МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)
червня 202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерні науки,
освітньо-професійної програми «Інформатика»
на тему: Інформаційна веборієнтована система для організаційної роботи студентського деканату
здобувача групи ІН – 03 Азаренка Олександра Володимировича

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

Олександр АЗАРЕНКО
(підпис)

Керівник, доцент,
кандидат фізико-математичних наук,
доцент Галина ОЛЕКСІЄНКО
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерні науки, освітньо-професійної програми «Інформатика»
здобувача групи ІН-03, Азаренка Олександра Володимировича

1. Тема роботи: «Інформаційна веборієнтована система для організаційної роботи студентського деканату»
затверджую наказом по СумДУ від *«22» квітня 2024_ р. № 0414-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 31 травня 2024_ року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для автоматизації процестів в ОСС. 3) Розробка веборієнтованої системи для організаційної роботи студентського деканату. 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «06» травня 2024 р.
Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	06.05.24-10.05.24	
2	*Огляд веборієнтована систем, що використовуються для автоматизації процесів студентського самоврядування*	11.05.24-15.05.24	
3	*Розробка інформаційної веборієнтованої системи*	16.05.24-26.05.24	
4	*Аналіз отриманих результатів*	26.05.24-28.05.24	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	29.05.24- 31.05.24	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 87 стр., 68 рис., 1 табл., 8 додатків, 21 використане джерело.

Обґрунтування актуальності теми роботи — обумовлена необхідністю створення інформаційної веборієнтованої системи для організаційної роботи органів студентського самоврядування, автоматизації та оптимізації багатьох процесів, таких як: збирання даних про студентів, управління кураторами, забезпечення зручного каналу комунікації між студентами та студентським деканатом. Використання сучасних технологій, таких як інтеграція з Telegram ботами, дозволяє значно підвищити ефективність роботи деканату, забезпечити своєчасну та достовірну інформацію, а також створити сприятливі умови для інтерактивного навчання та розвитку студентів.

Об’єкт дослідження — інформаційна система організаційної роботи студентського самоврядування.

Предмет дослідження – процеси автоматизації та оптимізації адміністративних функцій студентського деканату.

Мета роботи — Розробка інформаційної веборієнтованої система для організаційної роботи студентського деканату на основі платформи Telegram.

Методи дослідження — аналіз, проектування, розробка та тестування веборієнтованих систем, опитування та аналіз зворотного зв'язку від користувачів.

Результати — розроблено та впроваджено інформаційну веборієнтовану систему для організаційної роботи студентського деканату, що включає веб-додаток з функціоналом форм, систему управління кураторами, авторизацію через Telegram, статистику та довідкові матеріали. Додатково створено Telegram бота з багатофункціональними можливостями для інтерактивної взаємодії зі студентами, управління чатами, проведення ігор, та надання освітніх матеріалів.

ШТУЧНИЙ ІНТЕЛЕКТ, ІНФОРМАЦІЙНА СИСТЕМА, ВЕБ-РОЗРОБКА, TELEGRAM БОТ, PHP, АВТОМАТИЗАЦІЯ, СТУДЕНТСЬКИЙ ДЕКАНАТ

ЗМІСТ

ВСТУП 6
1. АНАЛІТИЧНИЙ ОГЛЯД 7
1.1 Поняття штучного інтелекту (ШІ) 8
1.2 Приклади використання ШІ в навчальному процесі 9
1.3 Аналіз проєктів студентського самоврядування 11
1.4 Постановка задачі 13
2 МЕТОД РОЗВ'ЯЗАННЯ ЗАДАЧІ 14
2.1 Інструментарій 14
2.1.1 Технології та інструменти для розробки веб-додатку 14
2.1.2 Інструментарій для розробки Telegram бота 15
2.1.3 База даних та веб-сервер 16
2.2 Інформаційна модель системи 16
2.3 Деталі реалізації Telegram бота 17
3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ 19
3.1 Прототип 19
3.2 База даних 20
3.3 Бібліотека Telegram бота 29
3.4 Функціональність 40
3.4.1 Взаємодія бота та веб-додатку 40
3.4.2 Telegram бот 45
3.4.3 Веб додаток 53
ВИСНОВКИ 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 67
ДОДАТОК А (bot.php) 69
ДОДАТОК Б (start.php) 72
ДОДАТОК В (loader.php) 73
ДОДАТОК Г (chat.php) 74
ДОДАТОК Д (chatMember.php) 78
ДОДАТОК Е (User.php) 80
ДОДАТОК Ж (chatGPT.php) 83

ДОДАТОК К (Схема бази даних) 86

ВСТУП

Обґрунтування вибору теми роботи обумовлено необхідністю створення ефективної інформаційної системи для організаційної роботи студентського деканату. В сучасних умовах автоматизація та оптимізація адміністративних процесів у вищих навчальних закладах стає важливою для забезпечення якісного управління та покращення освітнього процесу.

Актуальність. Інформаційні веборієнтовані системи для організаційної роботи студентських деканатів значно підвищують ефективність управління та комунікації між студентами і адміністрацією. Це дозволяє оперативно вирішувати питання, пов'язані з навчанням і адміністративними процесами, а також забезпечує зручний доступ до необхідної інформації.

Об’єкт дослідження. Інформаційна веборієнтована система організаційної роботи студентського самоврядування.

Предмет дослідження. Процеси автоматизації та оптимізації адміністративних функцій студентського деканату.

Гіпотеза. Впровадження розробленої інформаційної веборієнтованої системи дозволить значно покращити ефективність організаційної роботи студентського деканату, підвищити рівень комунікації зі студентами та оптимізувати адміністративні процеси.

Новизна. Розробка комплексної веборієнтованої системи, що включає інтеграцію з Telegram ботом для забезпечення багатофункціональної взаємодії зі студентами та автоматизації ключових адміністративних процесів.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибору методу розв’язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД

Розробка інформаційної системи дружньої до користувача в сучасних умовах стає дедалі важливішою для вищих навчальних закладів. Системи управління навчальним процесом (LMS), системи для збору даних про студентів, а також інструменти для комунікації та організації роботи значно підвищують ефективність адміністративних процесів. Сучасні інформаційні системи мають на меті забезпечення швидкого доступу до даних, інтеграцію з різноманітними сервісами та платформами, а також забезпечення безпеки та конфіденційності інформації [21].

Особливу роль в цьому відіграють технології штучного інтелекту (ШІ). Вони сприяють автоматизації багатьох рутинних завдань, таких як сортування та аналіз даних, надання персоналізованих рекомендацій студентам, а також вдосконалення процесів комунікації між викладачами та студентами [14]. Завдяки алгоритмам машинного навчання, системи можуть адаптуватися до потреб кожного користувача, підвищуючи таким чином ефективність навчального процесу та адміністративної роботи.

Студентам зручно отримувати інформацію в одному місці, а це саме Telegram, що дозволяє уникати постійного переходу між різними платформами, такими як Google Forms, Google таблиці, MIX рис. 1.1 та 1.2 (створений для взаємодії викладачів зі студентами, а не органів студентського самоврядування), Zoom [19] , Meet [20] рис. 1.3 та інші. Інтеграція ШІ у ці платформи робить їх ще більш потужними та гнучкими.

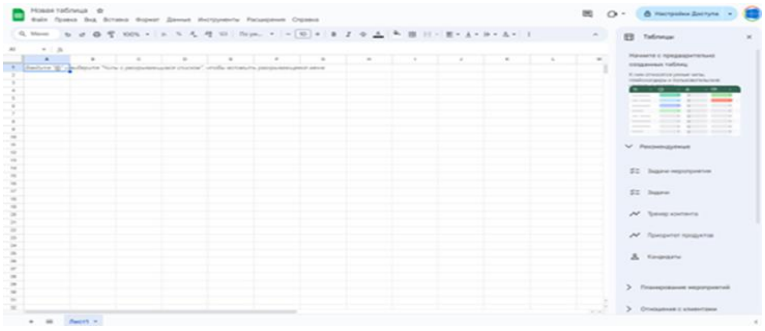


Рисунок 1.1 – Платформа Google таблиці

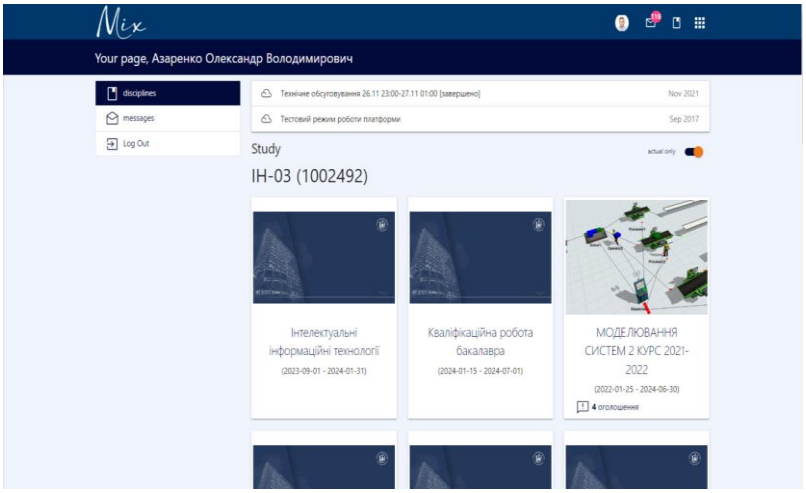


Рисунок 1.2 – Платформа MIX

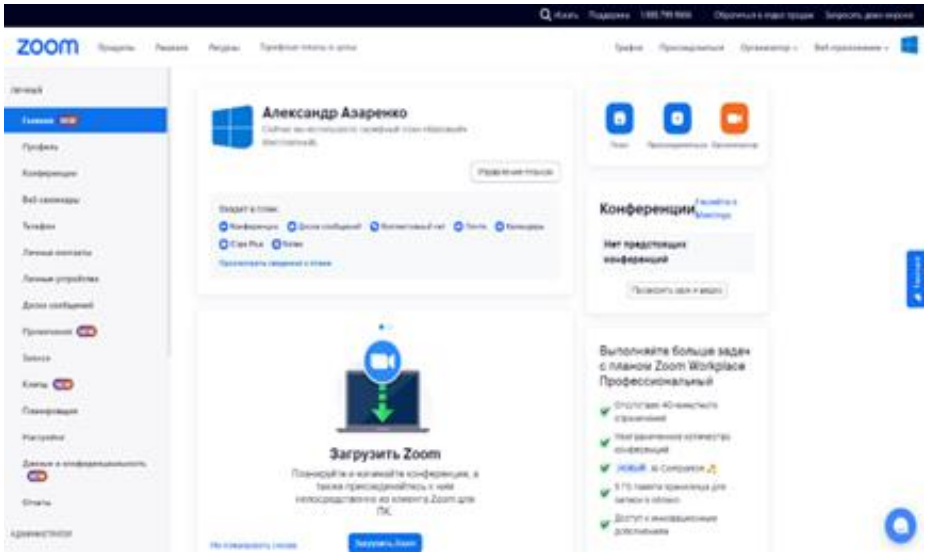


Рисунок 1.3 – Платформа Zoom

1.1 Поняття штучного інтелекту (ШІ)

Штучний інтелект використовує комп'ютери для імітації здатності людського розуму вирішувати проблеми та приймати рішення. Інакше кажучи, штучний інтелект — це метод програмування комп'ютера таким чином щоб він думав, як високоінтелектуальна людина. Останні кілька десятиліть з'явилося багато визначень штучного інтелекту. Цей термін часто застосовується до проєкту розробки систем, наділених інтелектуальними процесами, характерними для людей, такими як здатність міркувати,

розкривати зміст, узагальнювати чи отримувати уроки з минулого досвіду [21].

Штучний інтелект (ШІ) – це розділ інформатики, що займається створенням систем і програм, здатних виконувати завдання, які вимагають інтелектуальних зусиль від людини. Це можуть бути процеси навчання, міркування, розпізнавання мови, візуальне сприйняття, прийняття рішень та інші види інтелектуальної діяльності. ШІ базується на використанні алгоритмів і моделей машинного навчання, що дозволяє системам аналізувати великі обсяги даних, виявляти закономірності та робити прогнози [17].

Важливою складовою ШІ є здатність до самонавчання, що дозволяє системам покращувати свою ефективність з часом без прямого втручання людини. Штучний інтелект знаходить застосування у різних сферах, включаючи медицину, фінанси, освіту, транспорт, виробництво та багато інших, сприяючи автоматизації процесів та підвищенню ефективності роботи.

1.2 Приклади використання штучного інтелекту в навчальному процесі

Штучний інтелект відіграє ключову роль у сучасному навчальному процесі, надаючи нові можливості для покращення якості освіти та підвищення ефективності навчання. Одним з найяскравіших прикладів є використання ШІ для створення персоналізованих навчальних програм, які адаптуються до індивідуальних потреб кожного студента, враховуючи їхній рівень знань, інтереси та стиль навчання. Такі платформи, як Coursera та Khan Academy рис. 1.4 та 1.5, використовують алгоритми машинного навчання для аналізу даних про успішність студентів і надання їм персоналізованих рекомендацій щодо подальших курсів і навчальних матеріалів.

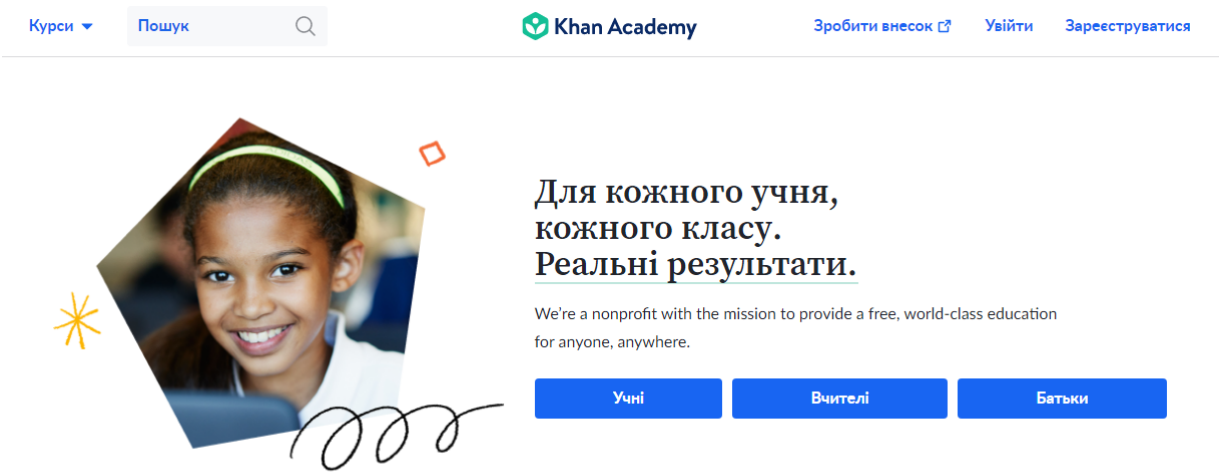


Рисунок 1.4 – Платформа Khan Academy

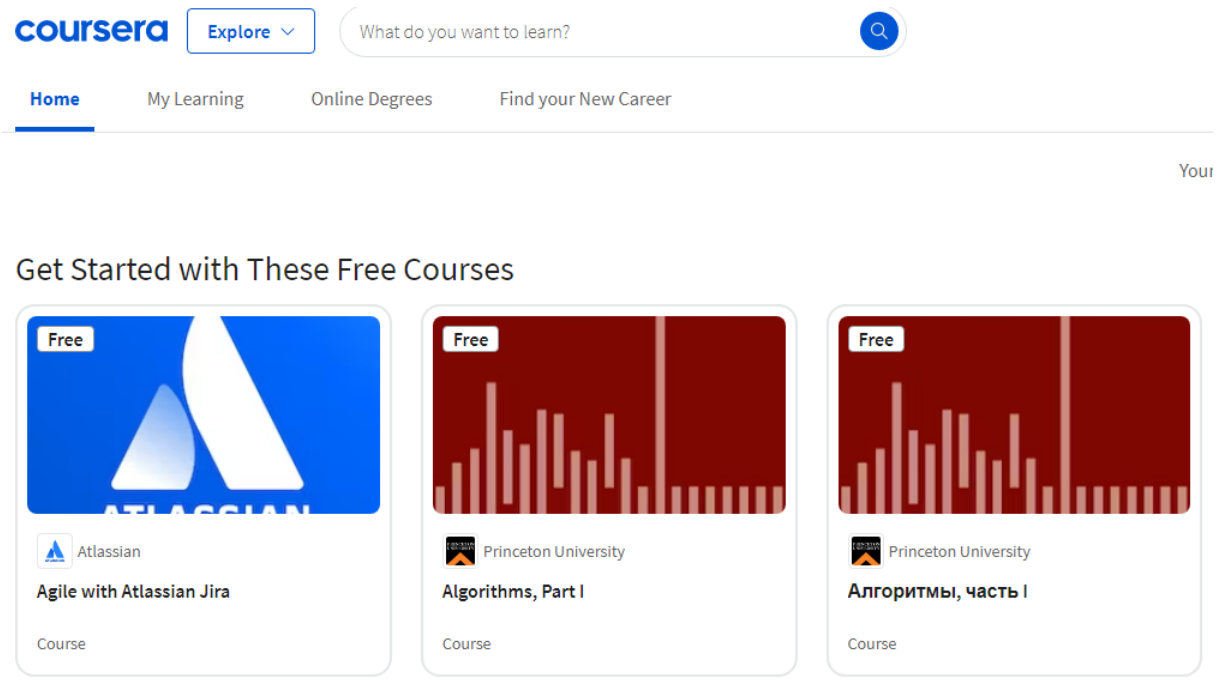


Рисунок 1.5 – Платформа Coursera

Іншим прикладом є використання ШІ для автоматизації оцінювання та зворотного зв'язку. Системи на основі ШІ можуть автоматично перевіряти та оцінювати письмові роботи, тести та завдання, забезпечуючи швидкий і об'єктивний зворотний зв'язок для студентів. Це не тільки знижує навантаження на викладачів, але й дозволяє студентам отримувати миттєві результати та рекомендації для покращення [18].

Чат-боти, що працюють на базі ШІ, також стають незамінним інструментом у навчальному процесі. Вони можуть відповідати на запитання студентів у режимі реального часу, надавати навчальні матеріали, допомагати в організації групових проектів та навіть проводити тестування знань. Наприклад, чат-боти в Telegram [1] або Microsoft Teams [16] можуть інтегруватися з навчальними платформами, забезпечуючи зручний та ефективний спосіб взаємодії між студентами та викладачами.

ШІ також використовується для аналізу великих обсягів даних з метою ідентифікації тенденцій та проблем у навчальному процесі. Це дозволяє адміністраціям навчальних закладів приймати обґрунтовані рішення щодо покращення навчальних програм та процесів. Наприклад, аналіз даних про відвідуваність, успішність та взаємодію студентів допомагає виявляти потенційні проблеми та розробляти стратегії для їх вирішення.

1.3 Аналіз проєктів студентського самоврядування

Існує ряд проєктів, спрямованих на автоматизацію адміністративних процесів у вищих навчальних закладах. Наприклад, Google Forms забезпечують зручний інструмент для збору даних та опитувань, однак не інтегрують ці дані з адміністративними системами університетів. Системи управління навчальним процесом, такі як MIX [15], забезпечують управління курсами та навчальним контентом, але не завжди включають інструменти для організаційної роботи органів студентського самоврядування.

Інші проєкти, такі як Microsoft Teams, надають інструменти для комунікації та співпраці, але не фокусуються на специфічних потребах органів студентського самоврядування. Всі ці системи мають свої переваги та недоліки, і нова розробка має поєднувати їхні кращі риси для задоволення потреб студентського деканату.

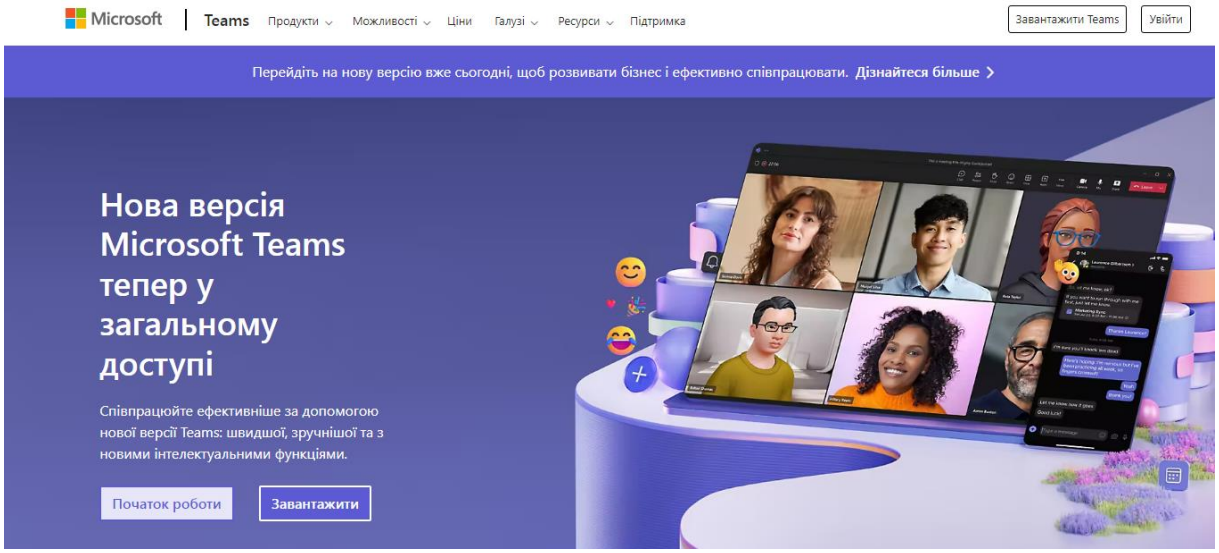


Рисунок 1.6 – Платформа Microsoft Teams

Таблиця 1.1 Порівняння проєктів, спрямованих на автоматизацію адміністративних процесів у вищих навчальних закладах

Сервіс	**Опис**	**Переваги**	**Недоліки**
1	**2**	**3**	**4**
MIX	Система управління курсами та навчальним контентом.	Зручна платформа для облаштування освітнього процесу.	Не має в собі ціль автоматизації роботи студентських органів самоврядування та взаємодії між студентами, не оптимізований для використання на смартфонах, не має додатку.
Google Forms	Онлайн-інструмент для створення анкет, опитувань і форм збору даних	Простота використання, доступність, можливість швидкого створення та поширення форм.	Відсутність інтеграції з адміністративними системами, обмежена функціональність для керування студентськими групами, недостатній рівень автоматизації, Обмежені можливості комунікації
Microsoft Teams	Потужна платформа для спільної роботи, яка забезпечує	Комунікація, Спільна робота, Доступ до ресурсів, Інтеграція з іншими сервісами.	Відсутність спеціалізованих функцій: орієнтованість на корпоративне

	інтеграцію багатьох інструментів для комунікації, співпраці та управління завданнями.		середовище. Складність інтеграції з Telegram, Недостатня гнучкість у налаштуваннях, Високі вимоги до апаратного забезпечення та інтернет-з'єднання. Потужна, але в цей же час складна система для позанавчальної взаємодії студентів.

1.4 Постановка задачі

Метою роботи є розробка комплексної інформаційної веборієнтованої системи для організаційної роботи органів студентського самоврядування, яка включає в себе інтеграцію з Telegram ботом та забезпечує автоматизацію ключових адміністративних процесів.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1) Проаналізувати сучасний стан інформаційних систем у вищих навчальних закладах;
2) Дослідити та порівняти існуючі аналогічні проєкти;
3) Розробити інформаційну модель системи;
4) Реалізувати веборієнтовану систему з інтеграцією з Telegram ботом;
5) Провести тестування системи та аналіз результатів її роботи.

2 МЕТОД РОЗВ'ЯЗАННЯ ЗАДАЧІ

Для успішного розв'язання задачі були використані сучасні веб-технології та програмні засоби. Система складається з двох окремих проектів – веб-додатка та Telegram бота, які взаємодіють між собою через спільну базу даних. Мова програмування backend частини веб-додатку та Telegram бота – PHP, бо це найпопулярніша мова програмування для веб-додатків. Її популярність обумовлена кількома ключовими факторами. По-перше, PHP є відносно простою у вивченні та використанні мовою, що робить її доступною для широкого кола розробників, від початківців до професіоналів. По-друге, PHP має широку підтримку в спільноті розробників, яка створила велику кількість бібліотек, фреймворків та інструментів, що спрощують та прискорюють процес розробки.

2.1 Інструметарій розробки

Для розробки інформаційно-аналітичної системи, яка включає веб-додаток та Telegram бота, були вибрані певні технології та інструменти.

2.1.1 Технології та інструменти для розробки веб-додатку

Для розробки веб-додатку було використано такі технології:

Frontend:

HTML, що є стандартною мовою розмітки для створення веб-сторінок. Він широко використовується у веб-розробці, забезпечуючи сумісність з усіма сучасними браузерами.

CSS дозволяє розділити структуру та стиль веб-сторінки, що спрощує підтримку та оновлення дизайну. Крім того, CSS забезпечує адаптивність інтерфейсу для різних пристроїв.

JavaScript, що є однією з найпопулярніших мов програмування для створення динамічних веб-сторінок. Він дозволяє реалізувати інтерактивні елементи та покращити користувацький досвід.

JQuery спрощує роботу з JavaScript, особливо при маніпуляції DOM-елементами, обробці подій та AJAX-запитах. Це дозволяє швидше і легше створювати інтерактивні веб-додатки.

Bootstrap для створення інтерфейсу користувача, що є одним з найпопулярніших CSS-фреймворків, який забезпечує швидку розробку адаптивних і сучасних інтерфейсів користувача. Він має вбудовані компоненти та стилі, що значно прискорює процес розробки.

Backend:

PHP для обробки запитів та управління даними, бо це одна з найпоширеніших серверних мов програмування для веб-розробки. Вона легко інтегрується з базами даних, має широку підтримку спільноти та велику кількість готових рішень і бібліотек.

2.1.2 Інструментарій для розробки Telegram бота

Для розробки Telegram бота було використано:

Мова програмування:

PHP та NodeJS з бібліотеками. Вони обрані бо дозволяють реалізувати різноманітні функціональні можливості бота. PHP використовується для обробки серверних запитів та взаємодії з базою даних, тоді як NodeJS забезпечує асинхронну обробку запитів, що підвищує продуктивність.

PHP бібліотеки:

Composer – пакетний менеджер PHP. Він спрощує управління залежностями проекту, дозволяючи легко додавати та оновлювати бібліотеки, що забезпечує гнучкість та ефективність розробки.

Process.php - для зручного керування фоновими скриптами.

Orhanerday\OpenAi - для взаємодією з API OpenAI, що дозволяє інтегрувати передові AI функції у Telegram бот.

NodeJS бібліотеки:

Mysql – для зв’язку з базами даних. Ця бібліотека забезпечує ефективну роботу з базами даних MySQL у NodeJS, що дозволяє обробляти запити та зберігати дані.

Express – для зручного створення та відправки запитів.

node-telegram-bot-api – для взаємодії з API Telegram у NodeJS частинах програми

2.1.3 База даних та веб-сервер

База даних:

MySQL для зберігання та управління даними. що є однією з найпопулярніших систем управління базами даних. Вона забезпечує високу продуктивність, надійність та підтримку транзакцій. Крім того, MySQL має велику спільноту користувачів і велику кількість документації.

Взаємодія з базою даних: PHPMyAdmin для забезпечення зручного адміністрування БД.

Веб сервер:

Apache2 є одним з найпоширеніших веб-серверів, який забезпечує надійність, масштабованість та підтримку різних модулів для розширення функціональності.

Постачальник CDN, захисту від DDOS атак та SSL сертифікатів: Cloudflare, Inc.

2.2 Інформаційна модель системи

Інформаційна модель системи включає в себе структуру даних, що зберігається у базі даних, та взаємозв'язки між різними компонентами системи. Основні таблиці бази даних включають дані про студентів, кураторів, форми та відповіді на них, а також інформацію для управління правами доступу та статистикою використання.

Детальна структура бази даних представлена в додатку К, вона відображає всі основні таблиці та взаємозв'язки між ними. Ця структура дозволяє централізовано зберігати та обробляти всі необхідні дані для

функціонування як веб-додатку, так і Telegram бота.

Архітектура системи включає три основні компоненти: клієнтську частину (веб-додаток), серверну частину (сервер додатка) та Telegram бота. Веб-додаток забезпечує інтерфейс для створення та управління формами, а також для керування кураторами та правами доступу. Telegram бот інтегрований у чати груп та факультетів, забезпечуючи функціонал для інтерактивної взаємодії зі студентами та органами самоврядування.

Компоненти системи взаємодіють через спільну базу даних, що дозволяє синхронізувати дані між веб-додатком та ботом. Така архітектура забезпечує високу гнучкість та масштабованість системи, дозволяючи легко додавати новий функціонал та розширювати можливості системи в майбутньому.

2.3 Деталі реалізації Telegram бота

Telegram бот було розроблено з використанням PHP бібліотеки для роботи з Telegram Bot API, яка була спеціально розроблена під цей проект, що дозволяє забезпечити широкий спектр функцій та інтеграцій. Основні компоненти бота включають:

KERNEL – Ядро бота, що зображене на рис. 2.1

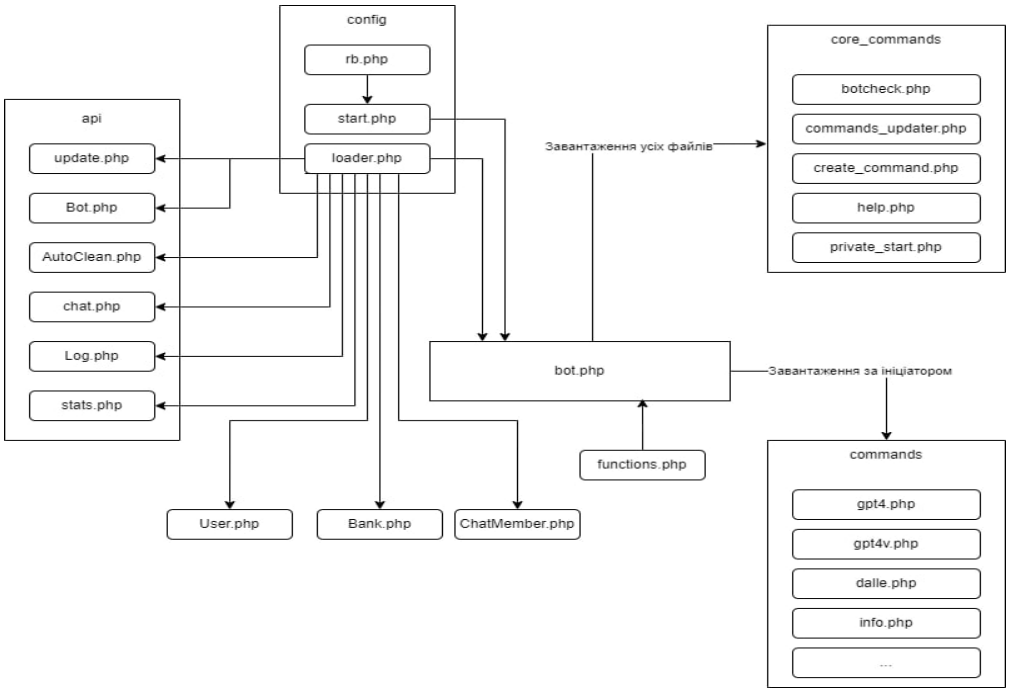


Рисунок 2.1 - Ядро Telegram бота

TG_BOT – ядро та усі додаткові файли: команди, бібліотеки, фонові процеси, без яких бот може працювати маючи мінімальний функціонал.

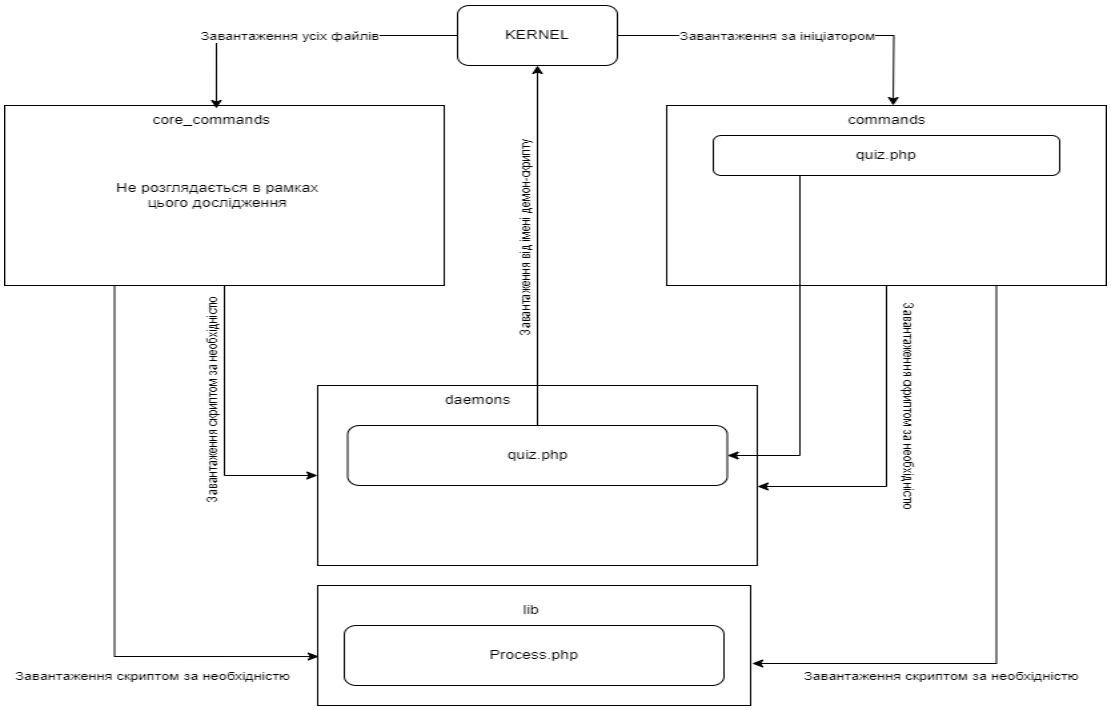


Рисунок 2.2 - Типова схема допоміжного функціонала разом з ядром

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Прототип

Формування вхідних даних є критичним етапом у розробці інформаційної системи. Вхідні дані, що використовуються у системі, надходять із різних джерел та повинні бути структуровані та оброблені для подальшого використання. Основні джерела вхідних даних включають:

- Студент
- Куратор
- Студентський деканат

Процес формування вхідних даних включає:

- Збирання даних. Використання форм у веб-додатку для збору даних про студентів, кураторів та відповіді на форми.
- Автоматизований збір даних. Telegram бот автоматично збирає дані про учасників чатів, їхні повідомлення та активність.
- Валідація даних. Перевірка введених даних на правильність та відповідність встановленим критеріям (наприклад, формат номера телефону, унікальність ідентифікатора користувача).
- Зберігання даних. Дані зберігаються у базі даних MySQL, що забезпечує централізоване управління та легкий доступ до інформації для всіх компонентів системи.

Розглянемо основні компоненти системи.

Команди бота

За команди бота відповідають три таблиці: actions, action_types та command_files. Вони автоматично заповнюються при оновленні списку команд, подібно до оновлення списку репозиторіїв у системах Linux. Дані беруться з коментарів, які обов'язково вказуються у кожному файлі команди.

Таблиця **actions** зберігає ініціатори для команд:

- **initiator** - ініціатор команди (в залежності від типу).
- **type** - посилання на тип ініціатора (таблиця action_types, яка в свою чергу зберігає назву ініціаторів. На даний момент має три типи: текст, callback та display).
- **file_id** - посилання на файл команди.
- **args** - мінімальна кількість аргументів для запуску команди. Якщо користувач не вкаже їх, то він отримає помилку та довідку з правильним синтаксисом команди.

Таблиця **command_files** зберігає інформацію про файли команд:

- **name** - назва команди.
- **filename** - назва файла на сервері, що відповідає за цю команду.
- **info** - опис, що робить ця команда для користувача.
- **syntax** - синтаксис написання команди.
- **rank** - мінімальний ранг для запуску команди.

Ініціатори в таблиці actions мають різні формати:

Якщо тип - текст, тоді бот перевіряє перше слово користувача в повідомленні і, якщо воно збігає, виконує відповідну команду. Display - перевірка по стану стейт машини (показчик user.display, використоввується для розуміння у якому режимі зараз знаходиться користувач при взаємодії з ботом). Callback - перевірка натискання inline кнопки.

3.2 База даних

Користувачі

Головна таблиця для зберігання інформації про користувачів - це **users**. Вона містить:

- **user_id** - ідентифікатор користувача.
- **telegram_id** - ідентифікатор користувача в Telegram.
- **nickname** та **username** - нікнейм та юзернейм користувача.
- **rank** - ранг користувача (є різні ранги від USER до OWNER, що визначають рівень прав та доступу).

- **tmp** - комірка для зберігання тимчасових даних при використанні бота.
- **display** - маркер стейт машини при використанні бота.
- **balance, balance_usd, diamonds** - різні типи віртуальних валют.
- **reg_date** - дата реєстрації користувача.
- **grp** - посилання на групу користувача.
- **avatar -** назва файлу аватара на сервері.
- **botcheck** - чи пройшов користувач перевірку анти-бот системи.
- **blacklist** - час розблокування, якщо користувач доданий до чорного списку.

З таблицею **users** пов'язані: **users_extra** - додаткові дані про таймаути між використаннями різних команд, **users_achievements –** зберігає отримані досягнення користувача, та **users_confidential** - конфіденційна інформація про користувача (ім'я, прізвище та номер телефону).

Групи

Таблиця **groups** зберігає:

- **name** – назву групи
- **gcode** - ідентифікатор групи в системі СумДУ, використовується для отримання розкладу занять.

Куратори

Таблиця **curators** зберігає:

- Посилання на користувача.
- Посилання на групу, яку курирує куратор.
- Посилання на чат групи.

Чати

Таблиця **chats** зберігає інформацію про чати:

- **tg_id** - ідентифікатор чату в Telegram.
- **title** - назву чату.
- **type** - тип чату (група, супергрупа, приватний чат чи канал).

- **botcheck** - чи вмикнена система перевірки користувачів антибот системою у чаті.
- **welcome_text** та **welcome_photo** - текст та фото для вітання нових користувачів.
- **rules_text** - правила чату.
- **aviator_started** - чи запущена гра "Авіатор" у чаті.
- **autoclean** - чи вмикнена система автоматичного видалення повідомлень бота.
- **autoclean_delay** - затримка перед видаленням повідомлень бота (в секундах).

Система автовидалення повідомлень бота

Щоб запобігти спаму, була спроектована система автовидалення повідомлень. За неї відповідає таблиця autoclean, яка зберігає:

- Посилання на чат.
- **tg_msg_id** - ідентифікатор повідомлення бота.
- **date** - дата та час повідомлення для розуміння, коли його необхідно видалити в залежності від налаштувань чату.

Учасник чату

Для зберігання інформації про користувача та окремий чат була створена таблиця **chatmembers**, яка дозволяє відстежувати участь користувача в різних чатах та його активність. Ось детальний опис полів таблиці:

- **user_id** - посилання на користувача, що є унікальним ідентифікатором користувача в системі.
- **chat_id** - посилання на чат, що є унікальним ідентифікатором чату в Telegram.
- **is_admin** - булеве значення, яке вказує, чи має користувач права адміністратора у цьому чаті (1 - має, 0 - не має).
- **last_update_status** - дата та час, коли останній раз було оновлено

статус адміністратора чату для цього користувача.

- **blacklist** - дата та час, коли користувача буде видалено із чорного списку бота у цьому чаті. Якщо користувач не має блокування, це поле буде містити значення NULL.
- **message_counter** - кількість повідомлень, надісланих користувачем у цьому чаті. Це поле допомагає відстежувати активність користувачів та їх внесок у спілкування в чаті.

Досягнення

Усі можливі досягнення зберігаються в таблиці **achievements**:

- **emoji –** емодзі символ досягнення
- **name –** назва досягнення
- **description –** опис за що отримане досягнення
- **telegraph –** стаття з описом в Telegram (для більш зручного перегляду на мобільних пристроях)

Налаштування

Налаштування усієї системи зберігаються у таблиці **settings**, наприклад, кількість монет в банку, скільки користувач отримує монет при реєстрації, id останнього коду, чи вимкнено режим дебагінгу та інше.

- **name** – назва налаштування
- **value** – значення

botcheck

- **user_id** – посилання на користувача
- **chat_id** – посилання на чат
- **checked** – чи пройшов користувач капчу
- **date** – дата та час перевірки

auth

Використоввується для авторизації користувачів у веб-додатку

- **user_id** – посилання на користувача
- **code** – код доступу

- **date** – дата та час генерації коду

checkscript

Зберігає неперевірений чи не виконаний код користувачів

- **user_id** – посилання на користувача
- **chat_id** – посилання на чат де було створено код
- **code** – PHP код
- **confirmed** – чи дозволено для виконання

Система створення нових команд користувачами.

Кожен користувач може запропонувати додати свою команду до бота, інформація про це зберігається в таблиці **checking_commands**:

- **info** – повна інформація у json про нову команду, включаючи код
- **user_id** – посилання на користувача
- **chat_id** – посилання на чат звідки була запропонавана команда

ChatGPT

У боті можна експерементувати з ChatGPT моделью GPT-4o, задаючи йому різні налаштування (Усі, котрі підтримує OpenAI API). Усю цю інформацію зберігають таблиці **chatgpt_conversations** та **chatgpt_messages**.

chatgpt_conversations представляє чати з ChatGPT:

- **user_id** – посилання на користувача
- **prompt** – промт для моделі
- **max_tokens** – максимальна кількість токенів, котру ChatGPT може витратити на генерацію повідомлення
- **temperature** – температура моделі (від 0 до 2, чим вище, тим модель більш креативна)
- **frequency_penalty та presence_penalty** – різні види штрафів за генерацію схожого детермінованого тексту.

chatgpt_messages зберігає усі повідомлення користувачів з ChatGPT:

- **conversation_id** – посилання на чат з chatgpt

- **message_id** – ідентифікатор повідомлення
- **message_text** – текст повідомлення
- **image** – назва файлу зображення на сервері (коли використовуються функції vision моделі)
- **role** – роль власника повідомлення у чаті (система чи користувач, чи chatgpt)
- **father_id** – рекурсійне посилання для можливості отримання історії чату

Вікторина

Раз на добу користувачі можуть показати свої знання у вікторинах зі своєї спеціальності. Питання для тестів генерує ChatGPT, а за правильні відповіді користувачі отримують віртуальну валюту.

За цю систему відповідають таблиці групи **quiz**

quiz_questions зберігає питання для вікторин (аби зменшити витрати за використання API вже пройдені питання одного користувача можуть потрапити іншому)

- **question** – текст питання
- **theme** – тема питання

quiz_answers зберігає варіанти відповідей на питання

- quiz_id – посилання на питання
- answer – текст варіанту відповіді
- correct – чи правильна ця відповідь

quiz_results зберігає результати виконання тестів користувачами

- **user_id** – посилання на користувача
- **question_id** – посилання на питання
- **success** – чи дана правильна відповідь

Біржа

Аби користувачі могли обмінювати різні типи віртуальних валют і таким чином формувати економіку – була створена біржа. За неї відповідає

таблиця **offers**.

offers зберігає:

- **user_id** – посилання на користувача, що створив оголошення
- **price** – ціна за 1 діамант
- **sale** – оголошення типу «продаж» чи «купівля»
- **turnover** – при досягненні якої кількості діамантів на балансі необхідно деактивувати оголошення
- **date** – дата та час створення оголошення

Перекази

У користувачів є можливість переказувати віртуальну валюту між собою. Щоб формувати підтвердження та історію операцій, була створена таблиця **transfers.**

- **from_user –** посилання на користувача, від кого переказ
- **to_user –** посилання на користувача, кому переказ
- **sum –** сума переказу
- **confirmed –** чи підтверджено переказ
- **date –** дата та час переказу

Статистика

Уся статистика по іграх зберігається у 3х таблицях групи **stats**, які мають схожу структуру:

- **user_id** – посилання на користувача
- **profit** – зміна балансу користувача в результаті гри
- **moves** – кількість ходів по полю
- **date** – дата та час гри

Форми

У веб-додатку є система форм, котру студентський деканат може використовувати для опитування студентів та подальшу взаємодію з ними. За це відповідають таблиці групи **forms**.

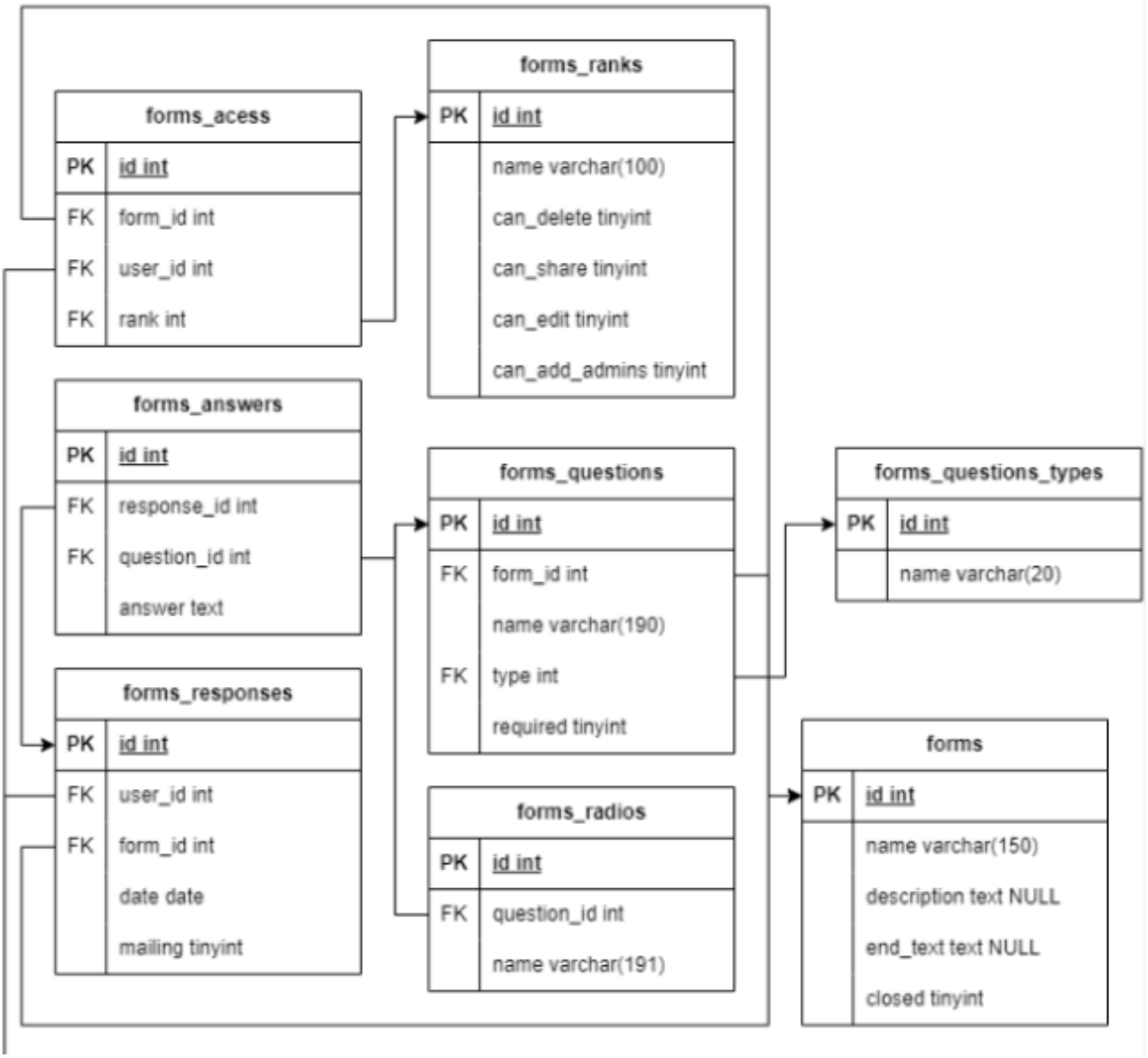


Рисунок 3.1 - Структура та зв’язки між таблицями бази даних для форм.

forms – основна таблиця, яка зберігає дані форми, а саме:

- name – назва форми
- description – опис форми
- end_text – текст після заповнення форми
- closed – чи закрита форма для нових відгуків

forms_questions відповідає за зберігання питань форми

- form_id – посилання на форму
- name – назва форми
- type – посилання на тип питання (таблицю forms_questions_types, яка зберігає назву типу, на даний момент це текст або radio (вибір варіанту

з запропонованих))

- required – чи обов’язкове це питання

forms_radios описує варіанти відповідей на питання, вона зберігає:

- question_id – посилання на форму
- name – текст варіанту відповіді

forms_respones описує відгуки від користувачів на форму

- user_id – посилання на користувача
- form_id – посилання на форму
- date – дата та час заповнення форми
- mailing – чи підписаний користувач на розсилку по цій формі

forms_answers зберігає відповіді на питання з форми

- response_id – посилання на відгук
- question_id – посилання на запитання
- answer – текст відповіді

forms_access зберігає рівень доступу редакторів до форми

- form_id – посилання на форму
- user_id – посилання на користувача
- rank – посилання на ранг доступу форм

forms_ranks зберігає ранги та рівні їх доступу до форм

- name – назва рангу
- can_delete – чи може цей ранг видаляти форму
- can_share – чи може надавати доступ до форми іншим користувачам
- can_edit – чи може редагувати форму
- can_add_admins – чи може надавати доступ адміністратора іншим користувачам.

3.3 Бібліотека Telegram бота

Розглянемо загальну структуру.

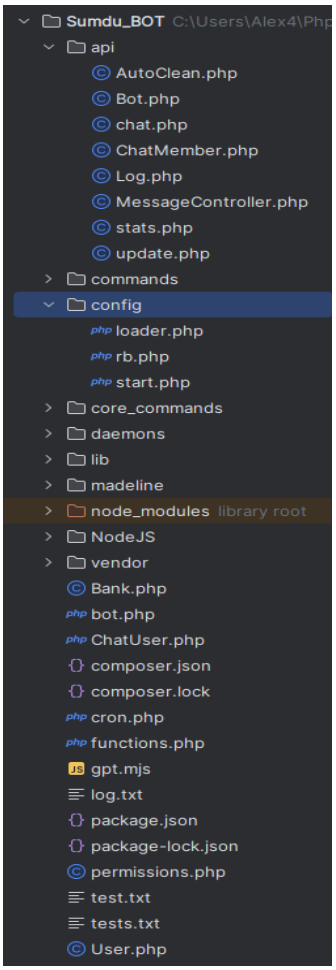


Рисунок 3.2 - Структура файлів бота

Вхідний файл - це bot.php

Підключення Налаштувань та Завантажувача

- **require_once 'config/start.php';**
- **require_once 'config/loader.php';**

Ці рядки включають основні конфігураційні файлі та завантажувач. Файл **start.php** містить глобальні налаштування, ініціалізацію з'єднання з базою даних MySQL і подібні "стартові" налаштування. **loader.php** використовується для підключення необхідних класів та бібліотек, які використовуватимуться у всьому коді бота.

Імпорт Просторів Імен

- **use api\{Bot as Bot, chat as chat, ChatMember as ChatMember, Log as Log, update as update};**

Обробка Вхідного Запиту

- **$request_json = file_get_contents('php://input');**
- **$request = json_decode($request_json, true);**

Ці рядки зчитують вхідний потік даних (від Telegram API), які передаються у форматі JSON, і конвертують їх у масив для подальшої обробки. Це фундаментальна частина бота, оскільки вона обробляє дані, які бот отримує від Telegram.

Ініціалізація Об'єктів та Перевірки

- Ініціалізація основних об'єктів (наприклад, **$bot = new Bot($bot_token);**, **$update = new update($request);**).

Перевірки (наприклад, **if (!update::$chat['id']) die('!chat');**), що забезпечують основні функції безпеки та валідації.

Обробка Команд

Останні рядки коду (**$dir = __DIR__.'/core_commands/';** і т.д.) відповідають за підключення та виконання файлів із папки **core_commands**. Це забезпечує виконання основних команд бота та включають системні команди, які мають працювати постійно.

Оптимізація Обробки Команд у Телеграм-Боті

В рамках розробки телеграм-бота було впроваджено передовий механізм обробки команд, який сприяє значному прискоренню реакції бота на запити користувачів. Основою цієї системи є розумне зчитування та асоціація команд, введених користувачами, з відповідними обробниками команд.

Суть методу полягає в тому, що при отриманні вхідного повідомлення від користувача, бот аналізує текст команди та швидко визначає, який саме файл має бути використаний для її обробки. Це досягається завдяки наступним крокам

Ініціалізація та Розбір Команд

Використовуючи рядок **$init = ['text' => $cmd[0], 'callback' => explode('_', $callback_data)[0], 'display' => explode('_', $display)[0]];**, бот визначає тип дії (текстова команда, callback, display) та відповідну команду або дію.

Вибір Обробника Команди

За допомогою циклу **foreach**, бот перевіряє, яка саме дія була ініційована, та визначає відповідний обробник команди з бази даних, використовуючи RedBeanPHP ORM (об'єктно-реляційне відображення).

Динамічне Підключення Обробника

Після ідентифікації потрібного файлу команди, використовується **require_once __DIR__.'/commands/'.$file['filename'];** для динамічного підключення цього файлу та виконання команди.

Цей метод дозволяє боту швидко реагувати на команди без необхідності прогону через велику кількість умовних операторів або перевірок, що забезпечує високу швидкість відповіді. Такий підхід не лише забезпечує ефективність обробки команд, але й робить код бота більш читабельним та легким для масштабування.

У порівнянні з традиційними методами, де кожна команда вимагає окремої перевірки та виклику відповідного обробника, впроваджена система дозволяє зменшити час відповіді бота на команди користувача, забезпечуючи високу продуктивність.

Аналіз Файлу Глобальних Налаштувань (start.php):

У розробці телеграм-бота одним з ключових елементів є налаштування глобального середовища, яке визначає основні параметри та змінні, необхідні для функціонування бота. Файл **start.php** служить цій меті, встановлюючи важливі конфігурації та ініціалізуючи з'єднання з базою даних. Розглянемо детальніше його складові.

Підключення ORM (Об'єктно-Реляційного Відображення) - RedBeanPHP:

- **require_once 'rb.php';**

Цей рядок підключає бібліотеку RedBeanPHP, що є простим та гнучким ORM рішенням для роботи з базами даних у PHP. Воно дозволяє використовувати об'єктно-орієнтований підхід при роботі з даними, спрощуючи процес взаємодії з базою даних.

Завантаження Токенів та Конфігураційних Даних

- **require_once __DIR__.'/../../tokens.php';**

Файл **tokens.php** містить важливі конфігураційні дані, такі як токен бота, ідентифікатор адміністратора, параметри підключення до бази даних тощо. Виокремлення цих даних у окремий файл додає додатковий рівень безпеки, оскільки ці параметри не зберігаються безпосередньо в основному коді.

Встановлення Глобальних Констант та Змінних

Встановлення констант та змінних, таких як **DOMAIN**, **bot_username**, **chat_for_checkcodes**, визначає основні параметри, які використовуватимуться у всій роботі бота. Це включає ім'я бота, домен для вебхуків та інші ключові елементи.

Ініціалізація З'єднання з Базою Даних

R::setup('mysql:host='.$db_host.';dbname='.$db_name, $db_login, $db_pass);

Цей рядок використовує RedBeanPHP для встановлення з'єднання з базою даних MySQL.

loader.php підвантажує до проекту необхідні бібліотеки та класи: update.php (зберігає інформацію про отримані від telegram дані), Bot.php, AutoClean.php, chat.php, Log.php, stats.php, ChatMember.php, Bank.php, User.php.

Аналіз Класу Bot з Файлу bot.php

Файл **bot.php** включає в себе визначення класу **Bot**, який є ключовою частиною телеграм-бота. Цей клас відповідає за взаємодію з Telegram Bot API і містить набір методів для різних функцій бота. Ось основні компоненти та їхні функції:

Ініціалізація Класу

Конструктор класу (**public function __construct($bot_token)**) приймає токен бота і зберігає його в приватній статичній змінній **self::$token**. Це дозволяє використовувати токен в усіх методах класу.

Запити до Telegram API

Метод **request** відправляє HTTP-запит до Telegram Bot API. Він формує URL з використанням токену бота, методу API, і параметрів, які передаються у запиті. Метод також обробляє відповідь від API, перевіряючи на наявність помилок.

Отримання Фотографій Профілю Користувача

Метод **getUserProfilePhotos** використовується для отримання фотографій профілю користувача. Він формує параметри запиту та викликає метод **request** для взаємодії з API.

Отримання Інформації про Файл

Метод **getFile** використовується для отримання даних про файл у Telegram. Цей метод може бути корисним для завантаження файлів, які надіслані користувачами.

Зберігання Файлу

Метод **storeFile** використовується для зберігання файлів, отриманих через Telegram, на сервері. Він формує URL для доступу до файлу, визначає розширення файлу та здійснює його копіювання у вказане місце зберігання.

Цей клас є фундаментальним для функціонування телеграм-бота, оскільки він забезпечує основні взаємодії з Telegram Bot API. Використання об'єктно-орієнтованого підходу у розробці класу **Bot** сприяє чистоті коду,

його модульності та легкості підтримки. Кожен метод класу має чітко визначену відповідальність, що дозволяє легко розширювати та адаптувати функціонал бота під різні потреби.

AutoClean.php зберігає інформацію про повідомлення, котрі потрібно буде видалити через деякий час, щоб не забруднювати групові чати повідомленнями бота та користувачів, котрі викликають ці команди

Аналіз Класу chat з Файлу chat.php

Файл **chat.php** визначає клас **chat**, який відіграє ключову роль у взаємодії з чатами в Telegram. Цей клас використовується для різноманітних операцій, пов'язаних із чатами, таких як відправка повідомлень, управління членами чату, редагування та видалення повідомлень. Ось детальний аналіз його основних компонентів та функцій:

Ініціалізація Класу

Конструктор (**__construct**) ініціалізує клас з ідентифікатором чату. Він також використовує RedBeanPHP для пошуку відповідного чату в базі даних.

Зберігання Інформації про Чат

Метод **storeChat** зберігає інформацію про новий чат у базі даних. Цей метод також використовується для надсилання повідомлень користувачам при їхньому першому зверненні до бота.

Відправка Повідомлень

Метод **sendMessage** відправляє текстові повідомлення у чат. Він включає параметри для кастомізації повідомлення, такі як розмітка, відповідь на конкретне повідомлення, вимкнення сповіщень тощо.

Редагування Тексту Повідомлення

Метод **editMessageText** дозволяє редагувати вже відправлені текстові повідомлення.

Управління Членами Чату

Методи **restrictChatMember**, **banChatMember**, **unbanChatMember**

надають можливості для управління правами користувачів у чаті, включаючи обмеження, блокування та розблокування.

Створення Запрошення в Чат

Метод **createChatInviteLink** використовується для створення посилань-запрошень у чат.

Відправка Фотографій та Видалення Повідомлень

Методи **sendPhoto** та **deleteMessage** надають функціонал для відправки фотографій та видалення повідомлень відповідно.

Відповідь на Callback-Запити

Метод **answerCallbackQuery** використовується для відповіді на callback-запити в інлайн-клавіатурі.

Клас **chat** відіграє важливу роль у взаємодії бота з чатами Telegram, надаючи гнучкість та множинність функцій для різних сценаріїв комунікації. Через використання цього класу, розробники можуть легко імплементувати складні функціональні вимоги, пов'язані з управлінням чатом і взаємодією з користувачами.

Класи Log та stats логують та зберігають статистику користування ботом

Клас **ChatMember** в **ChatMember.php** відіграє важливу роль у взаємодії з учасниками чату Telegram. Цей клас містить методи для управління інформацією про учасників, включаючи їх статуси, права доступу та інші відомості.

Конструктор та Ініціалізація

Конструктор класу (**__construct**) ініціалізується з ідентифікаторами користувача та чату. Він спробує завантажити існуючу інформацію про члена чату; якщо такої інформації немає, він автоматично створює новий запис.

Завантаження Даних Члена Чату

Метод **load** використовується для пошуку існуючих даних про члена чату в базі даних з використанням RedBeanPHP.

Створення Нового Члена Чату

Метод **newChatUser** створює новий запис для користувача у чаті, якщо він раніше не був зареєстрований. Метод також перевіряє статус користувача в чаті (адміністратор, створювач тощо).

Оновлення Даних Члена Чату

Метод **update** дозволяє оновлювати різні параметри члена чату, наприклад, статус у чорному списку, використовуючи ORM RedBeanPHP.

Перевірка Статусу Користувача в Чаті

Метод **getChatStatus** визначає, чи є користувач адміністратором або створювачем чату.

Додавання до Чорного Списку

Метод **addToBlackList** дозволяє додати користувача до чорного списку на певний час або нескінченно (за значенням **0**).

Клас **ChatMember** є важливим для адміністративного управління чатом та обробки даних про членів чату, надаючи функціональність для визначення ролей користувачів, їхнього управління та відстеження їхньої активності. Використання цього класу дозволяє боту автоматизувати багато процесів в чаті, таких як модерація та керування доступом.

Клас Bank.php забезпечує функціонування банку та внутрішньо ігрової валюти.

Аналіз Класу User з Файлу User.php

Файл **User.php** визначає клас **User**, який є важливим для управління даними користувачів телеграм-бота. Клас містить методи для завантаження, створення, оновлення та управління даними користувачів. Ось детальний опис ключових функцій цього класу:

Завантаження Даних Користувача

Методи **loadByTGID**, **loadByID**, **loadByNick**, **loadByUsername** використовуються для завантаження даних користувача з бази даних за різними ідентифікаторами, такими як Telegram ID, нікнейм чи ім'я користувача.

Створення Нового Користувача

Метод **newUser** створює новий запис користувача в базі даних на основі інформації, отриманої з Telegram (наприклад, з об'єкта **from**).

Оновлення Даних Користувача

Метод **update** дозволяє оновлювати різні параметри користувача, такі як баланс, статус у чорному списку тощо.

Локальне Сховище

Методи **LocalStorageSet**, **LocalStorageGet**, **LocalStorageClear** використовуються для управління тимчасовими даними користувача, які зберігаються у вигляді JSON-об'єктів.

Управління Балансом Користувача

Метод **addBal** змінює баланс користувача, інтегруючи з класом **Bank** для відстеження загального балансу системи.

Додавання до Чорного Списку

Метод **addToBlackList** дозволяє додати користувача до чорного списку всього бота на певний час або нескінченно (за значенням **0**).

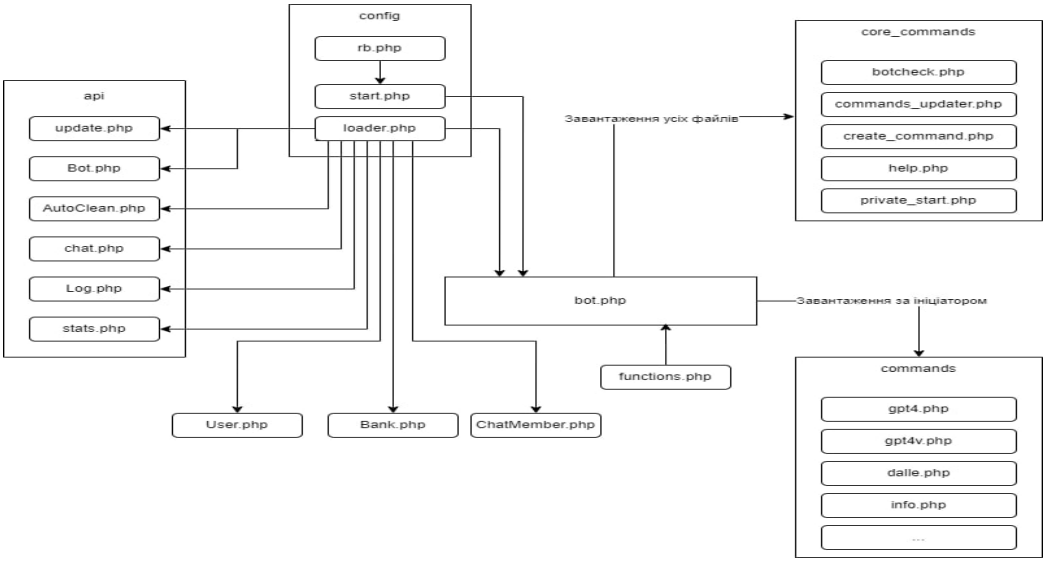


Рисунок 3.3 - Структура завантаження ядра бота

Створення команди та перевірка її роботи

Для прикладу, створемо команду !gpt4v, котра буде передавати наш запит в gpt4-vision-preview:

```
<?php
//
// Command: ChatGPT #
// Text: !gpt4v /gpt4v #
// Callback: gpt4v #
// Display: gpt4v #
// Info: ChatGPT 4 vision #
// Syntax: !gpt4v [Повідомлення з фото] #
// Args: 1 #
// Rank: USER #
//
```

… (Увесь код у додатку або у репозиторії GitHub)

Коментарі для Обробника Команд

Початок файлу містить коментарі, які визначають ключові параметри команди, такі як текст команди (**!gpt4v**), callback (**gpt4v**), display (**gpt4v**), інформацію про команду, синтаксис, кількість аргументів та рівень доступу користувача (**USER**). Ці коментарі важливі для обробника команд, який використовує цю інформацію для визначення, коли саме підключати цей файл.

Логіка Обробки Команди

Скрипт ініціюється за наявності певних тригерів (наприклад, перше слово у повідомленні - **!gpt4v**), ініціалізує об'єкт AutoClean, перевіряє баланс користувача і блокує одночасне виконання кількох таких самих скриптів.

За допомогою OpenAI API та відправки зображень, команда генерує відповіді, використовуючи модель ChatGPT 4 з підтримкою зображень.

Робота з Зовнішніми Файлами та API

Команда інтегрується з зовнішніми бібліотеками та API, зокрема з OpenAI, для обробки запитів, включаючи відправку зображень та обробку тексту.

Фінансові Операції

Команда враховує вартість використання API, віднімаючи відповідну суму з балансу користувача.

Обробка Помилок

У разі виникнення помилок під час виконання команди, скрипт здійснює відповідне логування та інформування користувача.

Перевірка роботи

Рисунок 3.4 - Тестування створеної команди

3.4 Функціональність

Проект є дуже великим та комплексним, тому в цьому розділі ми розглянемо його функціональність, поділивши її на окремі функціональні блоки. Кожен блок буде описаний з точки зору його основних можливостей та принципів роботи, не вдаючись у деталі реалізації кожної строки коду. Це дозволить зосередитися на загальній архітектурі та логіці системи, що полегшує розуміння її структури та взаємодії компонентів.

3.4.1 Взаємодія бота та веб-додатку

Авторизація у веб-додатку

Так як веб-додаток пов’язаний з ботом – вхід в аккаунт у веб-додатку виконується через Telegram бота. Існує 2 методи авторизації. За допомогою системи Telegram Widget [1] (надавши веб-додатку доступ до інформації про свій telegram аккаунт за допомогою oAuth2), або сгенерувавши одноразове посилання у самому боті за допомогою кнопки «Авторизація на порталі» у приватних повідомленнях з ботом. Якщо користувач не використає посилання – воно автоматично деактивується через 5 хвилин.

Тестування

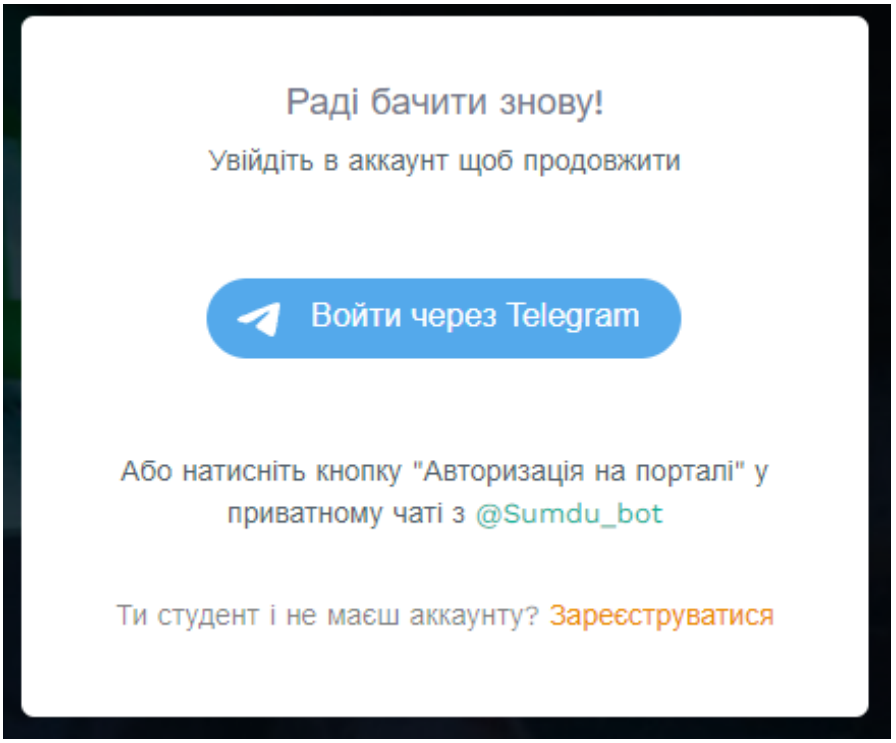


Рисунок 3.5 - Сторінка входу в обліковий запис.

Вхід за допомогою Telegram Widget

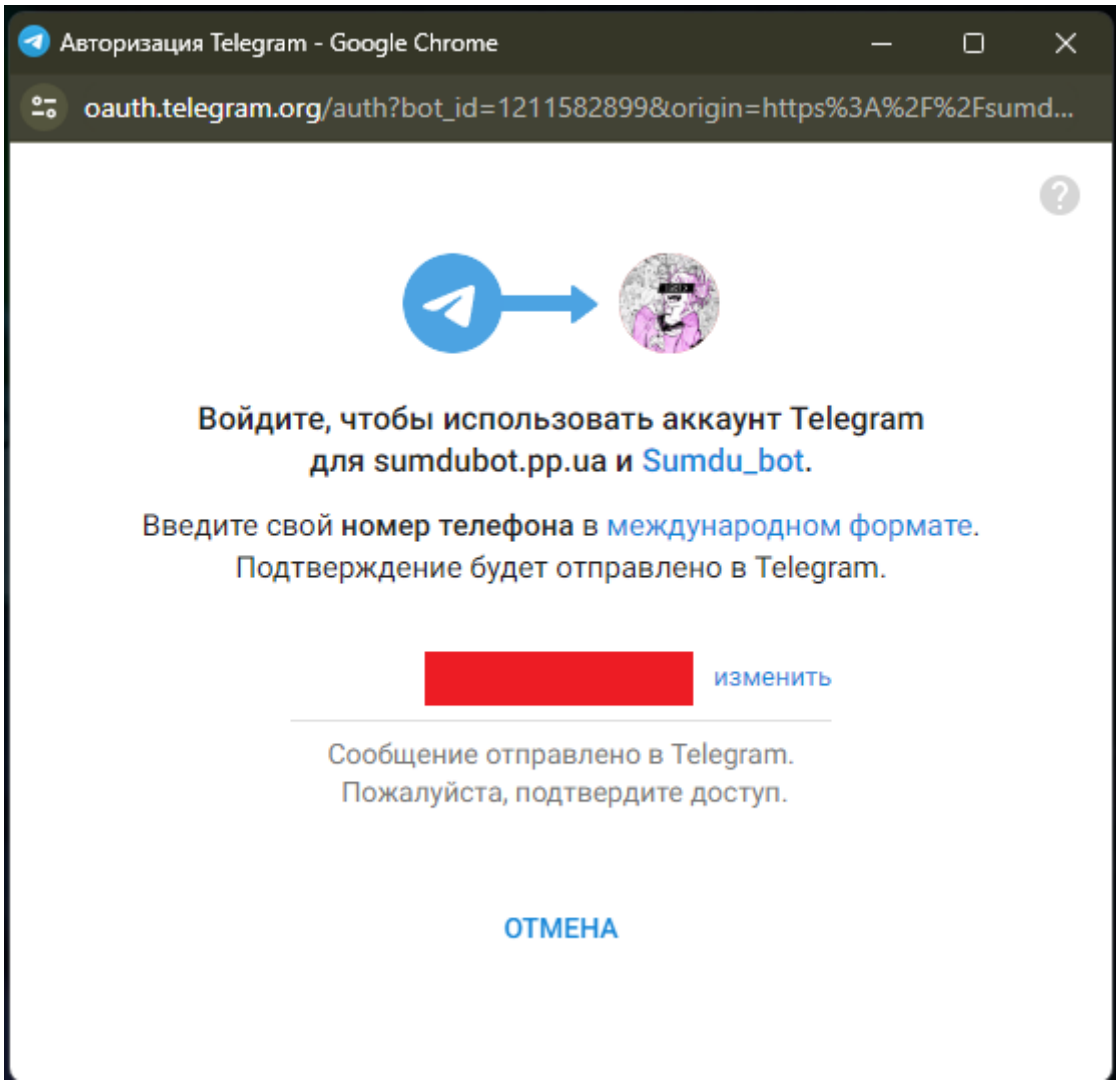


Рисунок 3.6 - Вхід за допомогою Telegram Widget

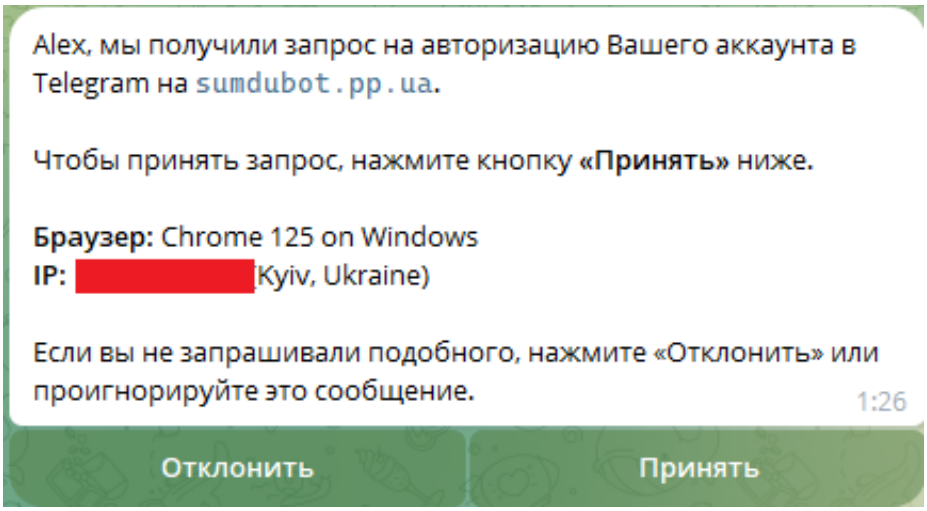


Рисунок 3.7 - Підтвердження входу за допомогою Telegram Widget

Авторизація за допомогою посилання через бота

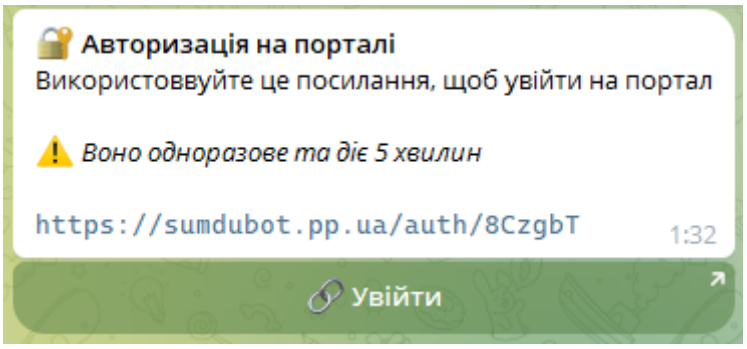


Рисунок 3.8 - Вхід за допомогою бота

Після будь якого виду авторизації, користувачу в приватні повідомлення надійде така інформація:

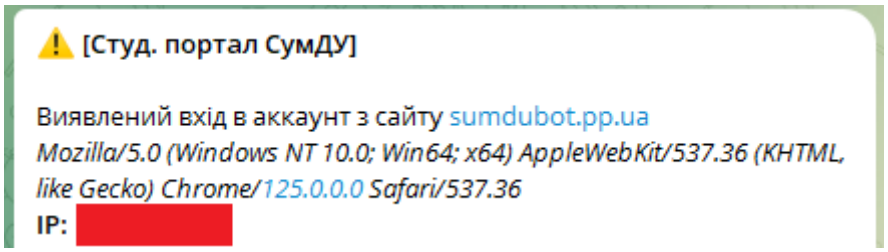


Рисунок 3.9 - Повідомлення про вхід в обліковий запис у веб-додатку

Система botcheck

Для захисту чатів від спам-атак ботами була розроблена ця система. Адміністратори чатів можуть деактивувати її за допомогою core команди /settings. Так як мітка проходження перевірки зберігається у профілі користувача – йому достатньо лише 1 раз пройти перевірку, після чого бот не буде його турбувати у усіх чатах. Логіка системи полягає в наступному: Коли користувач перший раз заходить у будь який чат де активована ця функція – йому буде обмежено доступ писати повідомлення та надано посилання та 5 хвилин на проходження перевірки. По посиланню користувач потрапляє у веб-додаток, де йому необхідно натиснути кнопку «Я людина», захищену системою Google reCAPTCHA 3ї версії, котра визначає бот це чи ні на основі поведінки користувача на усіх веб-додатках світу, де підключена reCAPTCHA. Якщо користувач проходить перевірку – з нього знімаються усі обмеження, інакше користувач блокується у чаті на

30 хвилин (Його може розблокувати адміністрація чату або бота).

Тестування

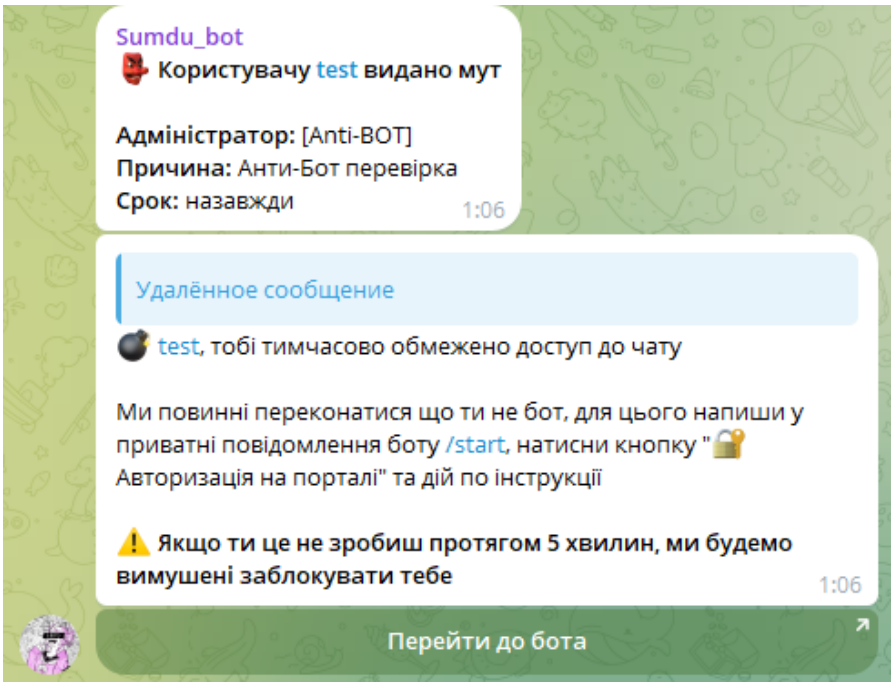


Рисунок 3.10 - Блокування від системи botcheck.

Після настискання на кнопку «Перейти до бота», користувач потрапляє в приватний чат з ботом, де йому одразу пропонується авторизуватися у веб-додатку:

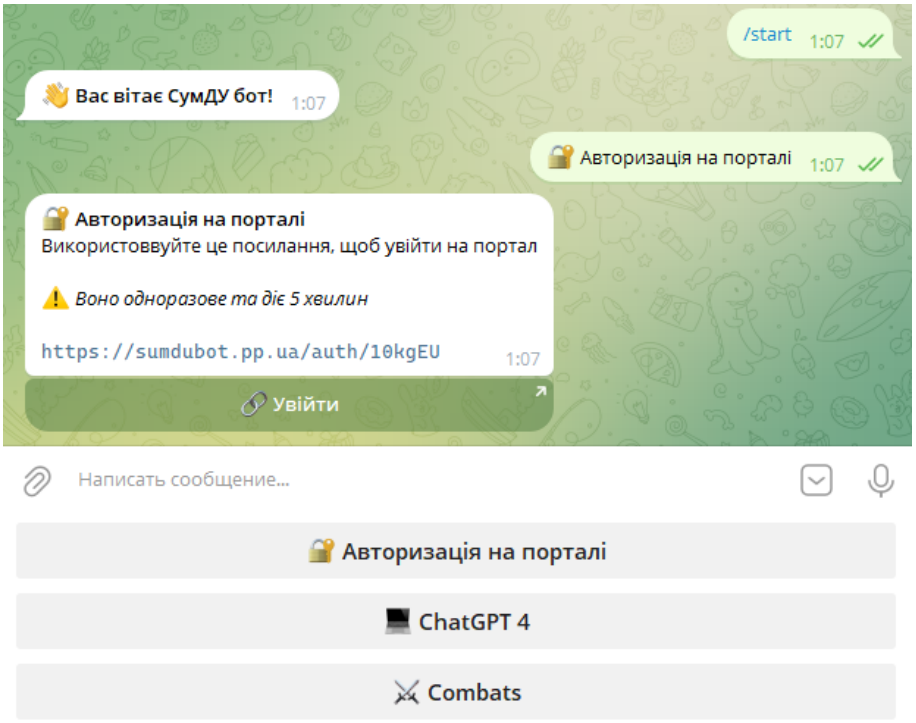


Рисунок 3.11 - Ініціалізація приватного чату з ботом після блокування системою BotCheck

При авторизації, користувач потрапляє на сторінку, де натискає

єдину кнопку «Я людина»

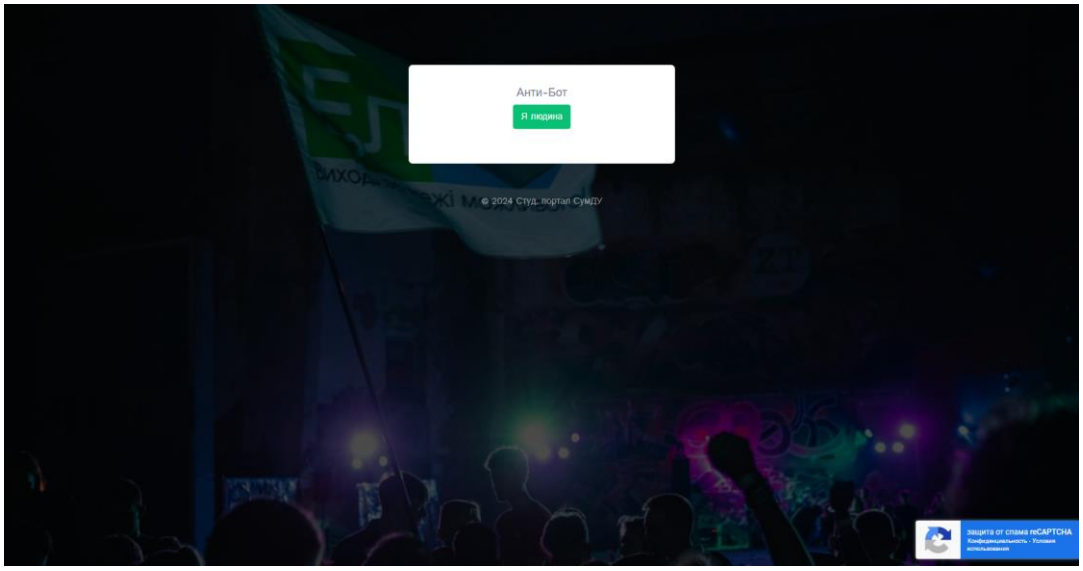


Рисунок 3.12 - Сторінка перевірки, яка захищена системою Google ReCaptcha

Після чого з користувача знімаються обмеження, а у веб-додатку пропонується додати більше інформації про себе

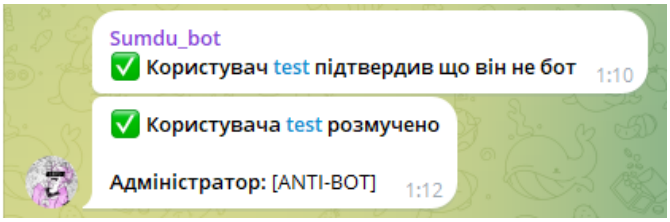


Рисунок 3.13 - Автоматичне розблокування після проходження BotCheck

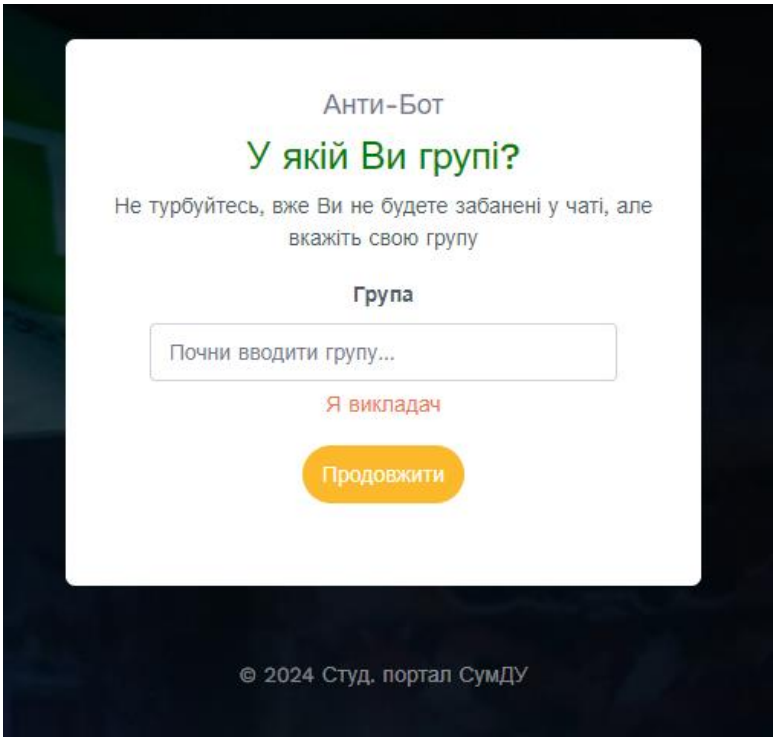


Рисунок 3.14 - Пропозиція додати більше інформації про себе

Список груп автоматично «підтягується» при заповненні:

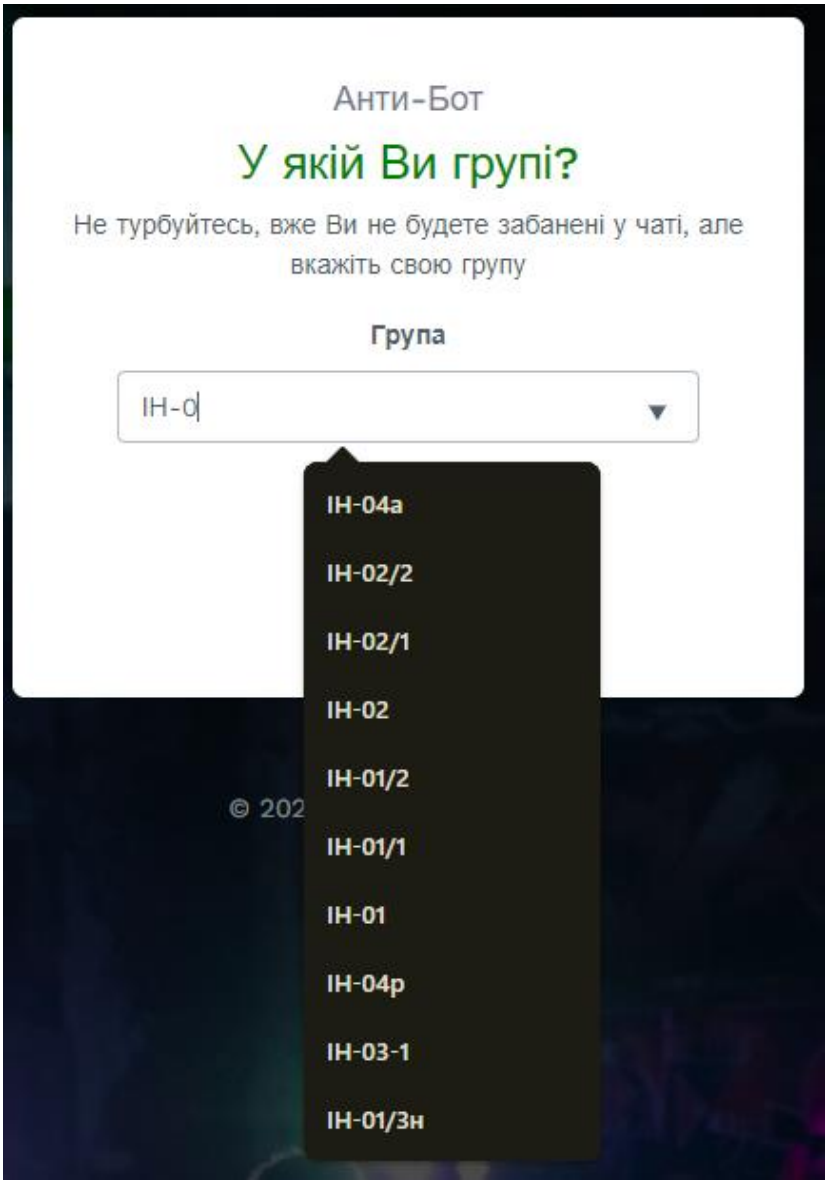


Рисунок 3.15 - Підвантажування списку груп.

Після цього процедура підтвердження вважається повністю завершеною.

3.4.2 Telegram Бот

Зосередимося на описі лише тих модулів, які безпосередньо стосуються теми роботи, залишаючи поза увагою розважальні та інші додаткові функції.

Функції обмеження порушників

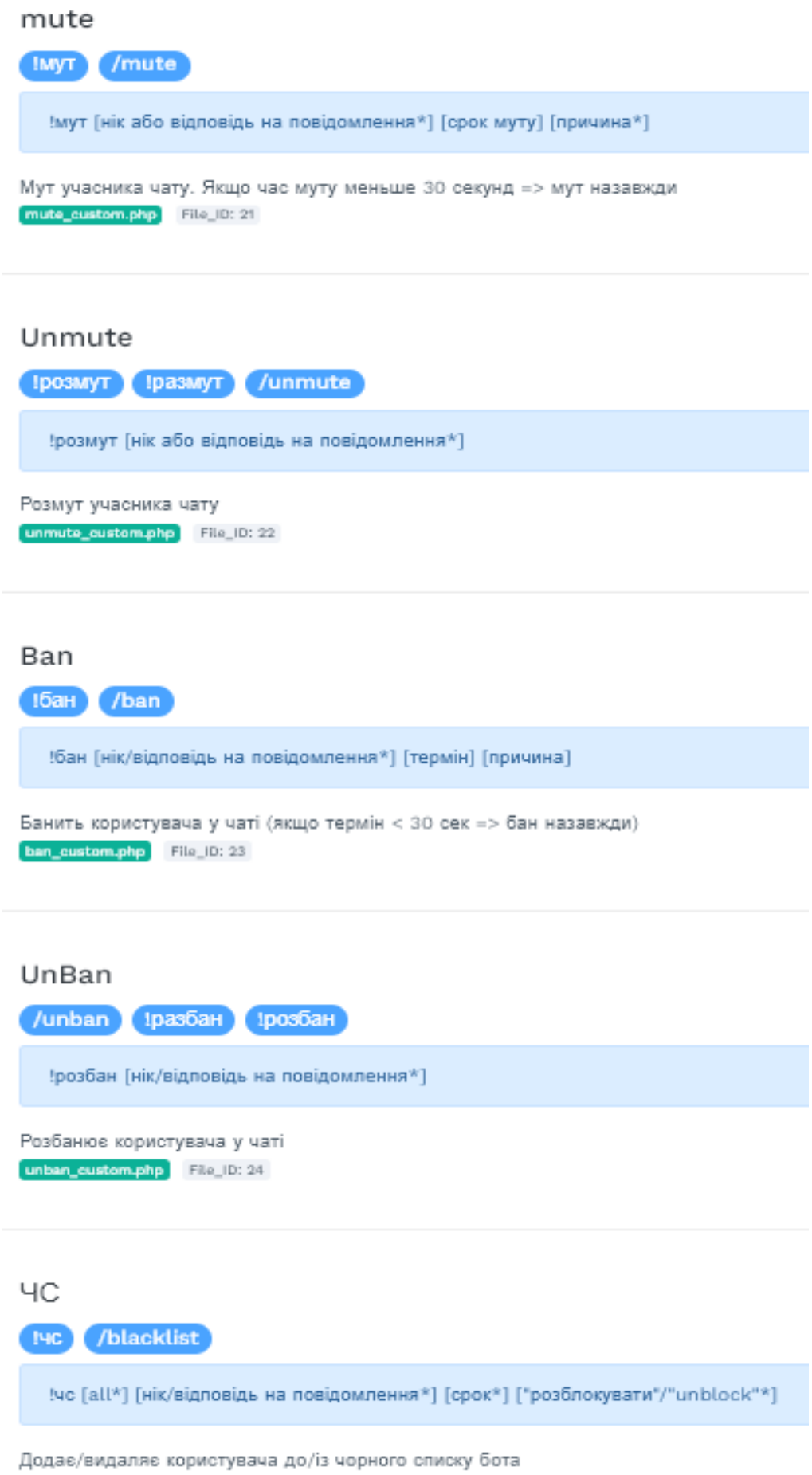


Рисунок 3.16 - Функції обмеження порушників.

Час вказується згідно формату ISO 8601 duration, тобто, наприклад, 3M3d3h3m3s розшифроввується як 3 місяця, 3 дні, 3 години, 3 хвилини, 3 секунди. У процесі воно конвертується у секунди, надсилається API запит до Telegram, та конвертується в звичайний вигляд для розуміння користувача

Тестування

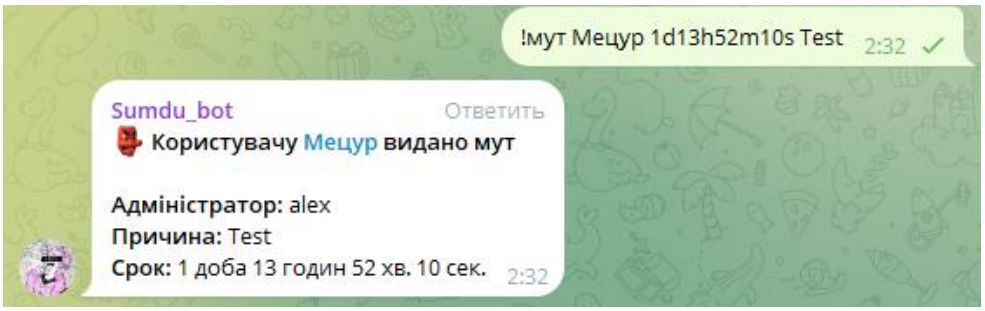


Рисунок 3.17 - Мут користувача.

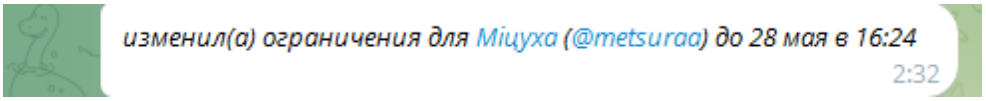


Рисунок 3.18 - Підтвердження факту муту від Telegram

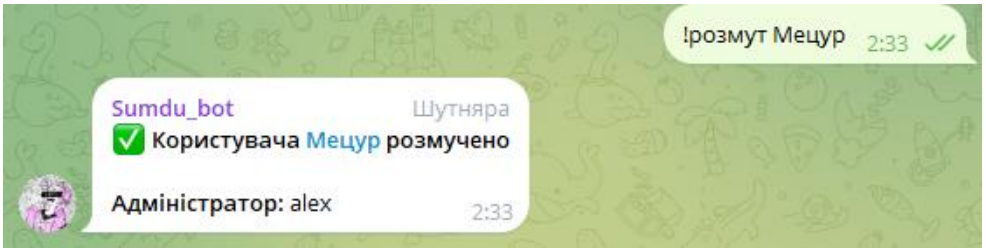


Рисунок 3.19 - Розмут користувача

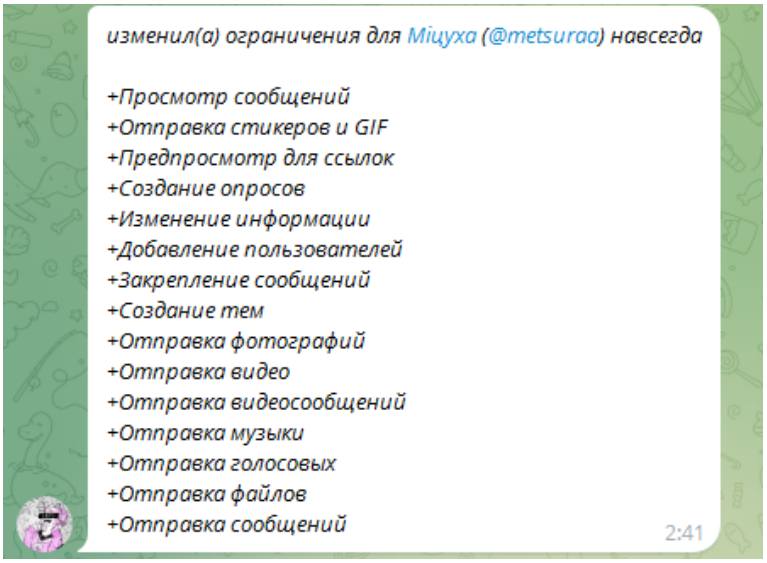


Рисунок 3.20 - Підтвердження розблокування від Telegram

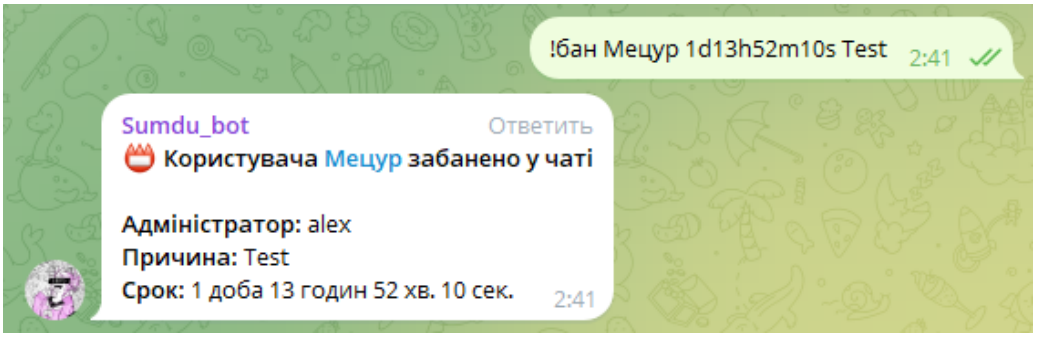


Рисунок 3.21 - Блокування та виключення користувача з чату

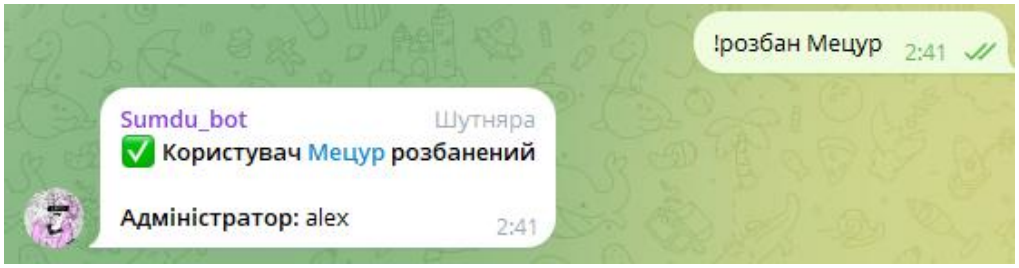


Рисунок 3.22 - Розблокування користувача

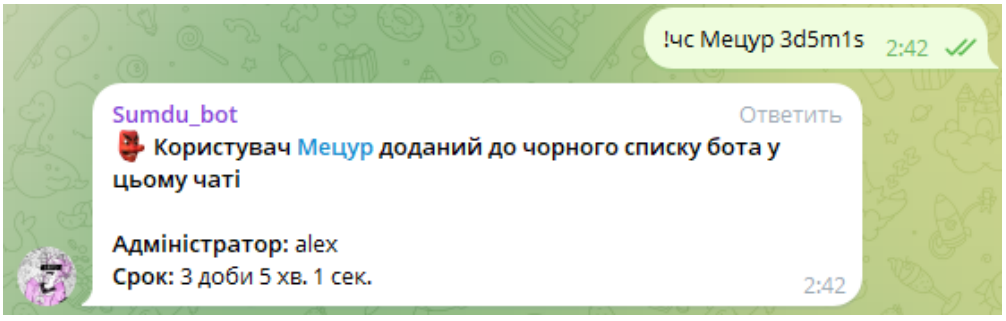


Рисунок 3.23 - Додання користувача до чорного списку бота у чаті (бот буде ігнорувати команди)

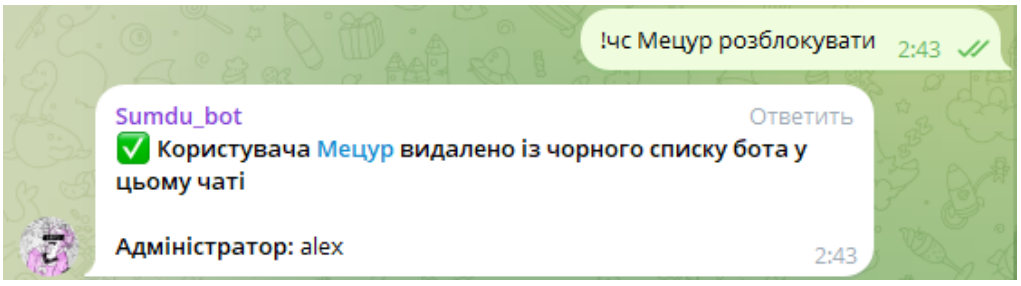


Рисунок 3.24 - Видалення користувача з чорного списку бота

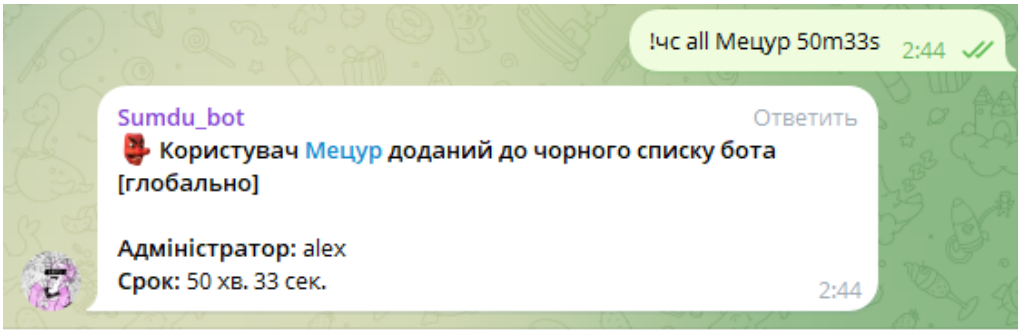


Рисунок 3.25 - Додання користувача до чорного списку бота в усіх чатах

Налаштування чатів

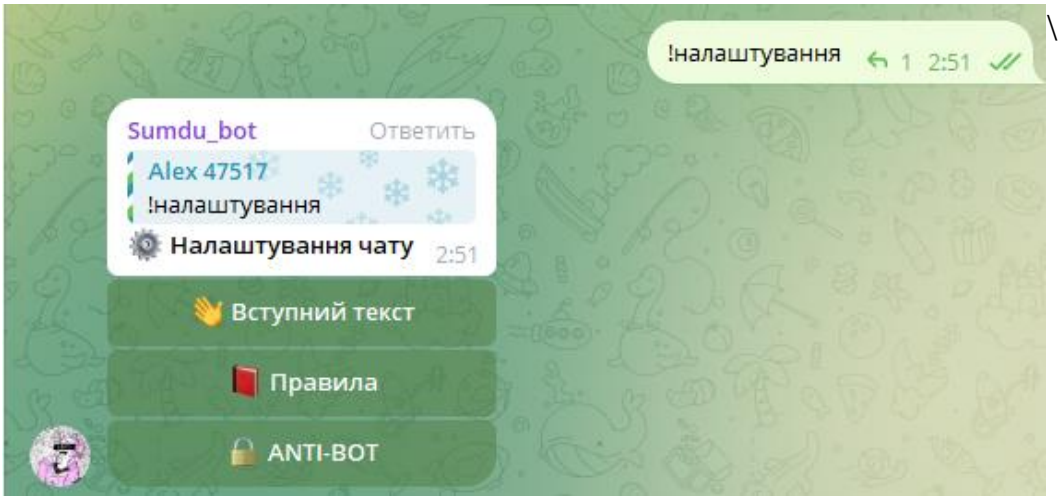


Рисунок 3.26 - Меню налаштування чатів

За допомогою кнопки «Вступний текст» можна налаштувати вступний текст та зображення до нього, який буде відображатися коли новий користувач приєднається до чату. За допомогою кнопки «Правила» встановлюються правила чату, які доступні усім користувачам за допомогою команди !правила або /rules. Кнопка «ANTI-BOT» активує або вимикає систему botcheck у цьому чаті.

Вікторини

Функція генерації питань для вікторин, розроблена для AI моделі GPT-4 Turbo, представляє собою ключовий інструмент у підтримці освітньої взаємодії. Вона забезпечує створення релевантних та змістовно насичених питань, що охоплюють різноманітні дисципліни. Це не тільки сприяє активізації когнітивної діяльності студентів, але й відкриває можливість для глибшого засвоєння знань через інтерактивну гру.

Із-за певних обмежень веб серверу та платформи Telegram – цей модуль має дві частини – «стартер» як команда, та «демон» - екземпляр якого виконується постійно на протязі проходження вікторини користувачем.

Вікторину можна проходити лише у приватних повідомленнях з ботом, також повідомлення стосовно вікторини захищені від пересилання користувачем.

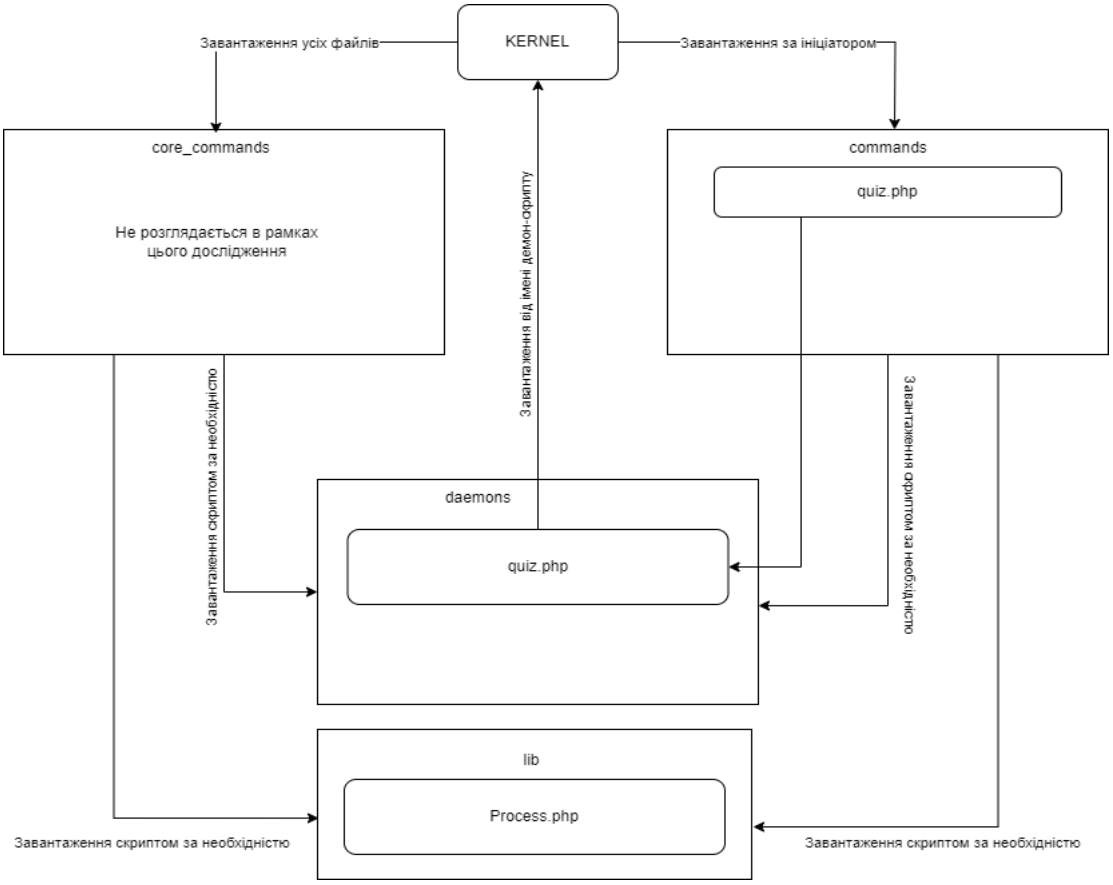


Рисунок 3.27 - Схема реалізації вікторин

Перевірка роботи

1. Користувач обирає тему вікторини.

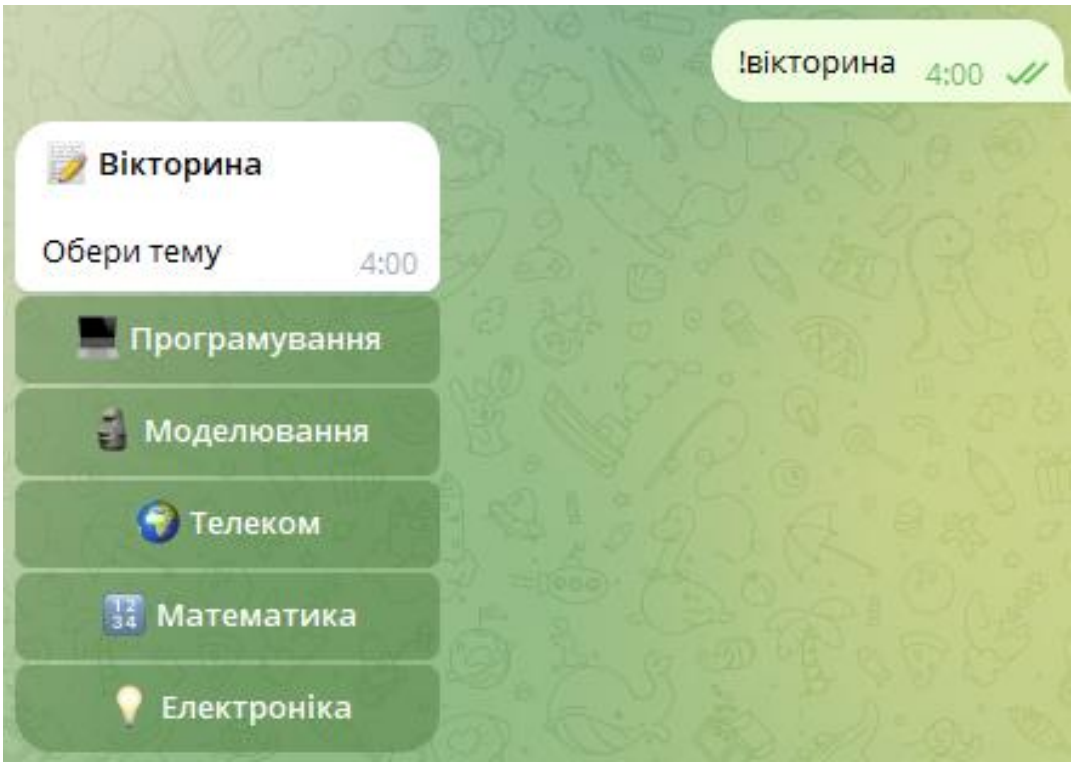


Рисунок 3.28 - Меню тем вікторин

2. Після вибору отримуємо питання.

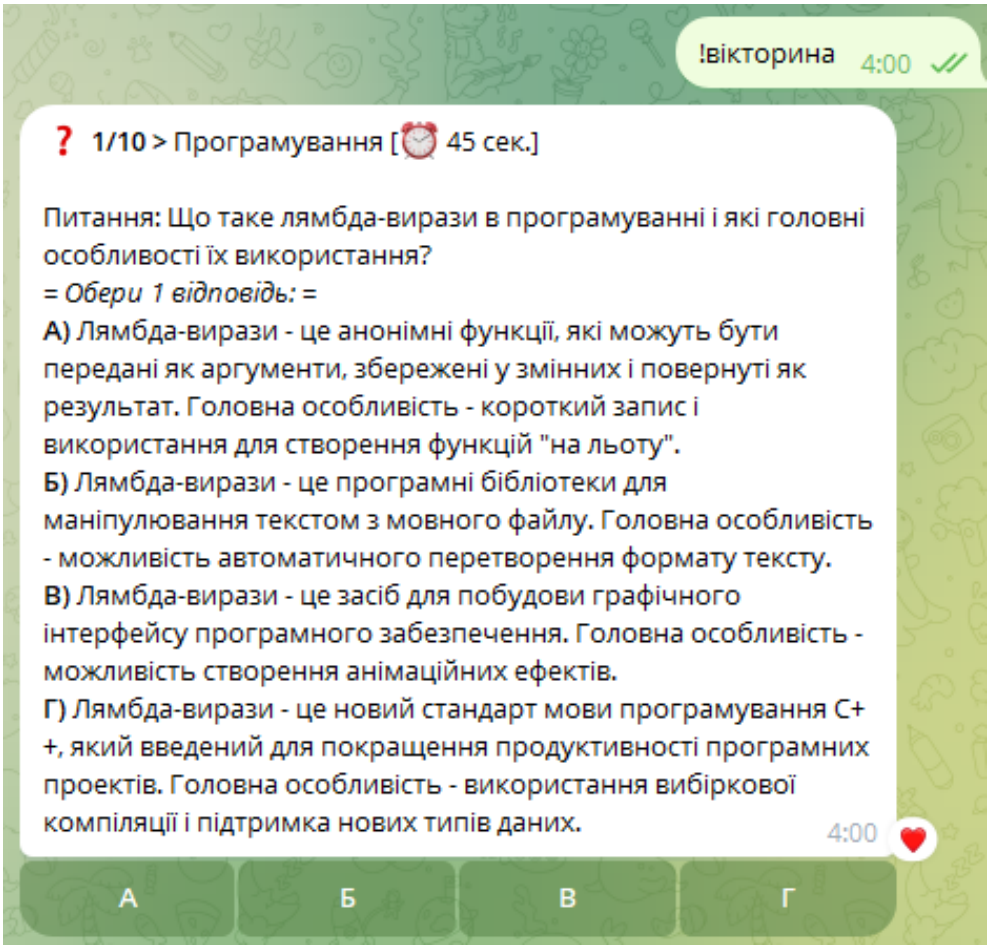


Рисунок 3.29 - Меню з питанням

3. Якщо користувач не відповідає за відведений час, повідомлення автоматично редагується та відображає правильну відповідь на питання.

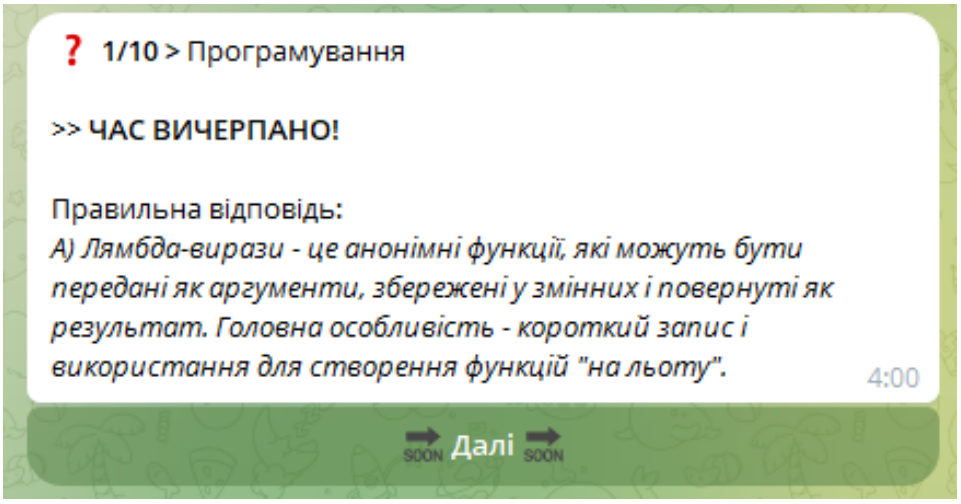


Рисунок 3.30 - Час вичерпано

4. У випадку правильної відповіді повідомлення формується наступним чином:

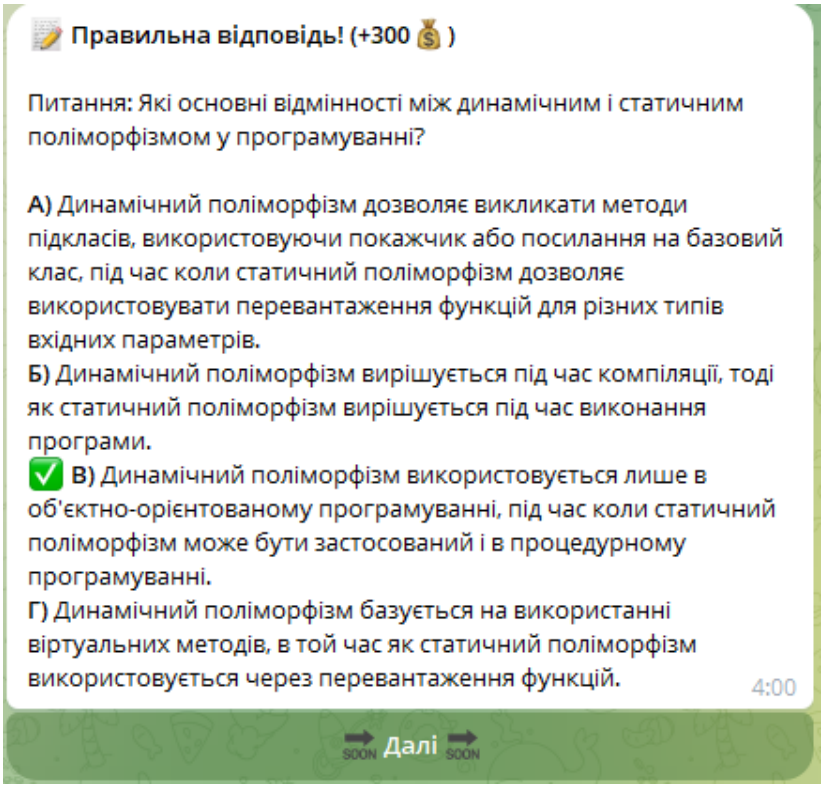


Рисунок 3.31 - Вказано правильну відповідь

5. У випадку неправильної відповіді, користувачу буде відображено що він обрав та яка відповідь була правильною.

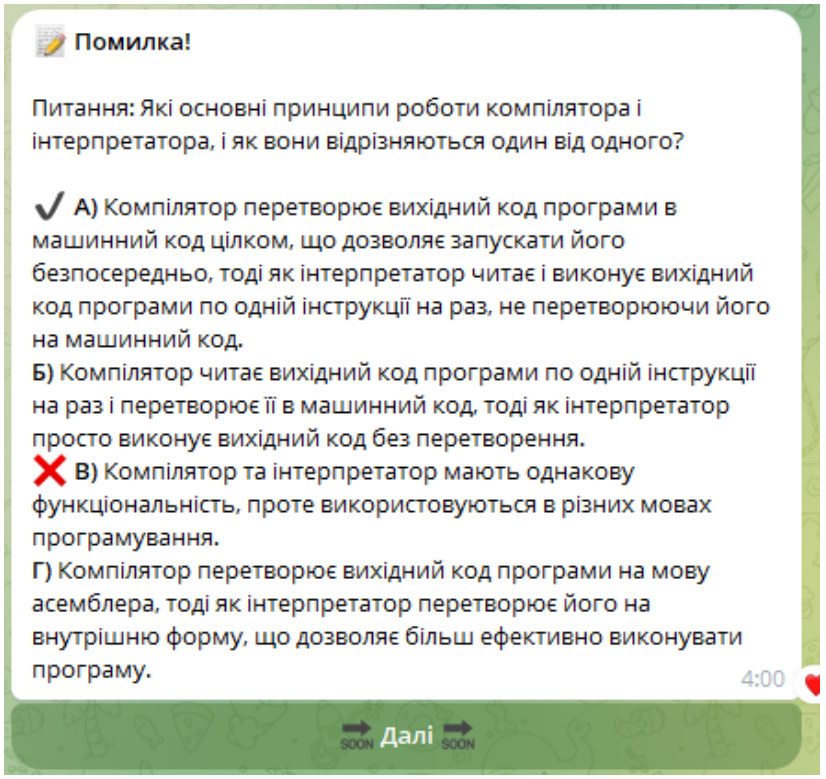


Рисунок 3.32 - Вказано помилкову відповідь

6. Під час генерації питання нейронною мережею, користувачу буде відображено наступне повідомлення:

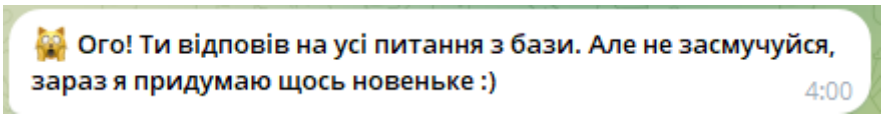


Рисунок 3.33 - Користувач вирішив усі питання з БД, генерація нового за допомогою функції ChatGPT

7. Після завершення вікторини користувач отримує подібне повідомлення:

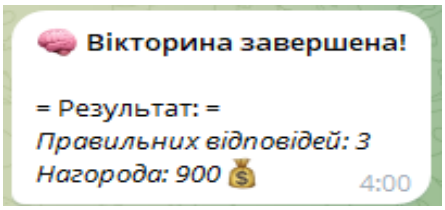


Рисунок 3.34 - Повідомлення про завершення вікторини

3.4.3 Веб-додаток

У веб-додатку кожен користувач може отримати бонус віртуальної валюти раз на добу, також там є функціонал форм, керування кураторами, довідка по боту, яка формується автоматично завдяки продуманій реалізації модульності команд, правила користування платформою та статистика по розважальним командам. Front-end частину веб-додатку розроблено за допомогою bootstrap, тому ми маємо адаптивність до мобільних пристроїв.

Розділ WIKI включає в собі довідку, правила та «статті», які користувачі можуть залишати у якості довідки.

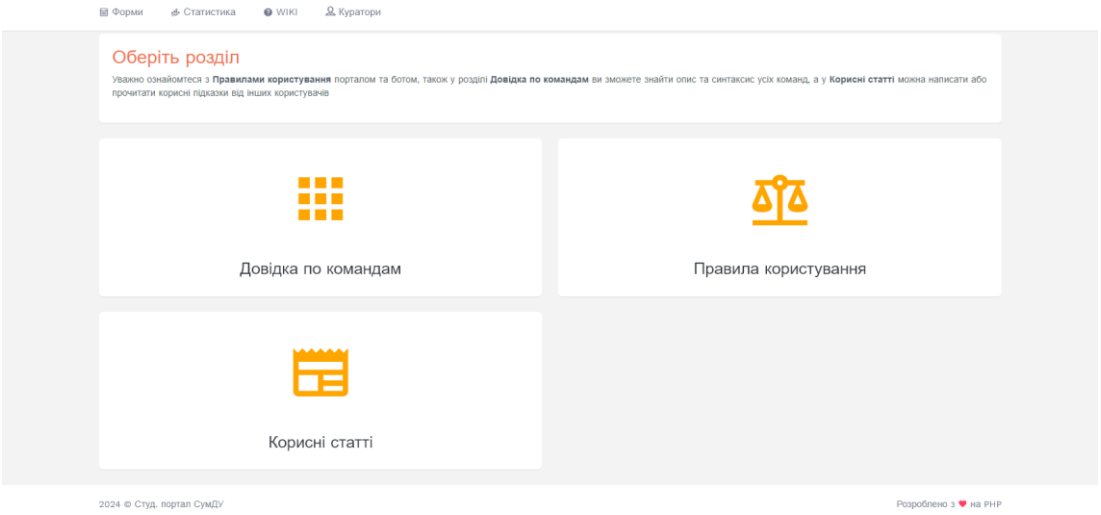
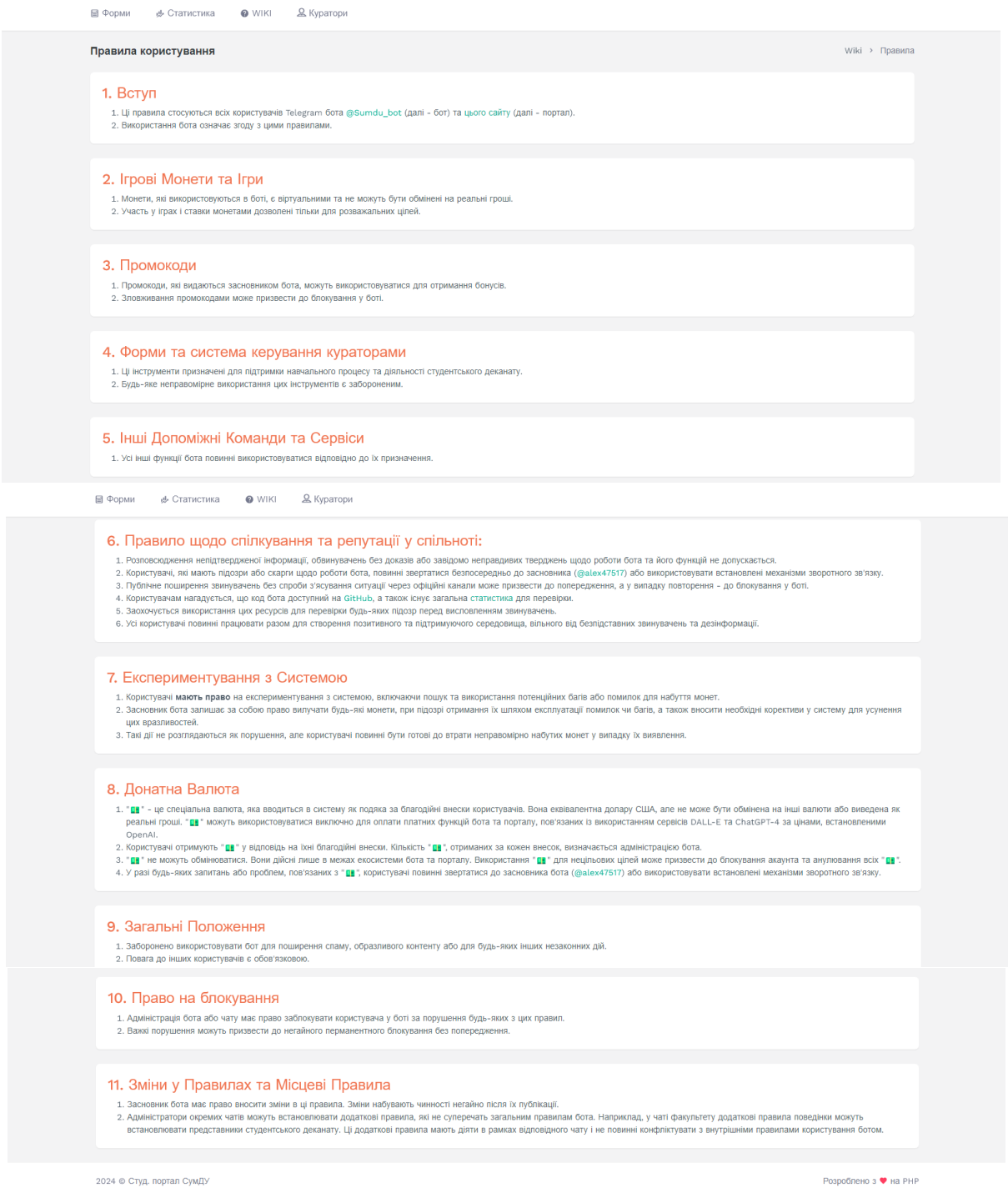


Рисунок 3.35 - Сторінка WIKI

Правила користування:

Це статична сторінка, де виписані правила користування платформою, які забов’язані виконувати усі користувачі.



Рисунки 3.36 – 3.38 - Правила користування платформою

Довідка

Ця сторінка формується повністю автоматично, з коментарів у початку файла з кодом, які необхідно обов’язково вказувати при створенні нової не системної команди для бота. На сторінці розписані усі ранги та які повноваження вони мають, а також спойлери з описом усіх доступних

команд для цього рангу та рангів вище.

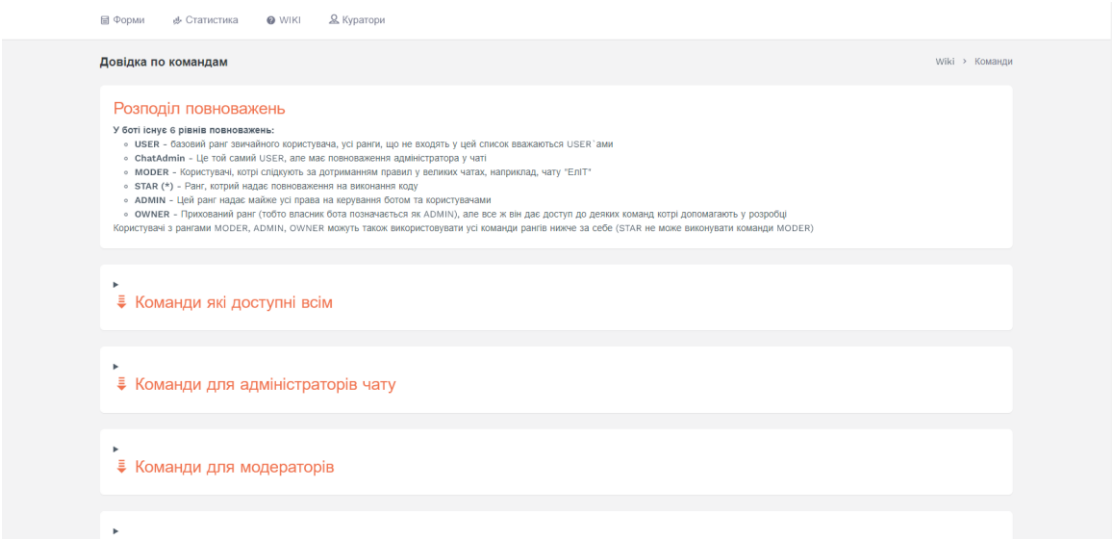


Рисунок 3.39 - Сторінка довідки

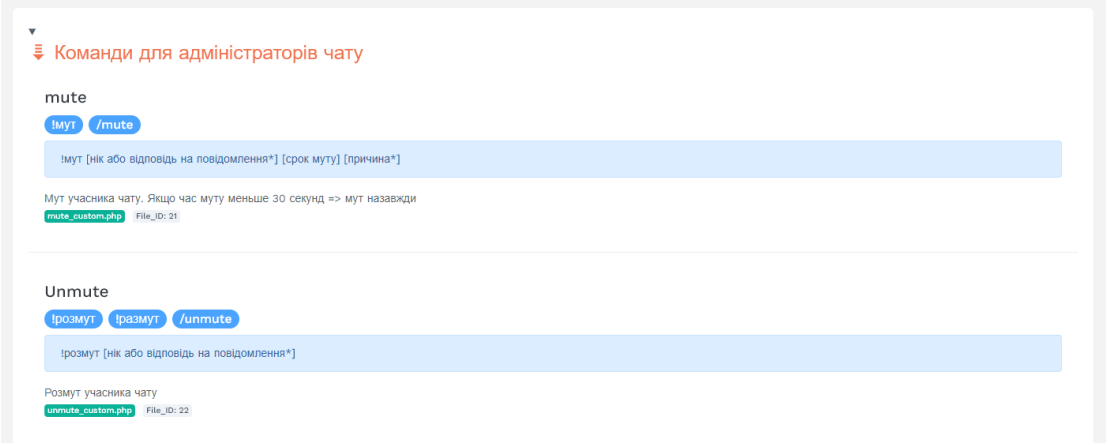


Рисунок 3.40 - Список команд для рангу

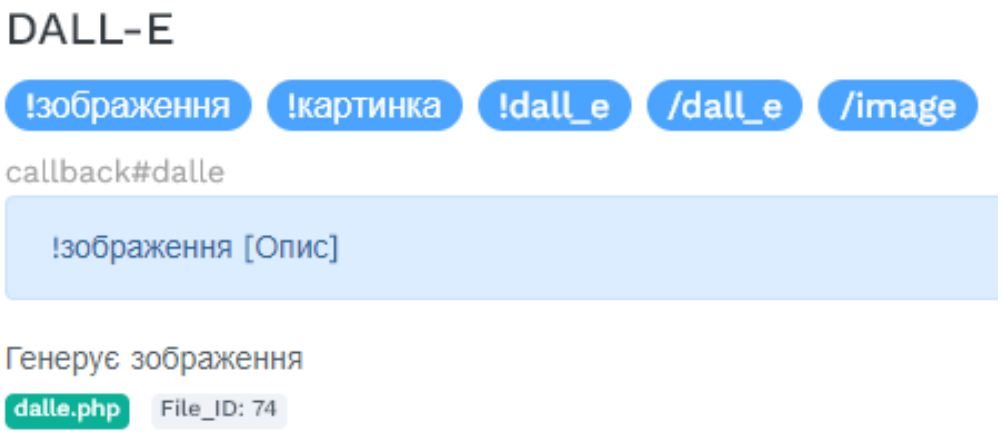


Рисунок 3.41 - Довідка про команду

У блоці кожної команди відображається назва її модулю, ініціатори,

синтаксис, опис, ім’я файлу, яке відповідає цьому модулю та ідентифікатор модуля у базі даних.

Форми

Одна з основних функцій, яка створена для автоматизації роботи студентського деканату, кураторів та старост груп.

На головній сторінці цього розділу ми можемо або створити нову форму, або переглянути створені.

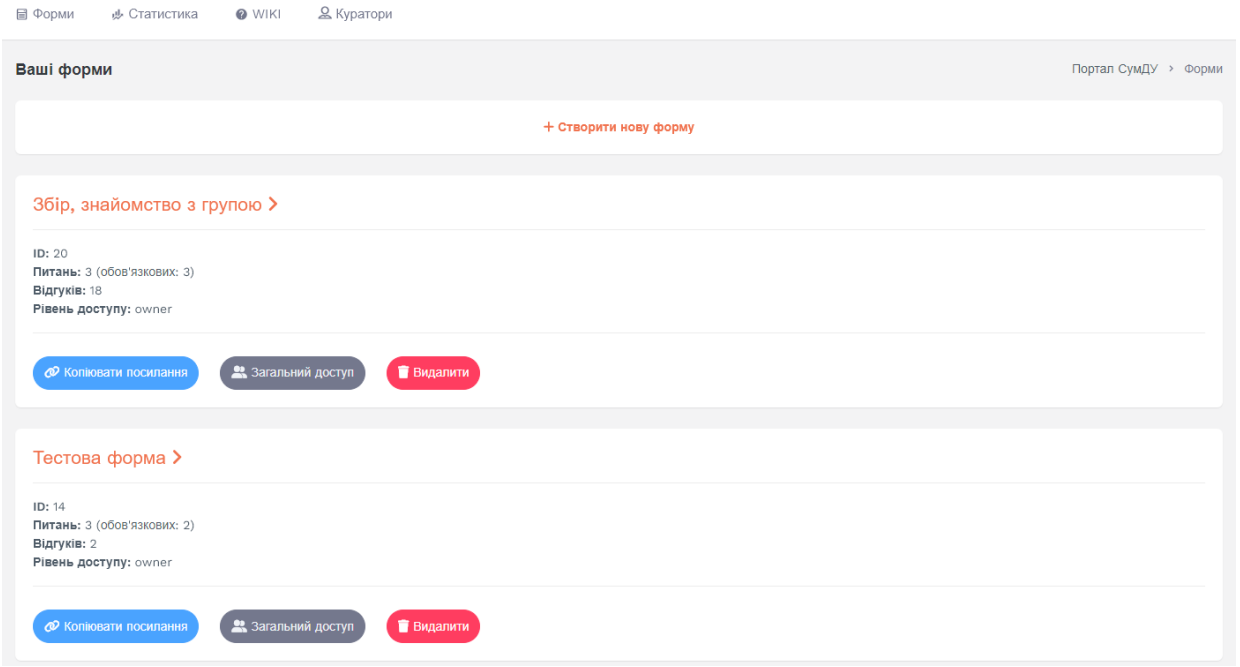


Рисунок 3.42 - Список форм користувача

Біля кожної форми є кнопки «Копіювати посилання на форму», «Загальний доступ» та «Видалити форму».

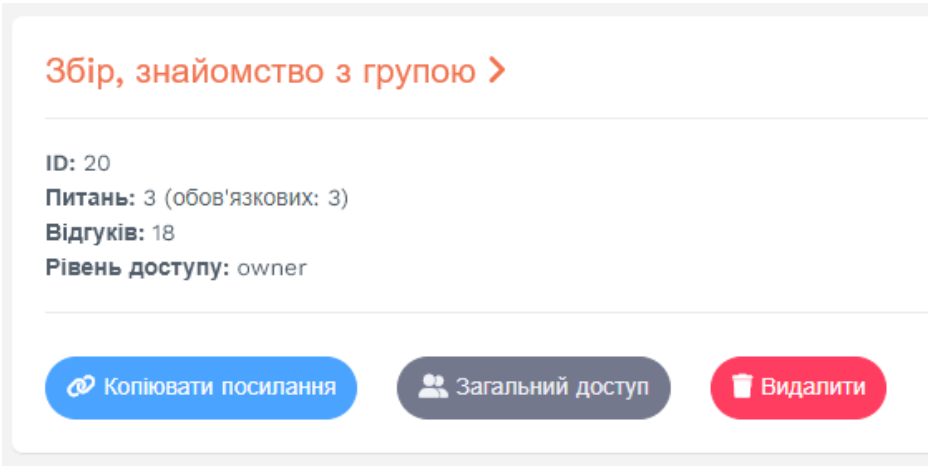


Рисунок 3.43 - Форма користувача

При натисканні на кнопку «Загальний доступ» відкривається

модальне вікно, де ми можемо або додати адміністратора, або видалити вже доданого.

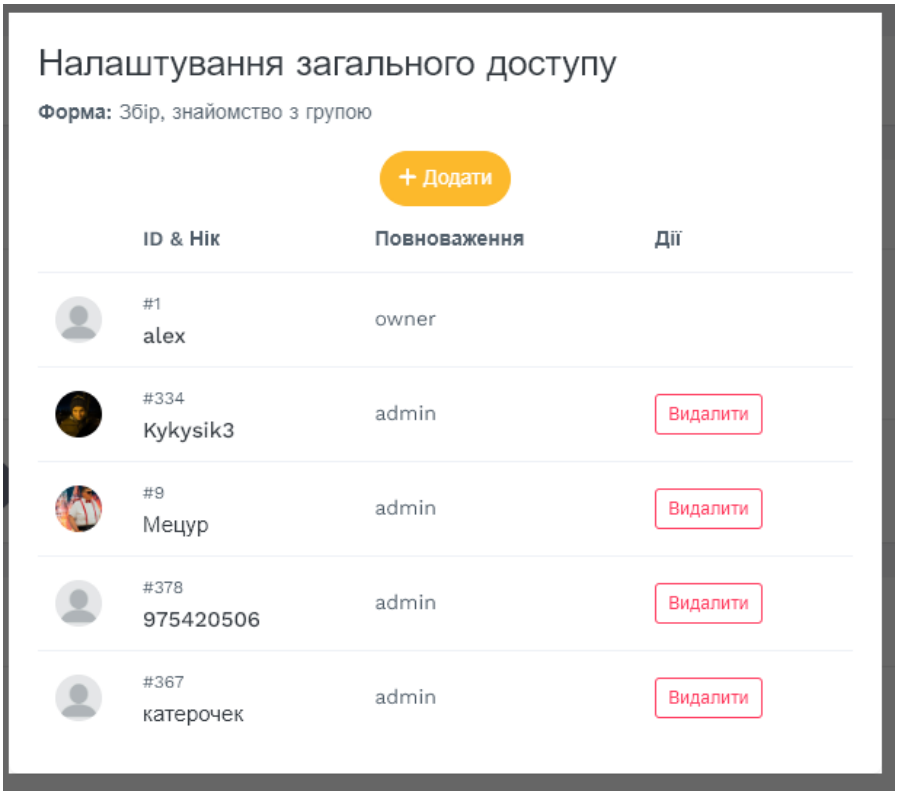


Рисунок 3.44 - Налаштування загального доступу до форми

Якщо ми натиснемо кнопку «Додати» - буде відкрито модальне вікно додання адміністратора або менеджера форми.

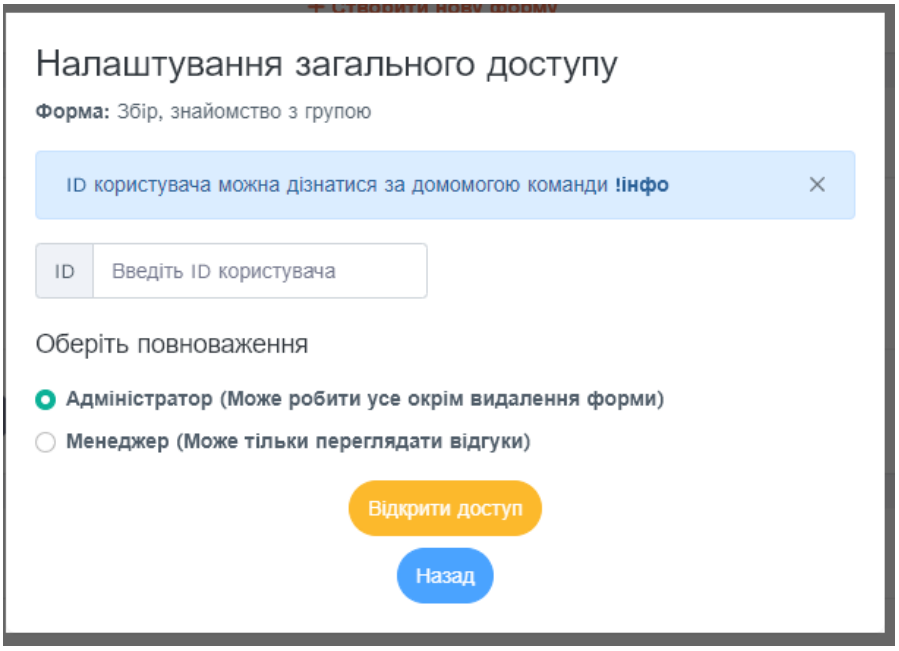


Рисунок 3.45 - Налаштування загального доступу до форми (додання адміністратора або менеджера)

Коли ми вводимо ID користувача – автоматично підтягується його

нік.

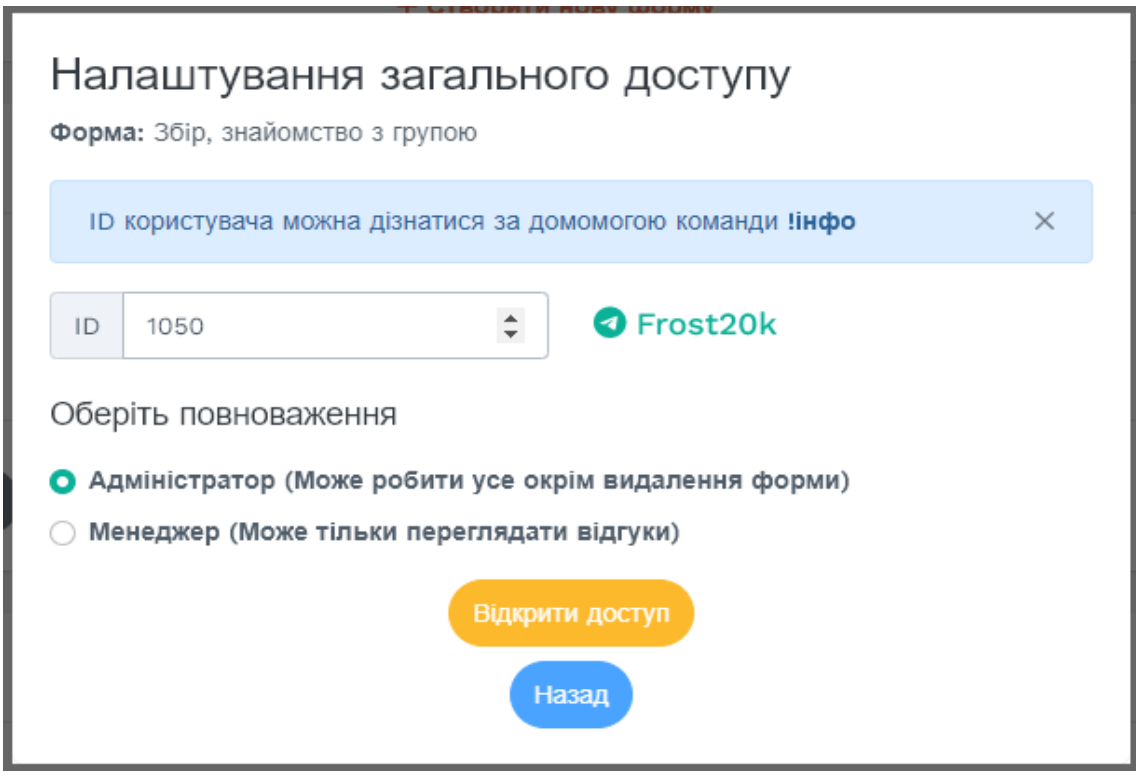


Рисунок 3.46 - Нік користувача автоматично підвантажується

Повернемося до списку форм та їх елементів керування. Кнопка «видалити» видаляє форму, а при натисканні на назву форми ми попадемо на сторінку з розширеним керуванням форми та таблицею відгуків на неї.

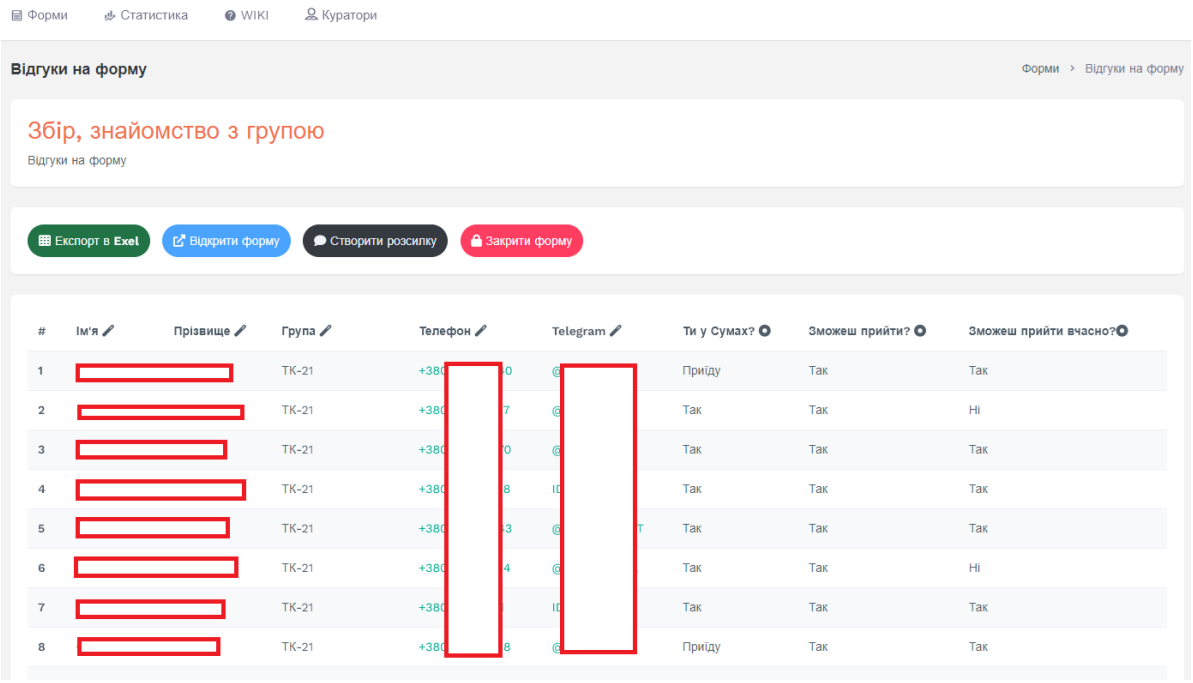


Рисунок 3.47 - Відгуки на форму

Аби зберегти конфіденційність користувачів – деякі дані було

приховано.

Зверху є розширені інструменти керування формою.

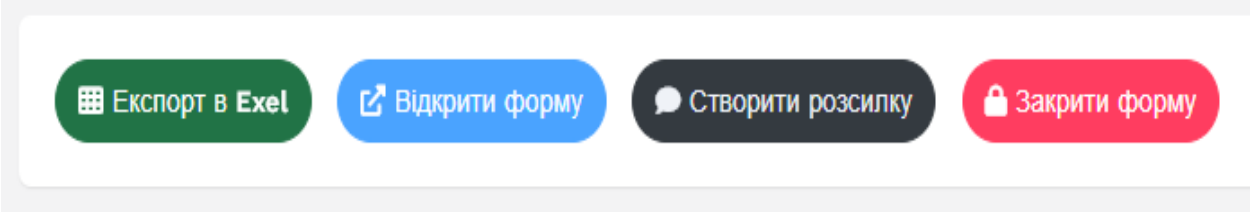


Рисунок 3.48 - Інструменти керування формою

«Експорт в Exel» – формує Exel файл та завантажує його на комп'ютер за допомогою бібліотеки PHP Office та API LibreOffice на сервері.

«Відкрити форму» - Відкриває форму у режимі заповнення користувачем.

«Створити розсилку» - надає можливість адміністраторам форми розіслати користувачам, які відгукнулись на неї, повідомлення в Telegram, яке надійде їм від імені бота.

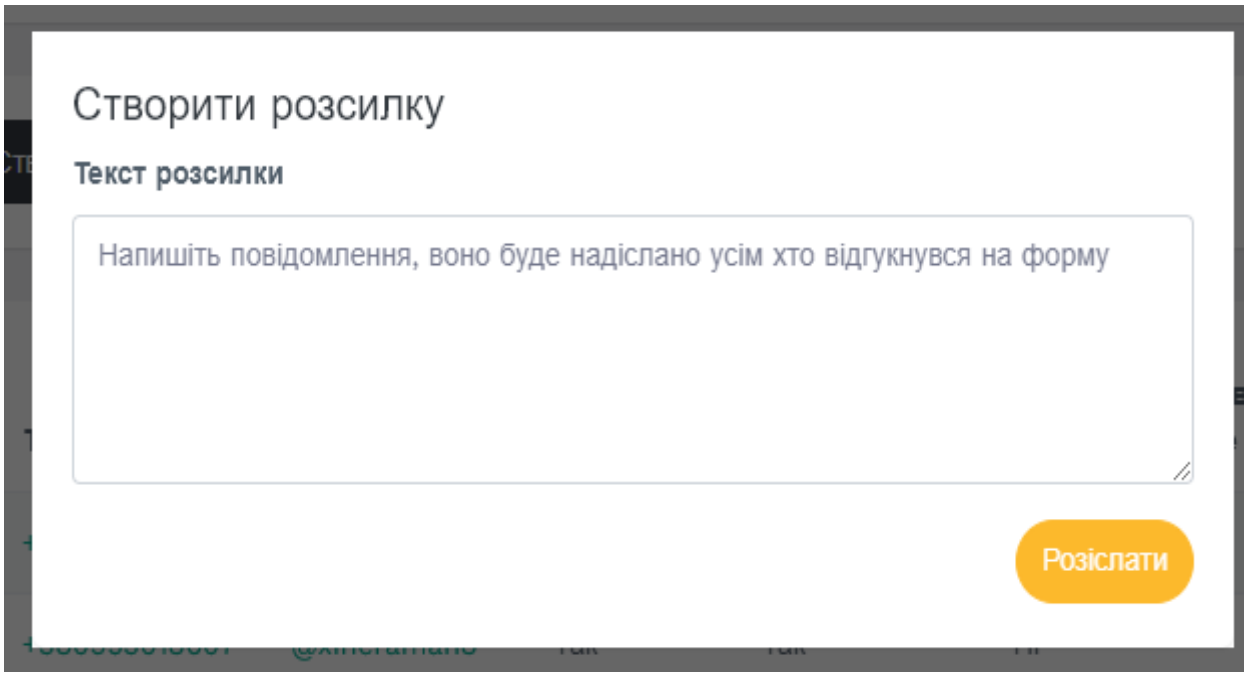


Рисунок 3.49 - Вікно створення розсилки

«Закрити» або «Публікувати форму» (в залежності від її стану) – закриває або відкриває форму для нових відгуків.

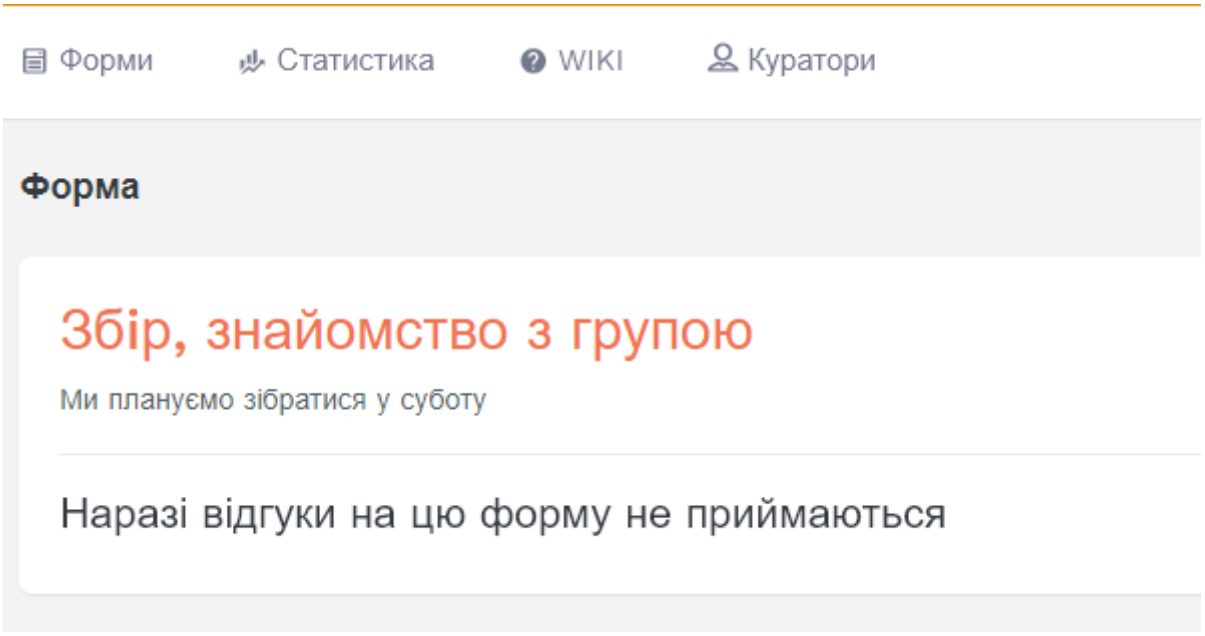


Рисунок 3.50 - Форма закрита для нових відгуків

Також на цій сторінці можна сформувати діаграму обравши потрібне питання з відповідями.

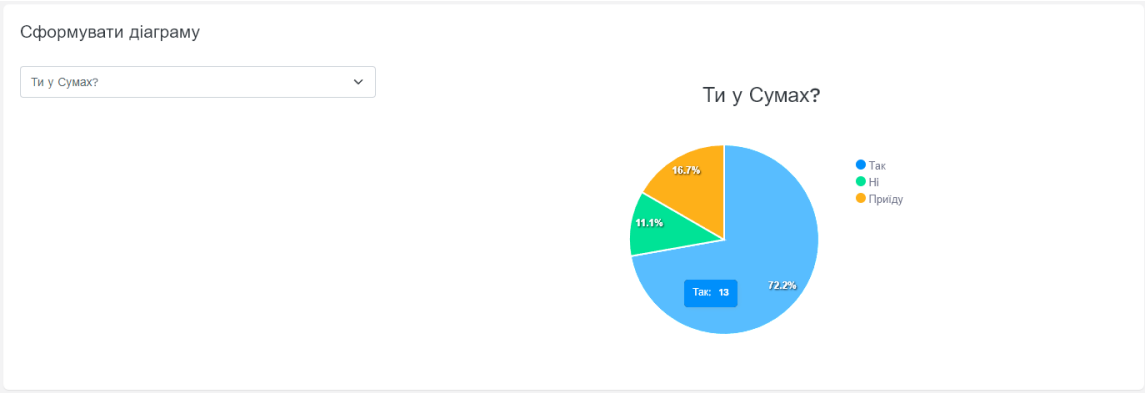


Рисунок 3.51 - Сформована діаграма для питання

Заповнення форми користувачем

Коли користувач потрапляє на форму – він одразу попереджується що після заповнення форми, її адміністраторам будуть автоматично надіслані його конфіденційні дані, а саме: номер телефону, ПІБ, групу.

Після заповнення користувачу буде відображено текст який ввів адміністратор при її створенні.

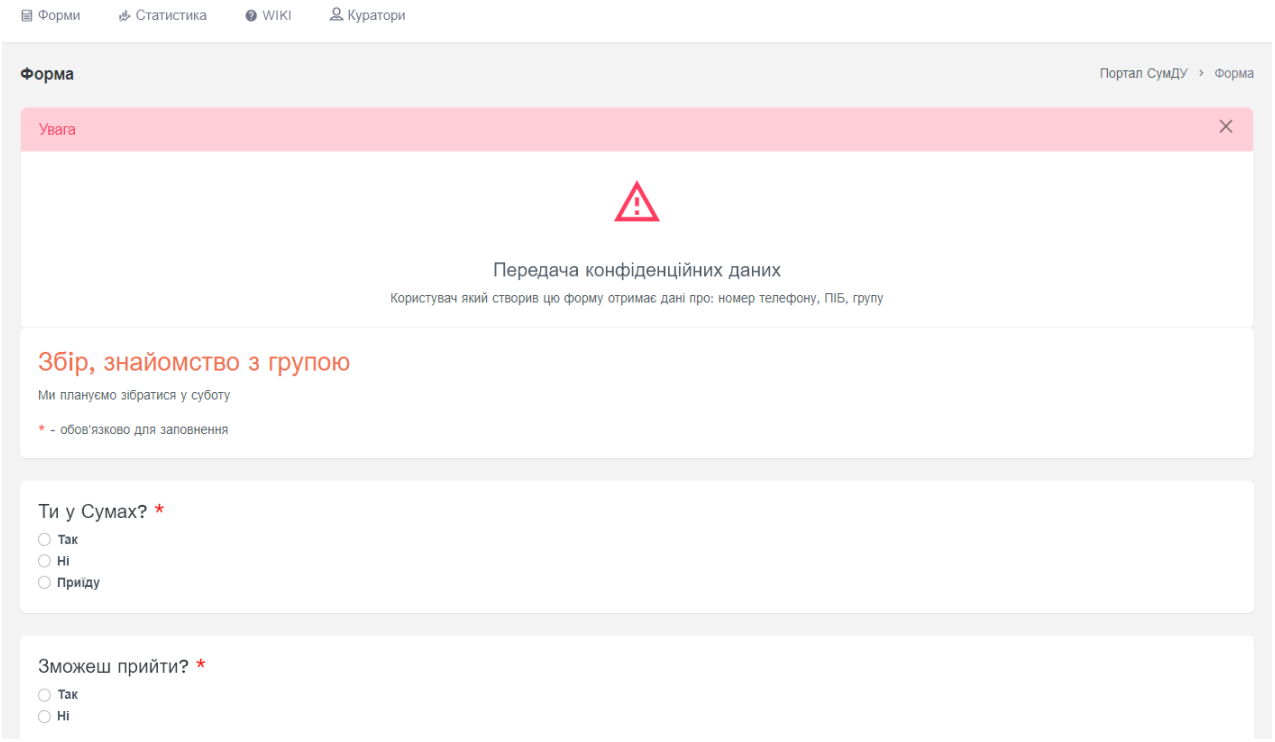


Рисунок 3.52 - Сторінка заповнення форми

Створення форми

На сторінці створення нової форми, користувач заповнює назву фоми, її опис, та формує питання.

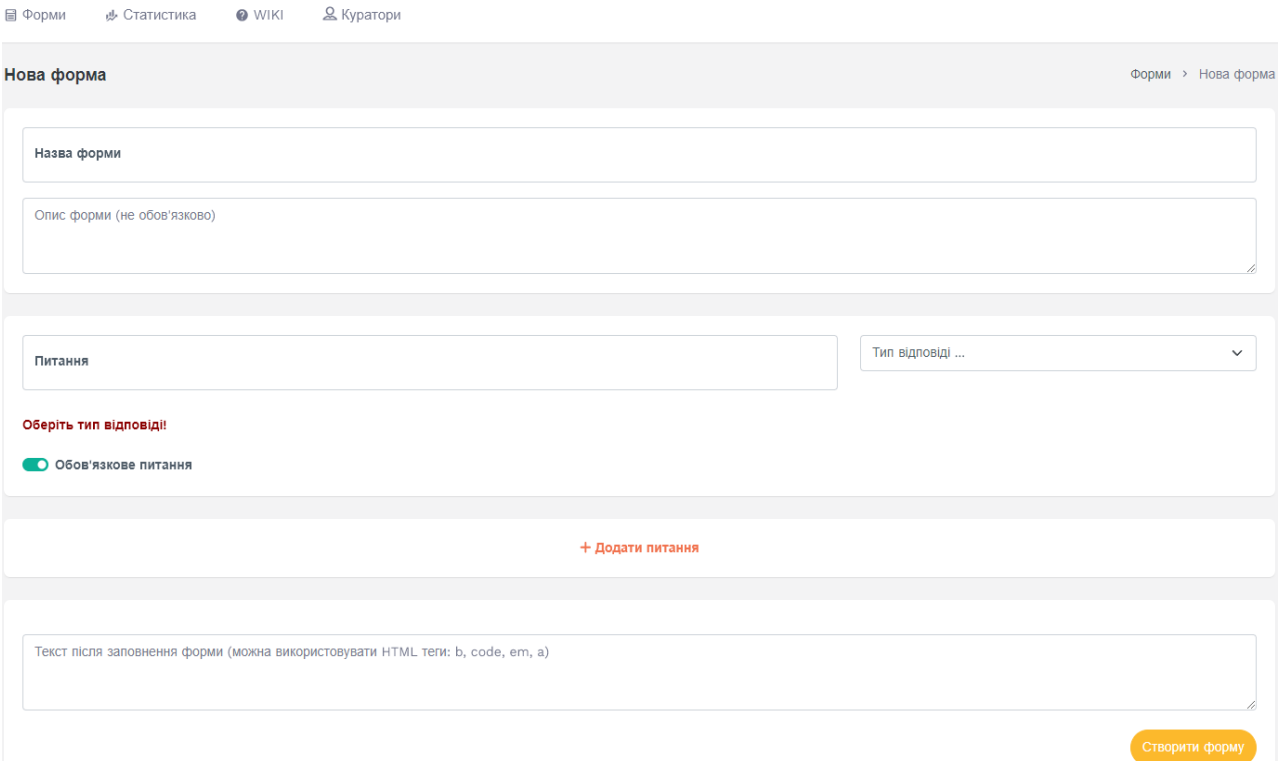


Рисунок 3.53 - Сторінка створення форми

На момент написання цієї роботи є 2 типи відповідей

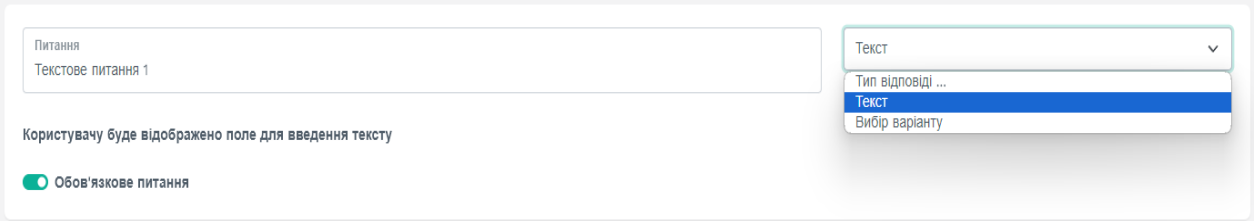


Рисунок 3.54 - Типи відповідей на питання

Текстова відповідь та «Вибір варіанту»

Також користувач може створювати обов’язкові питання та не обов’язкові для заповнення.

Якщо це не перше питання, тоді у користувача буде можливість видалити його.

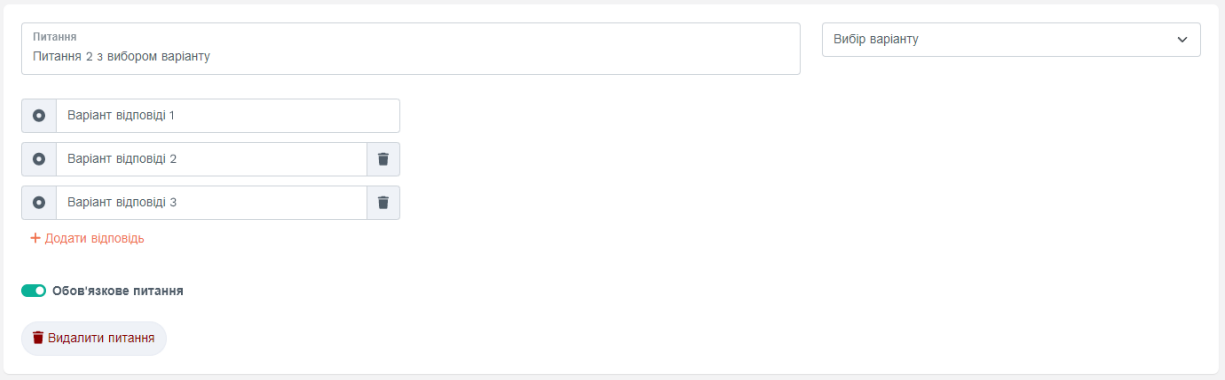


Рисунок 3.55 - Елементи керування питанням

У режимі створення питання з вибором варіанту – можна створювати завгодну кількість варіантів, та так само видаляти їх натиснувши на значок смітника.

У кінці користувач може написати текст після заповнення форми. Якщо тексту нема – буде відображено стандартний текст «Дякуємо за відповіді!»

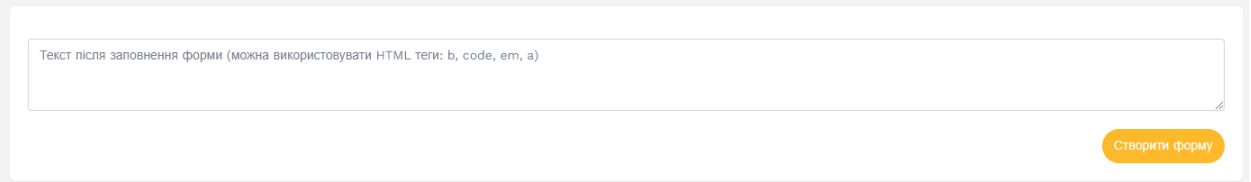


Рисунок 3.56 - Фо.рма вводу тексту після заповнення форми

Для запобігання XSS атак увесь html фільтрується у цій комірці, та пропускаються лише дозволені теги та атрибути.

Після натискання «Створити форму», користувач потрапляє на список форм.

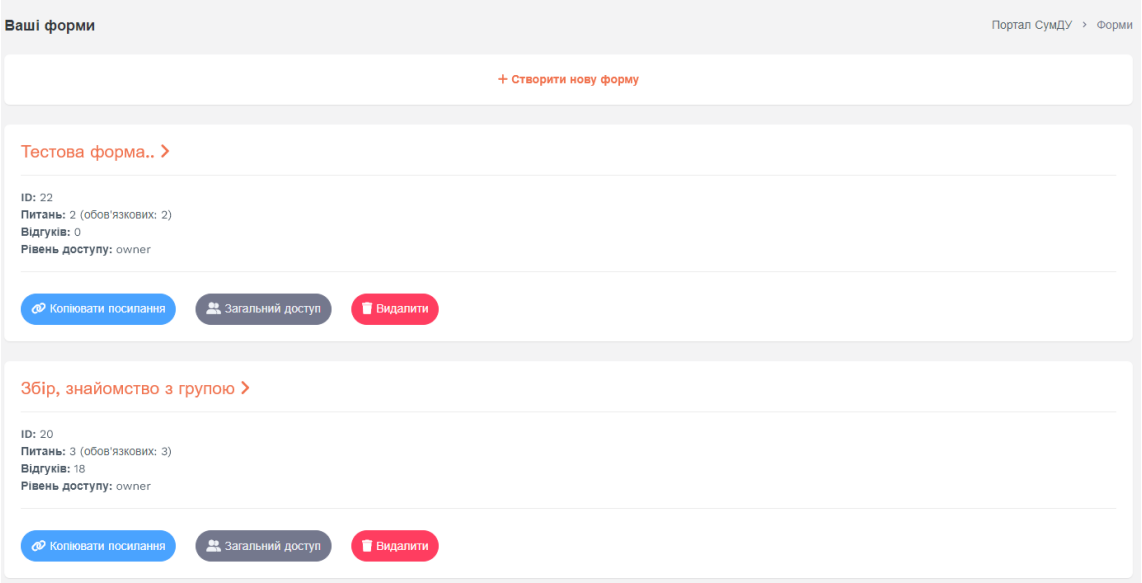


Рисунок 3.57 - Щойно створена форма у списку форм

Він отримує ранг owner у своїй формі.

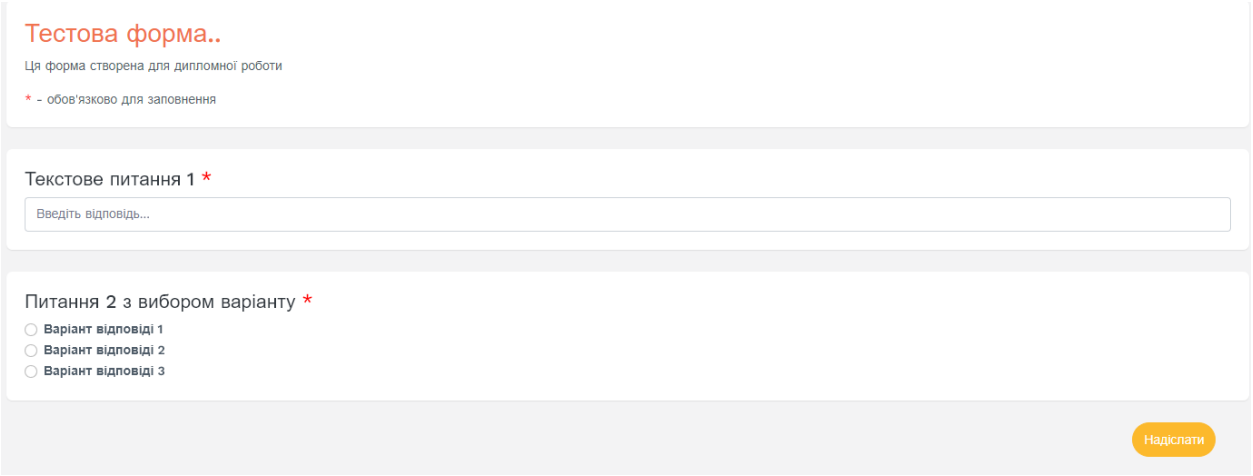


Рисунок 3.58 - Заповнення щойно створеної форми

Система керування кураторами

Також одна з ключових функцій автоматизації для студентського деканату. Деканат може видавати права на кураторство користувачам, а система в свою чергу автоматично зв’язує нових користувачів які вступають

в чат своєї групи з цією групою, також ці користувачі звільняються від перевірок botcheck. В подальшому деканат може створювати розсилки на необхідні групи.

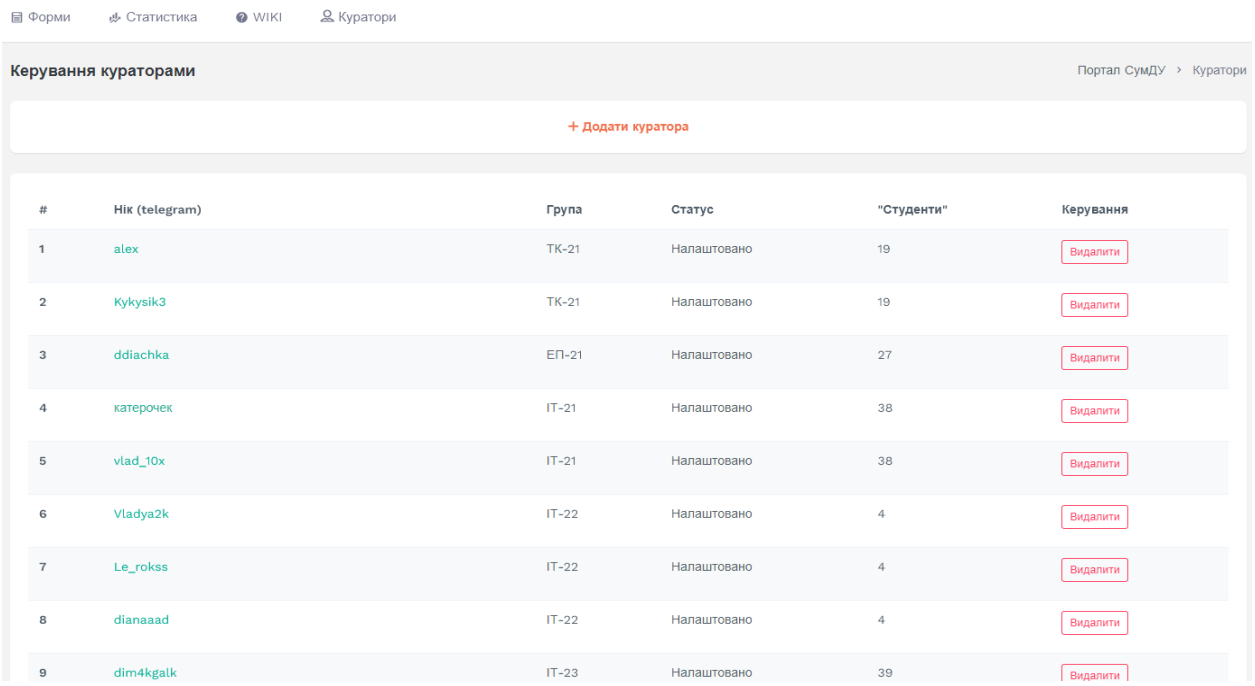


Рисунок 3.59 - Сторінка керування кураторами

Нові куратори також легко додаються, для цього необхідно натиснути на кнопку «Додати куратора», ввести групу якою він буде керувати та його нік. Також система автоматично «підтягує» назви груп та ніків.

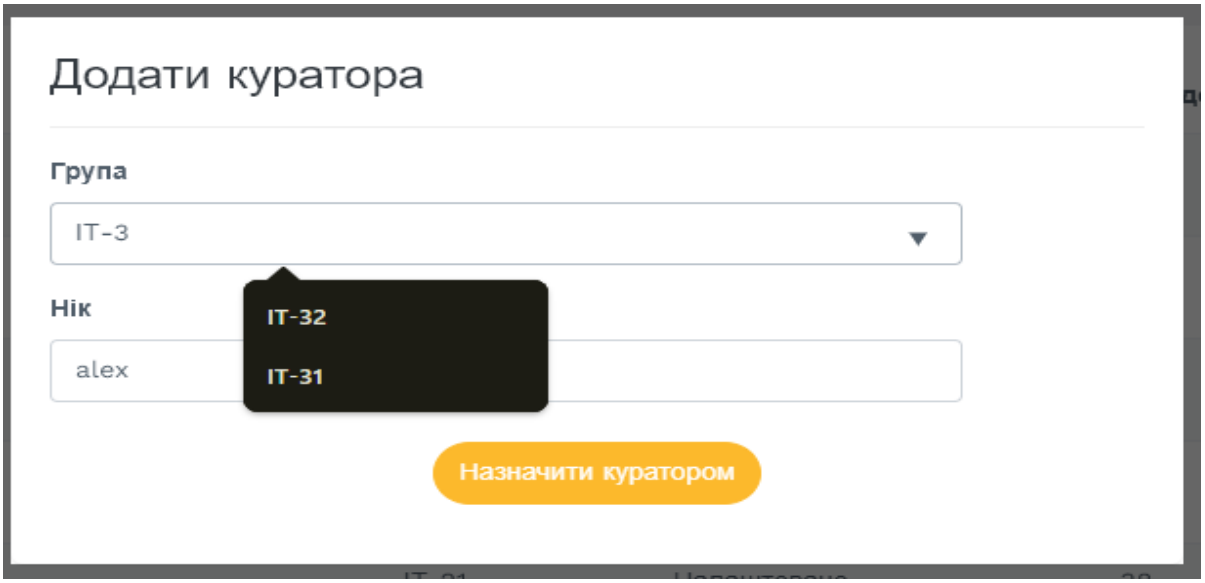


Рисунок 3.60 - Підвантаження груп та ніків користувачів

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було реалізовано великий та комплексний проект, спрямований на створення інформаційної веборієнтованої системи для організаційної роботи студентського деканату. Розроблена система включає в себе різноманітні функціональні компоненти, що значно полегшують та автоматизують процеси комунікації, управління та аналізу даних у студентському середовищі. Вона є гнучкою та масштабованою, що дозволяє легко додавати нові функції та адаптувати її під змінювані потреби користувачів. Впровадження такої системи сприяє покращенню якості управління та освітнього процесу в університетах, що є важливим кроком у напрямку інноваційного розвитку освітньої сфери.

Варто зазначити, що вся система була розроблена на чистому PHP з нуля без використання фреймворків. Це дозволило максимально гнучко підійти до вирішення специфічних задач проекту та забезпечити високу продуктивність і масштабованість коду. Крім того, розробка системи з нуля надала значний практичний досвід і дозволила глибоко зрозуміти, як працюють на низькому рівні фреймворки, наприклад Laravel. Це розуміння є цінним для подальшого розвитку в галузі веброзробки, оскільки воно дає змогу більш ефективно використовувати фреймворки, проектувати архітектуру та адаптуватись до специфічних потреб проектів.

На момент написання роботи, цей проект вже працює 4 роки. Декілька разів повністю змінювалась його архітектура на рівні ядра. Це дало глибоке розуміння проектування низькорівневої архітектури веб-програм. В останній версії ядра, яка працює вже 2 роки, застосовуються принципи модульності: ядро підключає лише необхідний йому модуль, не використовуючи непотрібний для конкретної операції код. Це підхід забезпечив підвищення продуктивності та ефективності роботи системи.

Наразі проектом користуються більше тисячі користувачів, тому у реалізації функціоналу враховувалися позитивні та негативні відгуки та їх

пропозиції щодо подальшого розвитку проекту. Це допомогло створити більш адаптовану та зручну для користувачів систему, що відповідає їх потребам та очікуванням.

Подальші перспективи розвитку. Подальший розвиток проекту включає:

- Оптимізацію існуючих функцій для підвищення ефективності та зручності використання.
- Додавання нових модулів та функціональних компонентів для розширення можливостей системи.
- Підвищення рівня безпеки та захисту даних користувачів.
- Розробку мобільного додатку для ще більш зручного доступу до функціоналу системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Telegram Bot API [Електронний ресурс]. - Режим доступу: https://core.telegram.org/bots/api (дата звернення 25.05.2024)

2. OpenAI API [Електронний ресурс]. - Режим доступу: https://platform.openai.com/docs/api-reference/introduction (дата звернення 25.05.2024)

3. Modern PHP: New Features and Good Practices - J. Josh Lockhart, O'Reilly Media, 2015

4. Design Patterns for Modern Web Applications, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - ACM Computing Surveys, 1995

5. MySQL High Availability - Charles Bell, Mats Kindahl, Lars Thalmann - O'Reilly Media, 2014 – 760 p.

6. Design and Implementation of a Scalable Information System for Large User Bases - James Smith, Linda Garcia, International Journal of Computer Applications, 2020.

7. Distributed Architectures for High-Performance Information Systems - Rajesh Gupta, Sunita Patil, Journal of Distributed and Parallel Databases, 2018

8. Scalable Design Patterns for PHP-Based Telegram Bots - John Doe, Jane Smith - Journal of Web Development and Technologies, 2021

9. Efficient Data Handling and Storage for PHP Telegram Bots - Rajesh Gupta, Sunita Patil - Journal of Data Management and Information Systems, 2018

10. Prompt Engineering or Fine Tuning: An Empirical Assessment of Large Language Models in Automated Software Engineering Tasks - Feng et al. – Cornell University, arXiv, 2023

11. Sample Design Engineering: An Empirical Study of What Makes Good Downstream Fine-Tuning Samples for LLMs - Zhou et al. - Cornell University, arXiv, 2024

12. Prompt Engineering or Fine-Tuning? A Case Study on Phishing

Detection with Large Language Models - Fouad Trad, Ali Chehab – MDPI 2024, режим доступу: https://www.mdpi.com/2504-4990/6/1/18 (дата звернення 25.05.2024)

13. GitHub цього проекту [Електронний ресурс]. - Режим доступу: - https://github.com/Alex47517/SumduBot_diploma (дата звернення 25.05.2024)

14. Stuart J. Russell, Peter Norvig. Artificial Intelligence: A Modern Approach. — 3. — Pearson, 2015.

15. MIX SumDU [Електронний ресурс]. - Режим доступу: https://mix.sumdu.edu.ua/ (дата звернення 25.05.2024)

16. Microsoft Teams [Електронний ресурс]. - Режим доступу: https://www.microsoft.com/uk-ua/microsoft-teams/(дата звернення 15.05.2024)

17. О. Мороз. Штучний інтелект // Філософський енциклопедичний словник / В. І. Шинкарук (гол. редкол.) та ін. — Київ : Інститут філософії імені Григорія Сковороди НАН України : Абрис, 2002. — с. 727.

18. Засоби штучного інтелекту: навч. посіб. / Р. О. Ткаченко, Н. О. Кустра, О. М. Павлюк, У. В. Поліщук; М-во освіти і науки України, Нац. ун-т «Львів. політехніка». — Львів: Вид-во Львів. політехніки, 2014. — 204 с.

19. Zoom [Електронний ресурс]. - Режим доступу: - https://zoom.us/ (дата звернення 10.05.2024)

20. Meet [Електронний ресурс]. - Режим доступу: https://meet.google.com/ (дата звернення 05.05.2024)

21. Miao, Fengchun, Holmes, Wayne, Ronghuai Huang, Hui Zhang: AI and education: guidance for policy-makers – UNESCO, 2021. – 45 с.

ДОДАТОК А (bot.php)

```
<?php
require_once 'config/start.php';
require_once 'config/loader.php';
use api\{Bot as Bot, chat as chat, ChatMember as ChatMember, Log as Log, update as update};
$request_json = file_get_contents('php://input');
$request = json_decode($request_json, true);
```

```
$bot = new Bot($bot_token);
$update = new update($request);
if (!update::$chat['id']) die('!chat');
if (update::$date + 20 < date('U')) die('Timeout');
$chat = new chat(update::$chat['id']);
if (!$chat->chat['id']) {  $chat->storeChat(update::$chat);
}
$debug_all = R::load('settings', 3);
if ($debug_all['value']) {
   $chat->sendMessage('⚙ [DEBUG/ALL]
'.var_export($request, true));
}
$user = new User();
if (update::$new_chat_member) {
   $from = update::$new_chat_member;
} else $from = update::$from;
if (!$user->loadByTGID($from['id'])) {
   $user->newUser($from);
   $curator = R::findOne('curators', 'chat_id = ?', [$chat->chat['id']]);
   if ($curator) {
      $user->update('botcheck', 1);
      $user->update('grp', $curator['grp']);
   }
}
$chatMember = new ChatMember($user->user['id'], $chat->chat['id']);
require_once 'functions.php';
require_once 'permissions.php';

$msg = update::$message['text'];
if ($msg[0] == '/') $msg = str_replace('@'.$bot_username, '', $msg);
if ($msg == '0' && !$user->user['display']) die();
$cmd = explode(' ', $msg);
$callback_data = update::$callback_data;
$display = $user->user['display'];
if ($msg) $log = $msg;
else $log = 'callback:'.$callback_data.' ('.$display.')';
Log::admin('MSG', ''.$chat->chat['id'].' | '.$chat->chat['title'].' | <a href="tg://user?id='.$user-
>user['tg_id'].'">'.$user->user['nick'].'</a>: '.$log);

//DEBUG:
if ($chat->chat_id == $admin_user_id) {
```

```
    $result = $chat->sendMessage(var_export($request, true));
}

if (($msg == '/start' or $msg == '/start@'.$bot_username) && $user->user['tg_id'] != $chat->chat['tg_id']) {
    menu();
}

//перевірка на ЧС
if ($user->user['blacklist']) {
    if (($user->user['blacklist'] - date('U')) > 0 or $user->user['blacklist'] == 1) {
        die();
    } else {
        $user->update('blacklist');
    }
}
if ($chatMember->chatMember['blacklist']) {
    if (($chatMember->chatMember['blacklist'] - date('U')) > 0 or $chatMember->chatMember['blacklist'] == 1) {
        die();
    } else {
        $chatMember->update('blacklist');
    }
}
$chatMember->update('message_counter', ($chatMember->chatMember['message_counter']+1));
//файли із core_commands будуть підключені усі без виключення
//це системні команди, котрі повинні постійно виконуватися
$dir = __DIR__.'/core_commands/';
$files = scandir($dir);
foreach($files as $file) {
    if (($file !== '.') and ($file !== '..'))
        include_once $dir."$file;
}

$init = ['text' => $cmd[0], 'callback' => explode('_', $callback_data)[0], 'display' => explode('_', $display)[0]];
foreach ($init as $key => $in) {
    if ($action) break;
    if ($in) $action = R::findOne('actions', '`initiator` = ? AND `type` = ?', [$in, $key]);
    if ($key == 'display' && $in && $action) $ex_display = explode('_', $display);
    if ($key == 'callback' && $in && $action) $ex_callback = explode('_', $callback_data);
}
if (!$action['file_id']) die();
$file = R::load('commandfiles', $action['file_id']);
```

```
switch ($file['rank']) {
    case 'OWNER': Permissions::Owner($user->user); break;
    case 'ADMIN': Permissions::Admin($user->user); break;
    case 'STAR': Permissions::Star($user->user); break;
    case 'MODER': Permissions::Moder($user->user); break;
    case 'ChatAdmin': Permissions::ChatAdmin($user->user); break;
}
if ($action['args'] && !$cmd[$action['args']]) args_error($action);
if ($action) {
    require_once __DIR__.'/commands/'.$file['filename'];
}
```

ДОДАТОК Б (start.php)

```
<?php
require_once 'rb.php';
require_once __DIR__.'/../../tokens.php';
// Вміст файлу tokens.php:
// $bot_token = 'BOT_TOKEN';
// $admin_user_id = [458746251];
// $db_host = 'DB_HOST';
// $db_name = 'DB_NAME';
// $db_login = 'DB_LOGIN';
// $db_pass = 'DB PASSWORD';
const DOMAIN = 'sumdubot.pp.ua';
$bot_username = 'Sumdu_bot';
$chat_for_checkcodes = -1001399681943;
R::setup( 'mysql:host='.$db_host.';dbname='.$db_name, $db_login, $db_pass);
```

ДОДАТОК В (loader.php)

```
<?php
require_once __DIR__.'/../api/update.php';
require_once __DIR__.'/../api/Bot.php';
require_once __DIR__.'/../api/AutoClean.php';
require_once __DIR__.'/../api/chat.php';
require_once __DIR__.'/../api/Log.php';
require_once __DIR__.'/../api/stats.php';
require_once __DIR__.'/../api/ChatMember.php';
require_once __DIR__.'/../Bank.php';
require_once __DIR__.'/../User.php';
```

ДОДАТОК Г (chat.php)

```
<?php
namespace api;
use R;
use api\{update as update, AutoClean as AutoClean};

class chat {
    public $chat;
    public $chat_id; //TELEGRAM ID
    function __construct($chat_id) {
        $this->chat_id = $chat_id;
        $chat = R::findOne('chats', 'tg_id = ?', [$chat_id]);
        if ($chat) {
            $this->chat = $chat;
            return true;
        } else return false;
    }
    public function storeChat($chat) {
        if ($chat['type'] == 'group' or $chat['type'] == 'supergroup') {
            $this->sendMessage('👋 Вас вітає <b>СумДУ бот</b>


<em>Для корректної роботи надайте права адміністратора боту</em>');
        } elseif ($chat['type'] == 'private') {
            $chat['title'] = 'PM';
            $this->sendMessage('👋 Вас вітає <b>СумДУ бот</b>');
        } else {
            die('Invalid chat type');
        }
        $new_chat = R::dispense('chats');
        $new_chat->tg_id = $chat['id'];
        $new_chat->title = $chat['title'];
        $new_chat->type = $chat['type'];
        R::store($new_chat);
        $this->chat = $new_chat;
        return true;
    }
    public function sendMessage($text, $reply_to_message_id = null, $reply_markup = null, $disable_notification
= true, $protect_content = false, $associative = false) {
        $text = str_replace($GLOBALS['bot_token'], '[secret]', $text);
```

```
    if ($reply_markup) $reply_markup = json_encode($reply_markup);
    $params = [
        'text' => $text,
        'chat_id' => $this->chat_id,
        'reply_to_message_id' => $reply_to_message_id,
        'reply_markup' => $reply_markup,
        'disable_notification' => $disable_notification,
        'allow_sending_without_reply' => true,
        'protect_content' => $protect_content,
        'parse_mode' => 'html',
    ];
    $result = Bot::request('sendMessage', $params, $associative);
    AutoClean::addToRemove($this->chat, $result->result->message_id);
    return $result;
}
public function update($param, $value = null) {
    $this->chat[$param] = $value;
    R::store($this->chat);
    return true;
}
public function getChatMember($user_id) {
    $params = [
        'user_id' => $user_id,
        'chat_id' => $this->chat_id,
    ];
    return Bot::request('getChatMember', $params);
}
public function editMessageText($text, $reply_markup, $message_id) {
    if (update::$with_photo) {
        $this->deleteMessage($message_id);
        return $this->sendMessage($text, null, $reply_markup);
    }
    if ($reply_markup) $reply_markup = json_encode($reply_markup);
    $params = [
        'message_id' => $message_id,
        'text' => $text,
        'parse_mode' => 'html',
        'chat_id' => $this->chat_id,
        'reply_markup' => $reply_markup
    ];
    return Bot::request('editMessageText', $params);
```

```
    }
    public function restrictChatMember($user_id, $permissions, $until_date) {
        $params = [
            'user_id' => $user_id,
            'chat_id' => $this->chat_id,
            'permissions' => json_encode($permissions),
            'until_date' => $until_date,
        ];
        return Bot::request('restrictChatMember', $params);
    }
    public function banChatMember($user_id, $until_date, $revoke_messages = false) {
        $params = [
            'user_id' => $user_id,
            'chat_id' => $this->chat_id,
            'revoke_messages' => $revoke_messages,
            'until_date' => $until_date,
        ];
        return Bot::request('banChatMember', $params);
    }
    public function unbanChatMember($user_id, $only_if_banned = true) {
        $params = [
            'user_id' => $user_id,
            'chat_id' => $this->chat_id,
            'only_if_banned' => $only_if_banned,
        ];
        return Bot::request('unbanChatMember', $params);
    }
    public function createChatInviteLink($name, $expire_date = null, $member_limit = null,
$creates_join_request = null) {
        $params = [
            'name' => $name,
            'chat_id' => $this->chat_id,
            'expire_date' => $expire_date,
            'member_limit' => $member_limit,
            'creates_join_request' => $creates_join_request,
        ];
        return Bot::request('createChatInviteLink', $params);
    }
    public function sendPhoto($photo, $caption = null, $reply_to_message_id = null, $reply_markup = null,
$disable_notification = true, $protect_content = false) {
        if ($reply_markup) $reply_markup = json_encode($reply_markup);
```

```
        $params = [
            'photo' => $photo,
            'caption' => $caption,
            'reply_to_message_id' => $reply_to_message_id,
            'reply_markup' => $reply_markup,
            'disable_notification' => $disable_notification,
            'protect_content' => $protect_content,
            'parse_mode' => 'html',
            'chat_id' => $this->chat_id,
            'allow_sending_without_reply' => true,
        ];
        return Bot::request('sendPhoto', $params);
    }
    public function deleteMessage($message_id) {
        $params = [
            'message_id' => $message_id,
            'chat_id' => $this->chat_id,
        ];
        return Bot::request('deleteMessage', $params);
    }
    public function sendDice($emoji) {
        $params = [
            'emoji' => $emoji,
            'chat_id' => $this->chat_id,
        ];
        return Bot::request('sendDice', $params);
    }
    public function answerCallbackQuery($text, $show_alert = false, $url = null) {
        $params = [
            'text' => $text,
            'callback_query_id' => update::$callback_id,
            'chat_id' => $this->chat_id,
            'show_alert' => $show_alert,
            'URL' => $url,
        ];
        return Bot::request('answerCallbackQuery', $params);
    }
}
```

ДОДАТОК Д (chatMember.php)

```
<?php
namespace api;
use R;
class ChatMember {
    public $chatMember;

    public function __construct($user_id, $chat_id) {
        $load_status = $this->load($user_id, $chat_id);
        if (!$load_status) {
            $this->newChatUser($user_id, $chat_id);
        }
        return true;
    }

    public function load($user_id, $chat_id) {
        $chatMember = R::findOne('chatmembers', 'user_id = ? AND chat_id = ?', [$user_id, $chat_id]);
        if ($chatMember) {
            $this->chatMember = $chatMember;
            return true;
        } else {
            return false;
        }
    }

    public function newChatUser($user_id, $chat_id) {
        $chatMember = R::findOne('chatmembers', 'user_id = ? AND chat_id = ?', [$user_id, $chat_id]);
        if ($chatMember) {
            $this->chatMember = $chatMember;
            return false;
        }
        $is_admin = $this->getChatStatus($user_id);
        $chatMember = R::dispense('chatmembers');
        $chatMember->user_id = $user_id;
        $chatMember->chat_id = $chat_id;
        $chatMember->is_admin = $is_admin;
        $chatMember->last_update_status = date('U');
        R::store($chatMember);
        $this->chatMember = $chatMember;
        return true;
```

```
    }

    public function update($param, $value = null) {
        $this->chatMember[$param] = $value;
        R::store($this->chatMember);
        return true;
    }

    public function getChatStatus($user_id) {
        global $chat;
        $user = R::load('users', $user_id);
        $chat_user = $chat->getChatMember($user['tg_id']);
        if ($chat_user->result->status == 'administrator' or $chat_user->result->status == 'creator') $is_admin =
true; else $is_admin = false;
        return $is_admin;
    }

    public function addToBlackList($time) {
        if ($time == 0) $store = 1; else $store = date('U')+round($time);
        $this->update('blacklist', ($store));
        return true;
    }
}
```

ДОДАТОК Е (User.php)

```
<?php
class User {
    public $user;
    public function loadByTGID($tg_id) {
        $user = R::findOne('users', 'tg_id = ?', [$tg_id]);
        if ($user) {
            $this->user = $user;
            return true;
        } else {
            return false;
        }
    }
    public function loadByID($id) {
        $user = R::load('users', $id);
        if ($user) {
            $this->user = $user;
            return true;
        } else {
            return false;
        }
    }
    public function loadByNick($nick) {
        $user = R::findOne('users', 'nick = ?', [$nick]);
        if ($user) {
            $this->user = $user;
            return true;
        } else {
            return false;
        }
    }
    public function loadByUsername($tg_id) {
        $user = R::findOne('users', 'username = ?', [$tg_id]);
        if ($user) {
            $this->user = $user;
            return true;
        } else {
            return false;
        }
    }
```

```
public function newUser($from) {
    $user = R::findOne('users', 'tg_id = ?', [$from['id']]);
    if ($user) {
        $this->user = $user;
        return false;
    }
    if ($from['username']) {
        $nick = $from['username'];
    } else {
        $nick = $from['id'];
    }
    Bank::add(4985);
    $user = R::dispense('users');
    $user->tg_id = $from['id'];
    $user->nick = $nick;
    $user->username = $from['username'];
    $user->rank = 'USER';
    $user->balance = 15;
    $user->reg_date = date('d.m.y');
    R::store($user);
    $this->user = $user;
    return true;
}
public function update($param, $value = null) {
    $this->user[$param] = $value;
    R::store($this->user);
    return true;
}
public function LocalStorageSet($key, $value) {
    $add = [$key => $value];
    if ($this->user['tmp']) {
        $tmp = json_decode($this->user['tmp'], true);
        $new_tmp = array_merge($tmp, $add);
    } else $new_tmp = $add;
    $this->update('tmp', json_encode($new_tmp));
    return true;
}
public function LocalStorageGet($key) {
    $tmp = json_decode($this->user['tmp'], true);
    return $tmp[$key];
}
```

```
    public function LocalStorageClear() {
        $this->update('tmp');
        return true;
    }
    public function addBal($add) {
        if (Bank::add($add*-1)) {
            $new_user_balance = $this->user['balance']+$add;
            $this->update('balance', $new_user_balance);
        } else {
            custom_error('Банк просить вибачення', 'Але на жаль він збанкрутував');
        }
    }
    public function addToBlackList($time) {
        if ($time == 0) $store = 1; else $store = date('U')+round($time);
        $this->update('blacklist', ($store));
        return true;
    }
}
```

ДОДАТОК Ж (chatGPT.php)

```
<?php
//
// Command: ChatGPT #
// Text: !gpt4v /gpt4v #
// Callback: gpt4v #
// Display: gpt4v #
// Info: ChatGPT 4 vision #
// Syntax: !gpt4v [Повідомлення з фото] #
// Args: 1 #
// Rank: USER #
//
require __DIR__ . '/../lib/Process.php';

use Orhanerday\OpenAi\OpenAi;
use api\{update as update, Log as Log, AutoClean as AutoClean, Bot as Bot};

$starter = ['!gpt4v', '/gpt4v'];
function roundUp($number, $precision = 3)
{

    $factor = pow(10, $precision);
   return ceil($number * $factor) / $factor;
}

if (in_array($cmd[0], $starter)) {
   AutoClean::save();
   if ($cmd[1]) {
      $message = str_replace($cmd[0] . ' ', '', $msg);
   }
   if ($user->user['balance_usd'] < 0.04) custom_error('Недостатньо коштів', 'Необхідно: 0.01 ⛽
У тебе: ' . $user->user['balance_usd'] . ' ⛽');
   $fp = fopen('/tmp/lockfile2', 'w+');
   if (flock($fp, LOCK_EX | LOCK_NB)) {
      $result = $chat->sendMessage('⌛ <b>ChatGPT</b> - Генерація відповіді...');
      $edit = $result->result->message_id;
      $tg_file_response = Bot::getFile(update::$photo_id);
      $path = $tg_file_response['result']['file_path'];
      $store_path = __DIR__.'/../../../sumdubot.pp.ua/img/tmp';
```

```
$filename = Bot::storeFile($path, $store_path, date('U').'_'.mt_rand(99, 999));
require_once '/../vendor/autoload.php';
$openai = new OpenAI('OPEN_AI_API_KEY');

$complete = $openai->chat([
   'model' => 'gpt-4-vision-preview',
   'messages' => [
      [
         "role" => "system",
         "content" => [[
            "type" => "text",
            "text" => "Ти ChatGPT 4, працюєш у SumduBot (Бот Сумского Державного Університету)
Тобі пише людина з ніком " . update::$from['username'] . "."
         ]]
      ],
      [
         "role" => "user",
         "content" => [
            [
               "type" => "image_url",
               "image_url" => "https://sumdubot.pp.ua/img/tmp/".$filename
            ],
            [
               "type" => "text",
               "text" => $message
            ]
         ]
      ],
   ],
   'max_tokens' => 2000,
]);

$prompt_tokens = json_decode($complete, true)['usage']['prompt_tokens'];
$completion_tokens = json_decode($complete, true)['usage']['completion_tokens'];
$result_price = roundUp(($prompt_tokens * 0.00001) + ($completion_tokens * 0.00003), 3);
$user->update('balance_usd', ($user->user['balance_usd'] - $result_price));
$chat->sendMessage('<b>З балансу <a href="tg://user?id=' . $user->user['tg_id'] . '">' . $user-
>user['nick'] . '</a> списано ' . $result_price . ' </b>

Input: <b>' . $prompt_tokens . ' tokens</b>
Output: <b>' . $completion_tokens . ' tokens</b>
```

```
' . $prompt_tokens . ' * 0.00001 + ' . $completion_tokens . ' * 0.00003 = ' . $result_price);
        $text = json_decode($complete, true)['choices'][0]['message']['content'];
        if (!$text) {
            $file = fopen('log.txt', 'w+');
            fwrite($file, var_export($openai->getCURLInfo(), true));
            fclose($file);
            $result = $chat->editMessageText('🔥 Виникла помилка при генерації
<code>' . var_export(json_decode($complete, true), true) . '</code>', null, $edit);
        }


        $result = $chat->editMessageText($text, null, $edit, 'Markdown');
        unlink($store_path.'/'.$filename);

    } else {
        echo 'Скрипт вже запущено';
    }

    fclose($fp);
}
```

Додаток К (схема бази даних)