до каналів Telegram

Схеми були визначені за допомогою Mongoose, з вкладеними піддокументами та зв'язками через посилання ObjectId. Індекси були додані для підтримки пошукових запитів за типом тесту, датою, ідентифікатором користувача та ключовим словом. MongoDB Compass використовувався як графічний інтерфейс для перевірки, налагодження та ручного редагування даних за потреби. Це було особливо корисно під час створення тестів та автоматизованого тестування ботів.

Основна розробка проводилася у WebStorm, IDE, яка забезпечувала глибоку підтримку React, JavaScript та Git. Такі функції, як живі шаблони, рефакторинг, структурні представлення та лінтування коду, підвищили продуктивність команди.

Цикл розробки включав:

1. Локальна розробка та налагодження за допомогою Vite та nodemon
2. Тестування API за допомогою скриптів Postman та curl
3. Безперервне розгортання з використанням комітів репозиторію GitHub
4. Ручне та користувацьке тестування в Telegram на iOS, Android та настільних комп'ютерах

Кінцева система була розгорнута на хмарних серверах з належними сертифікатами HTTPS, зворотними проксі-серверами та конфігураціями домену.

Підсумовуючи, поєднання SDK React, Express, MongoDB та Telegram, підкріплене інструментами структурованої розробки, дозволило нам створити сучасну, адаптивну та безпечну навчальну систему на базі Telegram з інтегрованими функціями управління та розповсюдження контенту.

3.2. Інтеграція з Telegram WebApp

Фундаментальною особливістю архітектури системи є її глибока та високоефективна інтеграція з платформою Telegram за допомогою використання Telegram Web Apps. Ця інтеграція не лише технічно надійна, але й орієнтована на користувача, що гарантує, що учні та адміністратори можуть взаємодіяти з системою безпосередньо в знайомому середовищі месенджера Telegram. Такий підхід усуває необхідність завантаження окремих програм, створення облікових записів або складних процедур адаптації, забезпечуючи безперебійний та зручний користувацький досвід на всіх основних платформах, включаючи iOS, Android, настільні комп'ютери та веб.

Інтеграція починається з офіційного SDK веб-додатку JavaScript від Telegram, який служить мостом між клієнтом Telegram та вбудованим веб-інтерфейсом. Цей SDK дозволяє застосунку:

1. Безпечно отримувати доступ до метаданих користувача Telegram, таких як ідентифікатор користувача, ім'я, прізвище, ім'я користувача та мова інтерфейсу
2. Отримувати безпечне корисне навантаження initData, підписане Telegram та використане для перевірки сеансу
3. Виявляти та реагувати на тему клієнта Telegram (світлу чи темну) для забезпечення нативного візуального досвіду
4. Використовувати примітиви інтерфейсу користувача Telegram, такі як головна кнопка, навігація назад та поведінка закриття, для забезпечення узгодженого відчуття

Після натискання кнопки «Відкрити застосунок» у чаті бота SDK ініціалізується, що дозволяє миттєво завантажувати контекст користувача. Це дозволяє негайно відображати персоналізований контент, не змушуючи користувача автентифікуватися через електронну пошту або сторонні сервіси. Якщо ідентифікатор Telegram недійсний або підроблений, система коректно обробляє помилку та запобігає несанкціонованому доступу.

Безпека є надзвичайно важливою в будь-якій програмі, яка має справу з даними користувачів. Автентифікація Telegram WebApp спирається на процес перевірки на стороні сервера, де отримані initData хешуються та порівнюються за допомогою рекомендованого алгоритму Telegram. Це захищає сеанс від підробки та гарантує, що фронтенд працює в перевіреному контексті.

Цей перевірений ідентифікатор Telegram використовується в усій системі як універсальний ідентифікатор для пошуку в базі даних, тестових завдань та керування доступом на основі ролей. Наприклад, якщо ідентифікатор Telegram позначено як адміністративний у базі даних, фронтенд автоматично вмикає додаткові подання, такі як панелі результатів користувача, редактори контенту або панелі розсилки повідомлень.

Усі HTTP-запити від фронтенду до бекенду містять ідентифікацію користувача, або через заголовки, або захищені параметри запиту. До них належать: ідентифікатор користувача для зв'язування тестових завдань, прапорці для визначення можливостей адміністратора, токени для обмеження доступу до захищених маршрутів.

Ця модель забезпечує баланс між зручністю використання та надійною перевіркою ідентичності, і все це без необхідності явного керування паролями або сеансами від користувача.

Однією з головних переваг побудови системи як Telegram WebApp є автоматична підтримка кількох типів пристроїв. Telegram гарантує, що WebApp завантажується однаково та передбачувано на:

1. Нативних додатках для Android та iOS
2. Telegram Desktop
3. Telegram Web (версія для браузера)

Інтерфейс React був стилізований та розроблений таким чином, щоб бути повністю адаптивним. На мобільних пристроях інтерфейс надає пріоритет великим кнопкам, безпечним для свайпів полям та вертикальному макетуванню. На робочому столі картки розгортаються горизонтально, а адміністратори отримують вигоду від покращеної видимості таблиць та панелей стану. Кожен екран у додатку був протестований та оптимізований для цієї кросплатформної узгодженості.

Telegram WebApp також забезпечує вбудовану підтримку жестів та взаємодій, таких як:

1. Проведення пальцем вниз для закриття додатка
2. Автоматичне повторне відкриття до останнього стану після повернення
3. Реагування на зміни внутрішньої колірної схеми Telegram на льоту
4. Це робить WebApp не просто ярликом до веб-сайту, а справжнім досвідом у додатку.
5. Взаємодія в реальному часі та обмін даними

Ключовою перевагою архітектури Telegram WebApp є її здатність функціонувати як односторінковий додаток (SPA). Це означає: що немає перезавантажень сторінок, усі переходи між екранами анімовані та миттєві, стан (наприклад, вибраний тест або сеанс користувача) зберігається в різних представленнях [13].

Використовуючи контекстний API React та локальний стан, додаток зберігає інформацію про сеанс користувача, поточний вибраний тест, поточні відповіді тощо. Зв'язок з серверною частиною відбувається через виклики fetch до кінцевих точок RESTful API, розміщених на сервері Node.js. Кожна відповідь

обробляється з зворотним зв'язком інтерфейсу користувача (наприклад, спінери завантаження, банери статусу).

Переваги використання Telegram WebApp:

Ця інтеграція дає численні технічні та бізнес-переваги:

1. Вбудоване керування ідентифікацією: Telegram обробляє автентифікацію
2. Узгоджене середовище: інтерфейс користувача залишається в рамках Telegram
3. Тісний зв'язок з доставкою контенту: публікації, кнопки та відповіді ботів відчуваються пов'язаними
4. Підвищена довіра та безпека: користувачі взаємодіють у відомому додатку, зменшуючи кількість відмов

Обійшовши потребу в магазинах додатків, команда розробників змогла швидше розгортати оновлення, повторювати відгуки без тривалих циклів перевірки та створювати освітній досвід, який здається нативним, але повністю веб-орієнтованим. Екосистема WebApp Telegram все ще зростає, і система розроблена, щоб розвиватися разом з нею. У майбутніх версіях додаток може реалізувати наприклад моделі навчання на основі послідовностей, пов'язані з активністю каналу

Рисунок 3.1 – Плановий вигляд того, як повинен виглядати застосунок.

На завершення, інтеграція Telegram WebApp дозволила системі досягти своїх цілей: миттєвої доступності, безпечної ідентифікації та кросплатформної сумісності. Після того, як ця інтеграція була надійно впроваджена, наступним логічним кроком було створення надійного клієнтського інтерфейсу для підтримки реальних навчальних випадків використання.

3.3. Реалізація клієнтського інтерфейсу

Клієнтський інтерфейс слугує основною точкою дотику для користувачів, які взаємодіють із системою всередині Telegram. Він розроблений як односторінковий додаток React (SPA) та включає кілька ключових модулів, призначених для учнів та адміністраторів. Ці модулі відповідають за навігацію тестами, їх виконання, зворотний зв'язок щодо результатів та адміністративний огляд.

Веб-додаток побудовано на основі простої та інтуїтивно зрозумілої навігації на основі вкладок, яка дозволяє користувачам перемикатися між такими режимами перегляду:

1. Головна / Каталог тестів – відображає доступні вікторини та ігри, згруповані за типом або предметом
2. Мої результати – відображає історію завершення тестів та бали
3. Профіль – відображає ідентифікацію користувача та містить інструменти спільного доступу
4. Адміністраторський розділ – реалізовано як окремий інтерфейс веб-застосунку та бота, доступний лише користувачам, чиї Telegram ID відповідають попередньо визначеному списку, що зберігається у файлі .env сервера. Цей адміністративний інтерфейс надає доступ до результатів тестування користувачів, інструментів публікації контенту та функцій керування тестами, незалежно від основного клієнтського веб-застосунку.

Навігація реалізована за допомогою внутрішнього стану React у поєднанні з умовним рендерингом, що дозволяє уникнути повного перезавантаження та покращує швидкість реагування.

Рисунок 3.2 – Реалізований вигляд головної сторінки з усіма тестами.

Тестові картки та категоризація:

1. Тести представлені у вигляді адаптивних карток, які містять:
2. Назву та короткий опис
3. Тема
4. Кольоровий значок, що вказує на категорію тесту (наприклад, вікторина, гра)
5. Додаткове зображення обкладинки для візуальної привабливості

Натискання на картку перенаправляє користувача на сторінку з детальною інформацією про тест, яка описує мету та дозволяє користувачеві розпочати.

Після того, як користувач розпочинає тест, система переходить у режим виконання тесту. Кожне питання відображається як окремий повноекранний блок із: текстом питання (і, за бажанням, зображенням), відповідями з кількома варіантами відповідей, кнопками, що можна натискати, для вибору відповіді, кнопкою «Далі» для продовження.

Вибрані відповіді зберігаються в локальному стані та надсилаються після завершення тесту. Інтерфейс обробляє перевірки (наприклад, вимогу вибору) та надає візуальний зворотний зв'язок.

Рисунок 3.3 – Демонстрація процесу проходження тесту з категорії “Діалог з фразами”, у якому необхідно вести розмову з різними персонажами використовуючи готові фрази.

Після завершення тестування користувачам відображається екран результатів, який включає:

1. Відсоток балів
2. Список правильних та неправильних відповідей

Цей екран закріплює знання та надає можливість для самоаналізу.

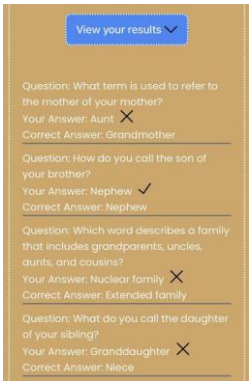

Рисунок 3.4 – Перегляд користувачем його відповідей після проходження тесту.

Розділ профілю містить інформацію про користувача Telegram (ім'я користувача) та посилання на додаток, яким можна поділитися. Це сприяє поширенню вірусу та взаємодії через мережу Telegram.

Рисунок 3.5 – Сторінка профілю користувача.

Для користувачів, чиї Telegram ID відповідають списку, визначеному у файлі конфігурації .env сервера, стає доступним окремий бот. Адміністративний статус не зберігається в базі даних, а визначається динамічно шляхом порівняння Telegram ID поточного користувача з цим попередньо визначеним списком. Це призводить до панелі адміністратора, яка включає:

1. Таблицю нещодавніх завершених тестів з фільтрами
2. Кнопки для перегляду детальних відповідей
3. Доступ для створення та керування вмістом тесту

Усі елементи відображаються за допомогою компонентів React та умовно стилізуються відповідно до ролей користувачів.

Рисунок 3.6 – Адміністративна сторінка з усіма користувачами.

Ця реалізація використовує сильні сторони React в управлінні станом, повторному використанні компонентів, умовному відображенні та динамічній маршрутизації. Використання модульних компонентів дозволило ізолювати логіку для відображення тестів, керування вводом користувача та представлення результатів без дублювання коду. Кожне представлення в додатку створено з урахуванням чіткості та зручності обслуговування, що полегшує його розширення в майбутніх версіях.

Інтерактивні функції, такі як навігація питаннями, фільтрація тестів та відображення результатів, отримують користь від синхронізації даних у реальному часі з API серверної частини. Елементи інтерфейсу користувача, такі як спінери завантаження, спливаючі вікна підтвердження та динамічні стани кнопок, додають рівні зворотного зв'язку та чіткості до взаємодії з користувачем. Адміністратори отримують додаткові компоненти інтерфейсу користувача, такі як модальні вікна для редагування контенту, таблиці даних з пагінацією та індикатори перевірки в реальному часі.

Підсумовуючи, інтерфейс користувача — це сучасний, адаптивний застосунок на базі React, покращений завдяки безшовній інтеграції в екосистему Telegram. Він утворює ядро системи, що дозволяє користувачам навчатися, взаємодіяти та відстежувати прогрес у візуально узгодженому та адаптивному середовищі.

3.4. Інтерфейс адміністрування

Адміністративний інтерфейс відіграє центральну роль в управлінні та нагляді за освітньою системою. Це життєво важливий інструмент для адміністраторів, який надає їм функціональність для керування контентом, моніторингу продуктивності користувачів, надсилання сповіщень та налаштування системних параметрів за потреби. Інтерфейс реалізовано за допомогою двох окремих компонентів: веб-додатку адміністратора та бота адміністратора, кожен з яких розроблений для задоволення різних випадків

використання, зберігаючи при цьому безперешкодну інтеграцію в екосистему Telegram. Відокремлення адміністративних функцій від системи, орієнтованої на користувача, гарантує, що адміністратори можуть зосередитися на завданнях управління, не втручаючись у процес навчання користувачів.

Веб-додаток адміністратора створено за допомогою React. Як односторінковий додаток (SPA), веб-додаток дозволяє адміністраторам переміщатися між різними розділами системи без оновлення сторінки, що робить користувацький досвід швидшим та більш плавним. Веб-додаток повністю відокремлений від користувацького додатка, що гарантує, що адміністративні функції не заважають освітньому досвіду звичайних користувачів.

Розробка веб-додатку адміністратора була зосереджена на створенні інтуїтивно зрозумілого, ефективного та простого у використанні інтерфейсу для адміністраторів. Додаток був розроблений для підтримки широкого спектру адміністративних завдань, гарантуючи, що інтерфейс користувача залишається чистим та організованим. Він був розроблений на основі основного веб-додатку (користувацького), тому технологічні рішення для його розробки повторюють “батьківські”, таким чином основні інструменти наступні

1. React
2. React Router
3. Стилізовані компоненти

Веб-додаток адміністратора надає адміністраторам низку функцій, зокрема:

Керування тестами – система дозволяє адміністраторам створювати нові тести, додаючи запитання, визначаючи відповіді та класифікуючи тести за темою, рівнем складності та типом (вікторина, гра тощо).

Рисунок 3.7 – Адміністративна сторінка для створення нового тесту.

Результати користувачів – цей розділ містить детальний аналіз результатів тестів користувачів. Адміністратори можуть переглядати окремі результати тестів, відстежувати прогрес користувачів з часом та аналізувати, на які запитання найчастіше дають неправильні відповіді.

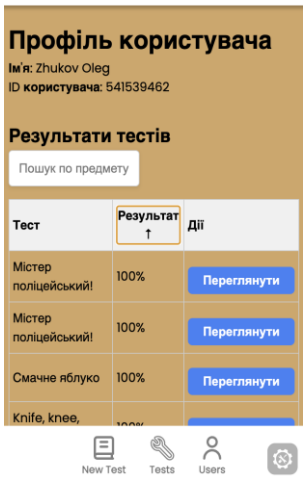

Рисунок 3.8 – Адміністративна сторінка з результатами тестування конкретного користувача.

Веб-додаток адміністратора має просту систему навігації на основі вкладок, що забезпечує легкий доступ до наступних розділів. Основні розділи:

Керування тестами – розділ, де можна створювати, оновлювати або видаляти тести. Адміністратори можуть додавати нові запитання, вибирати правильні відповіді та класифікувати тести за темою. Після того, як тести готові, їх можна опублікувати для проходження користувачами.

Результати користувачів – у цьому розділі відображаються детальні результати для окремих користувачів, включаючи результати тестів та дані про завершення.

Окрім веб-додатку адміністратора, бот адміністратора відіграє ключову роль в управлінні та взаємодії з системою в режимі реального часу. Бот адміністратора створено за допомогою фреймворку Telegraf.js, який дозволяє адміністраторам виконувати різні завдання безпосередньо в інтерфейсі Telegram, що дозволяє швидко та легко взаємодіяти з користувачами, надсилати повідомлення та керувати контентом. Деякі ключові функції бота адміністратора включають:

1. Надсилання оголошень – адміністратори можуть використовувати бота для надсилання оголошень або сповіщень певним користувачам або групам. Це може включати нові тести, важливі оновлення або загальні системні повідомлення.
2. Розподіл тестів – адміністратори можуть надсилати автоматичні або ручні запрошення на тестування користувачам через бота. Бот може надсилати нагадування користувачам, які ще не завершили свої тести.
3. Керування користувачами – бот дозволяє адміністраторам відстежувати поведінку користувачів, переглядати результати тестів та блокувати або розблоковувати користувачів за потреби.
4. Доступ до даних у режимі реального часу – бот надає адміністраторам доступ до важливих даних у режимі реального часу, таких як продуктивність користувачів, рівень завершення тестів та відгуки учнів.

3.5. Можливості бота Telegram

Бот Telegram відіграє центральну роль у функціональності системи, дозволяючи користувачам взаємодіяти з програмою, а адміністраторам – відстежувати прогрес, надсилати сповіщення та розповсюджувати контент. Він служить інтерфейсом для користувачів, адміністраторів та системних компонентів, оптимізуючи комунікацію та автоматизацію.

Рисунок 3.9 – Демонстрація початку роботи з нашим ботом.

Основна роль бота Telegram полягає в тому, щоб надати користувачам можливість взаємодіяти з навчальним контентом та проходити тести, не виходячи з програми Telegram. Бот виконує кілька ключових функцій, орієнтованих на користувача. Наприклад “запрошення на тестування” – бот надсилає користувачам запрошення розпочати тестування безпосередньо або через посилання, якими діляються в групових чатах чи каналах.

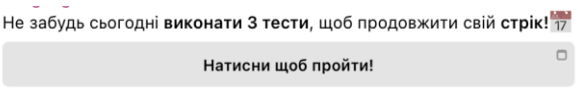

Рисунок 3.10 – Результат який надійшов у канал.

Бот Telegram також надає адміністраторам різноманітні потужні інструменти для керування системою. Ці інструменти включають:

1. Надсилання широкомовних повідомлень – адміністратори можуть використовувати бота для надсилання оголошень або сповіщень користувачам. Вони можуть включати нові тести, важливі оновлення або загальні системні повідомлення.
2. Надсилання нагадувань про тестування – бот дозволяє адміністраторам планувати та надсилати нагадування користувачам про завершення тестів, що очікують на виконання.

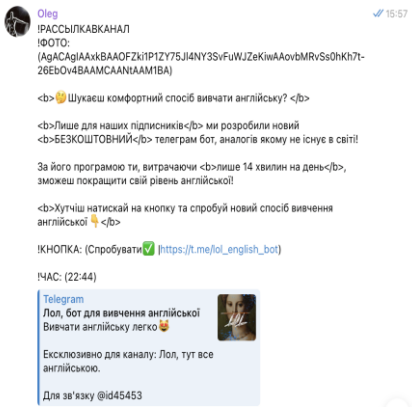

Рисунок 3.11 – Приклад надсилання тексту в бота адміністратором для публікації в каналі.

Рисунок 3.12 – Результат який надійшов у канал.

Бот працює паралельно з веб-додатком адміністратора, забезпечуючи адміністраторам додатковий досвід керування системою безпосередньо в Telegram.

Бот реалізовано за допомогою Telegraf.js, популярного фреймворку для створення ботів Telegram на Node.js. Бот взаємодіє з API серверної частини для обробки запитів, таких як надсилання повідомлень, збір тестових даних та отримання результатів користувачів. Деякі з основних компонентів бота включають:

1. Обробники повідомлень – функції, які прослуховують певні дані користувача та реагують відповідно, наприклад, надсилання тесту або надання зворотного зв'язку.
2. Керування станом – бот відстежує прогрес користувача в режимі реального часу, забезпечуючи належне керування кожним тестовим сеансом та точне зберігання відповідей.
3. Інтеграція API – бот взаємодіє з API серверної частини для отримання та надсилання даних, забезпечуючи синхронізацію фронтенду та серверної частини.

Бот також підтримує вбудовані кнопки та клавіатури для покращення взаємодії з користувачем, дозволяючи користувачам легко орієнтуватися в тестах та відповідати на запити. Серед переваг використання бота Telegram можна визначити наступні:

Миттєва доступність – бот дозволяє користувачам запускати тести, отримувати результати та взаємодіяти з навчальним контентом безпосередньо з Telegram, без необхідності відкривати окрему програму.

Зв'язок у режимі реального часу – бот забезпечує миттєве спілкування між користувачами та адміністраторами, що дозволяє отримувати негайний зворотний зв'язок та оновлення в режимі реального часу.

Залучення користувачів – використовуючи систему сповіщень Telegram, бот допомагає підтримувати зв'язок користувачів за допомогою нагадувань, звітів про хід роботи та сповіщень про новий контент.

Підсумовуючи, Telegram Bot є важливою частиною системи, що дозволяє користувачам безперешкодно взаємодіяти з освітнім контентом у Telegram. Він надає широкий спектр функцій як для користувачів, так і для адміністраторів, забезпечуючи ефективність, адаптивність та залученість системи. У наступному розділі ми розглянемо інтеграцію з серверною частиною, зосередившись на API, управлінні базами даних та потоці даних.

3.6. Публікація контенту через канал

Одним із важливих аспектів системи є можливість транслювати освітній контент, оголошення та запрошення на тестування користувачам через канали Telegram. Ця функція дозволяє адміністраторам взаємодіяти з користувачами у великих масштабах, гарантуючи, що вони отримуватимуть своєчасні оновлення та нагадування про нові тести чи навчальні матеріали.

Система дозволяє адміністраторам надсилати користувачам різні типи контенту через канали Telegram, зокрема:

1. Текстові повідомлення – адміністратори можуть надсилати звичайні текстові повідомлення, щоб інформувати користувачів про нові тести, оновлення системи або загальні освітні поради.
2. Зображення – адміністратори можуть надсилати зображення, що супроводжують освітній контент, такі як діаграми, схеми або фотографії, що стосуються певного тесту чи теми.
3. Відео – адміністратори можуть ділитися навчальними відео або рекламними матеріалами, щоб допомогти ефективніше залучати користувачів.

4. Вбудовані кнопки – бот може містити кнопки, які дозволяють користувачам виконувати дії безпосередньо в Telegram, такі як початок тесту, відвідування URL-адреси або навіть надання відгуків.

Трансляція контенту через канал Telegram є важливою функцією системи, оскільки вона гарантує, що користувачі завжди поінформовані та можуть діяти на основі наданої інформації. Адміністративний інтерфейс дозволяє адміністраторам планувати та керувати трансляціями, відстежувати надіслані повідомлення та обробляти помилки доставки. Це спрощує планування кампаній та оновлення користувачів без необхідності ручного втручання щоразу. Система розповсюдження є гнучкою, що дозволяє адміністраторам надсилати контент певним групам користувачів на основі їхніх інтересів або участі в певних тестах. Наприклад, система може надсилати цільові сповіщення користувачам, які не пройшли тест, або тим, хто набрав менше певного порогу.

Канали Telegram використовуються не лише для трансляції контенту, але й для створення більш інтерактивного навчального середовища. Використовуючи API ботів Telegram, система дозволяє адміністраторам автоматично надсилати заплановані повідомлення, взаємодіяти з користувачами в каналі та навіть запускати контент на основі дій користувачів (наприклад, проходження тесту).

Поєднання бота Telegram та каналів забезпечує потужний спосіб взаємодії з користувачами, забезпечуючи ефективну та інтерактивну доставку навчальних матеріалів. Адміністратори можуть використовувати канали для створення відчуття спільноти та підтримки мотивації користувачів за допомогою регулярних оновлень контенту.

У майбутніх версіях системи можуть бути представлені такі функції, як персоналізований розподіл контенту на основі результатів тестування користувачів або елементи гейміфікації для подальшого залучення користувачів. Це дозволить забезпечити більш динамічний потік контенту, де користувачі отримуватимуть контент, адаптований до їхніх потреб та прогресу в навчанні.

Використання каналів Telegram та Bot API забезпечує безперебійний досвід, де адміністратори та користувачі можуть легко взаємодіяти в єдиному середовищі. Можливість ефективно та результативно поширювати контент сприяє більш захопливому та інтерактивному навчанню.

Рисунок 3.13 – Приклад однієї з можливих публікацій контенту у каналі.

3.7. Логіка адміністративної системи

Адміністративна функціональність системи побудована як модульне, багатоінтерфейсне середовище, що складається з двох повністю незалежних, але синхронізованих компонентів: Admin WebApp та Admin Telegram Bot. Ці компоненти працюють поверх спільної бекенд-інфраструктури та використовують загальну базу даних MongoDB. Така архітектура забезпечує безперебійний потік даних, послідовну бізнес-логіку та гнучкість в адміністративних робочих процесах.

Admin WebApp розроблено як односторінковий додаток (SPA) на основі React, який пропонує повний інтерфейс для управління вмістом системи та відстеження її використання. Він доступний виключно для користувачів, чиї Telegram ID включені до файлу конфігурації безпечного середовища (.env). Доступ перевіряється через контекст Telegram WebApp, що усуває потребу в традиційних системах входу.

WebApp пропонує широкий спектр адміністративних можливостей, включаючи:

1. Створення нових тестів з користувацькими метаданими, такими як заголовок, тема та опис.
2. Додавання, редагування та видалення питань; зв'язування їх з одним або кількома тестами.
3. Налаштування візуального оформлення кожного тесту (кольорові теги, банери зображень).
4. Доступ до детальних звітів для кожного користувача, включаючи окремі відповіді, відсоткові бали та позначки часу.

Усі дії фронтенду виконуються через RESTful HTTP-запити до бекенд-сервісу, який розроблено в Node.js з використанням фреймворку Express.js. Бекенд обробляє автентифікацію через Telegram JWT, перевіряє корисні навантаження, взаємодіє з колекціями MongoDB (тести, питання, користувачі, test_passing) та повертає структуровані JSON-відповіді [14].

Рисунок 3.14 – Загальний зв’язок усіх контролерів та моделей бази даних, пов’язаних з адміністративними функціями між собою.

Інтерфейс є модульним, з такими компонентами, як TestEditor, QuestionForm та ResultsTable, реалізованими з використанням сучасних шаблонів React, включаючи перехоплювачі та контекст для управління станом. Такий дизайн забезпечує зручність обслуговування та швидку розробку нових функцій.

Щоб доповнити можливості Admin WebApp, було реалізовано спеціальний Admin Telegram Bot за допомогою Telegraf.js. Цей бот дозволяє адміністраторам швидко та неформально взаємодіяти з системою, що особливо корисно в ситуаціях, коли мобільні пристрої в першу чергу використовуються, або коли комп'ютер недоступний.

Адміністраторський бот включає такі функції:

1. Надсилання масових або цільових сповіщень (наприклад, тестові запуски, нагадування, оновлення).
2. Проведення кампаній розсилки з текстовими, графічними або відеоповідомленнями.
3. Автоматичний збір та відображення кількості успішних доставок та помилок.

Бот налаштовано на перевірку особи адміністратора на основі порівняння Telegram ID та працює через HTTPS через той самий сервер, що й WebApp. Усі повідомлення, надіслані ботом, реєструються, а результати оновлюються для забезпечення відстеження та можливості аудиту.

Систему було розроблено для забезпечення максимальної зв'язності та мінімальної надмірності між двома адміністративними інтерфейсами. Веб-додаток адміністратора та бот адміністратора працюють на основі спільної логіки та спільних даних, що забезпечує послідовний та синхронізований досвід незалежно від точки доступу.

Типовий робочий процес адміністратора може включати таку послідовність:

1. Адміністратор створює новий тест у веб-додатку адміністратора. Метадані тесту та посилання на запитання зберігаються в колекціях тестів та запитань.
2. Адміністратор перемикається на бота адміністратора та надсилає повідомлення, що містить повідомлення про новий тест. Бот генерує кнопки клавіатури та надсилає повідомлення користувачам.
3. Користувачі проходять тест у веб-додатку Telegram. Їхні результати (відповіді, бал, позначки часу) зберігаються в колекції test_passing.
4. Потім адміністратор може переглядати сукупні дані у веб-додатку.

У більш складних сценаріях адміністратори можуть одночасно запускати кампанії з обмеженим часом, переглядати активність користувачів у режимі реального часу або модерувати відповіді, використовуючи обидва інтерфейси одночасно. Ця архітектура з двома інструментами підтримує гнучке, масштабоване та послідовне адміністрування. Він забезпечує адміністраторам як потужність, так і портативність, не жертвуючи цілісністю чи синхронізацією системи. Оскільки обидва інструменти покладаються на одні й ті самі кінцеві точки API, додавання функцій в одному місці одразу приносить користь іншому. Архітектура також підтримує майбутні вдосконалення, такі як:

1. Інтеграція з хмарними панелями інструментів (наприклад, Grafana) для глибшої аналітики
2. Контроль доступу на основі ролей для кількох рівнів адміністратора
3. Інструменти модерації або підтримки в додатку (наприклад, позначення спроб користувачів)

Загалом, ця адміністративна система забезпечує баланс між точністю та гнучкістю, забезпечуючи як структуроване управління контентом, так і адаптивну комунікацію, що критично важливо для управління масштабними освітніми середовищами в Telegram.

Для кращої ілюстрації структури системи та потоку даних слід включити такі діаграми:

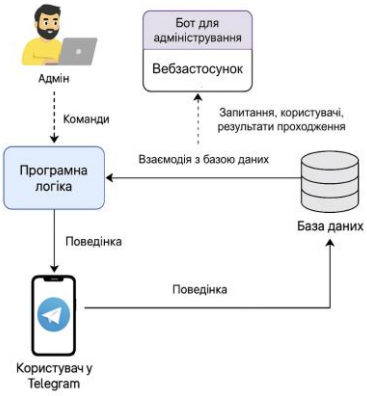

Рисунок 3.15 – Умовна загальна структура проєкту.

Ця діаграма показує, як взаємодіють Telegram Bot, Telegram WebApp (клієнт), Admin WebApp, Backend API (Node.js + Express) та MongoDB. Вона включає потоки запитів/відповідей, автентифікацію через контекст Telegram та спільне використання колекцій (користувачі, тести, test_passing, розподіл).

Рисунок 3.16 – Демонстрація алгоритму створення тесту адміністратором.

Ця діаграма окреслює типовий сценарій:

1. Адміністратор відкривши адміністративний WebApp через відповідного бота заповнює форму створення тесту.

2. Дані про тест відправляються на сервер де проходять перевірку.
3. Тест зберігається у БД, повертаючи на клієнт відповідне повідомлення про успіх, або помилку.

Рисунок 3.17 – Демонстрація алгоритму створення адміністратором розповсюдження в канал, або бота.

Ця діаграма зосереджена на тому, як повідомлення, зображення або відеоконтент розповсюджуються за допомогою схеми розповсюдження. Вона включає взаємодію між ботом, адміністративною панеллю та користувачами, а також показує.

3.8. Тестування, налагодження та відгуки користувачів

Тестування та налагодження є критично важливими кроками у забезпеченні якості, стабільності та надійності програми. У цьому розділі описано стратегії тестування, що використовуються під час розробки, інструменти налагодження, що використовуються, та те, як відгуки користувачів враховуються в ітеративному процесі розробки.

Було проведено різні типи тестування, щоб переконатися, що система працює належним чином та забезпечує безперебійний користувацький досвід. Стратегії тестування включали:

Модульне тестування – окремі компоненти та функції були протестовані, щоб переконатися, що кожна частина системи поводиться належним чином.

Наприклад, компоненти React були протестовані, щоб перевірити, чи правильно вони відображаються на основі введених користувачем даних або даних серверної частини [15].

Інтеграційне тестування – тести були проведені, щоб перевірити, наскільки добре різні частини системи працюють разом. Це включало тестування взаємодії між фронтендом та серверною частиною, перевірку того, чи дані були правильно отримані з бази даних та відображені користувачам.

Комплексне (E2E) тестування – було протестовано повні робочі процеси, щоб переконатися, що користувач може безперешкодно виконувати завдання, такі як запуск тесту, надсилання відповідей та отримання відгуків. Для автоматизації цих тестів використовувалися такі інструменти, як Cypress або Puppeteer, що імітували дії користувачів у Telegram WebApp [16].

Ручне тестування – окрім автоматизованого тестування, проводилося ручне тестування, щоб забезпечити функціональність системи та виявити потенційні проблеми, які могли пропустити автоматизовані тести. Це включало дослідницьке тестування, під час якого відбувалась взаємодія з системою неструктурованим чином, щоб виявити будь-які приховані помилки.

Налагодження є важливою частиною процесу розробки. Для швидкого виявлення та виправлення проблем використовувалися такі інструменти та методи:

Chrome DevTools – цей інструмент відіграв важливу роль у налагодженні проблем як фронтенду, так і серверної частини, дозволяючи розробникам перевіряти мережевий трафік, помилки JavaScript та стилі елементів. Консоль у Chrome DevTools широко використовувалася для реєстрації та відстеження проблем під час виконання [17].

Проміжне програмне забезпечення для ведення журналу – на сервер Express було додано спеціальне проміжне програмне забезпечення для ведення журналу, щоб

відстежувати вхідні запити, реєструвати помилки та контролювати продуктивність API.

Налагодження Telegraf.js – бібліотека Telegraf.js, що використовується для бота Telegram, також надавала інструменти налагодження для реєстрації взаємодії ботів та відповідей API, що було корисним для виявлення проблем у режимі реального часу.

Окрім технічного тестування, відгуки користувачів відіграли важливу роль у формуванні розробки системи. Систему спочатку було випущено невеликій групі бета-тестерів, яких попросили надати відгуки щодо функціональності, зручності використання та будь-яких виявлених помилок. На основі отриманих відгуків було внесено покращення в таких областях:

Покращення зручності використання: відгуки тестувальників виділили області, де можна покращити інтерфейс користувача, такі як організація тестових карток та зрозумілість інструкцій. Ці зміни були скориговані, щоб зробити систему більш інтуїтивно зрозумілою та простішою в навігації.

Виправлення помилок: під час тестування було виявлено кілька помилок, включаючи незначні збої інтерфейсу користувача та проблеми з серверною частиною, пов'язані з отриманням даних. Їх було оперативно виправлено та повторно протестовано для забезпечення належної функціональності.

Процеси тестування, налагодження та зворотного зв'язку забезпечили стабільність, зручність використання та повну функціональність системи. Завдяки ретельному тестуванню та постійному вдосконаленню на основі відгуків користувачів, система була оптимізована як для продуктивності, так і для зручності використання. Завдяки цим процесам система добре підготовлена до розгортання, а майбутні оновлення продовжуватимуть розширювати її можливості на основі постійних потреб користувачів та відгуків.

Висновки за розділом 3

У цьому розділі ми розглянули повний процес впровадження освітньої системи, від інструментів розробки та впровадження на стороні клієнта до логіки виконання тестів та адміністративного інтерфейсу.

Ми дослідили, як Telegram WebApp був бездоганно інтегрований у платформу Telegram, що дозволяє користувачам отримувати доступ до освітнього контенту безпосередньо в інтерфейсі Telegram. Основні функції системи, такі як створення тестів, взаємодія з користувачем та зворотний зв'язок у режимі реального часу, були розроблені для забезпечення захопливого та інтерактивного навчального процесу. Admin WebApp та Admin Bot надають адміністраторам гнучкі можливості управління в режимі реального часу для розповсюдження контенту, аналізу результатів користувачів та моніторингу системи, все в рамках екосистеми Telegram.

Ретельне тестування, налагодження та процеси зворотного зв'язку з користувачами гарантували, що система залишається стабільною, масштабованою та адаптивною, задовольняючи як потреби користувачів, так і операційну ефективність.

Завдяки цим функціям система не лише пропонує зручну для користувача та масштабовану платформу для вивчення мов, але й надає адміністраторам потужні інструменти для керування контентом, моніторингу прогресу та спілкування з користувачами.

ВИСНОВКИ

Цей кваліфікаційний проект зробив особливий акцент на розробці та впровадженні надійної, гнучкої та масштабованої адміністративної підсистеми в рамках ширшої освітньої платформи, інтегрованої з Telegram. Основною метою було забезпечити повний контроль над управлінням контентом, моніторингом тестування та взаємодією з користувачами з адміністративної точки зору, все в розподіленому середовищі реального часу.

Була розроблена адміністративна структура через два основні інтерфейси:

1. Admin WebApp — повнофункціональний односторінковий додаток React, розроблений для поглибленого управління контентом та результатами.
2. Admin Telegram Bot — легкий, але потужний командний інтерфейс для взаємодії та комунікації в реальному часі, побудований за допомогою Telegraf.js.

Обидва інструменти взаємодіють через спільний бекенд, побудований на Node.js (Express), та працюють з єдиною моделлю даних MongoDB. Це забезпечує узгодженість, синхронізацію та єдину бізнес-логіку в обох точках входу.

Реалізовано автентифікацію адміністратора безпечним методом, яка керується шляхом порівняння ідентифікатора користувача Telegram із попередньо визначеними значеннями в конфігурації .env, що спрощує контроль доступу без необхідності використання традиційних систем входу.

Впроваджено набір інструментів розповсюдження навчального контенту, сповіщень, включаючи можливість надсилання широкомовних повідомлень через бота адміністратора з інтерактивними кнопками, вбудованими медіа та запланованим часом. Автоматизоване відстеження доставки з реєстрацією в режимі реального часу sentCount, errorsCount та зворотного зв'язку щодо повідомлень безпосередньо в базу даних для адміністративного перегляду.

Двоінтерфейсний доступ до зведеної інформації, де швидкі огляди та оновлення статусу можна отримувати в Telegram, тоді як глибший аналіз залишається доступним у веб-додатку.

Розроблена адміністративна підсистема, в рамках цього проєкту, не є додатковою функцією — це є центральним операційним ядром усієї платформи. Вона займається створенням, підтримкою та стратегічним розповсюдженням освітнього контенту. Система керує залученням користувачів та надає аналітичні дані для прийняття рішень. Також немало важливим є підтримка як проактивних, так і реактивних робочих процесів, дозволяючи адміністраторам керувати кампаніями, модерувати дані та оптимізувати контент.

Завдяки модульній архітектурі, що орієнтована на API, та сильній інтеграції між веб-додатком та Telegram Bot, ця підсистема гарантує, що освітній процес не лише керується, але й стратегічно підтримується потужними та добре структурованими інструментами адміністрування. Система є перспективною та готова до розширення, покращення аналітики та постійного розвитку, що відповідає освітнім цілям та зростанню кількості користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. M. Weller. Twenty Years of Edtech., 2018 [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://oro.open.ac.uk/55708/ (дата звернення: 27.04.2025).
2. T. Laiby, B. Subramanya. A Comprehensive Overview of Telegram Services - A Case Study, 2022 [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://d1wqtxts1xzle7.cloudfront.net/115309128/1847-libre.pdf?1716700831=&response-content-disposition=inline%3B+filename%3DA_Comprehensive_Overview_of_Telegram_Ser.pdf&Expires=1746800399&Signature=epHMBD72TQav87fnW3GgWUmlSpJ3amNPIIjZ3DXIKUyfiEssGSfol0M5Yt-VIUOoCAReYg50MAAYCO-p0f9WTMUSj86FpQb1-gBjfUwfNt2mMOjU-Wybr44WoIm4H6GaNEIe6qS46n-DvusHSDgI~zDIhk6JHQp4ckUMqT3bQ~~dH-1YjFxSxDTNvVhX4JTT9LZdm1Bi70R3OqBvOdpcYnohFl52oVWUGUbyDPZ55EXElzcPYPB5tr2kxkAz~EKOhfqNWjdJZE6l5IAuD~HDj1wZNbXOpkcZr9Ppu0Zdi-UTbMenFaRp2PGAqMeJldcezIv6xY4Q0rpx7VWC7hdSNQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA (дата звернення: 27.04.2025).
3. M. Shortta , S. Tilakb , I. Kuznetcovaa , B. Martensc , A. Babatunde, Gamification in mobile-assisted language learning: a systematic review of Duolingo literature from public release of 2012 to early 2020, 2021 [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://www.tandfonline.com/doi/full/10.1080/09588221.2021.1933540#abstract (дата звернення: 29.04.2025).

4. A. Banks, E. Porcello. Learning React: Modern Patterns for Developing React Apps, 2020 [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://books.google.com.ua/books?hl=ru&lr=&id=tDjrDwAAQBAJ&oi=fnd&pg=PR2&dq=react&ots=DDJ3hmkK_0&sig=iRGCSyPVbgzIXIO3G-zdgUNoAj8&redir_esc=y#v=onepage&q=react&f=false (дата звернення: 30.04.2025).
5. C. Peters. Building Rich Internet Applications with Node.js and Express.js, 2017 [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://uol.de/f/2/dept/informatik/ag/svs/download/reader/reader-seminar-ws2016.pdf#page=18 (дата звернення: 02.05.2025).
6. A. Chauhan, A Review on Various Aspects of MongoDb Databases, 2019 [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://d1wqtxts1xzle7.cloudfront.net/60661268/a-review-on-various-aspects-of-mongodb-databases-IJERTV8IS050031201909 21-10051-a8lu8n-libre.pdf?1569063684=&response-content-disposition=inline%3B+filename%3DIJERT_A_Review_on_Various_Aspects_of_Mon.pdf&Expires=1746798237&Signature=QP1opYlar2fputDGFNgjRtogsp-Sn7dIYC0GEdJdCbLZ8SSo5pSzt4WzRwJBRGMbOYysrII3VFcNKXBvwYMJpfa~BoPFpqxcqO~cYIrTyOBPA1OTiXsKjh4SW8Bm1OvC09rj5b0NifZyA2DifF5jFOTZ6iw7WM0bzk83kxP1VobMu1LgMkyU8qbHmirQtr2HHYCI7iN2h3Y8~dYtabTJrx-IUvAk5E0AsUR1C2qgxVIop~6mMZGNg0YX~Cht0bRC-FmhbkhAAb1NJ3OVZDK~c1qUPCv4gZJjKieqp2mcanwX1eq7UcKOvL3gQa0MX0jq1GoLMJQBAqCTZzSAPcX8nA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA (дата звернення: 06.05.2025).
7. Y. Perez-Riverol ,L. Gatto,R. Wang,T. Sachsenberg,J. Uszkoreit,F. da Veiga Leprevost,C. Fufezan,T. Ternent, S. J. Eglen,D. S. Katz,T. J. Pollard,A.

Konovalov,R. M. Flight,K. Blin,J. A. Vizcaíno , Ten Simple Rules for Taking Advantage of Git and GitHub, 2016 [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004947 (дата звернення: 03.05.2025).

8. Telegram APIs (офіційна документація) [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://core.telegram.org/ (дата звернення: 04.05.2025).
9. Telegram Mini Apps (офіційна документація) [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://core.telegram.org/bots/webapps (дата звернення: 05.05.2025).
10. Hamidli, Nasrullah. "Introduction to UI/UX design: key concepts and principles." Preuzeto 28 (2023): 2024.[Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://d1wqtxts1xzle7.cloudfront.net/99497290/nnesirr-libre.pdf?1678118545=&response-content-disposition=inline%3B+filename%3DIntroduction_to_UI_UX_Design_Key_Concept.pdf&Expires=1748277276&Signature=F5YuozETLRtpVlZVT0PaEGdmxz7zB8aY-99Z7yZNGG~1cTdbhwo6XG94yjvXzrJeYJ~DgmIKc8GrhVo0oqjWZg~cR1tMCkeS6YKo2icDYSzeE9LHRPYlN-NCXZStC0CUl~NKd0XeWg0tNdExarqfgi0tLyxKy5NUGGexmQ5F5UD9832gVJGCxZKuOh3WUWefrShGzwnuDORutdDcCs6czscqu-YGU8omvO~MpxmOahmPzrG7ky7NDmdNZ955awFCaSbsJms8XTl4FgjgJyfp860992ncDgoB4cB60-coF0n00Cu2OdTaiE6KazPrEsLBuxQDmJiFjkEMec~iyQvhcg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA (дата звернення: 05.05.2025)

11. EHSAN, Adeel, et al. RESTful API testing methodologies: Rationale, challenges, and solution directions. Applied Sciences, 2022.[Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://www.mdpi.com/2076-3417/12/9/4369 (дата звернення: 06.05.2025)
12. Phaltankar, Amit, et al. MongoDB Fundamentals: A hands-on guide to using MongoDB and Atlas in the real world. Packt Publishing Ltd, 2020. [Електронна книга]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://books.google.com.ua/books?hl=uk&lr=&id=uaEQEAAAQBAJ&oi=fnd&pg=PP1&dq=MongoDB+Atlas&ots=NAlMdL1n_L&sig=JjPeiw5-45WsGSuvsKZj9QBzgFs&redir_esc=y#v=onepage&q=MongoDB%20Atlas&f=false (дата звернення: 09.05.2025)
13. Jonathan, Ricky. "Development of Front-End Web Applications Utilizing Single Page Application Framework and React. js Library." International Journal Software Engineering and Computer Science (IJSECS) 3.3 (2023). [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://www.journal.lembagakita.org/ijsecs/article/view/1943 (дата звернення: 10.05.2025)
14. Adam, Stenly Ibrahim, Jimmy H. Moedjahedy, and Jeremiah Maramis. "RESTful web service implementation on Unklab information system using JSON web token (JWT)." 2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS). IEEE, 2020.[Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://ieeexplore.ieee.org/abstract/document/9320801 (дата звернення: 13.05.2025)
15. Da Costa, Lucas Fernandes. Testing JavaScript Applications. Simon and Schuster, 2021.[Електронна книга]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://books.google.com.ua/books?hl=uk&lr=&id=JTgzEAAAQBAJ&oi=fnd

&pg=PR13&dq=module+testing+javascript&ots=IEEfW9GrJt&sig=PlrKftJCLUcmHLHhibulT7OvMQQ&redir_esc=y#v=onepage&q=module%20testing%20javascript&f=false (дата звернення: 13.05.2025)

16. Di Meglio, Sergio. "End-to-End Testing in Web Environments: Addressing Practical Challenges." 2025 IEEE Conference on Software Testing, Verification and Validation (ICST). IEEE, 2025.[Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://ieeexplore.ieee.org/abstract/document/10988957 (дата звернення: 14.05.2025)
17. Moreno, José Miguel, Narseo Vallina-Rodriguez, and Juan Tapiador. "Chrowned by an extension: abusing the Chrome DevTools protocol through the debugger API." 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P). IEEE, 2023. [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://ieeexplore.ieee.org/abstract/document/10190532 (дата звернення: 14.05.2025)

ДОДАТКИ

Додаток А

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

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Бакалавр**
Галузь знань: 15 – Автоматизація та приладобудування
Спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітня програма «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«02» жовтня 2024року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Жукова Олега Ігоровича
(прізвище, ім'я, по батькові студента)

1. Тема роботи **АВТОМАТИЗОВАНИЙ КРОСПЛАТФОРМНИЙ ЗАСТОСУНОК ДЛЯ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ З ІНСТРУМЕНТАМИ АДМІНІСТРУВАННЯ.**

керівник роботи **Хруслов Максим Михайлович, кандидат фізико-математичних наук, доцент**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від ***№ 4101-5/962 від 16.04.2025***

2. Строк подання студентом роботи ***30 травня 2025 року***

3. Перелік питань, які потрібно розробити:
1. Аналіз сучасних підходів до розробки автоматизованих крос-платформних застосунків для вивчення іноземних мов та їх адміністрування.
2. Розгляд концептуальних моделей крос-платформеного застосунку для вивчення іноземної мови та необхідних інструментів адміністрування.
3. Дослідження принципів тестування розробленої системи, для оцінки точності та надійності результатів.
4. Визначення можливих обмежень і проблем при використанні створеної моделі, таких як якість навчальних матеріалів та їх рівень.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1.	Затвердження теми роботи	05.09.2024 - 02.10.2024
2.	Літературний огляд з проблематики розробки автоматизованих крос-платформених застосунків	03.10.2024 - 15.10.2024
3.	Огляд методів розробки застосунків для мульти-платформеного користування з можливістю їх адміністрування	16.10.2024 - 30.10.2024
4.	Розробка концепції моделі автоматизованої лінгвістичної системи спрямованої на вивчення мов та впровадження інструментів адміністрування	01.11.2024 - 01.12.2024
5.	Тестування системи	12.12.2024 - 20.12.2024
6.	Аналіз результатів тестування та визначення ефективності	21.12.2024- 31.12.2024
7.	Підготовка рекомендацій та вдосконалення моделі	01.01.2025- 01.02.2025
8.	Підготовка і оформлення звітних матеріалів. Написання статті за матеріалами кваліфікаційної роботи.	02.02.2025- 31.03.2025-
9.	Підготовка і оформлення звітних матеріалів та додатків кваліфікаційної роботи. Оформлення списку літератури	01.04.2025- 15.04.2025-
10.	Оформлення пояснювальної записки кваліфікаційної роботи відповідно вимогам до звітів про НДР.	16.04.2025- 30.04.2025-
11.	Оформлення звіту про переддипломну практику	01.05.2025- 29.05.2025-
12.	Представлення кваліфікаційної роботи керівнику та рецензенту	30.05.2025-

5. Дата видачі завдання ***02 жовтня 2024 року.***

Студент О.І. Жуков _____
ініціали, прізвище підпис

Керівник роботи М.М. Хруслов _____
ініціали, прізвище підпис

Додаток Б

Затверджую

_____

«_____»_____2024 р.

Технічне завдання

на розробку системи

«Автоматизований кросплатформний застосунок для вивчення іноземної мови з інструментами адміністрування».

1.	Введення	1.1 Назва роботи – Автоматизований кросплатформний застосунок для вивчення іноземної мови з інструментами адміністрування. 1.2. Галузь застосування: Інформаційні технології
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – Автоматизація та комп'ютерно інтегровані технології 2.2. Завдання на кваліфікаційну роботу бакалавра №4101-5/962 від «16» квітня 2025 року (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1. Мета розробки: створення системи для автоматизації процесу тестування в рамках освітнього процесу, інтегрованої з Telegram WebApp для зручного доступу користувачів, а також адміністрування контенту та аналізу результатів через адміністративні інструменти, що функціонують як окремий бот та WebApp. Система передбачає інтеграцію з платформою Telegram, що дозволяє зробити освітні тести доступними без необхідності додаткових установок і реєстрацій. 3.2. Призначення розробки: призначення цієї розробки є створення технологічного рішення, яке підвищить безпеку шляхом автоматичного створення та управління навчальними тестами, а також забезпечить аналітику результатів для

		покращення навчального процесу. Система інтегрована з Telegram, що забезпечує легкий доступ до контенту та ефективне адміністрування. 3.3. Вхідні дані: вхідні дані для системи включають наступне: Дані користувача: збираються безпосередньо через Telegram, включаючи ідентифікатор Telegram користувача, ім'я та мовні налаштування. Ці дані використовуються для персоналізації користувацького досвіду та відстеження прогресу. Вміст тесту: включає тестові питання, відповіді, мультимедійні елементи (зображення, аудіо) та метадані (наприклад, назва тесту, опис, рівень складності). Тестові дані зберігаються в базі даних та керуються адміністраторами. Результати тесту: результати кожного тесту, проведеного користувачами, включаючи відсоток правильних відповідей, час, витрачений на виконання тесту, та індивідуальні варіанти відповідей. Ці дані зберігаються для аналізу та зворотного зв'язку. Дані адміністратора: список ідентифікаторів Telegram адміністраторів, що зберігається у файлі конфігурації .env, визначає доступ до інтерфейсу адміністратора. 3.4. Вихідні дані розробки: Розроблена система пропонує кілька ключових функцій: Створення та керування тестами: Адміністратори можуть створювати, редагувати та класифікувати тести, додаючи мультимедійні елементи, такі як зображення чи аудіо, для покращення залучення користувачів. Взаємодія з користувачем: користувачі можуть отримувати доступ до тестів безпосередньо через Telegram, проходити їх та отримувати негайний відгук про свою ефективність. Аналіз результатів: система надає детальну інформацію про ефективність користувачів, включаючи результати тестів, окремі відповіді та

		час виконання, що дозволяє адміністраторам виявляти тенденції та коригувати контент за потреби. Розповсюдження контенту: система дозволяє адміністраторам транслювати оновлення, нагадування про тести та навчальні матеріали через канали Telegram для користувачів. Сповіщення в режимі реального часу: система надсилає користувачам сповіщення в режимі реального часу про нові тести, терміни здачі або оновлення системи через Telegram.
4.	Технічні вимоги до програмного виробу	4.1. Функціональні вимоги: Система повинна відповідати таким функціональним вимогам: 1. Автентифікація користувачів: автентифікувати користувачів за допомогою їхнього Telegram ID для безпечного доступу. 2. Керування тестами: адміністратори повинні мати можливість створювати, редагувати та керувати тестами, включаючи запитання та відповіді. 3. Виконання тестів: користувачі повинні мати можливість проходити тести через Telegram WebApp та надсилати відповіді. 4. Розрахунок результатів: надавати користувачам їхні бали та відгуки після завершення тесту. 5. Відстеження прогресу: відстежувати продуктивність користувачів та історію тестів. 6. Розповсюдження контенту: адміністратори можуть транслювати тести та сповіщення через канали Telegram. 7. Панель адміністратора: адміністратори можуть керувати тестами та переглядати результати користувачів за допомогою Admin WebApp та бота Telegram. 8. Зберігання даних: зберігати дані користувачів, тестовий контент та результати в базі даних.

		9. Сповіщення: надсилати сповіщення про тести та оновлення в режимі реального часу. 10. Міжплатформна сумісність: система повинна функціонувати на різних платформах Telegram (мобільна, настільна, веб). 11. Масштабованість: підтримувати збільшення завантаження користувачів та контенту з часом. 12. Безпека: забезпечення безпечної передачі даних та автентифікації. 4.2. Нефункціональні вимоги: 1. Система також повинна відповідати наступним нефункціональним вимогам для забезпечення продуктивності, зручності використання та ремонтопридатності: 2. Продуктивність: система повинна реагувати на дії користувача (надсилання тесту, розрахунок результатів тощо) протягом 2 секунд, щоб забезпечити безперебійну роботу користувача. 3. Масштабованість: система повинна мати можливість обробляти до 10 000 одночасних користувачів без зниження продуктивності. 4. Доступність: система повинна мати 99% безвідмовної роботи, щоб користувачі завжди могли отримувати доступ до контенту та проходити тести без перерв. 5. Зручність використання: інтерфейс користувача повинен бути інтуїтивно зрозумілим, а всі функції (наприклад, проходження тесту, перегляд результатів) повинні бути простими в навігації. 6. Безпека: система повинна забезпечувати конфіденційність даних користувача, впроваджуючи шифрування конфіденційної інформації та безпечну автентифікацію через Telegram.

		7. Сумісність: система повинна бути сумісною з усіма сучасними версіями Telegram (iOS, Android, Desktop, Web). 8. Ремонтопридатність: код повинен бути модульним та добре документованим, щоб забезпечити майбутні оновлення та легкість налагодження. 9. Резервне копіювання та відновлення: система повинна реалізовувати регулярне резервне копіювання даних та забезпечувати механізми відновлення даних у разі системного збою. 10. Локалізація: система повинна підтримувати кілька мов з можливістю додавання нових мов за потреби. 4.3. Вимоги до інтеграції: Щоб забезпечити ефективну роботу системи та задоволення потреб користувачів і адміністраторів, необхідно виконати такі вимоги до апаратного та програмного забезпечення: Вимоги до апаратного забезпечення 1. Сервер: хмарний сервер з щонайменше 2 ядрами процесора та 4 ГБ оперативної пам'яті для обробки серверних процесів та взаємодії з Telegram. 2. Сховище: мінімум 50 ГБ сховища для бази даних та системних файлів з можливістю масштабування за потреби. 3. Мережа: стабільне інтернет-з'єднання з мінімальною пропускною здатністю 1 Мбіт/с для обробки обміну даними в режимі реального часу між користувачами та серверною частиною. Вимоги до програмного забезпечення 1. Операційна система: система повинна бути сумісною з серверами на базі Linux (Ubuntu, CentOS тощо) для оптимальної продуктивності. 2. Node.js: сервер серверної частини повинен використовувати Node.js (версії 14 або

		вище) для запуску програми та обробки асинхронних процесів. 3. MongoDB: система використовує MongoDB (версії 4 або вище) для зберігання даних користувачів, тестового контенту та результатів. Для масштабованості рекомендується хмарна база даних MongoDB Atlas. 4. Telegram API: додаток інтегрується з Telegram Bot API та Telegram WebApp SDK для забезпечення зв'язку та взаємодії з користувачем у режимі реального часу. 5. Фронтенд: клієнтський додаток створено за допомогою React (версії 17 або вище) та вимагає сучасного браузера (Chrome, Firefox, Safari) для оптимальної продуктивності. 6. Фреймворк бекенду: бекенд розроблено за допомогою Express.js для обробки RESTful API-запитів. 7. Документація API: система повинна надавати документацію API за допомогою таких інструментів, як Swagger або Postman, для легкої взаємодії з бекендом. Архітектура системи розроблена з урахуванням хмарних технологій та масштабованості, що гарантує можливість розширення апаратних та програмних ресурсів залежно від зростання кількості користувачів та обсягу контенту. 4.4. Вимоги до безпеки: Хоча система розроблена гнучкою та масштабованою, існують певні обмеження, які необхідно враховувати для забезпечення безперебійної роботи та продуктивності: 1. Масштабованість: система повинна мати можливість масштабуватися, щоб враховувати зростаючу кількість користувачів та тестів. Однак масштабованість залежить від серверної інфраструктури, і зі збільшенням кількості

		користувачів можуть знадобитися додаткові ресурси (наприклад, процесор, оперативна пам'ять, сховище). 2. Залежність від мережі: система покладається на стабільне інтернет-з'єднання для зв'язку в режимі реального часу між клієнтом та сервером. Повільне або нестабільне інтернет-з'єднання може призвести до затримок в обробці запитів та негативно вплинути на взаємодію з користувачем. 3. Обмеження Telegram API: інтеграція з Telegram Bot API та WebApp SDK має певні обмеження, такі як обмеження швидкості повідомлень, що надсилаються через бота, обмежене сховище даних для повідомлень та мультимедіа, а також обмеження на налаштування інтерфейсу WebApp. Ці обмеження необхідно враховувати під час проектування функцій комунікації та взаємодії з користувачем системи. 4. Сумісність з браузерами: хоча система розроблена для роботи в сучасних браузерах, можуть бути незначні відмінності у відображенні веб-застосунку залежно від використовуваного браузера (наприклад, Chrome, Firefox, Safari). Забезпечення сумісності з усіма підтримуваними браузерами є пріоритетом, але деякі певні функції можуть працювати по-різному на різних платформах. 5. Сумісність з мобільними пристроями: система розроблена для роботи як на мобільних, так і на настільних пристроях, але певні функції, такі як завантаження файлів або відображення великих таблиць даних, можуть потребувати оптимізації для мобільних користувачів, щоб забезпечити безперебійну роботу.

		6. Зберігання даних: хоча MongoDB пропонує гнучкість у зберіганні неструктурованих даних, можуть існувати обмеження щодо обсягу даних та складності запитів у міру зростання системи. Важливо впроваджувати належні стратегії індексації та оптимізації даних, щоб забезпечити ефективне отримання та зберігання даних. 7. Безпека: система повинна дотримуватися найкращих практик безпеки даних, включаючи шифрування конфіденційної інформації та безпечну автентифікацію через Telegram. Однак зовнішні фактори, такі як поведінка користувачів (наприклад, обмін інформацією про обліковий запис), можуть призвести до потенційних вразливостей. Ці обмеження слід враховувати як на етапі впровадження, так і на етапі майбутньої розробки проекту, щоб забезпечити надійність та ефективну роботу системи за різних умов.
5.	Вимоги до програмної документації	1. Для належного розуміння, використання та обслуговування системи потрібна вичерпна документація. Наступні типи документації є важливими: 2. Посібник користувача: чіткі інструкції для користувачів щодо взаємодії із системою, включаючи проходження тестів та перегляд результатів. 3. Посібник адміністратора: інструкції для адміністраторів щодо керування тестами, користувачами та контентом, а також використання веб-додатку та бота адміністратора. 4. Документація API: детальний опис кінцевих точок API серверної частини, форматів запитів/відповідей та обробки помилок за допомогою таких інструментів, як Swagger або Postman.

<table>
<tr><td></td><td></td><td colspan="2">5. Архітектура системи: огляд архітектури системи, включаючи діаграми, що зображують взаємозв'язок між фронтендом, серверною частиною, базою даних та компонентами Telegram.
6. Документація розробки: документація кодової бази для розробників, включаючи інструкції щодо участі в проекті, налаштування середовища та структури коду.</td></tr>
<tr><td>6.</td><td>Вимоги до техніко-економічних показників</td><td colspan="2">Програмна документація для продукту «Система навчального тестування з інтеграцією Telegram» повинна містити:
1. Технічну специфікацію на розробку системи (повинна бути представлена у Додатку В пояснювальної записки до роботи).
2. Методологію управління тестуванням та розрахунку результатів (повинна бути представлена у розділах 3.8 пояснювальної записки до роботи).
3. Опис системи (повинен бути представлений у розділі 2 та 3 пояснювальної записки до роботи).</td></tr>
<tr><td rowspan="7">7.</td><td rowspan="7">Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>21.12.2024 – 25.01.2025</td><td>Аналіз наукової літератури та сучасних рішень з розробки крос-платформених освітніх застосунків.</td></tr>
<tr><td>25.01.2025 – 02.02.2025</td><td>Формування вимог до технічних характеристик.</td></tr>
<tr><td>02.02.2025 – 02.02.2025</td><td>Розробка технічного завдання.</td></tr>
<tr><td>02.02.2025 – 03.02.2025</td><td>Розробка архітектури системи, вибір стеку технологій, структурування моделі даних.</td></tr>
<tr><td>03.02.2025 – 30.03.2025</td><td>Розробка клієнтської та серверної частин програмного забезпечення.</td></tr>
<tr><td></td><td></td></tr>
</table>

		30.03.2025 – 05.04.2025	Тестування функціоналу застосунку, перевірка стабільності, відлагодження.
		05.04.2025 – 27.04.2025	Підготовка технічного звіту
		27.04.2025 – 27.05.2025	Розробка пояснювальної записки.
		12.05.2025 – 31.05.2025	Представлення кваліфікаційної роботи керівнику та рецензенту.
		31.05.2025	Оформлення пояснювальної записки та підготовка презентації.
8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки комп'ютерної моделі виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в примірнику на CD-R компакт-диску.	

Виконавець

Студент групи КУ-41

Жуков О. І.

Замовник

к. ф.-м. н., доц.

Хруслов М. М.

Додаток В

Програма і методика випробувань програмного виробу

«Автоматизований кросплатформний застосунок для вивчення іноземної мови з інструментами адміністрування»

1. **Об’єкт випробувань**

1.1. Назва розробленого прототипу: «Автоматизований кросплатформний застосунок для вивчення іноземної мови з інструментами адміністрування».

1.2. Галузь застосування: Інформаційні технології

1.3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. **Мета випробувань**

Перевірка відповідності функціональні можливості системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. **Загальні положення**

3.1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

3.2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп’ютерного класу кафедри в період роботи атестаційної комісії.

3.3. Обсяг випробувань

Приймальні випробування програмного виробу проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавці та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

4.1. Функціональні вимоги:

1. Створення тестів

Система повинна дозволяти адміністраторам створювати, змінювати та публікувати тести, що містять текст, питання з вибором однієї правильної відповіді та додаткові мультимедійні елементи (зображення, аудіо).

2. Взаємодія з користувачем

Користувачі повинні мати доступ до тестів, надсилати відповіді та отримувати зворотний зв'язок у режимі реального часу щодо своєї роботи.

3. Розрахунок результатів

Система повинна автоматично розраховувати бали користувачів, надавати детальний зворотний зв'язок та дозволяти повторне складання тестів.

4. Можливості адміністратора

Адміністратори повинні мати доступ до детальних звітів про роботу користувачів, включаючи бали, тенденції та результати окремих тестів.

5. Розповсюдження контенту

Система повинна дозволяти адміністраторам транслювати оновлення, нові тести та оголошення користувачам через канали Telegram.

6. Запрошення на тестування

Система повинна дозволяти адміністраторам надсилати персоналізовані запрошення на тестування або нагадування певним користувачам.

4.2. Нефункціональні вимоги:

1. Міжплатформна сумісність

Система повинна функціонувати на всіх платформах Telegram (iOS, Android, Desktop).

2. Безпека

Система повинна забезпечувати безпечну автентифікацію через Telegram та гарантувати конфіденційність даних користувачів.

3. Масштабованість

Система повинна бути масштабованою для обробки до 10 000 одночасних користувачів.

4. Швидкість реагування

Система повинна завантажувати контент (тести, результати) протягом 3 секунд.

5. Доступність для користувачів

Система повинна бути доступною на мобільних пристроях та настільних комп'ютерах, пропонуючи адаптивний інтерфейс.

4.3. Вимоги до інтеграції:

1. Інтеграція з Telegram

Система повинна бути повністю інтегрована з WebApp SDK та Bot API Telegram для забезпечення безперебійної взаємодії з користувачами.

2. Інтеграція з Backend API

Фронтенд (Telegram WebApp) та бекенд (Node.js + MongoDB) повинні взаємодіяти через безпечні виклики RESTful API для отримання даних користувачів, надсилання тестів та розрахунку результатів.

3. Синхронізація даних

Синхронізація відповідей користувачів, результатів тестів та оновлень контенту між Telegram та бекендом у режимі реального часу.

4.4. Вимоги безпеки:

1. Шифрування даних

Усі конфіденційні дані (інформація про користувача, результати тестів) мають бути зашифровані під час передачі.

2. Бот та автентифікація користувачів

Система повинна гарантувати, що лише авторизовані користувачі можуть отримати доступ до функцій адміністрування через бота Telegram, а валідація користувачів здійснюється за допомогою безпечних токенів.

3. Цілісність даних

Система повинна регулярно створювати резервні копії даних користувачів, щоб запобігти втраті через системний збій.

5. Вимоги до документації програмного забезпечення

Документація до програмного забезпечення для продукту «Автоматизований кросплатформний застосунок для вивчення іноземної мови з інструментами адміністрування» повинна містити:

1. Опис основних вимог та функціональності системи (подається як Додаток В пояснювальної записки до роботи).

2. Програма тестування та методологія розробленого програмного забезпечення (подається як Додаток В пояснювальної записки до роботи).

3. Опис архітектури розробленої системи (подається в розділах 2 та 3 пояснювальної записки до роботи).

6. Засоби і порядок випробувань

6.1. Інструменти випробування

Для тестування системи використовувалися такі інструменти:

1. Модульне тестування:

Компоненти системи (наприклад, фронтенд-компоненти React, маршрути бекенд-API) тестувалися за допомогою Jest та бібліотеки React Testing Library для компонентів React. Ці інструменти дозволяли тестувати окремі елементи функціональності окремо.

2. Наскрізне (E2E) тестування:

Для тестування всього потоку користувача, включаючи проходження тестів та генерацію результатів, для автоматизації взаємодії з користувачем використовувався Cypress. Інструмент імітував фактичну поведінку користувача, таку як надсилання тестів, отримання результатів та навігація додатком.

3. Тестування API:

Кінцеві точки бекенд-API тестувалися за допомогою Postman, щоб переконатися, що всі RESTful-сервіси правильно реагують на очікувані дані. Документацію Swagger також було інтегровано для кращого управління та тестування API.

4. Ручне тестування:

Окрім автоматизованого тестування, було проведено ручне тестування для перевірки реального користувацького досвіду в Telegram WebApp та Bot.

6.2. Процедура тестування

Процедура тестування була виконана наступним чином:

1. Попереднє налаштування:

Процес тестування розпочався з налаштування сервера, бази даних та API бота Telegram. Було налаштовано тестове середовище та згенеровано початкові дані користувачів для імітації реального використання.

2. Модульне тестування:

Кожен системний модуль був протестований окремо за допомогою бібліотеки тестування Jest та React. Розробники провели ці тести, щоб підтвердити, що кожна функція (наприклад, створення тесту, розрахунок результату) працює належним чином.

3. Інтеграційне тестування:

Після модульного тестування було проведено інтеграційне тестування, щоб переконатися, що компоненти працюють разом правильно. Це включало перевірку взаємодії між фронтендом, серверною частиною та базою даних.

4. Наскрізне тестування:

Cypress був використаний для імітації реальної взаємодії користувачів у Telegram WebApp. Ці тести гарантували, що користувачі можуть успішно орієнтуватися в системі, проходити тести та отримувати результати. Тести також перевіряли наявність крайніх випадків, таких як повільне інтернет-з'єднання або невдалі запити API.

5. Ручне тестування:

Систему вручну протестувала група бета-користувачів, зосередившись на зручності використання та продуктивності в реальному часі в Telegram. Відгуки про це тестування допомогли виявити будь-які проблеми з інтерфейсом користувача або неочікувану поведінку.

6.3. Перевірка функціональності

Функціональність системи було перевірено за допомогою кількох раундів тестування. Основний функціонал, що підлягав перевірці, включав:

1. Створення та керування тестами:

Адміністратори можуть створювати нові тести, додавати запитання, визначати відповіді та класифікувати тести за предметом або рівнем складності. Цей функціонал було перевірено шляхом створення різних тестів в адміністративній панелі та забезпечення видимості тестів для користувачів у веб-додатку Telegram.

2. Взаємодія з користувачем:

Користувач повинен мати можливість проходити тести, надсилати відповіді та отримувати зворотний зв'язок у режимі реального часу. Здатність системи відстежувати відповіді, надавати зворотний зв'язок та розраховувати бали була ретельно перевірена.

3. Зворотній зв'язок у режимі реального часу:

Після того, як користувач завершить тест, зворотний зв'язок (включаючи відсотковий бал та правильні/неправильні відповіді) відображається негайно. Цю функцію було перевірено вручну шляхом надсилання відповідей та перевірки правильності надання зворотного зв'язку.

Рис. В.1 – Тест

4. Панель адміністратора:

Адміністратори могли переглядати результати користувачів, керувати вмістом тестів та надсилати сповіщення користувачам. Зручність використання панелі адміністратора перевірялася шляхом створення, редагування та видалення тестів, а також забезпечення відображення цих змін у системі.

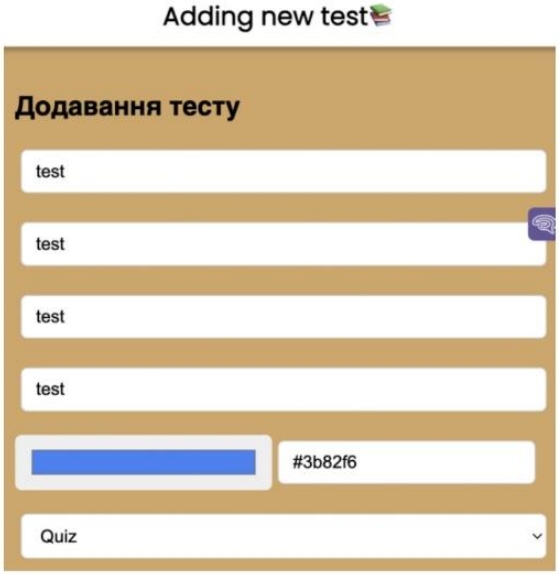

Рис. В.2 – Тестуємо створення тесту.

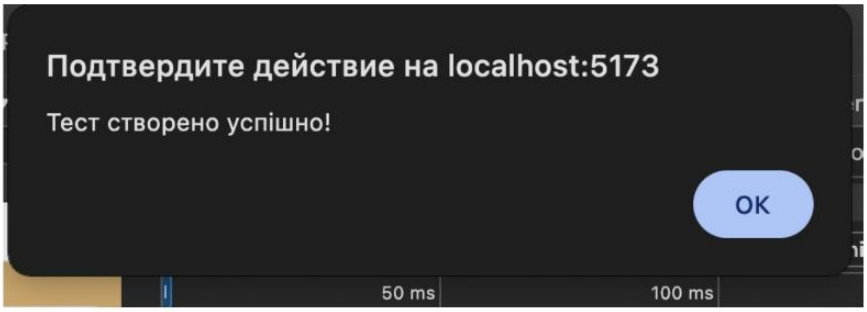

Рис. В.3 – Отримуємо повідомлення.

Рис. В.4 – Перевіряємо результати у Network.

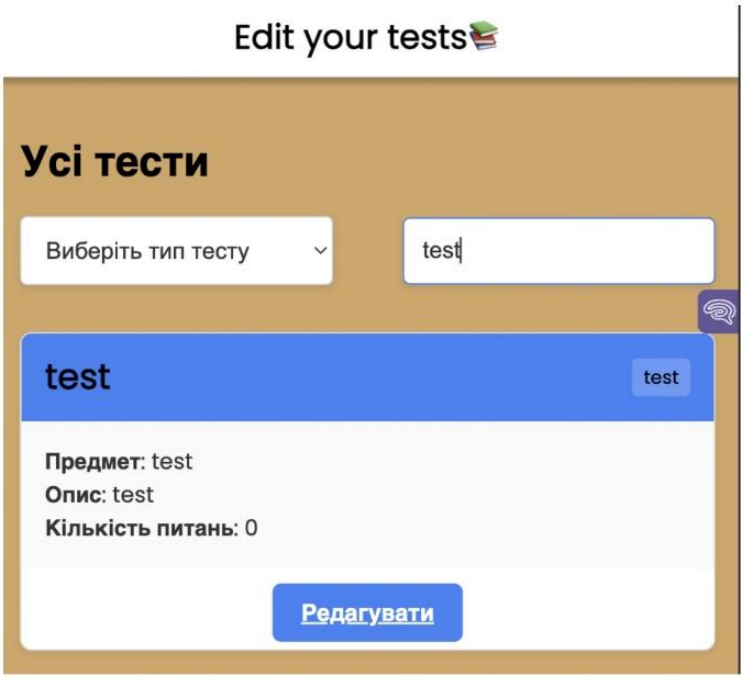

Рис. В.5 – Шукаємо створений тест на сторінці з усіма.

6.4. Перевірка інтеграції

Інтеграцію системи було перевірено для забезпечення безперебійної роботи всіх її компонентів:

1. Інтеграція фронтенду та бекенду: фронтенд (React WebApp) та бекенд (Node.js + Express API) були інтегровані та протестовані разом. Запити API від веб-додатку були перевірені шляхом перевірки даних відповідей та забезпечення відображення правильних результатів.
2. Зв'язок між Telegram-ботом та веб-додатком: Взаємодію між Telegram-ботом та веб-додатком було ретельно протестовано. Коли користувач взаємодіє з ботом, тестові дані та результати безперешкодно обмінюються між ботом та серверною системою. Синхронізацію даних між Telegram WebApp та ботом було протестовано за допомогою реальних сценаріїв користувачів.

Рис. В.6 – Бачимо, що WebApp отримав InitData, внаслідок чого вивів ім'я користувача в Telegram

1. Інтеграція з базою даних:

Базу даних MongoDB було протестовано, щоб забезпечити правильне зберігання та отримання даних користувачів, тестових даних та результатів. Також було перевірено час відгуку системи та здатність керувати великими обсягами тестових даних.

2. Розповсюдження контенту через канали:

Було протестовано можливість трансляції контенту (наприклад, запрошень на тестування та оголошень) через канали Telegram. Ця функціональність була вирішальною для забезпечення своєчасного отримання користувачами сповіщень та можливості реагувати на них.

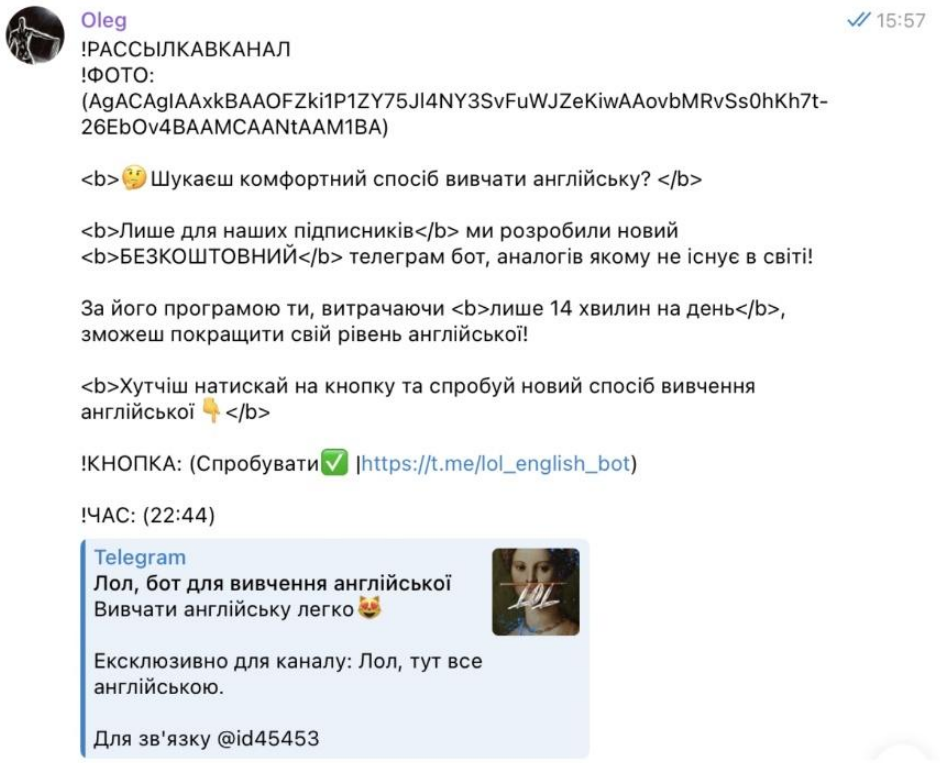

Рис. В.7 – Надсилання повідомлення в канал

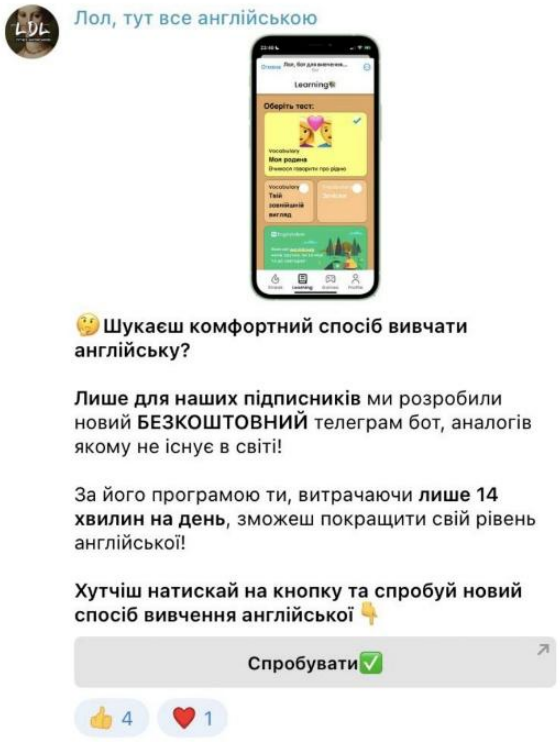

Рис. В.8 – Отримане повідомлення вже в каналі

6.5. Висновок

Процеси тестування, налагодження та перевірки забезпечили стабільність, надійність та належне функціонування системи. Було протестовано та підтверджено, що всі основні функції працюють коректно, включаючи створення тестів, взаємодію з користувачем, зворотний зв'язок щодо результатів та адміністративні функції. Майбутні цикли тестування будуть зосереджені на масштабованості, безпеці та покращенні взаємодії з користувачем.

Студент О.І. Жуков _____

ініціали, прізвище підпис