Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Остапчука Микити Сергійовича
Прізвище, ім'я, по батькові студента(ки)

на здобуття ступеня вищої освіти Бакалавр

«Телеграм-бот для розкладу занять в університеті»
Назва теми

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.200252.01.18.ПЗ

Виконав студент IV курсу, група ІПЗ-20-1 _____ Микита ОСТАПЧУК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник канд. техн. наук, доцент _____ Оксана ЯШИНА
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. техн. наук, доцент _____ Юрій ФОРКУН
Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

12 червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
02 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Остапчука Микити Сергійовича
Прізвище, ім'я, по батькові студента

1. Тема роботи Телеграм-бот для розкладу занять в університеті

Керівник роботи Яшина Оксана Миколаївна канд. техн. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 08.01.2024 р. № 6
2. Строк подання студентом роботи на кафедру 01.06.2024 р.
3. Вихідні дані до роботи Методичні матеріали до кваліфікаційної роботи
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проєктування програмного забезпечення, програмна реалізація та тестування чат-бота.
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення:
1. Діаграма варіантів використання
2. Діаграма декомпозиції розробки бота
3. Схема бази даних

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Форкун Ю. В., доцент кафедри ІПЗ	30.05.24	06.06.24
Антиплагіат	Форкун Ю. В., доцент кафедри ІПЗ	30.05.24	06.06.24

7. Дата видачі завдання « 02 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проєктування, визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12– 31.12.2023	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розроблення технічного завдання	01.01 – 20.02.2024	
3 Проєктування програмного забезпечення	21.02 – 20.03 2024	
4 Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 – 30.04.2024	
5 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 – 25.05.2024	
6 Попередній захист КвР	травень 2024	Згідно графіка
7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2024	
8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2024	

Студент _____ Микита ОСТАПЧУК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник роботи _____ Оксана ЯШИНА
Підпис Ім'я, ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Телеграм-бот для розкладу занять в університеті».

Автор роботи: Остапчук Микита Сергійович.

Керівник роботи: Яшина Оксана Миколаївна.

Пояснювальна записка: 115 с., 47 рис., 1 табл., 5 дод., 30 джерел.

Графічна частина: 3 креслення.

БОТ, ЧАТ-БОТ, TELEGRAM, TELEGRAM BOT API, PYTHON, SQLITE, VISUAL STUDIO CODE, РОЗКЛАД.

Метою кваліфікаційної роботи є створення чат-бота для факультету інформаційних технологій, який допоможе студентам зручно знаходити необхідну інформацію та отримувати відповіді на запитання з мінімальними витратами часу.

У кваліфікаційній роботі досліджено взаємодії користувача з існуючими системами обміну миттєвими повідомленнями. Проведено огляд ключових факторів взаємодії та алгоритмів, покладених в основу чат-ботів. Описано принципи, які є основними для створення чат-ботів. Наведено актуальність розробки телеграм бота та застосування його в студентському житті. Розглянуто проблеми проєкту, які вирішує розроблення чат-боту. Показано детальний опис практичної частини роботи, результат проведеної роботи, а також обґрунтовано вибір використаних технологій.

Для розробки програмної системи використано мову програмування Python, середовище розробки Visual Studio Code, система керування базами даних SQLite.

У результаті був розроблений Телеграм-бот для інформаційної підтримки навчального процесу у месенджері Telegram з використанням Telegram Bot API.

12.06.2024
Дата


Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.200252.01.18.ПЗ	Пояснювальна записка	115		
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А3	КвРІПЗ.200252.01.18.Е8	UML-діаграма варіантів	1		
			використання			
5	А3	КвРІПЗ.200252.01.18.Е8	Схема бази даних	1		
6	А3	КвРІПЗ.200252.01.18.Е8	Діаграма декомпозиції	1		
			розробки бота			

Змн.	Арк.	№докум.	Підпис	Дата	КвРІПЗ.200252.01.18.ПЗ		
Виконав		Остапчук М. С.		12.06	Телеграм-бот для розкладу занять в університеті	Літ.	Арк. 1 / Аркушів 1
Керівник		Яшина О. М.		12.06			
Н. контр.		Форкун Ю. В.		12.06	Відомість документів	ХНУ, ІПЗ-20-1	
Зав. каф.		Бедратюк Л. П.		12.06			

ЗМІСТ

ВСТУП

Зростання використання чат-ботів у якості сучасного інструменту комунікації пояснюється тим, що вони все частіше використовуються в різних галузях для встановлення спілкування з інтернет-користувачами. Особливу популярність чат-боти здобули, коли їх інтегрували у месенджери та соціальні мережі, зокрема, у Telegram та Facebook, де з'явилися цілі вітрини та магазини чат-ботів.

У цій роботі під чат-ботами розуміються спеціальні програми, які імітують людську мовленнєву поведінку та виступають віртуальними співрозмовниками в онлайн-спілкуванні з одним або кількома користувачами.

Актуальність цієї роботи визначається ситуацією на ринку інформаційних технологій, який щороку поповнюється новими програмними продуктами, вебсервісами та мобільними застосунками. Ці інструменти не лише сприяють комунікації, але й допомагають зміцнити бренд університету як освітнього центру. Чат-бот спрощує рутинні завдання студентів, такі як отримання інформації про розклад занять, дзвінків, зміни в розкладі та інше. Головна перевага чат-бота полягає в тому, що всі ці можливості об'єднані на платформі єдиного месенджера.

Метою кваліфікаційної роботи є створення чат-бота для факультету інформаційних технологій, який допоможе студентам зручно знаходити необхідну інформацію та отримувати відповіді на запитання з мінімальними витратами часу. Для досягнення цієї цілі були поставлені наступні завдання:

− дослідити визначення та поняття чат-ботів, а також коротку історію їх виникнення;

− провести пошук і порівняння наявних аналогів чат-ботів;

− визначити вимоги до програмного забезпечення;

− розробити та впровадити створене програмне забезпечення.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Програми для інтернет-комунікації, такі як чат-боти, базуються на передових технологіях, зокрема штучному інтелекті, нейронних мережах та аналізі великих даних (Big Data). Завдяки цьому чат-боти можуть забезпечувати спілкування, максимально наближене до людського, що дозволяє ефективно вирішувати різноманітні комерційні та маркетингові завдання.

Термін «чат-бот» походить від двох англійських слів: «to chat», що означає невимушену розмову в Інтернеті, і «bot» (робот), тобто машина. Отже, це роботи, створені для взаємодії з користувачами онлайн. Ці програми також називають віртуальними співрозмовниками або програмами-співрозмовниками, всі ці терміни є синонімами.

На сьогоднішній день в літературі можна знайти численні визначення чат-ботів, але аналіз цих визначень вказує на їхню схожість. Це свідчить про те, що дослідники мають досить чітке уявлення про цю технологію. З точки зору користувачів, спілкування з ботом виглядає як звичайне листування з реальною людиною.

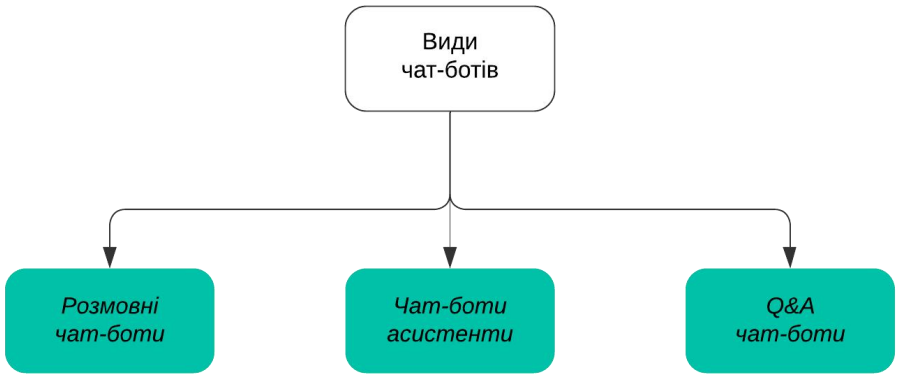


Рисунок 1.1 – Види чат-бот застосунків

Чат-боти − це розумні програми, що працюють у месенджерах і виконують різноманітні завдання. Їх можна класифікувати за різними параметрами: складністю, принципом дії, метою використання та функціональністю. Сучасні чат-боти можуть бути як простими, відповідаючи на базові запити, так і складними, забезпечуючи комплексну підтримку користувачів та інтеграцію з іншими системами (Рисунок. 1.1).

Існують два основних типи чат-ботів залежно від рівня складності:

− Чат-боти, що працюють на основі правил та шаблонів. Вони мають обмежений функціонал, але є досить простими у розробці та можуть ефективно розпізнавати повідомлення та команди;

− Чат-боти, що використовують машинне навчання та мають штучний інтелект. Ці боти мають більший функціонал і краще аналізують отримані повідомлення. Вони постійно вдосконалюються під час взаємодії з користувачами, розширюючи свої можливості. Однак їх розроблення вимагає більше зусиль та ресурсів.

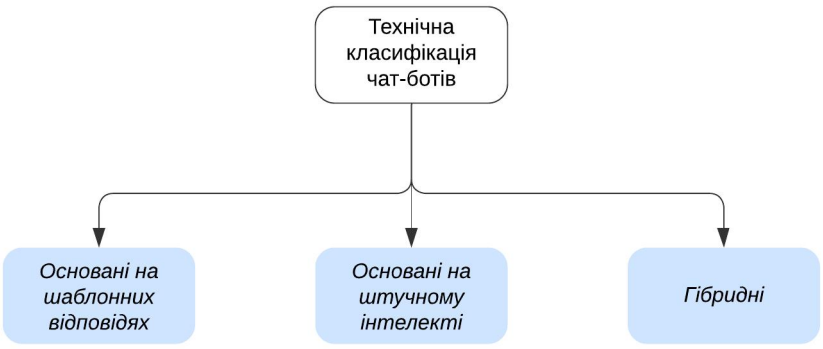


Рисунок 1.2 – Технічна класифікація чат-ботів

Перший тип чат-ботів взаємодіє з користувачами, використовуючи наперед визначені команди та ключові слова. Кожна команда має бути розроблена окремо з використанням регулярних виразів або інших методів аналізу тексту. Якщо користувач висловлює запитання, не враховуючи ключові слова, бот може відповісти повідомленням типу «вибачте, я не зрозумів». Такі боти зазвичай використовуються в ситуаціях, коли потрібна

точна відповідь на певне запитання, вони допомагають у вирішенні конкретних завдань.

Недоліком цих чат-ботів є їх обмежена здатність взаємодіяти з користувачами, оскільки вони реагують лише на визначені шаблони та команди, що створені розробником. Це може обмежувати різноманітність спілкування. Проте, перевагами такого підходу є простота створення та низькі витрати. Ці боти можуть бути корисними в ситуаціях, коли потрібно виконати обмежений набір завдань, таких як відповіді на типові запитання або виконання простих команд пошуку. Їхнє використання може значно полегшити рутинні завдання і покращити користувацький досвід.

Другий тип чат-ботів використовує штучний інтелект, що дозволяє їм спілкуватися більш природною мовою та надавати адекватні відповіді на запити. Ці боти зберігають всі розмови для подальшого аналізу, що сприяє їхньому вдосконаленню.

Основною перевагою таких чат-ботів є їхня функціональність і точність відповідей, які створюють враження живого спілкування. Крім того, завдяки здатності до обробки великих обсягів даних, такі чат-боти можуть забезпечувати персоналізований досвід для кожного користувача. Вони аналізують попередні взаємодії, щоб пропонувати індивідуальні рішення та рекомендації, що підвищує задоволеність клієнтів і їхню лояльність. Впровадження чат-ботів з використанням штучного інтелекту також сприяє ефективнішому управлінню робочими процесами всередині компанії, знижуючи навантаження на працівників і дозволяючи зосередитися на більш стратегічних завданнях. Проте, на їхню розробку потрібно більше фінансових та людських ресурсів.

Зазвичай, чат-боти використовуються для автоматизованого спілкування з користувачами, щоб швидко надавати їм актуальну інформацію, що зменшує потребу в спілкуванні з живим оператором або менеджером компанії (Рисунок 1.3).

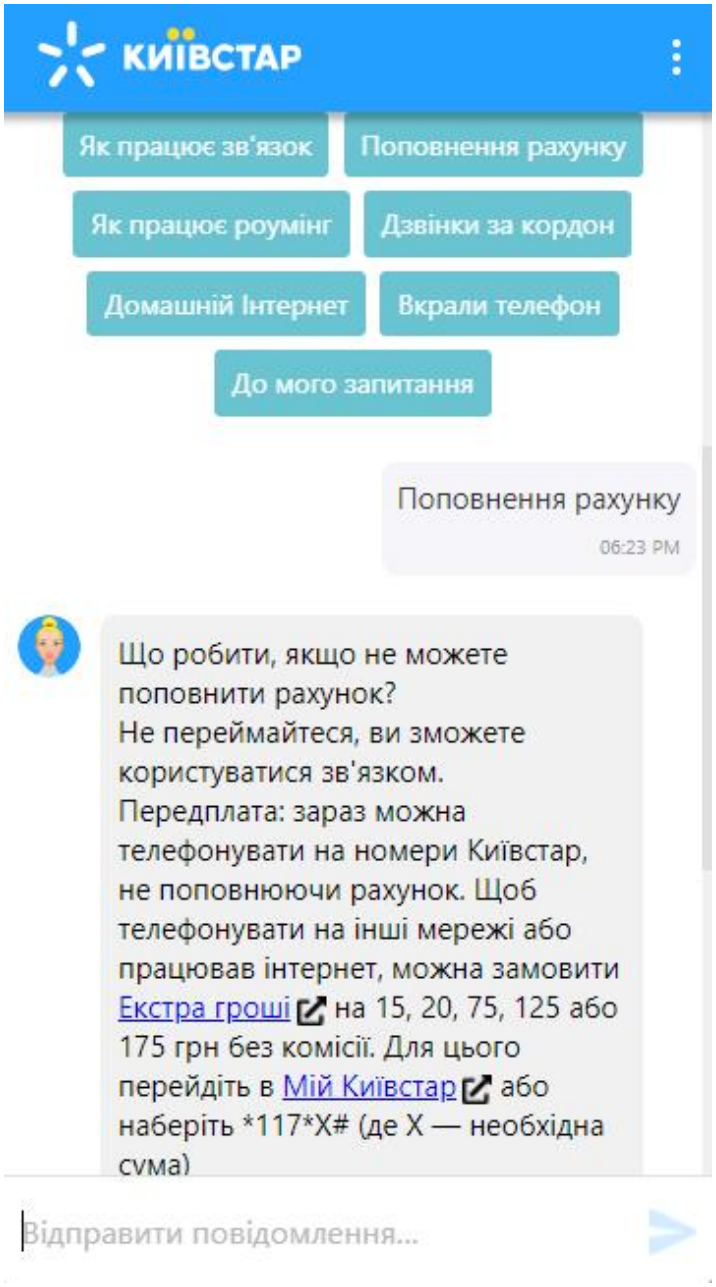


Рисунок 1.3 – Приклад листування з чат-ботом

Чат-бот - це програма, яка працює всередині месенджера, забезпечуючи можливість відповідати на запитання та навіть ставити їх самостійно. Крім цього, чат-боти можуть виступати у ролі цифрових помічників , допомагаючи виконувати різноманітні команди, надавати корисні рекомендації та здійснювати пошук інформації за вказаними параметрами. Також вони можуть бути інтегровані безпосередньо у вебсайти через спеціальні віджети, що розширює їх доступність для користувачів (Рисунок 1.4).

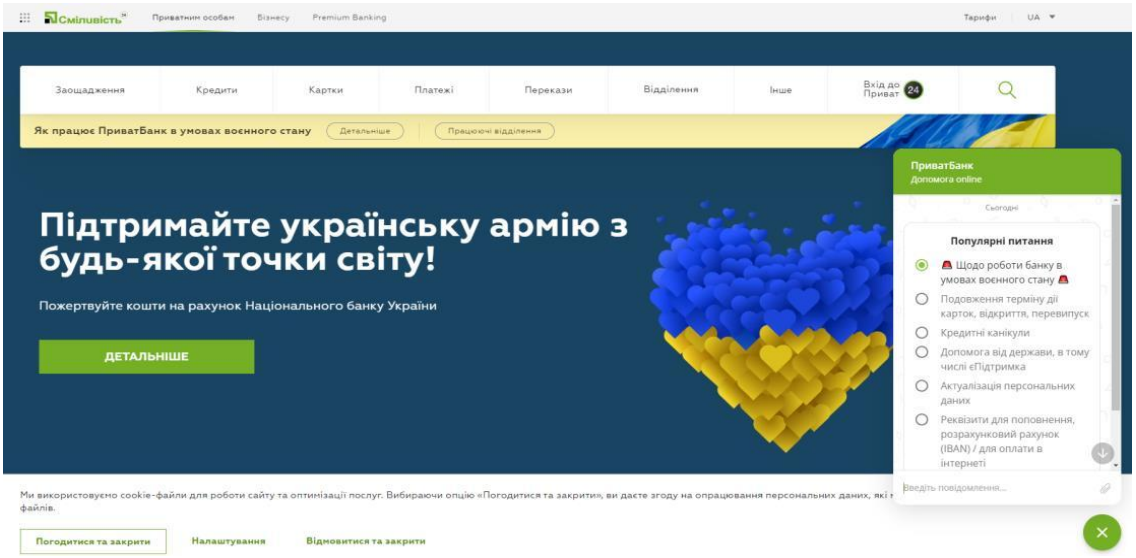


Рисунок 1.4 – Приклад чат-бота в форматі віджету

Отже, чат-бот - це програма, яка автоматизує спілкування з користувачами за допомогою текстових повідомлень. Вона може виконувати різноманітні функції в залежності від своїх можливостей та призначення, такі як надання відповідей на запитання, підтримка клієнтів, обробка замовлень, нагадування про події, рекомендації та оплата. Проте найкраще чат-боти проявляють себе в автоматизації рутинних завдань, забезпеченні швидкої та ефективної клієнтської підтримки, вирішенні типових питань, наданні доступу до інформації про продукти або послуги, а також у виконанні стандартизованих та повторюваних операцій.

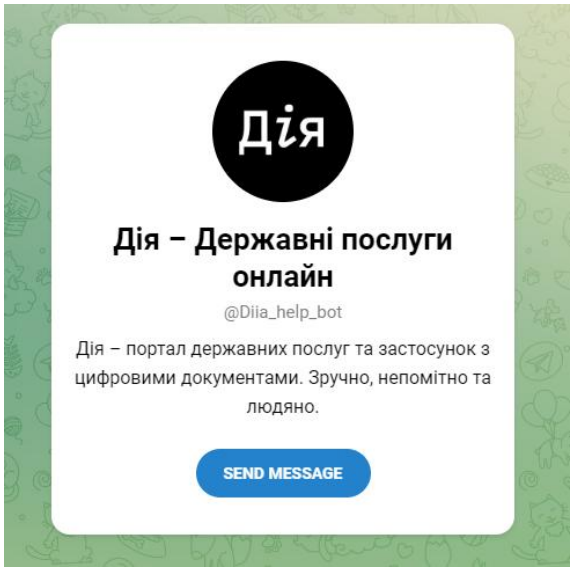


Рисунок 1.5 – Приклад чат-бота державного застосунку Дія

Найкорисніші чат-боти використовують технології машинного навчання, що дозволяє їм навчатися від людей. Ці програми призначені для використання користувачами, тому їх розроблення включає активну участь людей на всіх етапах.

В останні роки парадигма «messaging-as-an-interface» набуває шаленої популярності на тлі збільшення використання чат-ботів. Цей підхід відрізняється своєю простотою, миттєвістю та можливістю спілкуватися з великою кількістю людей одночасно. Зараз багато компаній використовують соціальні мережі як ефективний канал для взаємодії зі своїми клієнтами, що допомагає їм залучати та зберігати аудиторію.

Ще однією причиною популярності є те, що багатьом користувачам зручніше використовувати письмове спілкування, ніж голосове, оскільки воно дозволяє зосередитися на суті питання та швидко отримати відповідь.Це також дає їм можливість проводити розмови у зручний для них час, без очікування вільного оператора або поспішного телефонного дзвінка.

Чат-бот - це програма, що здатна взаємодіяти з багатьма користувачами одночасно, що дає можливість охоплювати широкий аудиторний спектр. Взаємодія з чат-ботом може відбуватися різними способами: через текстові повідомлення, голосові команди, жести та натискання на елементи інтерфейсу, якщо ці можливості підтримуються конкретним ботом.

Протягом останніх кількох років чат-боти стали не лише інструментом обслуговування клієнтів, а й ефективним рішенням для підвищення якості обслуговування та оптимізації бізнес-процесів на підприємствах. Варто відмітити, що різні чат-боти можуть мати різний рівень інтелектуальних можливостей: від базових моделей, що надають стандартні відповіді на запитання, до складних штучно-інтелектуальних систем, які враховують контекст користувача та навчаються на основі його взаємодії, адаптуючись до його унікальних потреб та стилю комунікації.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Чат-бот для факультету інформаційних технологій буде виконано українською мовою. Тому, під час аналізу наявних рішень, було обрано найпопулярніші та найбільш необхідні боти українських компаній.

Однією з найпопулярніших можливостей, які боти можуть надати, є змога спілкуватися та взаємодіяти з іншими мобільними застосунками та сервісами.

Прикладом такого бота є RailwayBot (Рисунок 1.6), який допомагає відслідковувати, знаходити та купувати квитки на поїзд. За допомогою бота можна встановити моніторинг на певний поїзд, дату, час і тип місця. Як тільки з'явиться квиток у наявності, бот надішле сповіщення. Алгоритм дії схожий на придбання квитків на uz.booking.gov.ua. Бот реалізовано на основі кнопок для введення міста відправлення та прибуття.

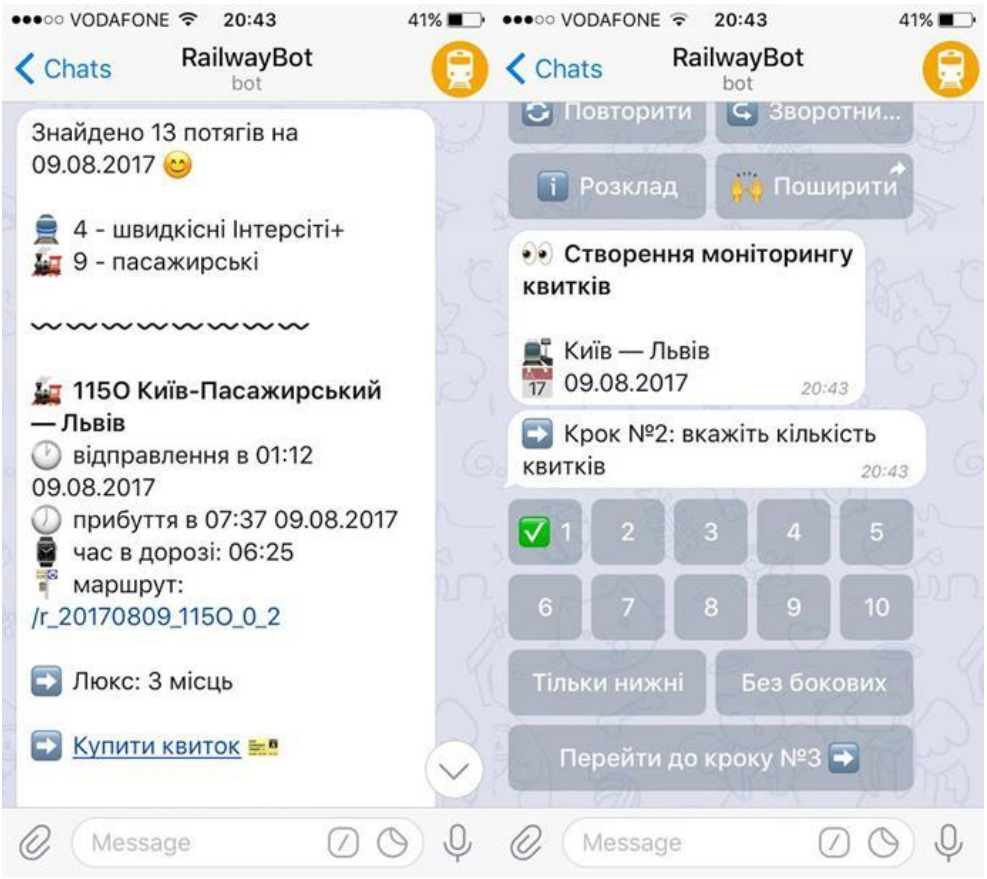


Рисунок 1.6 – RailwayBot

PrivatBankBot (Рисунок 1.7) - бот від ПриватБанку, який спрощує процес здійснення платежів. Це зручний спосіб переказу коштів без необхідності відкривати застосунок. Щоб надіслати кошти іншому користувачу, йому також потрібно мати активованого бота.

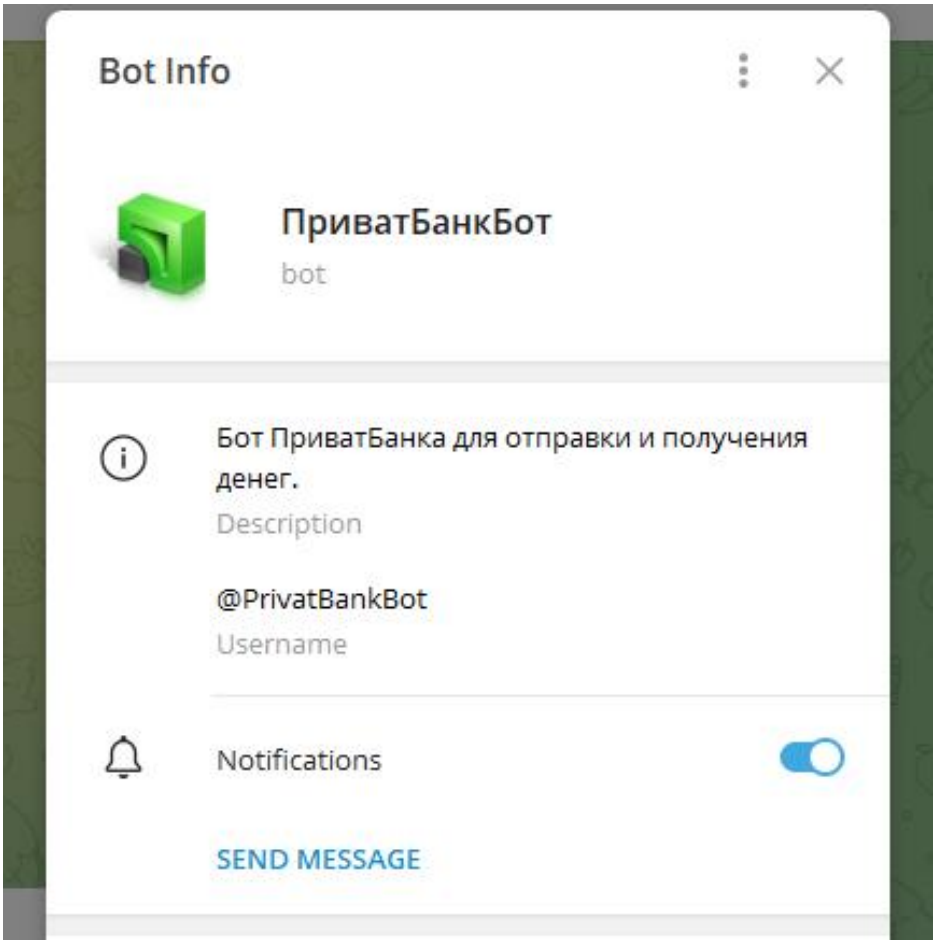


Рисунок 1.7 – PrivatBankBot

NovaPoshtaBot (Рисунок 1.8) - це чат-bot, призначений для контролю та відстеження відправлень, їх статусу та пошуку найближчого відділення.

QuickOsintBot - це bot, що надає можливість отримувати інформацію про людей за допомогою їхніх номерів телефонів, електронних адрес або інших ідентифікаторів. Цей бот широко відомий завдяки своїм можливостям знаходити особисті дані, такі як профілі в соціальних мережах, адреси та іншу публічно доступну інформацію.

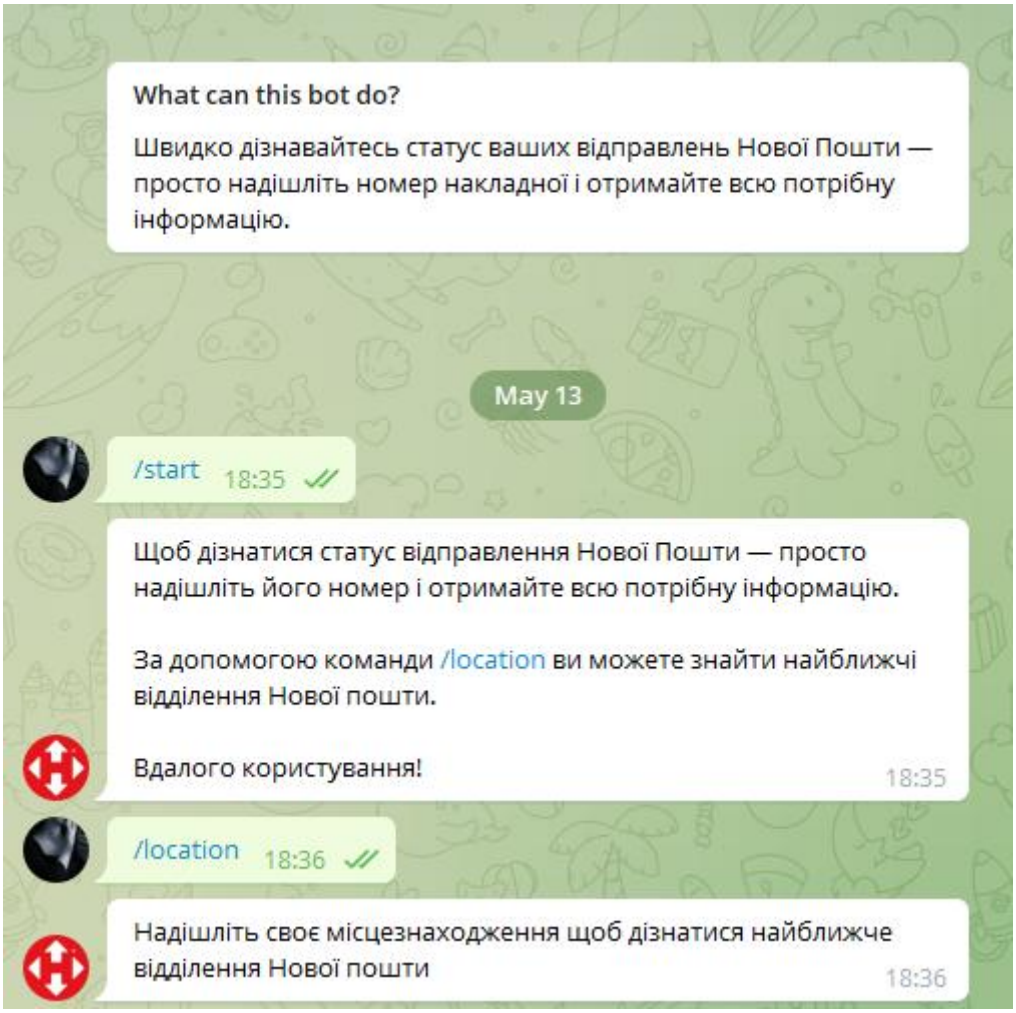


Рисунок 1.8 – NovaPoshtaBot

Оцінивши наявні рішення, було враховано, що швидкість бота має вирішальне значення. Якщо очікування триває кілька секунд, бот втрачає свою перевагу перед пошуком у Google чи відкриттям застосунків. Штучно-інтелектні боти більше нагадують реальну людську комунікацію, тому мають перевагу над скриптовими ботами. Водночас, гібридні боти є більш зручними у використанні. Представлення готових варіантів спрощує комунікацію та зменшує кількість дій, що потрібно виконувати користувачу.

Останні роки відзначаються відновленою популярністю месенджерів, що стали невід'ємною частиною нашого цифрового життя. Месенджер - це програма для обміну повідомленнями, яка дозволяє користувачам обмінюватися текстовими, голосовими та відео повідомленнями у режимі реального часу через інтернет. З появою смартфонів та широкого

розповсюдження мобільних застосунків, месенджери стали популярною альтернативою традиційним текстовим SMS повідомленням, завдяки своїй доступності та можливостям спілкування у різних форматах.

В даний час існує безліч месенджерів для обміну повідомленнями, які задовольняють різноманітні потреби користувачів. Наприклад, такі відомі програми, як Viber та Facebook Messenger, є лише декількома з численних варіантів. Крім них, на ринку існує багато інших застосунків, спрямованих на різні категорії користувачів - від фанатів стікерів до зайнятих фахівців, які надають перевагу аспектам безпеки, а також геймерів, які шукають спеціалізовані платформи для спілкування та гри.

На графіку (Рисунок 1.9) представлено кількість активних користувачів у найпопулярніших месенджерах станом на квітень 2024 року, за даними дослідження сайту statista.com.

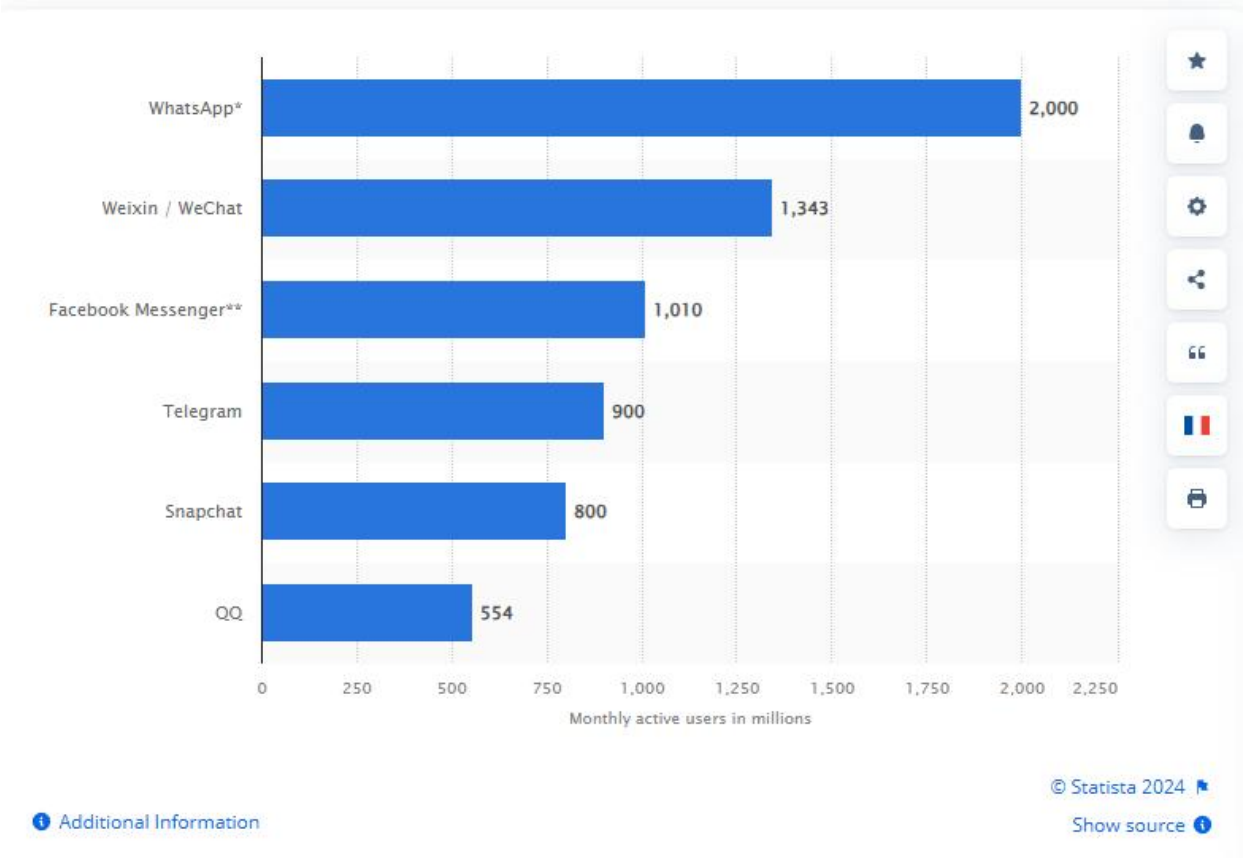


Рисунок 1.9 – Кількість користувачів у найпопулярніших месенджерах

Базовий функціонал майже всіх месенджерів однаковий, оскільки включає в себе надсилання коротких повідомлень. Проте різноманітність починається, коли мова йде про додаткові функції, рівень безпеки та зручність використання. Тому визначити найкращий месенджер складно, оскільки кожен з них має свої переваги.

Найфункціональнішим на території України можна вважати Telegram (Рисунок. 1.10), завдяки його хмарному зберіганню чатів, наскрізному шифруванню, каналами та ботами.

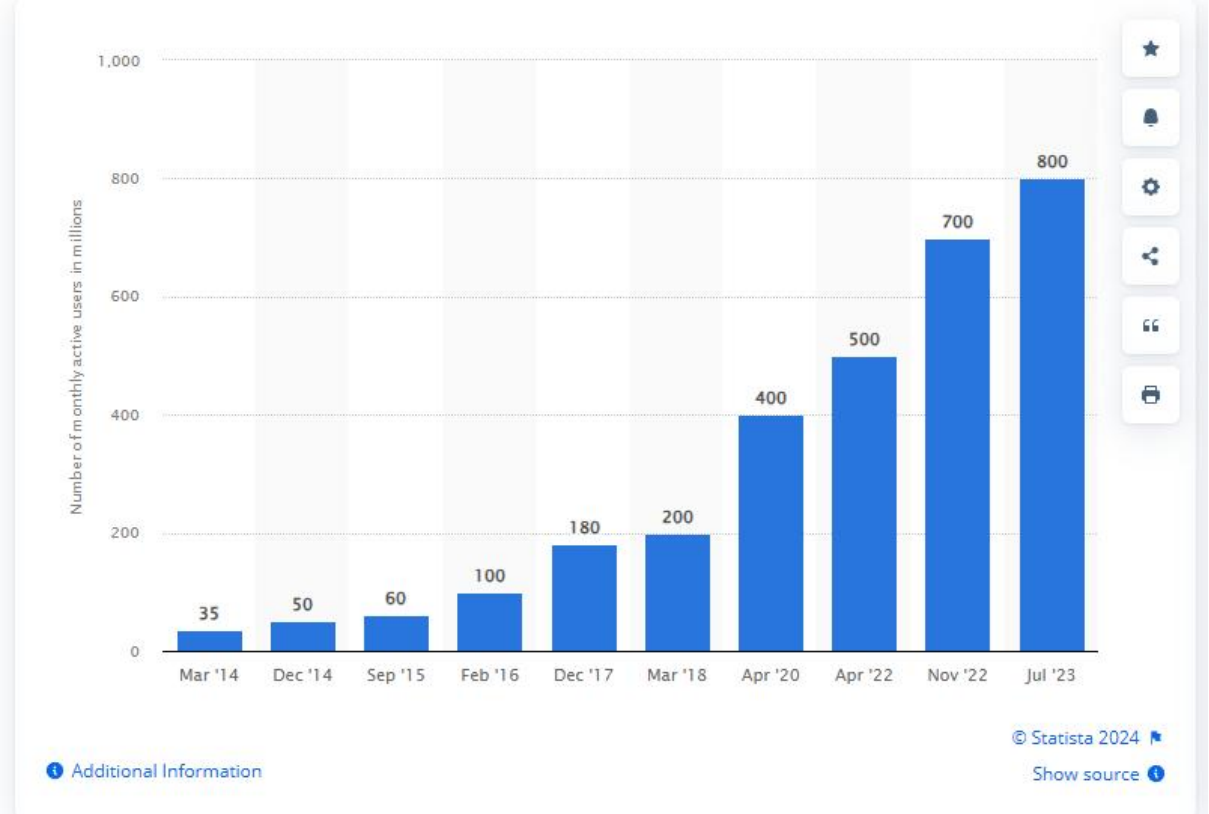


Рисунок 1.10 – Кількість активних користувачів Telegram

Telegram пропонує можливість використання наскрізного шифрування для користувачів. У разі хмарних чатів і груп, дані шифруються прямо між клієнтом і сервером, що перешкоджає провайдерам та іншим третім сторонам у мережі отримати доступ до цих даних.

Користувачі можуть відправляти як текстові, так і голосові повідомлення, здійснювати відео та голосові виклики, а також обмінюватися необмеженою кількістю зображень, документів (обмеження до 2 ГБ на файл),

а також місцезнаходженням. Користувачі також можуть стежити за каналами.

У січні 2021 року кількість активних користувачів Telegram перевищила 500 мільйонів щомісяця. Це був найбільш завантажуваний застосунок у всьому світі.

Безпека - це ще одна перевага Telegram як месенджера. Для захисту облікового запису можна встановити двофакторну автентифікацію, що робить його значно більш надійним у випадку спроби несанкціонованого доступу. Всі дані, що передаються через Telegram, а також вже збережені на його серверах, шифруються. Компанія заявляє, що не передає ключі шифрування нікому, включаючи державні органи безпеки.

Крім того, Telegram пропонує можливість спілкуватися у секретних чатах, де трафік не передається через сервери компанії, а прямує між учасниками безпосередньо. В таких чатах також можна встановити таймер самознищення для повідомлень. Якщо потрібно видалити історію листування, це можна зробити, стираючи як свої, так і повідомлення співрозмовника. Після того, як його пристрій з'єднається з Інтернетом, вони автоматично зникнуть без сліду.

Отже, питання розробки та наукового супроводження чат-бота на платформі Telegram, на поточний час, є актуальним.

1.3 Визначення функціональних та нефункціональних вимог

Функціональні вимоги визначають та описують, які завдання повинна виконувати система та які функції має надавати користувачам. Після аналізу обраної теми були визначені наступні функціональні вимоги:

– Розклад занять: Чат-бот повинен надавати можливість студентам переглядати розклад занять на певний день. Графік повинен бути доступний для кожної групи студентів і відображати інформацію про аудиторії, викладачів і час проведення занять.

– Оповіщення про зміни в розкладі: Чат-бот має надсилати сповіщення

студентам про будь-які зміни в розкладі занять, такі як скасування занять, зміна аудиторії або часу проведення. Це допоможе студентам бути в курсі оновленої інформації про їхні заняття.

– Підтримка різних форматів запитів: Чат-бот повинен бути здатний розуміти різні формати запитів від користувачів, такі як текстові запити або навігація по кнопкам. Це забезпечить зручність користувачам у використанні чат-бота.

– Закріплення курсу та групи: Студент повинен мати можливість вибрати і закріпити за собою курс та групу. Це дозволить отримувати персоналізований розклад швидко та зручно.

– Розсилка розкладу занять: Чат-бот повинен надсилати нагадування студентам за певний час до початку заняття (наприклад, за годину або за день). Це допоможе студентам не пропустити заняття. Також бот повинен надсилати студентам розклад занять на поточний день. Повідомлення повинні містити інформацію про предмет, час і місце проведення занять, а також ім’я викладача.

Для повного опису системи недостатньо лише функціональних вимог, тому потрібно брати до уваги нефункціональні вимоги, які глобально поділяються на наступні категорії:

– практичність. Відповідає за те, щоб програмне забезпечення було простим та легким у використанні для будь-якого користувача, не завдавало зайвих труднощів та було інтуїтивно зрозумілим;

– надійність. Відповідає за роботу програмного забезпечення, та описує як система повинна себе поводити у разі помилок чи збоїв в роботі та зберегти важливі дані навіть у цьому випадку;

– продуктивність. Визначає характеристики системи під час взаємодії з користувачами, тобто наступні характеристики - відповідає за оптимальний час на відповідь запита користувача, кількість користувачів, яку одночасно може прийняти система та визначає які системні ресурси їй на це знадобляться;

– можливість обслуговування. Означає можливість легко модифікувати, змінювати програмне забезпечення для введення нового функціоналу або з метою виправлення помилок.

Виходячи з цього можна визначити необхідні нефункціональні вимоги до системи:

– система повинна бути надійною та доступною для використання користувачами в будь-який момент часу;

– в разі збоїв в роботі програмне забезпечення повинно адекватно реагувати на це та намагатись самостійно відновити роботу;

– в разі помилок важливе збереження інформації в базі даних;

– система має бути побудована таким чином, щоб в майбутньому для неї було легко створювати нові модулі та функції або виправляти код з метою покращення;

Визначивши функціональні та нефункціональні вимоги, побудуємо діаграму варіантів використання (Рисунок 1.11).

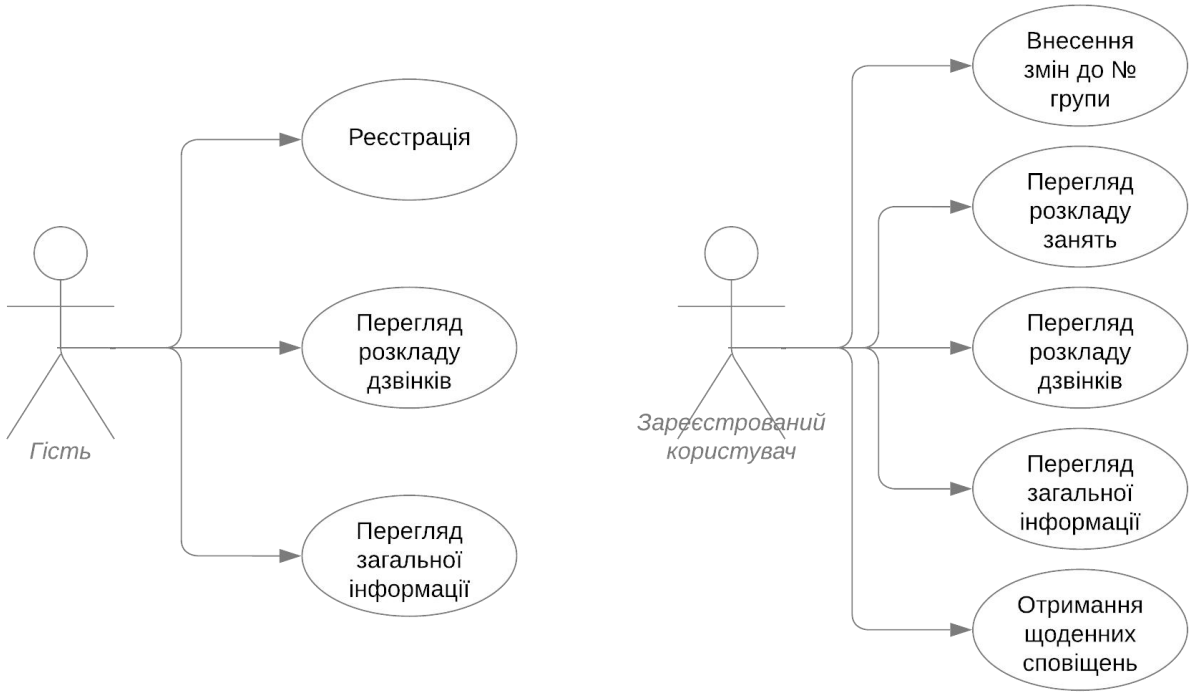


Рисунок 1.11 – Діаграма варіантів використання

1.4 Постановка задачі

Головним завданням кваліфікаційної роботи було створення чат-бота на основі месенджера Telegram, призначеного для інформування студентів про розклад занять. Перед початком проєктування та розробки телеграм-бота необхідно встановити основні вимоги та їх детальний опис. У попередньому розділі була проведена аналіз предметної області та розглянуті наявні аналоги, щоб переконатися в доцільності розробки.

Сучасні методи навчання та доставки інформація до кінцевого споживача постійно вимагають досконалостей у будь-яких сферах життя, починаючи від продуктових магазинів і закінчуючи великими державними підприємствами. Те, що було актуально 5 років тому, вже не користується популярністю, або втрачає аудиторію. Донесення актуальної інформації до студентів у стислі терміни, а саме зміна в розкладі, великий захід (добровільне здавання крові або день народження університету) та інше, вимагає негайного оповіщення. Єдиний спосіб доставки інформації до студента не завжди зручний, а часто забирає більше часу, ніж справді могло б. Тому одним із головних завдань у цій галузі є проблема актуалізація подачі інформації під сучасні платформи зв'язку в суспільстві.

Чим більше студентів вступають і навчаються, тим важливіше охопити всю аудиторію за короткий час. Студенти, через свій вік, активно освоюють і використовують нові технології, зокрема месенджер Telegram. Оскільки не всі студенти постійно перевіряють сайти університету та офіційні джерела, виникає потреба в централізації всієї необхідної та важливої інформації.

Отже, метою даної роботи є створення проєкту для інформаційної підтримки студентів на базі месенджера Telegram. Розроблена система повинна забезпечити зручний сервіс для студентів, які навчаються в університеті.

Виходячи з поставлених вище цілей, потрібно розробити Telegram-bot, який:

– буде інформувати про будь-які зміни розкладу та формат зберігання цих даних;

– зберігатиме отримані дані у спеціально створеній базі даних, оптимізованій для такого зберігання;

– певним чином зберігатиме дані кожного користувача та розклад занять кожної підгрупи;

– дозволятиме відстежувати поточний час та налаштовувати сповіщення для користувачів бота;

– надаватиме інформацію про розклад дзвінків.

1.5 Висновки до розділу 1

Перший розділ кваліфікаційної роботи присвячений проблемам проєкту, які вирішує розроблення чат-боту. Детально розглянуто наявні види ботів, їх поширення та використання. Вивчено приклади успішних рішень та шляхи їх реалізації.Також перший розділ акцентує увагу на алгоритмах, покладених в основу чат-ботів. Описано принципи, які є основними для створення чат-ботів. Проведено аналіз для вибору алгоритму за яким буде створюватись продукт. Розглянуто який результат буде досягнуто за допомогою обраних принципів.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

На початку розробки будь-якої системи або програмного забезпечення кожен розробник аналізуючи висунуті вимоги до проєкту має обрати стек технологій та засобів для реалізації продукту.

На цей вибір можуть впливати багато факторів, але з них можна виділити деякі основні, а саме:

– насамперед це цілі та вимоги до самої системи, наприклад – операційні системи, які повинні підтримувати програму та бути з нею сумісними, швидкість роботи, надійність, масштабованість, тощо. Розробник аналізує їх та виділяє технології, за допомогою яких можливо досягти потрібного результату;

– не менш важливий фактор це вміння, навички та досвід роботи розробника з технологіями виділеними з попереднього пункту. Зрозуміло, що при рівних умовах завжди надають перевагу технологіями в яких є більший досвід, тому що це впливає на швидкість та якість створення програмного забезпечення;

– також до уваги потрібно брати актуальність використання технології в момент розробки продукту та враховувати її перспективи в майбутньому, тому що використання застарілих технології може привести до нестабільної роботи системи через декілька років.

2.1 Аналіз та проєктування архітектури системи

Архітектура системи - це високорівневий план або концептуальна модель, що описує структуру, компоненти, взаємозв'язки та принципи організації системи. Це визначає загальну конфігурацію системи та її складові частини, що визначається на ранніх етапах розробки для забезпечення її ефективності, надійності та розширюваності.

Архітектура чат-ботів для Telegram включає ряд компонентів:

– Клієнт Telegram: Користувачі взаємодіють з ботом через клієнт Telegram, який надає інтерфейс для відправки повідомлень та отримання відповідей.

– Сервер: На віддаленому сервері запускається програмне забезпечення, яке обробляє запити від клієнта Telegram. Цей сервер містить код бота і відповідає за обробку вхідних повідомлень та відправку відповідей.

– API Telegram: Telegram надає API, який дозволяє розробникам створювати, налаштовувати та взаємодіяти з ботами. API надає методи для відправки та отримання повідомлень, управління підписниками, а також для роботи з різними функціями ботів.

– Бізнес-логіка бота: Це код, який визначає, як бот буде обробляти вхідні запити та генерувати відповіді. Це може включати обробку команд, аналіз текстових повідомлень, виконання дій на основі вхідної інформації тощо.

– Сховище даних (за бажанням): У деяких випадках може знадобитися зберігання даних, таких як інформація про користувачів, історія повідомлень тощо. Для цього можна використовувати різні бази даних або інші методи зберігання даних.

– Додаткові сервіси (за бажанням): Залежно від функціональності бота можуть знадобитися додаткові сервіси, такі як сервіси обробки природної мови, сервіси сповіщень тощо.

Загалом, архітектура чат-ботів для Telegram ґрунтується на клієнт-серверній моделі з використанням API Telegram для взаємодії між клієнтом (користувачем) і сервером (ботом).

Клієнт-серверна архітектура - це модель розподіленої обчислювальної системи, в якій задачі розбиваються на дві основні складові частини: клієнти, які надсилають запити, та сервери, які надають відповіді. Клієнти і сервери зазвичай взаємодіють через мережу, таку як Інтернет. На рисунку 2.1 зображена схема архітектури:

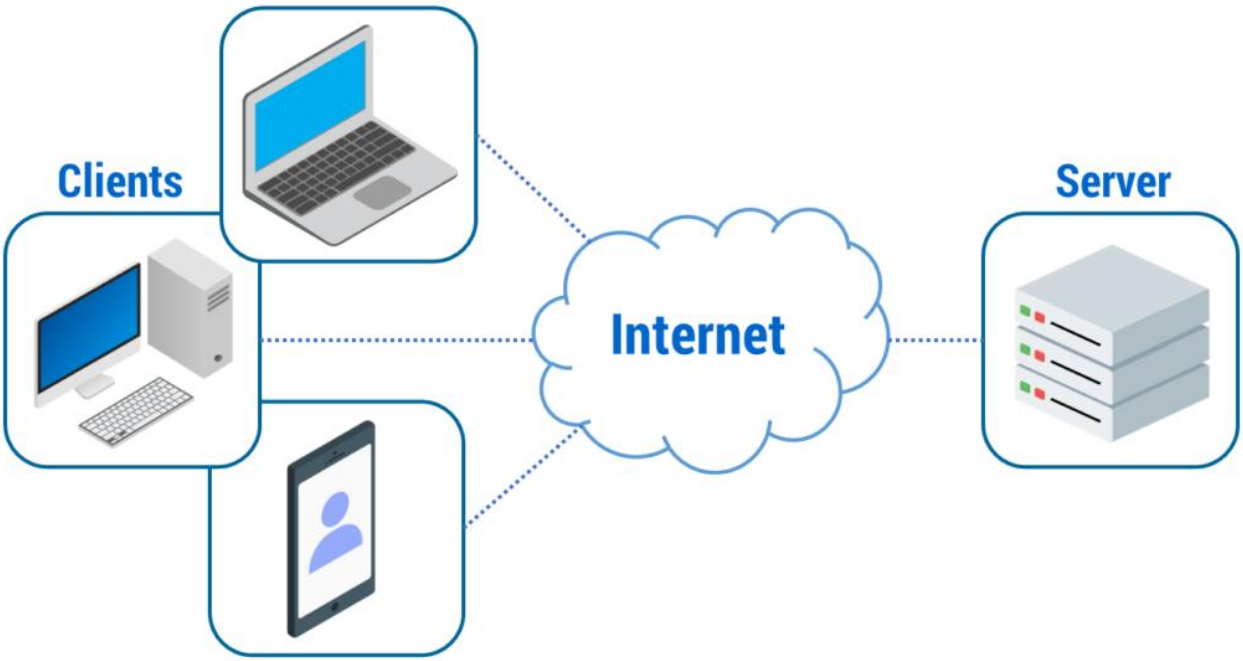


Рисунок 2.1 – Схема клієнт-серверної архітектури

Основні типи архітектури клієнт-сервер включають:

– Централізована архітектура;

– Розподілена архітектура;

– Тонкий клієнт - товстий сервер;

– Сервер-клієнт з розділенням по рівнях (Layered Architecture);

– Клієнт-сервер з використанням кешування.

Ці типи архітектур можуть бути використані в залежності від потреб та характеристик конкретного проєкту. Кожен з них має свої переваги та недоліки, які потрібно враховувати при проєктуванні системи.

Дотримання вказаного архітектурного стилю (клієнт-сервер) дозволить створити систему, яку буде легко оновлювати та масштабувати, змінювати без вимикання системи.

2.2 Проєктування бота на основі стандартів IDEF0 та UML

Метод IDEF0 дозволяє розробнику зобразити уявлення про процес,

входи (I), елементи керування (C) над процесом, виходи (O) та механізми (M), що діють на процес (вони разом називаються ICOM).

Кожна з чотирьох сторін функціонального блоку має своє певне значення (роль), при цьому:

− верхні стрілки являють собою елементи управління;

− ліва сторона є вхідними значеннями, які провокують початок процесу;

− права сторона є вихідними значеннями, тобто результат процесу в кінці;

− нижня сторона означає механізми, які використовуються для виконання завданого процесу.

На рисунку 2.2 зображено вигляд контекстної діаграми процесу розробки бота в Telegram.

Відповідно до контекстної діаграми має сенс розробити діаграму декомпозиції моделі. Діаграма декомпозиції існує для більш детального розгляду процесу, зазначеного в контекстній діаграмі, за рахунок розбиття головного процесу на додаткові процеси. Створення діаграми декомпозиції дає можливість описати процес розробкия бота. Це встановить розуміння і порядок виконання дії, що допоможе як у майбутньому процесі розробки, так і для розгляду “замовника”, щоб визначитися з термінами виконання завдань.

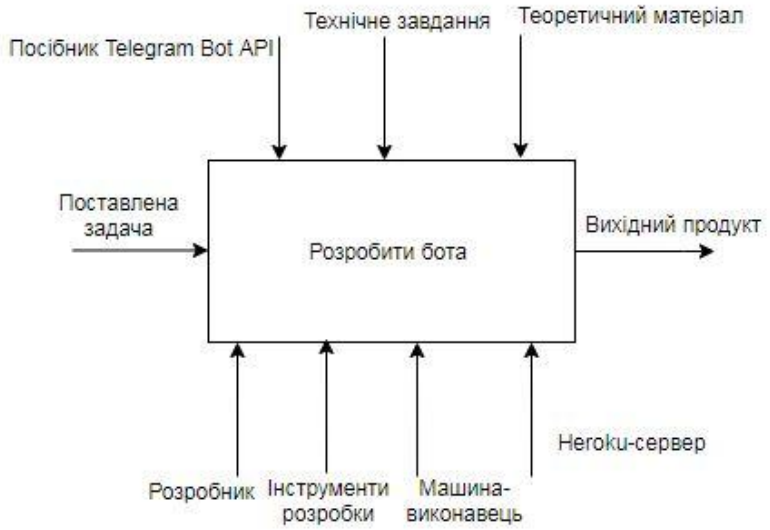


Рисунок 2.2 – Контекстна діаграма процесу розробки бота

Оскільки в контекстній діаграмі вказаний процес розроблення бота, то діаграма декомпозиції буде описувати, як ті чи інші фактори та механізми впливають на різні процеси в ході виконання роботи. Представлення діаграми декомпозиції зображено на Рисунок 2.3.

Діаграма розбиває розроблення боту на такі основні етапи.

− Реєстрація бота у Telegram.

− Процес розробки.

− Тестування бота.

Після успішного проведення тестування робота вважається виконаною. Так як діаграма декомпозиції побудована та визначено головні етапи розробки, далі було розроблено діаграму взаємодії.

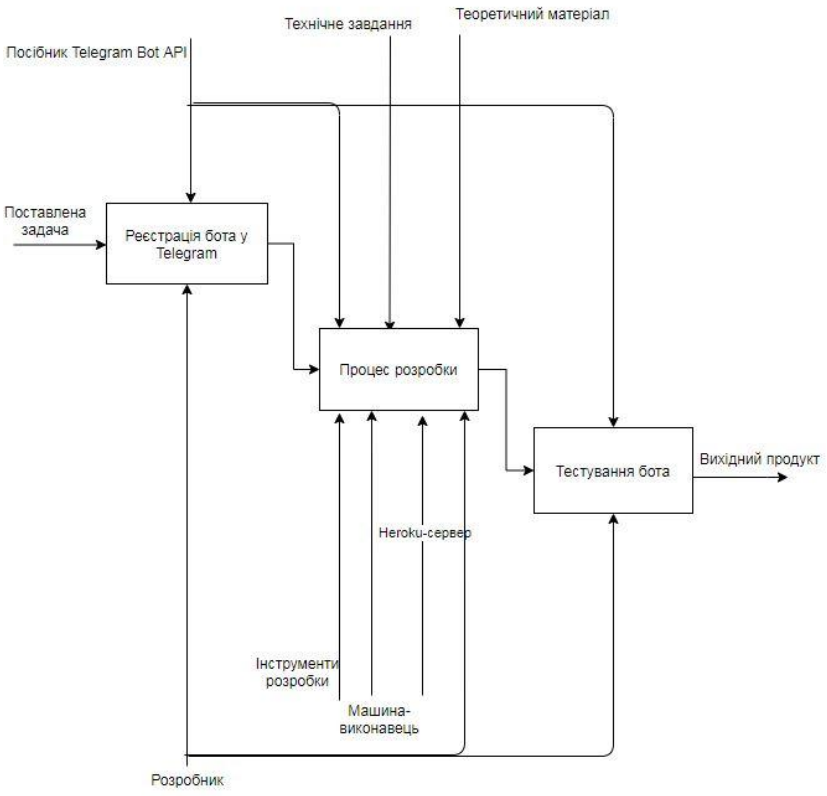


Рисунок 2.3 – Діаграма декомпозиції розробки бота

Наступна діаграма зображує яким чином бот перевіряє наявність нових зв’язків у системі. Для того, щоб бот відповів на повідомлення користувача йому необхідно час від часу звертатися до інтерфейсу, та перевіряти нові запити користувачів. Діаграма перевірки оновлень бота зображена на Рисунок 2.4.

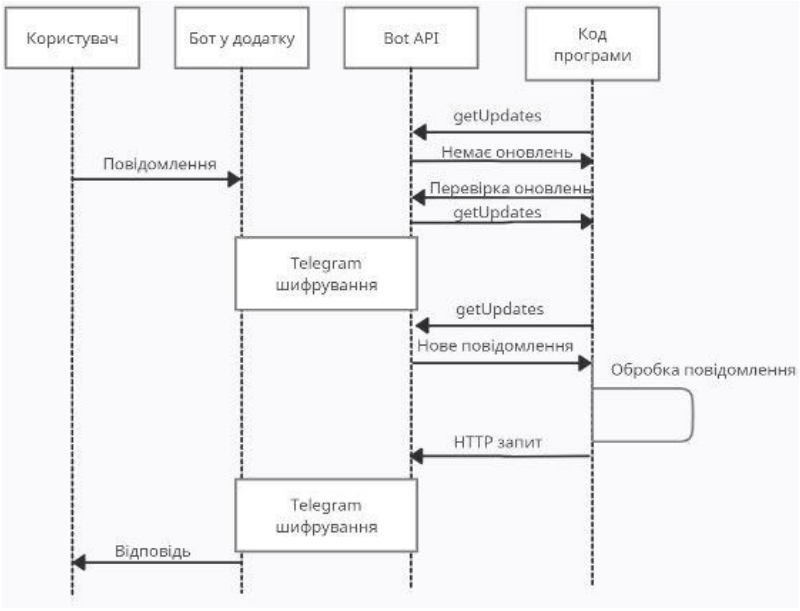


Рисунок 2.4 – Перевірка нових повідомлень

2.3 Аналіз та вибір типу бази даних, проєктування структури бази даних

База даних (БД) — це організована сукупність даних, яка зберігається та керується таким чином, що забезпечує легкий доступ, управління та оновлення цих даних. Бази даних використовуються для зберігання інформації, яка може бути використана для різноманітних цілей, включаючи зберігання інформації про користувачів, транзакції, товари, документи тощо.

Перелік основних типів баз даних:

− Реляційні бази даних (Relational Databases)

− Мережеві бази даних (Network Databases)

− Ієрархічні бази даних (Hierarchical Databases)

− Об'єктно-орієнтовані бази даних (Object-Oriented Databases)

− Бази даних з відкритим кодом (Open Source Databases)

− Розподілені бази даних (Distributed Databases)

− Хмарні бази даних (Cloud Databases)

Для чат-бота розкладу занять в університеті найбільш підходящим типом бази даних є реляційна база даних (Relational Database). Ось чому:

Причини вибору реляційної бази даних:

1. Структурованість даних:

− Розклад занять включає добре структуровані дані, такі як назви предметів, викладачі, аудиторії, дати та час занять. Реляційні бази даних дозволяють легко організувати ці дані у вигляді таблиць з чіткими відношеннями.

2. Нормалізація:

− Реляційні бази даних підтримують нормалізацію, що дозволяє зменшити дублювання даних та забезпечити цілісність даних. Наприклад, таблиці для студентів, викладачів, курсів та розкладу можуть бути зв'язані між собою за допомогою зовнішніх ключів.

3. SQL:

− Використання SQL (Structured Query Language) для запитів та маніпулювання даними забезпечує гнучкість у отриманні необхідної інформації. Це особливо корисно для складних запитів, таких як отримання розкладу для конкретного студента чи викладача.

4. Масштабованість та продуктивність:

− Сучасні реляційні бази даних, такі як PostgreSQL чи MySQL, добре масштабуються і можуть ефективно обробляти великі обсяги даних та численні запити від користувачів.

Альтернативи та їх недоліки:

1. NoSQL бази даних:

− Документно-орієнтовані (MongoDB) або ключ-значення (Redis) бази даних можуть бути корисні для деяких специфічних застосувань, але вони не забезпечують тієї ж ступені нормалізації та підтримки складних запитів, як реляційні бази даних.

− Вони більше підходять для гнучких та неструктурованих даних.

2. Об'єктно-орієнтовані бази даних:

− Можуть бути занадто складними для відносно простих структур даних, як розклад занять.

3. Ієрархічні та мережеві бази даних:

− Ці типи баз даних менш поширені та складні у використанні для більшості сучасних застосувань.

Висновок: Реляційна база даних є оптимальним вибором для чат-бота розкладу занять в університеті завдяки своїй структурованості, підтримці складних запитів, нормалізації та можливостям забезпечення цілісності даних.

Нижче наведена схема бази даних (Рисунок 2.5). Вона складається з 7-ти таблиць.

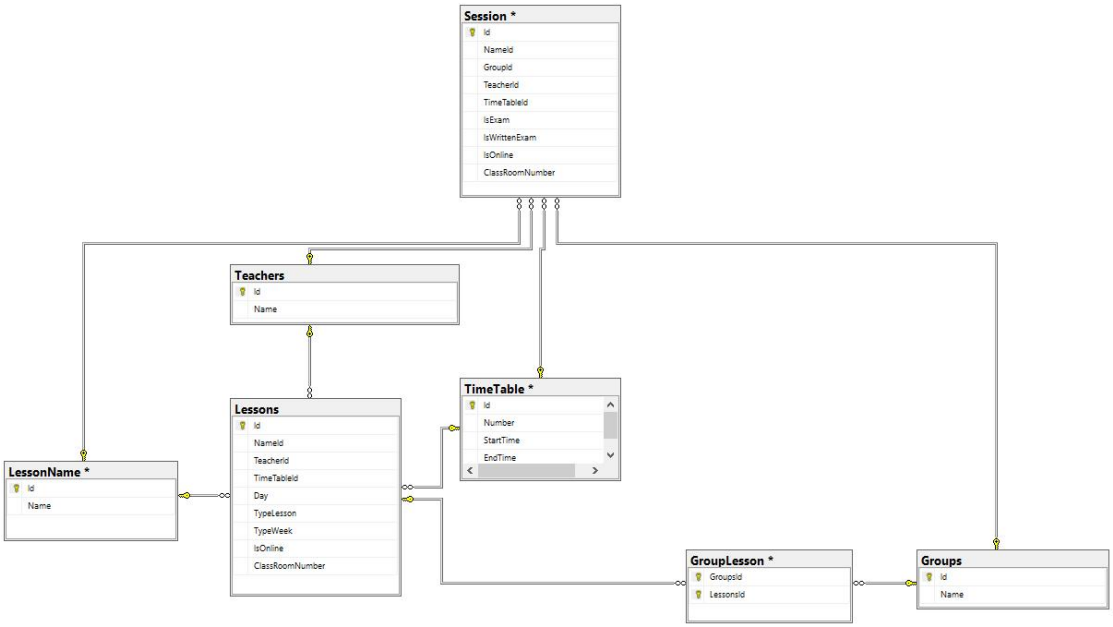


Рисунок 2.5 - Схема бази даних

2.4 Аналіз та вибір технологій і методів реалізації системи

Перед вибором інструментів розробки, слід познайомитися з прикладним програмним інтерфейсом (API), що пропонує месенджер для розробників. Наразі доступні два головних інструменти API для використання сервісів Telegram: Telegram Bot API і Telegram API. Перший призначений для розробки чат-ботів, другий дозволяє створювати повністю настроювані телеграм-клієнти. Telegram Bot API є надбудовою над Telegram

API, тому використання Bot API можливе без глибокого розуміння протоколу MTProto, який використовується.

Для його функціонування використовується проміжний сервер з HTTPS інтерфейсом, який шифрує трафік і забезпечує зв'язок з Telegram API. Bot API спрощує розробку програм, які використовують інтерфейс Telegram для виконання коду на локальному сервері. Користувачі можуть взаємодіяти з ботами, надсилаючи їм повідомлення, команди та вбудовані запити.

Робота будь-якого бота полягає у тому, що він постійно відправляє запити на сервер і регулярно отримує оновлення. Це можна зробити двома способами. По-перше, застосовуючи вебхуки, коли сервер робить зворотний виклик на визначений URL. По-друге, можна просто надсилати запити до Telegram і отримувати постійні відповіді.

Зверніть увагу - отримувати повідомлення про нові події в роботі та інші події можна лише один раз. Тому, якщо дані чату вважаються важливими, вам треба самостійно зберігати список чатів та історію повідомлень. Якщо ви випадково втратите цю інформацію, відновити її буде неможливо.

У відміну від Bot API, де отримувати оновлення можна лише один раз, у Telegram API це обмеження можна обійти, використовуючи кілька клієнтів. У такому випадку bot отримуватиме всі оновлення на кожному із запущених клієнтів. Крім того, у Bot API немає можливості надсилання повідомлень всім користувачам одночасно.

2.4.1 Вибір мови програмування

Першим пунктом вибору програмних засобів вважається мова програмування. Мова програмування – це мова, яка створена для взаємодії з комп’ютером шляхом написання команд і їх виконання безпосередньо на машині.

Бота в застосунку Telegram можна написати будь-якою мовою, якою володіє розробник.

Сьогодні, коли обчислювальні потужності персональних комп'ютерів і серверів стали досить високими, з'явився величезний попит на інтерпретовані мови програмування. Адже крім запуску самої програми необхідний запуск інтерпретатора, що вимагає додаткових ресурсів. Саме такою мовою і є Python.

Python - це мова програмування, створена для взаємодії з комп’ютером шляхом написання команд і їх виконання. Python дуже популярна серед розробників завдяки своїм численним перевагам.

Переваги Python:

− Гнучкість: Python дозволяє розробникам легко розширювати функціональність і адаптувати код до різних потреб.

− Можливість розширення: Завдяки величезній кількості бібліотек і фреймворків, Python можна використовувати для вирішення будь-яких завдань.

− Простота синтаксису: Чистий і зрозумілий код без зайвих дужок і виразів.

− Інтерпретованість: Інтерпретатор Python існує для всіх популярних платформ і за замовчуванням входить у більшість дистрибутивів Linux.

− PEP (Python Enhancement Proposal): Єдиний стандарт для написання коду, який підтримує його читабельність і підтримуваність.

− Open Source: Код інтерпретатора Python є відкритим, що дозволяє будь-кому брати участь у його розвитку.

− Ком'юніті: Дружнє і підтримуюче ком'юніті, яке допомагає вирішувати проблеми.

Недоліки Python:

− Продуктивність: Як інтерпретована мова, Python поступається в продуктивності іншим мовам, але це можна компенсувати використанням C реалізацій для проблемних ділянок коду.

− Синтаксис: Може бути незвичним для розробників, які переходять з інших мов програмування.

− Динамічна типізація: Вимагає більше ресурсів через внутрішнє кешування.

− Global Interpreter Lock (GIL): Обмежує продуктивність багатопотокових програм.

Python залишається однією з найбільш затребуваних мов програмування завдяки своїй гнучкості, простоті та потужності. Вона активно використовується у веброзробці, Machine Learning, Data Science та багатьох інших сферах.

Враховуючи вищезазначене, можна стверджувати, що всі недоліки мови з залишком нівелюються його перевагами, які набагато вагоміші в сьогоднішніх реаліях.

Python — це відмінна мова програмування як для навчання, так і для реальної розробки. Він допомагає вирішити величезний спектр завдань!

2.4.2 Вибір середовища розробки

Середовище розробки (IDE, Integrated Development Environment) – програмне забезпечення, що містить в собі можливості для реалізації написання коду, автоматизації програм.

При виборі середовища розробки важливо зробити акцент на орієнтованість щодо мови програмування. Адже чим більший функціонал надає середовище розробки, тим більший потенціал стосовно реалізації бота, і все залежить лише від фантазії і ідей, як саме складати програму.

При виборі IDE також важливо враховувати його підтримку та активність спільноти розробників. Популярні середовища мають активні форуми, блоги, та регулярні випуски оновлень, що дозволяє отримувати швидку підтримку та вирішення проблем.

Далі розглянуто деякі середовища розробки (Рисунок 2.6).

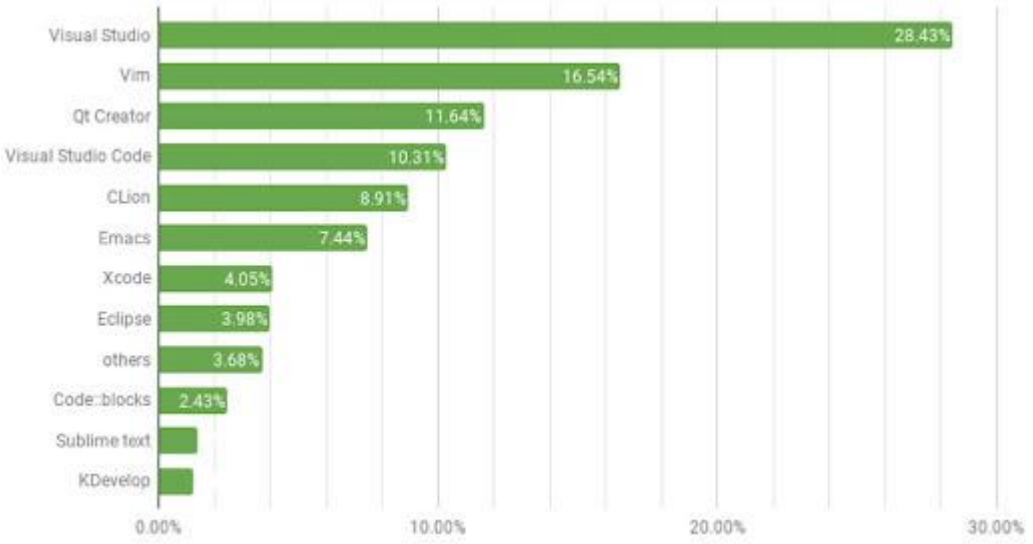


Рисунок 2.6 – Найпопулярніші IDE

1. Visual Studio Code

− Visual Studio Code є одним з найпопулярніших та найбільш широко використовуваних середовищ розробки на сьогодні. Це безкоштовне інтегроване середовище розробки, яке підтримує багато мов програмування, включаючи JavaScript, TypeScript, Python та багато інших. Він має розширення, що дозволяють розширити його функціональність, що забезпечує широкі можливості для розробки.

2. IntelliJ IDEA

− IntelliJ IDEA є іншим дуже популярним середовищем розробки, особливо для Java-розробки. Воно підтримує багато мов програмування, включаючи Java, Kotlin, Groovy, Scala та багато інших. IntelliJ IDEA має потужну систему рефакторингу, що дозволяє легко змінювати код та покращувати його.

3. Eclipse

− Eclipse є одним з найбільш популярних середовищ розробки для Java-розробки. Воно підтримує багато мов програмування, таких як C++, Python, PHP та інших. Eclipse має широкий набір інструментів для аналізу коду та підтримує системи контролю версій, такі як Git.

4. Atom

− Atom є безкоштовним відкритим середовищем розробки, який був розроблений компанією GitHub. Воно має широкі можливості для налаштування та підтримує багато мов програмування, включаючи JavaScript, HTML, CSS, Python та інші.

5. Sublime Text

− Sublime Text є дуже швидким та легким середовищем розробки з великою кількістю плагінів та розширень. Це дуже популярне середовище розробки для JavaScript, а також підтримує Python, HTML, CSS та інші мови програмування.

6. Aptana Studio

− Aptana Studio є відкритим та безкоштовним середовищем розробки для веброзробки, що підтримує багато мов програмування, таких як HTML, CSS, JavaScript, PHP, Ruby та інших. Воно має потужний інтерфейс зі зручними інструментами для редагування та аналізу коду.

Проаналізувавши різноманітні середовища розробки мов програмування, які зараз активно використовуються програмістами, можна зазначити, що кожне з них має свої сильні та слабкі сторони. Враховуючи особливості даного проєкту, прийшли до висновку, що для нього найзручнішим вибором буде «Visual Studio Code».

2.4.3 Вибір серверної платформи для доступу до бота

Для підтримки функціонування бота, щоб він мав можливість одразу взаємодіяти з користувачами, та відповідати на їх запити стає необхідним його розміщення на сервері. З цією функцією вдало впорається ресурс Heroku. Сервіс Heroku має власну систему контролю версій Git, а також надає доступ до хмарних обчислень.

Для запуску бота на сервер даного сервісу необхідно встановити HerokuCLI та провести операцію зі входу до особистого кабінету за допомогою команди «herokulogin». Після чого ввести ще кілька команд для

перенесення деяких файлів з сервера на комп'ютер. При вдалому встановленні зв'язку між сервером та виконавчою машиною, у терміналі PyCharm з'явиться повідомлення про вдале підключення. У результаті чого користувачі бота не будуть чекати довгої відповіді на запити.

2.4.4 Вибір системи управління базами даних

SQLite - це легка вбудовувана реляційна система управління базами даних, яка не потребує окремого серверу. Це програмна бібліотека, яка надає доступ до бази даних за допомогою стандартного мовного інтерфейсу SQL. SQLite надає значний підмножину функцій SQL Server, такі як створення таблиць, вставка, оновлення та видалення даних, індексація, транзакції тощо.

Подібно до SQL Server, SQLite також працює з мовою запитів SQL, але не має окремого сервера баз даних. Вона просто зберігає базу даних в одному файлі на диску. SQLite також доступний на різних платформах, включаючи Windows і Linux, і може бути легко вбудованим в різні застосунки.

Хоча архітектура SQLite і відрізняється від SQL Server, вони обидва працюють з SQL і можуть виконувати багато подібних завдань. Нижче наведена архітектура SQLite (Рисунок 2.7).

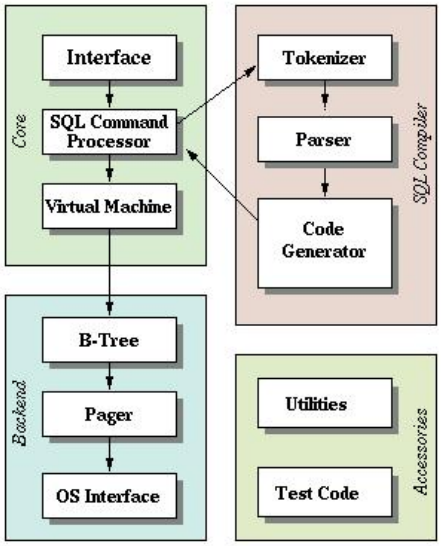


Рисунок 2.7 – Архітектура SQLite

Основний компонент SQLite - це SQLite Core. Він представляє собою ядро SQLite, яке включає в себе всі основні варіанти роботи з базами даних. SQLite Core забезпечує обробку SQL-запитів, керування даними, оптимізацію запитів, роботу з індексами, кешуванням даних, а також зчитування та запис даних на диск. Ядро є основою для роботи з базами даних SQLite і забезпечує їх основний функціонал.

2.5 Висновки до розділу 2

У цьому розділі було розглянуто поняття чат-бота, які будуть використовуватися в розробці, показано функції, наведена класифікація всіх існуючих на даний момент чат-ботів. Далі було обґрунтовано вибір програмного забезпечення для реалізації чат-бота, за результатами якого були обрані такі засоби: мова програмування Python, середовище розробки Visual Studio Code, месенджер Telegram та база даних SQLite.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЧАТ-БОТА

3.1 Реєстрація телеграм бота

Перед тим як приступити до розробки системи, необхідно зареєструвати телеграм-бота та отримати ключ доступу до BOT API, що дозволить керувати ботом.

Оскільки Telegram є месенджером, реєстрація бота проводиться в чаті через офіційного телеграм-бота - BotFather. Всі операції, пов'язані з налаштуванням ботів, здійснюються виключно через нього.

Щоб знайти BotFather, введіть у пошуку чатів запит «BotFather» (Рисунок 3.1) і виберіть бота з відповідним іменем та офіційною відміткою, що підтверджує його безпечність для користування.

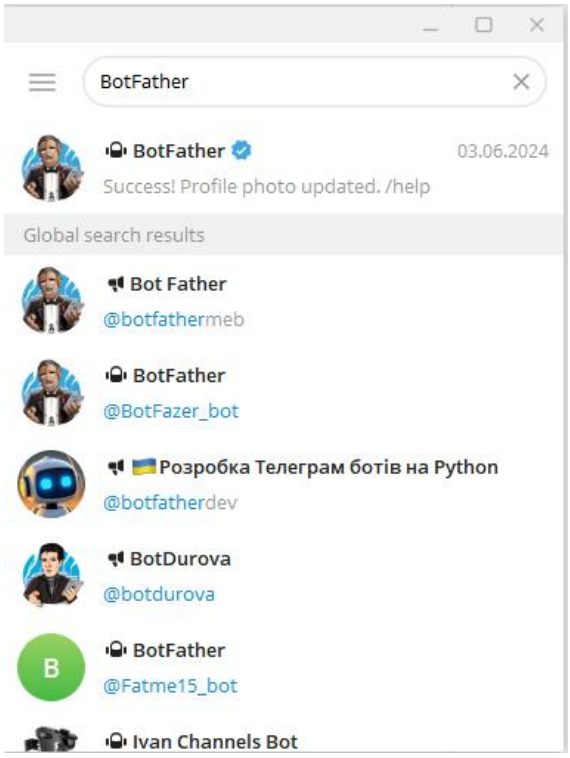


Рисунок 3.1 – Використання пошуку в Telegram для знаходження бота @BotFather

Далі слід відкрити чат з ботом і відправити команду «/start». Після цього відобразиться перелік доступних команд для налаштування ботів (Рисунок 3.2).

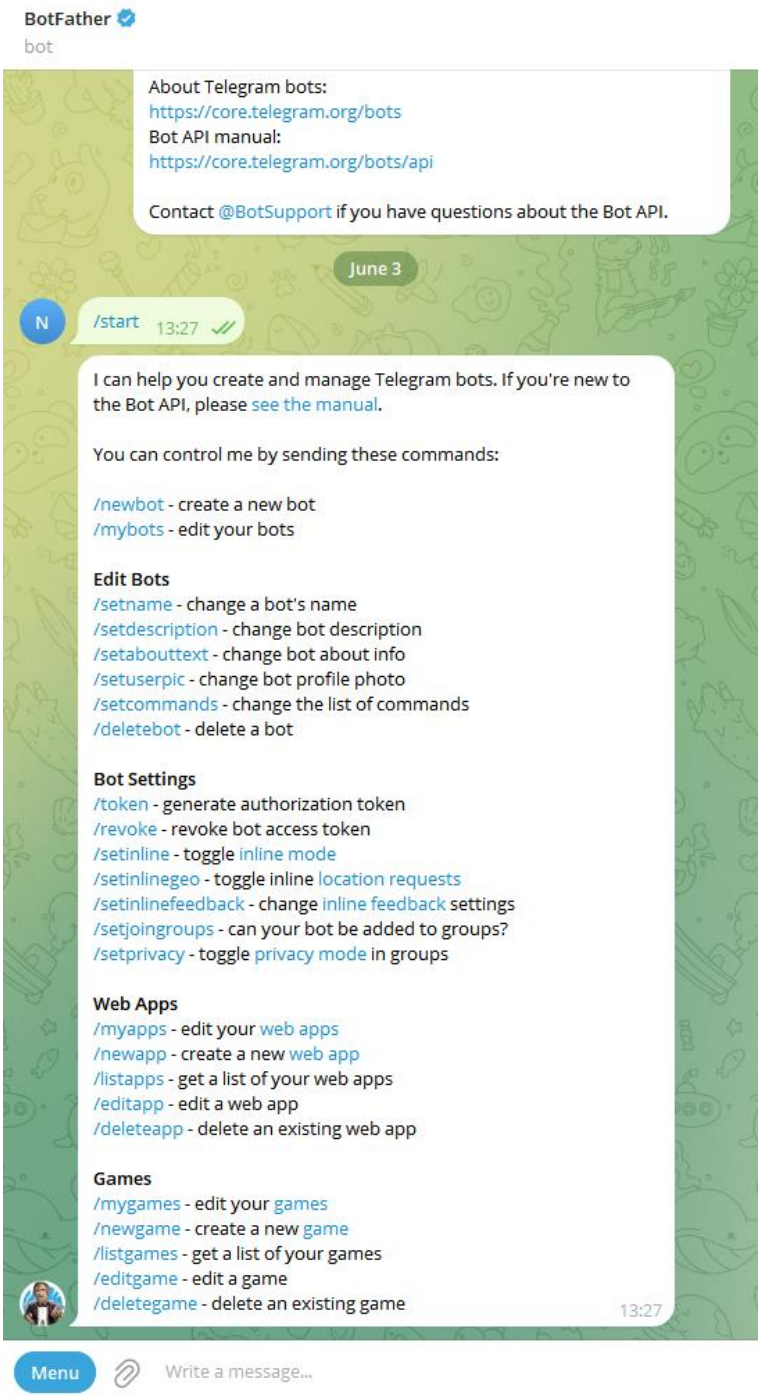


Рисунок 3.2 – Команди для налаштування бота

Серед команд, необхідних для налаштування бота, варто виділити такі:

– «newbot»: створює нового бота;

– «mybots»: надає інформацію про вже існуючі боти користувача, а також дозволяє їх налаштовувати та редагувати;

– «setname»: змінює ім’я бота;

– «setdescription»: дозволяє редагувати опис бота, який відображається при першому відкритті чату з ним.

– «setabouttext»: редагує інформацію про бота на його профілі;

– «setuserpic»: змінює фотографію бота;

– «setcommands»: додає нові команди, що запускають запрограмовані дії;

– «deletebot»: повністю видаляє бота;

– «token»: отримання ключа для керування ботом через BOT API;

– «set_inline»: вмикає інлайн режим для бота;

– «setjoingroups»: дозволяє або забороняє додавати бота до чатів;

– «setprivacy»: вмикає або вимикає читання ботом групових чатів;

– «revoke»: видаляє старий ключ доступу та створює новий, корисний у випадку компрометації токена.

– «setinlinefeedback»: налаштовує зв'язок для інлайн-відповідей;

– «setinlinegeo»: вмикає геолокацію для інлайн-відповідей;

– «setinlinequery»: дозволяє налаштовувати запити в інлайн-режимі;

– «setinlinefeedback»: дозволяє налаштовувати повідомлення зворотного зв'язку для інлайн-відповідей.

– «setgame» - додає можливість створення ігор для вашого бота;

– «set_sticker_set» - надає можливість додавати набори стікерів до бота.

– «setinlinegeo» - вмикає геолокацію для інлайн-відповідей;

Щоб створити нового бота, відправте команду «/newbot» до BotFather. Спочатку оберіть ім'я для бота, яке є важливим, оскільки це перше, що бачить користувач. Ім'я повинно бути простим і запам'ятовуватися, але водночас нести корисну інформацію. Далі виберіть унікальну назву, яка обов’язково повинна містити слово «bot», щоб користувачі розуміли, що це не реальна людина. Після виконання цих кроків, основна частина реєстрації завершена, і ботом можна користуватися, отримавши ключ доступу до Telegram Bot API (Рисунок 3.3). Рекомендується додати більше інформації про чат-бота, щоб користувачі розуміли його призначення та творця.

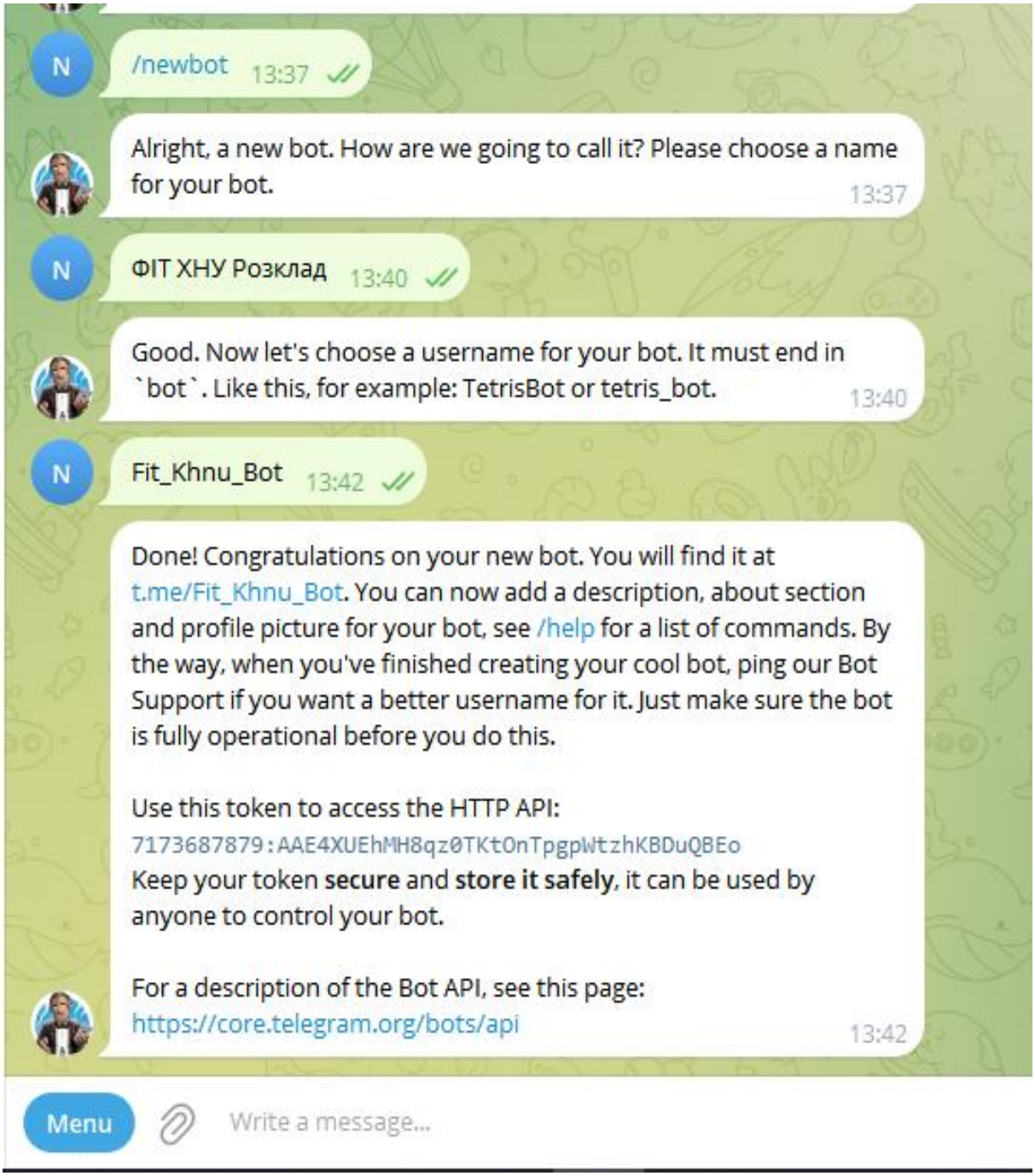


Рисунок 3.3 – Порядок створення нового бота та отримання доступу до Telegram Bot API

Далі можна перейти за посиланням і подивитися на бота, який надалі буде наповнюватися командами і набуде остаточного вигляду. Наразі він виглядає як порожній лист.

Натиснувши на іконку бота, можна переглянути профіль, як у звичайного користувача (Рисунок 3.4).

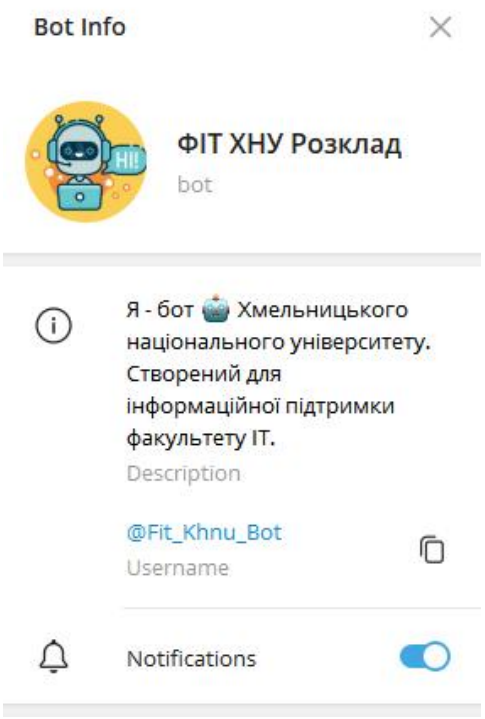


Рисунок 3.4 – Інформація про створеного бота

3.2 Розроблення програмних модулів

Програмні модулі – це автономні частини програмного забезпечення, які виконують конкретні функції та можуть бути незалежно розроблені, перевірені та використовувані у різних проєктах. Вони є основними будівельними блоками модульного програмування.

Структура проєкту, реалізованого для отримання дипломного програмного продукту зображена на рисунку 3.5:

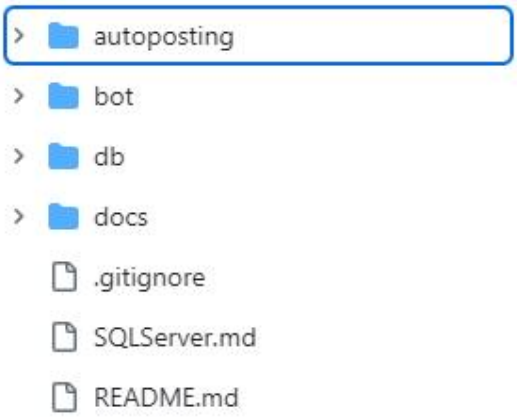


Рисунок 3.5 – Структура проєкту

Проаналізувавши можливі варіанти оформлення інтерфейсу вікна чату було зроблено висновок, що необхідно додати кнопки, для зручного переходу від одного блоку інформації до іншого. Це дає можливість створити якісну навігацію по боту для користувачів. Для початку потрібно з'ясувати різницю між видами кнопок у месенджері телеграм.

Reply Keyboard Markup – це шаблони повідомлень. Таким чином бот задає питання користувачу, у вигляді тексту і пропонує варіанти відповіді для подальшого переходу по архітектурі боту. В можливості користувача входить як самостійний набор з клавіатури, так і натиснення готових кнопок в інтерфейсі бота. Даний формат клавіатури не може містити ніяку інформацію. А реакцією на натиск цієї кнопки, буде лише відправка того змісту, який написаний безпосередньо на самій кнопці. Приклад створення та такої кнопки зображений нижче (Рисунок 3.6).

```python
from telegram import ReplyKeyboardMarkup
from telegram.ext import Updater, CommandHandler

# Функція для обробки команди /start
def start(update, context):
    # Створюємо клавіатуру з кнопкою "Дізнатися розклад занять"
    keyboard = [['Дізнатися розклад занять']]
    reply_markup = ReplyKeyboardMarkup(keyboard, resize_keyboard=True)
    # Відправляємо повідомлення з клавіатурою
    update.message.reply_text('Привіт! Якщо ви хочете дізнатися розклад занять, натисніть

def main():
    # Створюємо екземпляр Updater та передаємо токен вашого бота
    updater = Updater("YOUR_BOT_TOKEN", use_context=True)

    # Отримуємо об'єкт диспетчера для реєстрації обробників
    dp = updater.dispatcher

    # Додаємо обробник команди /start
    dp.add_handler(CommandHandler("start", start))

    # Запускаємо бота
    updater.start_polling()

    # Зупиняємо бота при натисканні Ctrl+C
    updater.idle()

if __name__ == '__main__':
    main()
```

Рисунок 3.6 – Приклад написання шаблонних кнопок

Така кнопка може лише виводити до чату текст, що написаний на ній, слугує більше для звичайних чат-ботів, задачею яких є виконання найпростіших функцій, таких як спілкування з користувачем (Рисунок 3.7).

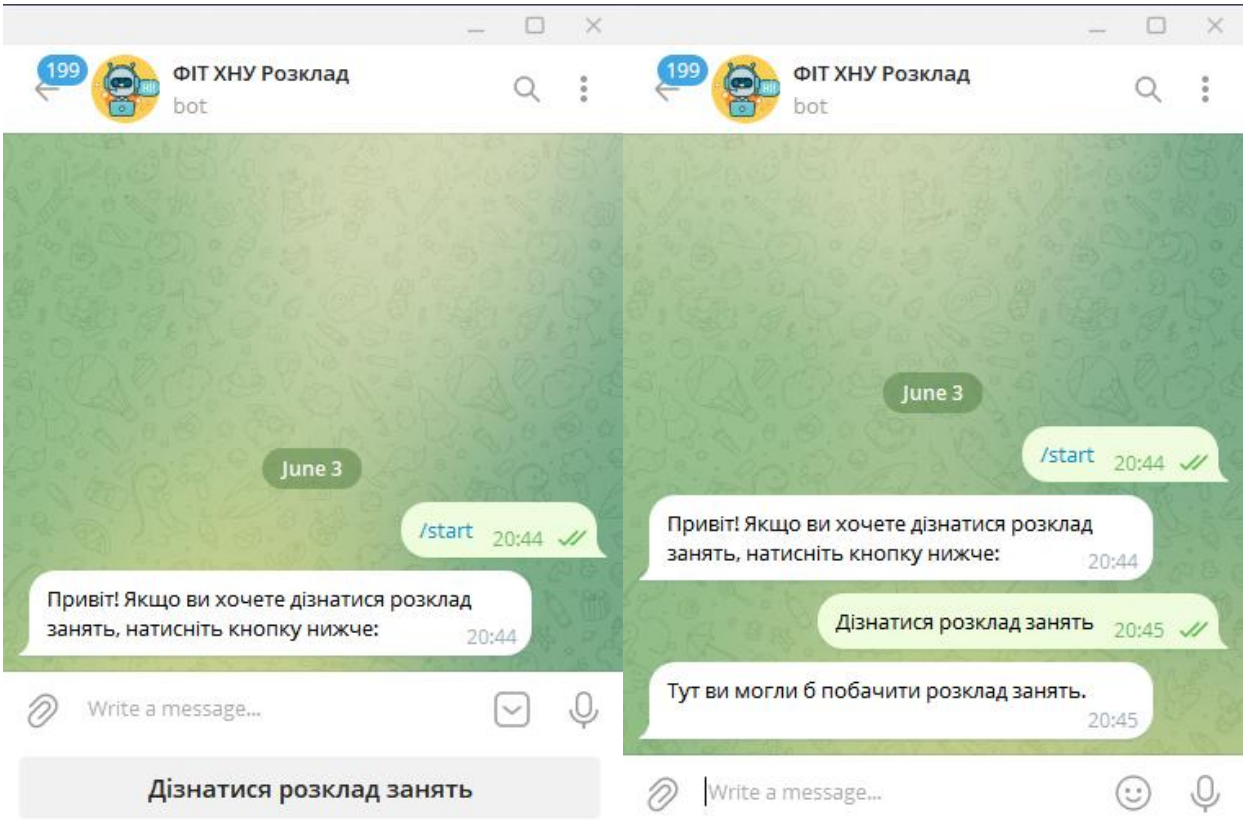


Рисунок 3.7 – Зовнішній вигляд шаблонної кнопки

Вона знаходиться у меню керування, і виглядає дійсно як звичайна клавіатура, якою люди користуються пишучи повідомлення друзям, або шукаючи якісь статті в Інтернеті.

Inline Keyboard Markup – являє собою клавіатуру з великою кількістю опцій для роботи. З її допомогою можна виконувати набагато складніші функції. Вона отримує зв’язок з повідомленням, або командою, з яким була відправлена. Функціонал цих кнопок дозволяє переходити за посиланнями в телеграм, пересуватися на зовнішні ресурси, робити форми вводу, для проведення опитувань, і багато інших можливостей. Приклад створення такої кнопки зображений нижче (Рисунок 3.8).

```python
from telegram import InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import Updater, CommandHandler, CallbackQueryHandler

# Функція для обробки команди /start
def start(update, context):
    # Створюємо інлайн кнопки для вибору курсу
    keyboard = [
        [InlineKeyboardButton("1 курс", callback_data='1'), InlineKeyboardButton("2 курс",
        [InlineKeyboardButton("3 курс", callback_data='3'), InlineKeyboardButton("4 курс",
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)
    # Відправляємо повідомлення з кнопками
    update.message.reply_text('Оберіть ваш курс:', reply_markup=reply_markup)

# Функція, що обробляє натискання на кнопку
def button(update, context):
    query = update.callback_query
    query.answer()
    # Відправляємо повідомлення з вибраним курсом
    query.edit_message_text(text=f"Ви обрали {query.data} курс.")

def main():
    # Створюємо екземпляр Updater та передаємо токен вашого бота
    updater = Updater("YOUR_BOT_TOKEN", use_context=True)

    # Отримуємо об'єкт диспетчера для реєстрації обробників
    dp = updater.dispatcher

    # Додаємо обробник команди /start
    dp.add_handler(CommandHandler("start", start))

    # Додаємо обробник для кнопки
    dp.add_handler(CallbackQueryHandler(button))

    # Запускаємо бота
    updater.start_polling()

    # Зупиняємо бота при натисканні Ctrl+C
    updater.idle()

if __name__ == '__main__':
    main()
```

Рисунок 3.8 – Приклад коду «inline» кнопок

Метод створення цих кнопок, як зазначено в назві, передбачає їхнє розміщення безпосередньо в чат-вікні, відрізняючись лише заокругленими прямокутними формами. Зовнішній вигляд таких кнопок приблизно відповідає зображенню на рисунку 3.9.

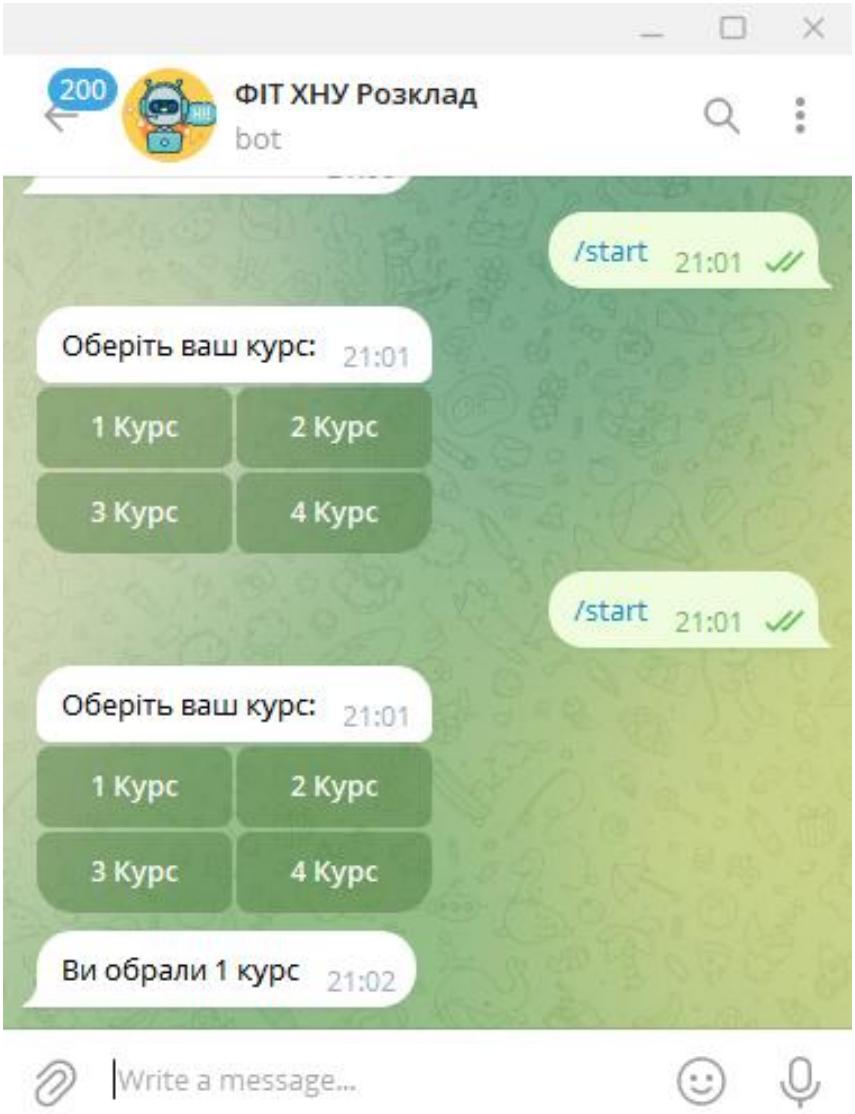


Рисунок 3.9 – Зовнішній вигляд «inline» кнопок

Процес вибору програмного забезпечення та способів реалізації було успішно завершено. Далі розроблено бота, функціонал якого описано в наступному розділі.

– Додано кнопку «/start» для виклику головного меню бота;

– головне навігаційне меню, з доступом до розкладу занять/дзвінків;

– кнопка загальна інформація, яка описує завдання бота;

– досліджено велику кількість ресурсів та джерел для збору структурованої інформації та якісної реалізації кваліфікаційної роботи;

– додано додаткові навігаційні кнопки для повернення на один крок назад або на головну сторінку чат-бота.

Повернемося до структури проєкту! Папка «bot» (Рисунок 3.10) містить у собі серце чат-бота - його логіку. Тут ви знайдете Reply Keyboard Markup та Inline Keyboard Markup, які відповідають за те, як користувач взаємодіє з ботом. Також тут зберігаються всі змінні та їх чітко організовані взаємозв'язки, що забезпечують безперебійну роботу бота!

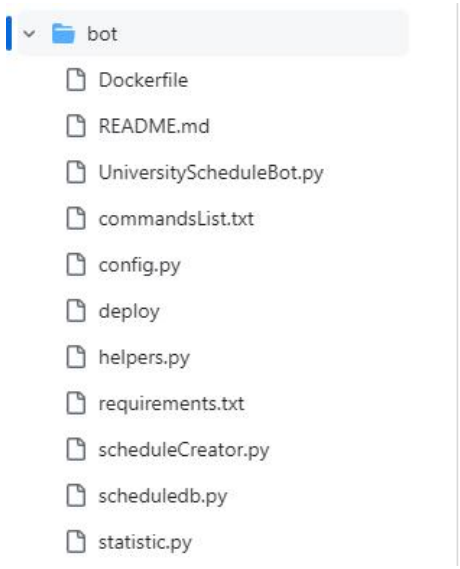


Рисунок 3.10 – Структура папки «bot»

Наступним важливим компонентом чат-боту є система автоматизованої розсилки розкладу занять. Тому було вирішено зберігати всі необхідні файли для цього розділу в папці під назвою «autoposting» (Рисунок 3.11).

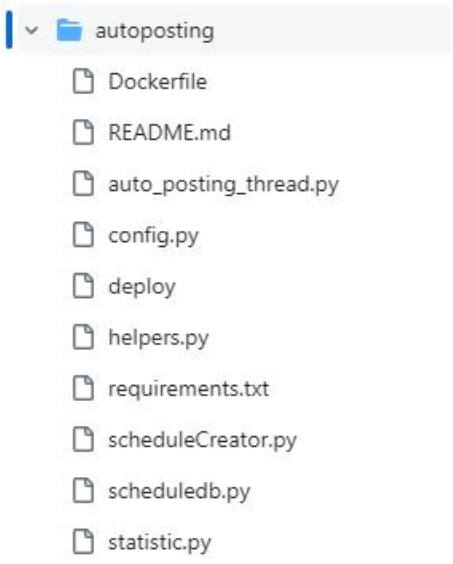


Рисунок 3.11 – Структура папки «autoposting»

Для побудови таблиць бази даних створена папка «db», в якій можна додавати дані, що стосуються бази даних (Рисунок 3.12).

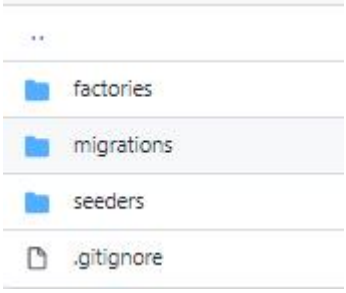


Рисунок 3.12 – Структура папки «db»

– migrations: директорія для створення таблиць і їх полів;
– seeders: директорія для генерування тестових даних у потрібному обсязі;
– factories: директорія для створення фальшивих даних, яка пов'язана з seeders.

3.3 Керівництво користувача

При першому запуску користувача зустрічає вікно (Рисунок 3.13), яке містить загальну інформацію про чат-бота. Для продовження роботи необхідно натиснути кнопку «START», щоб перейти до наступного меню!

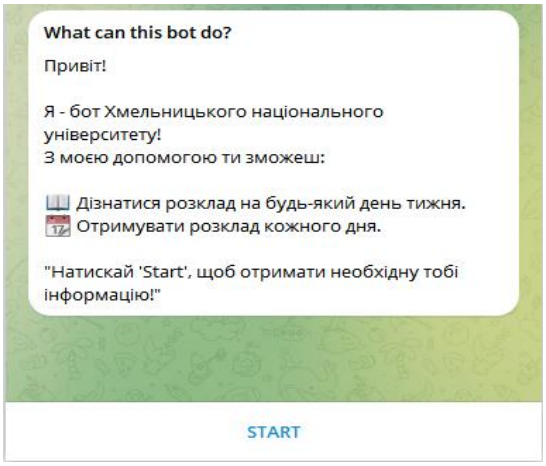


Рисунок 3.13 – Загальна інформацією про чат-бота

Після натискання кнопки «START» користувача перенесе на головну сторінку чат-бота (Рисунок 3.14), де він матиме можливість обрати

необхідний пункт з меню. Для цього потрібно натиснути на одну з трьох вбудованих кнопок.

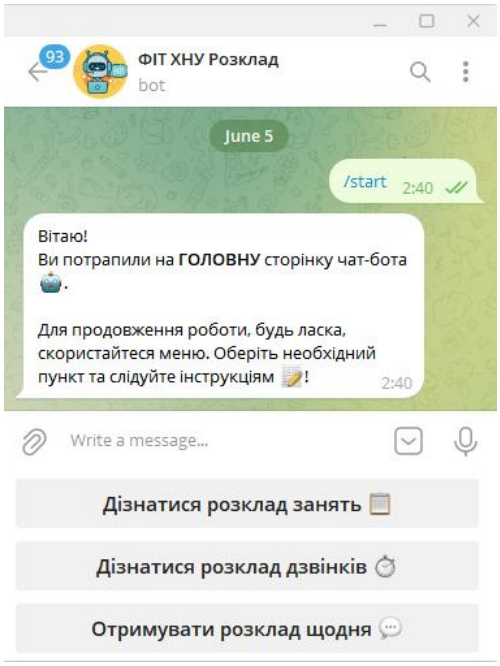


Рисунок 3.14 – Головна “сторінка”

Після натискання кнопки «Дізнатися розклад занять», бот автоматично запитає користувача про курс, на якому він навчається (Рисунок 3.15).

Рисунок 3.15 – Меню для вибору вашого курсу

Користувачові пропонується обрати курс, щоб продовжити: натисніть

відповідну кнопку або введіть цифру в чат. Після обрання курсу система автоматично перенаправить до меню вибору групи (Рисунок 3.16).

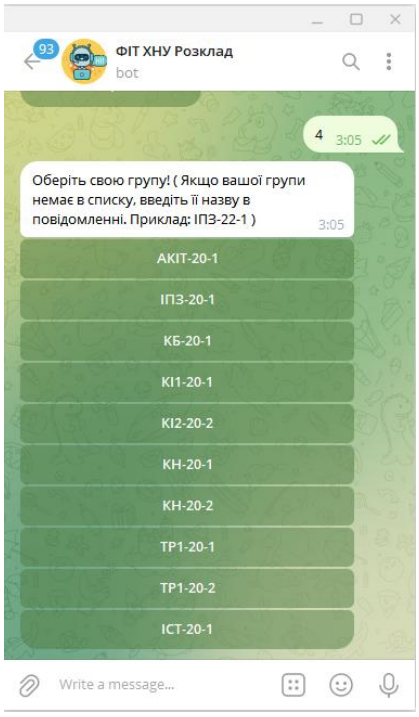


Рисунок 3.16 – Меню для вибору вашої групи

Користувачеві необхідно обрати свою групу із списку. Якщо групи немає у переліку, введіть її назву у повідомленні, наприклад: "ІПЗ-20-1". Важливо використовувати українську мову та дотримуватись прикладу.

Якщо операція виконана правильно, вас зустріне меню для підтвердження вашого вибору (Рисунок 3.17)!

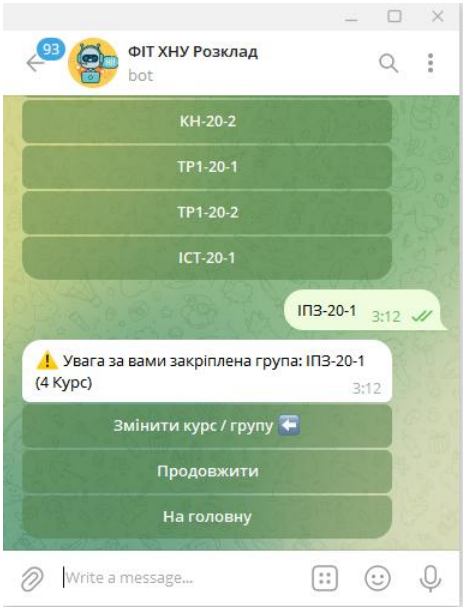


Рисунок 3.17 – Інформація про ваш курс та групу

Далі потрібно обрати наступний крок: змінити курс та групу, повернутися на головну сторінку чат-бота або продовжити і отримати розклад занять.

Після натискання кнопки «Продовжити» користувачеві буде запропоновано вибрати день тижня для отримання розкладу занять (Рисунок 3.18).

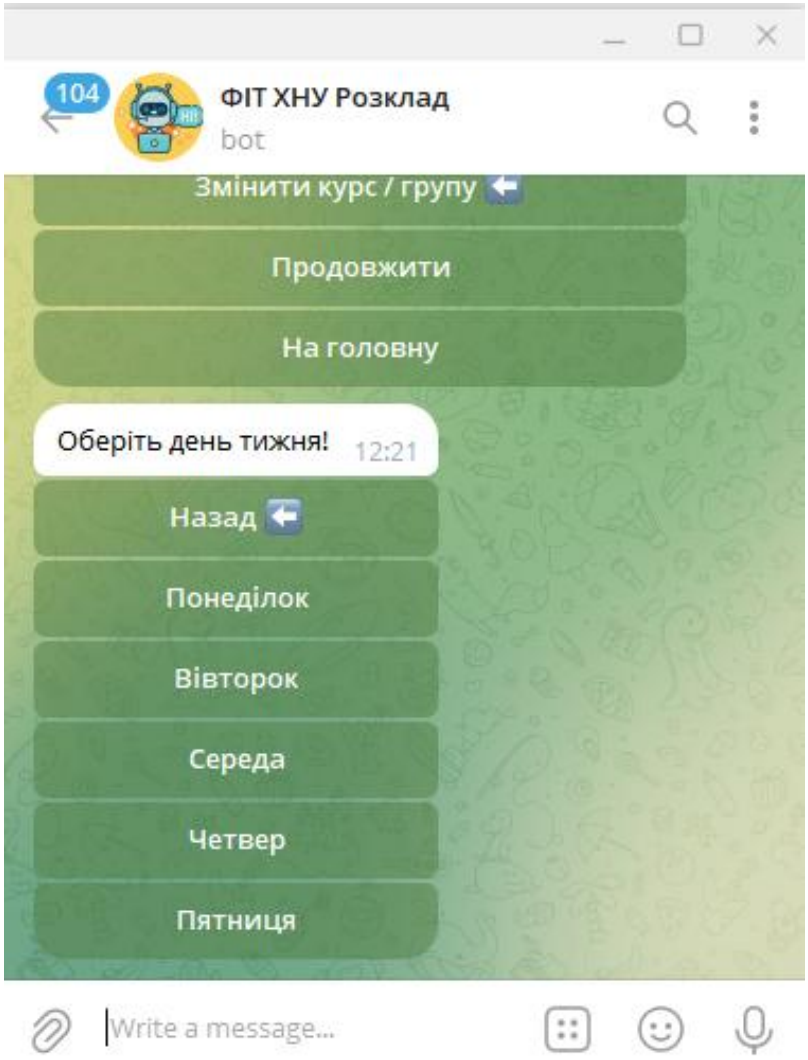


Рисунок 3.18 – Меню з вибором дня тижня

У фінальній частині сюжету для пункту "Дізнатися розклад занять" користувачеві надається можливість обрати день, щоб отримати розклад занять або повернутися на попередній крок за допомогою кнопки "Назад". При відсутності помилок у процесі вибору курсу та групи, користувач отримає розклад занять на вибраний день (див. Рисунок 1.19).

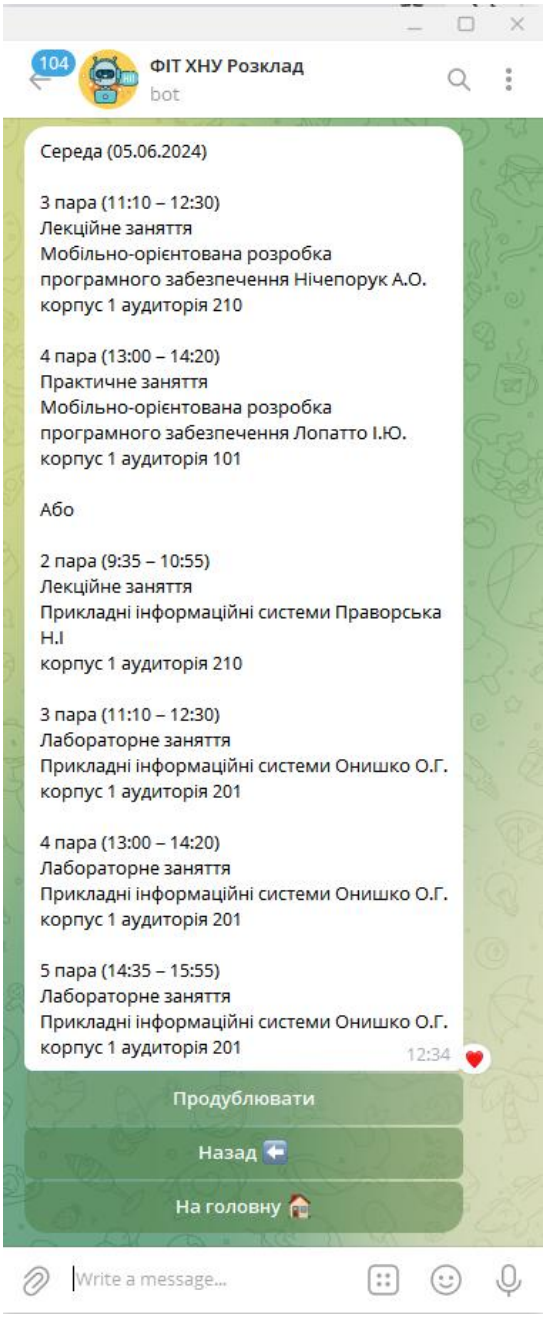


Рисунок 3.19 – Згенерований розклад занять

Під текстовою інформацією розташовані дві кнопки для навігації: "Назад" та "На головну". Відвідувачеві належить зробити вибір між двома діями: повернутися на головну сторінку чат-бота або повернутися на попередній крок.

Після натискання кнопки "На головну" бот автоматично перенаправить користувача на головну сторінку (Рисунок 3.20), де можна обрати інший варіант пункту з меню.

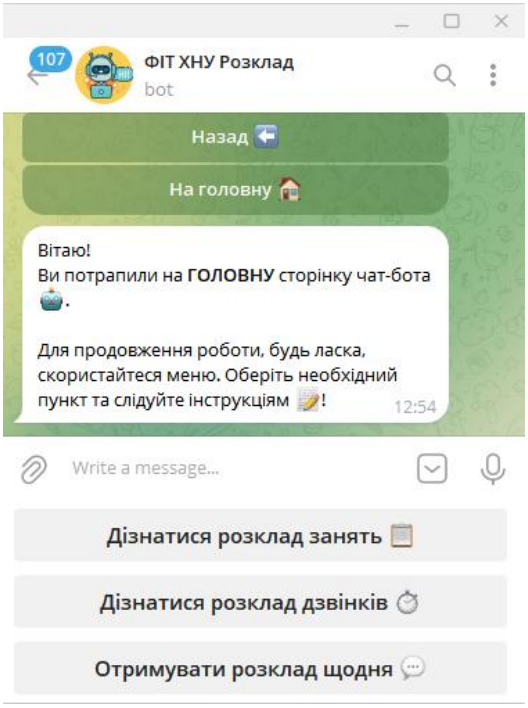


Рисунок 3.20 – Головна “сторінка”

Якщо користувач прагне скористатися чат-ботом знову, йому не доведеться проходити реєстрацію повторно. Система повідомить, що він вже зареєстрований і прив'язаний до певного курсу та групи (Рисунок 3.21), спрощуючи процес повторного входу до системи.

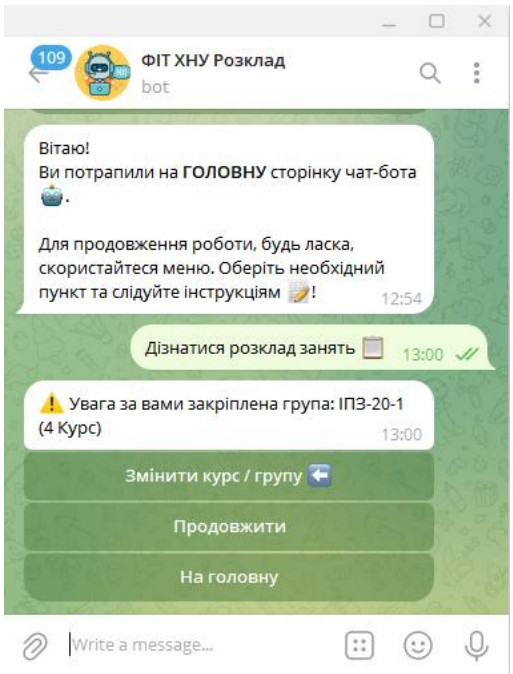


Рисунок 3.21 – Повідомлення про ваш курс та групу

Далі користувачеві необхідно вибрати наступний крок: він може змінити курс та групу, повернутися на головну сторінку чат-бота або продовжити і отримати розклад занять.

У випадку, якщо користувач спробує ввести щось або запитати у чат-бота поза "сценарієм", він отримає повідомлення про помилку (Рисунок 3.22), та бот автоматично перенаправить його в головне меню.

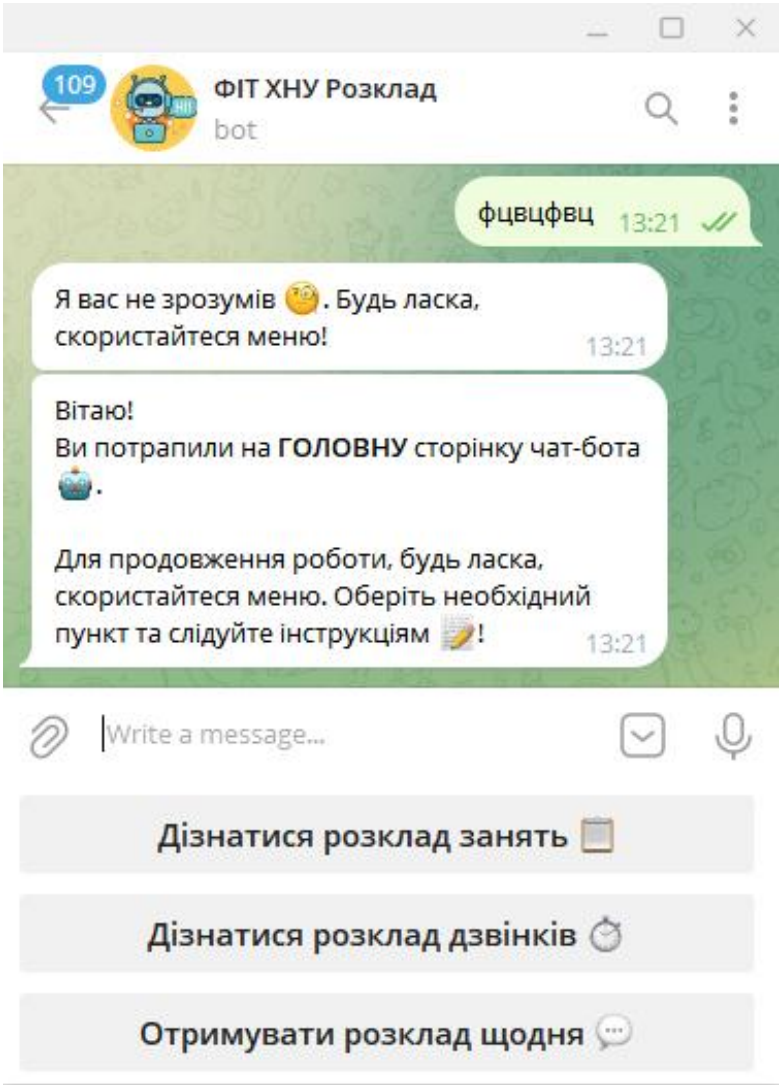


Рисунок 3.22 – Повідомлення про помилку

Після натискання кнопки «Отримати розклад дзвінків» бот автоматично надасть відвідувачу всю необхідну інформацію. Під виведеним текстом з'явиться кнопка «Назад», яка перенаправить до головного меню (Рисунок 3.23).

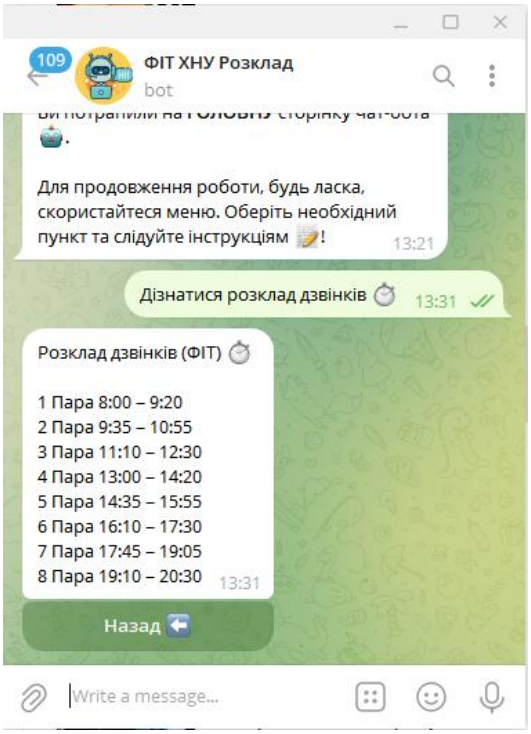


Рисунок 3.23 – Розклад дзвінків

Якщо користувач вирішить звернутися до чат-бота з питанням або повідомленням, що не входить в рамки "сценарію", він отримає повідомлення про те, що йому вже був наданий розклад дзвінків (Рисунок 3.24). Крім цього, бот автоматично перенаправить його до головного меню.

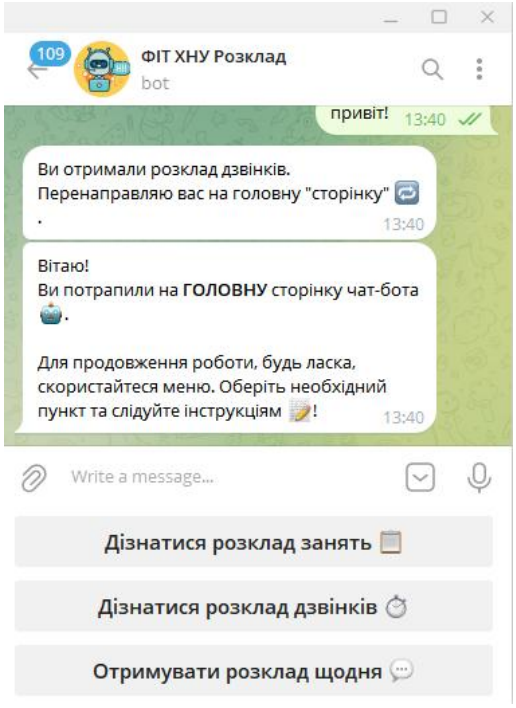


Рисунок 3.24 – Повідомлення про “помилку”

Останнім пунктом головного меню є кнопка з назвою «Отримувати розклад щодня».

Цей пункт меню дозволяє користувачеві підписатися на отримання щоденного розкладу. Після увімкнення цієї функції відвідувач отримуватиме автоматичні повідомлення з розкладом на кожний день. Це зручна функція для тих, хто хоче бути в курсі розкладу занять без необхідності кожен раз запитувати його у бота. Крім того, користувач може вимкнути цю функцію в будь-який час, якщо вона вже не потрібна.

Після натискання кнопки «Отримувати розклад щодня» бот надасть можливість увімкнути або вимкнути сповіщення про щоденний розклад (рис 3.25).

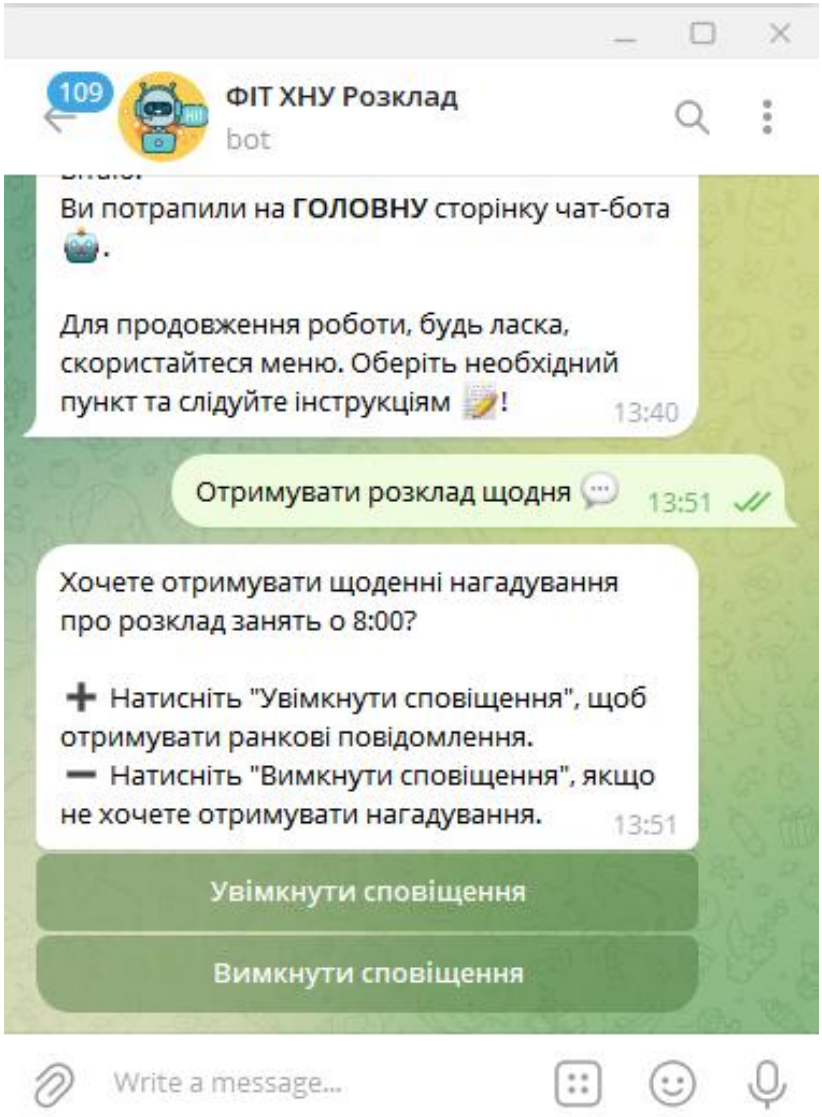


Рисунок 3.25 – Меню “щоденні нагадування”

Якщо користувач обере пункт під назвою "Увімкнути сповіщення", бот повідомить його про те, що він успішно увімкнув щоденні сповіщення. Робот також автоматично поверне його у головне меню (рисунок 3.26)..

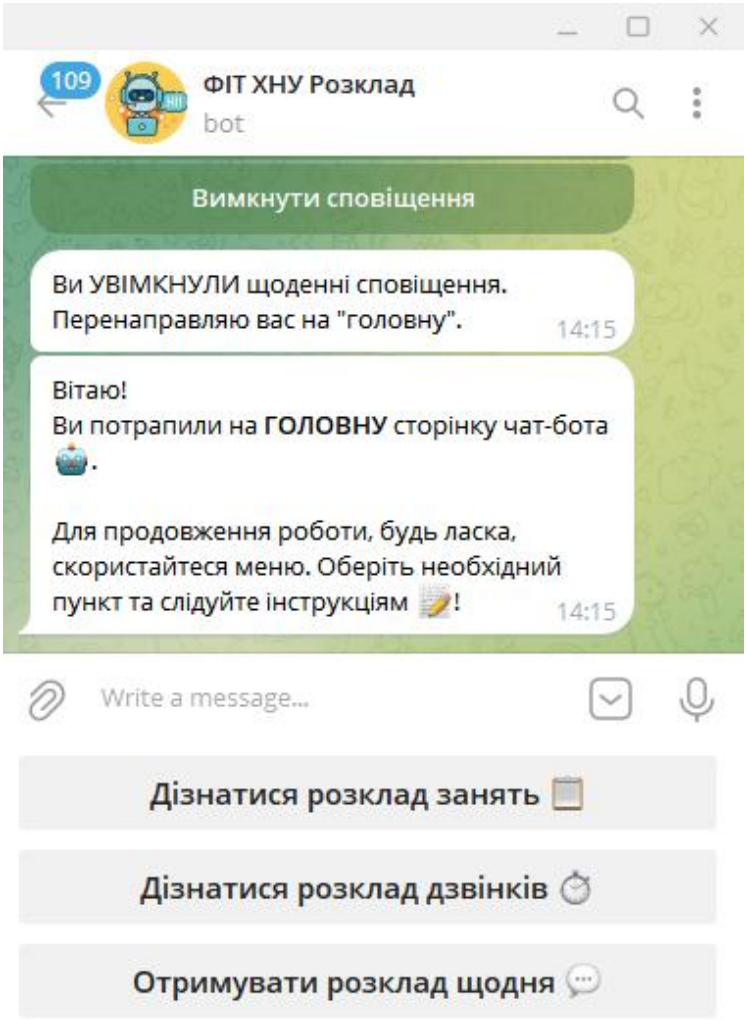


Рисунок 3.26 – Повідомлення про активацію сповіщень.

3.4 Вимоги до технічних та програмних засобів

Оскільки чат-бот був створений на платформі месенджера Telegram, мінімальні технічні характеристики пристрою повинні відповідати вимогам цього месенджера. Мінімальні вимоги для завантаження Telegram можуть відрізнятися в залежності від операційної системи, проте вони, в цілому, досить незначні.

– Android: Потрібна версія Android не нижче 4.1.

– iOS: Підтримується iOS версії 9.0 та вище.

– Комп'ютери: Для використання Telegram на комп'ютерах можна використовувати вебверсію або офіційні десктопні застосунки для Windows, macOS та Linux.

Операційна система:

– Для Windows: Windows 7 або новіша версія.

– Для macOS: OS X 10.10 Yosemite або новіша версія.

– Для Linux: Системи на базі Ubuntu 12.04+, Fedora 21+, Debian 8+ або аналогічні.

Процесор і пам'ять:

– Мінімум двоядерний процесор.

– Рекомендується наявність хоча б 2 ГБ оперативної пам'яті (RAM).

Місце на диску:

– Мінімум 100 МБ вільного місця на жорсткому диску для встановлення застосунку.

Інтернет-з'єднання:

– Наявність стабільного інтернет-з'єднання для завантаження застосунку та обміну повідомленнями.

Графічна карта (для деяких функцій):

– Будь-яка сучасна графічна карта, яка підтримує відображення GUI.

Навіть на пристроях зі старішими версіями операційних систем або на пристроях з обмеженими технічними характеристиками Telegram може працювати, але деякі функції можуть бути обмежені або недоступні.

3.5 Тестування програмної системи

Тестування чат-бота відбувалося одночасно з написанням коду для виправлення помилок. Основним етапом тестування було оцінювання інтерфейсу, перевірка роботи бота, швидкості відповіді на запити користувачів та коректності алгоритму навігації кнопок.

Першим етапом у тестуванні є перевірка відповідей бота на запити користувача. Для цього, перейшовши до онлайн JSON-редактора, було відправлено запит на отримання інформації про бота. Після введення запиту в рядок браузера та натискання кнопки відправлення, сторінка браузера відображає невеликий JSON-код (Рисунок 3.27). Після цього отриманий код було скопійовано та вставлено в онлайн редактор для подальшого перегляду інформації за вказаним запитом (Рисунок 3.28).

Онлайн редактор - це вебверсія редактора коду, яка дозволяє виконувати прості програми та перевіряти правильність написаного коду. У цьому випадку використання даного редактора дозволило переглянути інформацію про бота, яку надав сервер Telegram після отримання нашого запиту. Команда «getMe» показала, що сервер знайшов бота, що свідчить про те, що бот має зв'язок з сервером і доступний для всіх користувачів месенджеру Telegram.

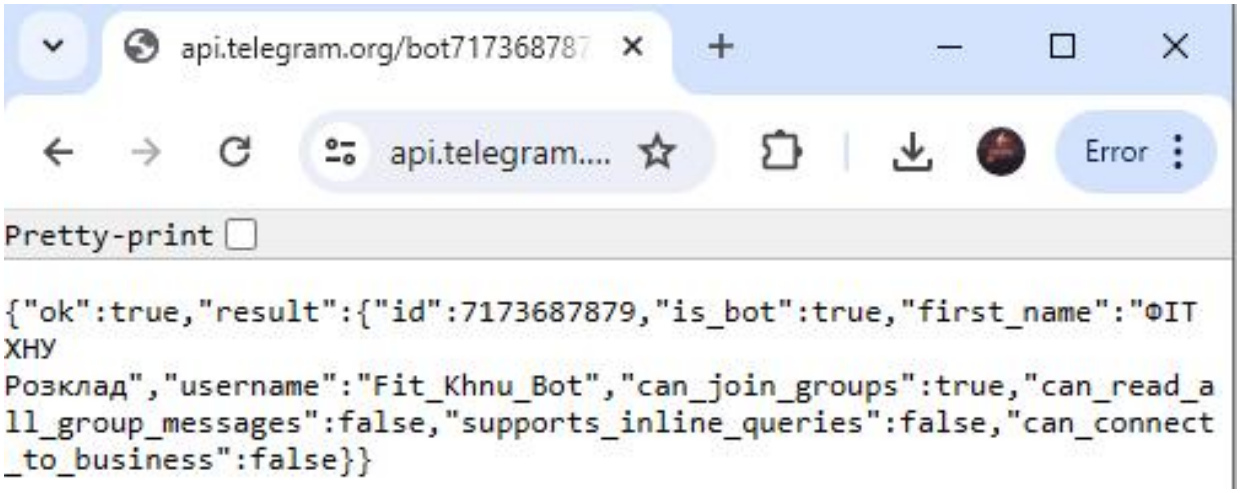


Рисунок 3.27 – Відповідь сервера телеграм на поданий запит

Після копіювання цього коду в редактор та його виконання, стає очевидним більш розкритий зміст, представлений у вигляді послідовностей (Рисунок 3.28).

```
data = {
    "ok": True,
    "result": {
        "id": 7173687879,
        "is_bot": True,
        "first_name": "ФІТ ХНУ Розклад",
        "username": "Fit_Khnu_Bot",
        "can_join_groups": True,
        "can_read_all_group_messages": False,
        "supports_inline_queries": False,
        "can_connect_to_business": False
    }
}
```


Рисунок 3.28 – Дерево ланцюгів відповіді сервера

Після аналізу коду, виявлено, що бот успішно співпрацює з сервером Telegram, що свідчить про правильне налаштування та підключення. Сервер повертає відповідь «ok»: true, підтверджуючи успішне виконання запиту, та надає необхідну інформацію. У разі неправильної відповіді, сервер видав би помилку (Рисунок 3.29).

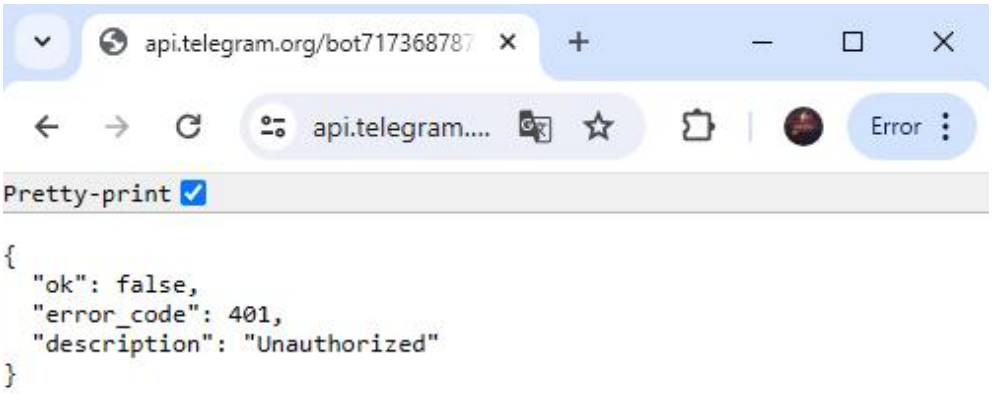


Рисунок 3.29 – Відмова сервера на запит

Видалення кількох символів з коду запиту дозволяє побачити, що сервер не може знайти бота та відповісти на запит. Це відбувається, коли сервер не здатний розпізнати бота.

При помилці HTTP 401 Unauthorized сервер повертає код відповіді, який вказує, що запит не може бути виконаний через відсутність необхідних персональних даних для доступу до ресурсу.

Після перевірки взаємодії бота з сервером Telegram і підтвердження його доступності в мережі, проводиться тестування навігації та логіки команд в інтерфейсі.

Для цього, відкривши застосунок Telegram на комп'ютері або смартфоні, необхідно знайти бота через пошуковий рядок, ввівши його тег (Рисунок 3.30).

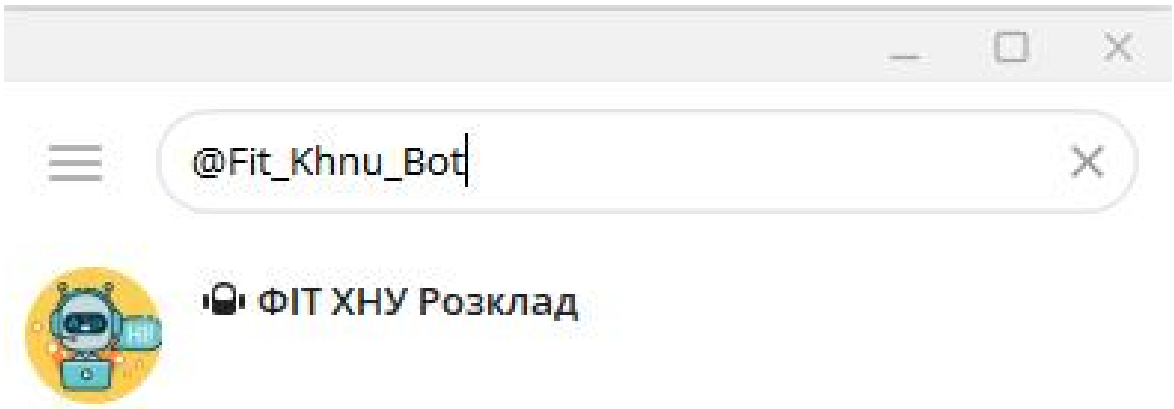


Рисунок 3.30 – Тег бота у месенджері

Знаходячись у чаті потрібного бота, написавши команду для початку (/start), почалася перевірка функціонування кнопок та коректності відповідей бота.

Відповідь бота на першу команду зайняла кілька секунд, що може свідчити про громіздкість коду. Результат був коректним, від-образивши заплановане головне меню. Меню складається з текстового блоку та кількох inline-кнопок для подальшої навігації застосунком.

Аналіз часу відправки повідомлень показує, що бот відповів користувачу в ту ж хвилину. Оскільки кількість користувачів наразі невелика, чат-бот швидко надає результати. Проте зі збільшенням кількості запитів час очікування може зрости, тому слід шукати альтернативні рішення для обробки.

Аналіз всіх можливих сценаріїв навігації в інтерфейсі чат-бота було протестовано в попередньому розділі під назвою “Інструкція користувача”.

3.6 Висновки до розділу 3

У цьому розділі було детально розглянуто основні етапи створення телеграм бота та пов'язані з цим технічні аспекти. Спочатку була описана процедура реєстрації телеграм бота, яка є необхідною для його подальшої розробки та інтеграції (підрозділ 3.1). Далі було приділено увагу розробці програмних модулів (підрозділ 3.2), що є ключовим етапом у забезпеченні функціональності бота.

Керівництво користувача (підрозділ 3.3) надає докладні інструкції щодо використання бота, що полегшує його експлуатацію кінцевими користувачами. У підрозділі 3.4 розглянуто вимоги до технічних та програмних засобів, що необхідні для коректної роботи бота, що допомагає забезпечити стабільність та ефективність системи.

Завершальним етапом було тестування програмної системи (підрозділ 3.5), яке є критично важливим для виявлення та усунення можливих недоліків у роботі бота, забезпечуючи таким чином його надійність та відповідність заданим вимогам.

ВИСНОВКИ

В ході виконання цієї кваліфікаційної роботи було створено телеграм бот для відображення розкладу занять факультету ІТ Хмельницького національного університету, що вирішує важливу задачу автоматизації та спрощення доступу до інформації про заняття для студентів.

На першому етапі дослідження було проведено змістовний аналіз предметної області, визначено її структурні та функціональні особливості. Було також виконано аналіз наявного програмно-технічного забезпечення предметної області, що дало змогу сформулювати чіткі завдання для розробки системи.

На етапі проєктування програмного забезпечення було проведено аналіз і проєктування архітектури системи з використанням стандартів IDEF0 та UML. Ретельно проаналізовано та обрано тип бази даних, спроєктовано її структуру.

На етапі програмної реалізації та тестування було здійснено реєстрацію телеграм бота та розроблено необхідні програмні модулі. Підготовлено керівництво користувача, визначено вимоги до технічних та програмних засобів. Заключним етапом стало тестування програмної системи, яке підтвердило її працездатність та відповідність поставленим вимогам.

Результати розробки чітко демонструють, що створений телеграм бот успішно впорався з основною метою - надати студентам зручний та ефективний доступ до розкладу занять. Система виявилася надійною, зручною у використанні та легкою в адмініструванні, що підтверджує її доцільність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Стаття «SQL vs ORM». URL: https://habr.com/ru/company/pgdayrussia/blog/328690/ (дата звернення 15.02.2024).

2. Документація Telegram Bot API. URL: https://core.telegram.org/bots/api#available-methods (дата звернення 20.02.2024).

3. Стаття «Virtual Environments and Packages». URL: https://docs.python.org/3/tutorial/venv.html (дата звернення 25.02.2024).

4. Стаття «Створення Telegram Bot». URL: https://codeguida.com/post/410 (дата звернення 01.03.2024).

5. Python Developers Guide. URL: https://devguide.python.org/ (дата звернення 05.03.2024).

6. Python Secrets Module. URL: https://pynative.com/python-secrets-module/ (дата звернення 10.03.2024).

7. Стаття «State Design Pattern». URL: https://sourcemaking.com/design_patterns/state (дата звернення 15.03.2024).

8. What are Chatbots and Why are They Becoming so Popular? URL: https://www.uctoday.com/contact-centre/what-are-chatbotsand-why-are-theybecoming-so-popular/ (дата звернення 20.03.2024).

9. Python and fast HTTP clients. URL: https://julien.danjou.info/python-and-fast-http-clients/ (дата звернення 25.03.2024).

10. Документація JSON. URL: https://www.json.org/json-en.html (дата звернення 30.03.2024).

11. Natural Language Processing with Python. URL: http://www.nltk.org/book/ (дата звернення 04.04.2024).

12. Документація бібліотеки pyTelegramBotAPI. URL: https://github.com/eternnoir/pyTelegramBotAPI (дата звернення 09.04.2024).

13. What is a Container? URL: https://www.docker.com/resources/what-container (дата звернення 14.04.2024).

14. Документація бібліотеки pyTelegramBotAPI. URL: https://github.com/eternnoir/pyTelegramBotAPI (дата звернення 15.04.2024).

15. Стаття «Telegram Bots». URL: https://core.telegram.org/bots (дата звернення 15.04.2024).

16. Credence Research. Прогнозування ринку чат-ботів. URL: https://www.credenceresearch.com/report/chatbots-market (дата звернення 15.04.2024).

17. Python Developers Guide. URL: https://devguide.python.org/ (дата звернення 05.03.2024).

18. Kantar. Рейтинг мобільних додатків за січень 2021. URL: https://tns-ua.com (дата звернення 20.04.2024).

19. Jet Brains. IDE для професійної розробки на Python. URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата звернення 25.04.2024).

20. Heroku Dev Center. URL: https://devcenter.heroku.com/ (дата звернення 26.05.2024).

21. Документація Telegram Bot API. URL: https://core.telegram.org/bots/api#available-methods (дата звернення 28.04.2024).

22. Стаття «Virtual Environments and Packages». URL: https://docs.python.org/3/tutorial/venv.html (дата звернення 02.05.2024).

23. Python and fast HTTP clients. URL: https://julien.danjou.info/python-and-fast-http-clients/ (дата звернення 06.05.2024).

24. Бібліотеки Python та їх особливості. URL: https://spacelab.ua/articles/biblioteki-python-i-ih-osobennosti/ (дата звернення 08.05.2024).

25. Найпопулярніші бібліотеки ШІ та машинного навчання Python. URL: https://proit.org.ua/naipopuliarnishi-bibliotieki-shi-ta-mashinnogho-navchannia-python/ (дата звернення 12.05.2024).

26. Путівник мовою програмування Python. URL: https://pythonguide.rozh2sch.org.ua/ (дата звернення 16.05.2024).

27. How to get Telegram bot API token. URL: https://www.siteguarding.com/en/how-to-get-telegram-bot-api-token (дата звернення 20.05.2024).

28. Що таке SqLite?. URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-sqlite/ (дата звернення 24.05.2024).

29. Документація бібліотеки pyTelegramBotAPI. URL: https://github.com/eternnoir/pyTelegramBotAPI (дата звернення 26.05.2024).

30. What is a Container? URL: https://www.docker.com/resources/what-container (дата звернення 28.05.2024).

ДОДАТОК А

(Обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

1 Загальні відомості.

Метою кваліфікаційної роботи є створення чат-бота для факультету інформаційних технологій, який допоможе студентам зручно знаходити необхідну інформацію та отримувати відповіді на запитання з мінімальними витратами часу.

1.1 Найменування замовника та розробника.

Замовником розробки визначено виконавця проєкту – кафедру інженерії програмного забезпечення.

Розробник програмного забезпечення - студент групи ІПЗ-20-1 Остапчук Микита.

1.2 Термін надання послуг.

Даний програмний продукт потрібно розробити протягом п`яти місяців з 08.01.2024 по 01.06.2024.

2. Вимоги до програмного продукту

2.1 Функціональні вимоги:

Розклад занять: Чат-бот повинен надавати можливість студентам переглядати розклад занять на певний день. Графік повинен бути доступний для кожної групи студентів і відображати інформацію про аудиторії, викладачів і час проведення занять.

Оповіщення про зміни в розкладі: Чат-бот має надсилати сповіщення студентам про будь-які зміни в розкладі занять, такі як скасування занять, зміна аудиторії або часу проведення. Це допоможе студентам бути в курсі оновленої інформації про їхні заняття.

Підтримка різних форматів запитів: Чат-бот повинен бути здатний розуміти різні формати запитів від користувачів, такі як текстові запити або навігація по кнопкам. Це забезпечить зручність користувачам у використанні чат-бота.

Закріплення курсу та групи: Студент повинен мати можливість вибрати і закріпити за собою курс та групу. Це дозволить отримувати персоналізований розклад швидко та зручно.

Розсилка розкладу занять: Чат-бот повинен надсилати нагадування студентам за певний час до початку заняття (наприклад, за годину або за день).

Це допоможе студентам не пропустити заняття. Також бот повинен надсилати студентам розклад занять на поточний день. Повідомлення повинні містити інформацію про предмет, час і місце проведення занять, а також ім’я викладача.

2.2 Нефункціональні вимоги

– практичність. Відповідає за те, щоб програмне забезпечення було простим та легким у використанні для будь-якого користувача, не завдавало зайвих труднощів та було інтуїтивно зрозумілим;

– надійність. Відповідає за роботу програмного забезпечення, та описує як система повинна себе поводити у разі помилок чи збоїв в роботі та зберегти важливі дані навіть у цьому випадку;

– продуктивність. Визначає характеристики системи під час взаємодії з користувачами, тобто наступні характеристики - відповідає за оптимальний час на відповідь запита користувача, кількість користувачів, яку одночасно може прийняти система та визначає які системні ресурси їй на це знадобляться;

– можливість обслуговування. Означає можливість легко модифікувати, змінювати програмне забезпечення для введення нового функціоналу або з метою виправлення помилок.

Виходячи з цього можна визначити необхідні нефункціональні вимоги до системи:

– система повинна бути надійною та доступною для використання користувачами в будь-який момент часу;

– в разі збоїв в роботі програмне забезпечення повинно адекватно реагувати на це та намагатись самостійно відновити роботу;

– в разі помилок перш за все важливе збереження всієї інформації в базі даних;

– система має бути побудована таким чином, щоб в майбутньому для неї було легко створювати нові модулі та функції або виправляти код з метою покращення;

3. Вимоги до структури системи

Архітектура:

– Клієнт-серверна архітектура.

– Використання мікросервісів для масштабованості.

– Серверна частина на основі Python та фреймворку pyTelegramBotAPI.

Компоненти:

– Модуль інтеграції з Telegram API.

– Модуль обробки запитів користувачів.

– База даних для зберігання розкладу занять (SQLite).

– Адміністративна панель для оновлення розкладу.

4. Вимоги до інформаційної та програмної сумісності.

Інформаційна сумісність:

– Підтримка стандартних форматів даних (JSON, CSV) для імпорту та експорту розкладу.

Програмна сумісність:

– Підтримка останніх версій операційних систем Windows, Linux.

5. Вимоги до маркування й упакування

Маркування:

– Вказати назву проєкту та контактні дані розробника на всіх упаковках.

– Забезпечити відповідність маркування стандартам безпеки та конфіденційності даних.

6. Вимоги до транспортування і зберігання.

Повинні дотримуватися правила експлуатації Flesh-носія, а саме: Забороняється:

– примусовий витяг накопичувача з порту USB до завершення будь-якої виконуваної операції або примусове переривання операції;

– спроби форматування накопичувача не в тій файлової системи, яка існувала в початковому стані пристрою в момент його придбання (робота з довільно обраної файлової системою може бути не передбачена в структурі контролера флешпам’яті);

– виконання з цікавості тестової операції запису у флеш-пам’ять, після чого накопичувач не видаляється з системи за допомогою механізму безпечного вилучення (виправлення недоліку

– повторне форматування пристрою в вихідної файлової системи або використання режиму форсованої зупинки);

– після безпечного видалення пристрій з флеш-пам’яттю не впізнав в файловому менеджері і спеціалізованих програмах (причина нестачі - було виконано безпечне видалення накопичувача, але пристрій не було своєчасно вилучено з порту USB; виправлення браку - остаточне вилучення пристрою з порту USB).

7. Вимоги щодо взаємодії та інтеграції з іншими системами

Взаємодія:

– Можливість інтеграції з іншими месенджерами за потреби.

Інтеграція:

– Використання стандартних протоколів (HTTP/HTTPS, REST) для обміну даними.

7. Стадії та етапи розробки програмного забезпечення

Таблиця А.1 – Стадії та етапи розробки

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.22 – 31.01.22	Обґрунтування необхідності розробки програми	Характеристика та опис ПЗ; підстави для розробки і призначення ПЗ; функціональні вимоги до розроблюваної системи; порядок контролю і приймання ПЗ.
Ескізний проєкт 01.02.22 – 14.02.22	розроблення ескізного проєкту	Визначення структури вхідних і вихідних даних; попередній вибір технологій; визначення потрібних алгоритмів.

Продовження таблиці А.1

Технічний проєкт 15.02.22 – 28.02.22	розроблення технічного проєкту	Затвердження структури вхідних і вихідних даних; розроблення алгоритмів; розроблення структури програми; вибір технологій.
Робочий проєкт 01.03.22 – 10.04.22	розроблення програмного забезпечення	Реалізація програмного забезпечення; виправлення помилок; тестування.
розроблення програмної документації 11.04.22 – 20.04.22	розроблення документації для програмного забезпечення	розроблення документації користувача, інструкції по запуску та зупинці ПЗ
Тестування системи 21.04.22 – 30.04.22	Проведення тестування програмного забезпечення	розроблення методики тестування; проведення основних тестів; коректування програмного забезпечення
Здача проєкту	Здача програми замовнику	Передача вихідного коду та документації замовнику

ДОДАТОК Б

(Обов’язковий)

ДІАГРАМИ

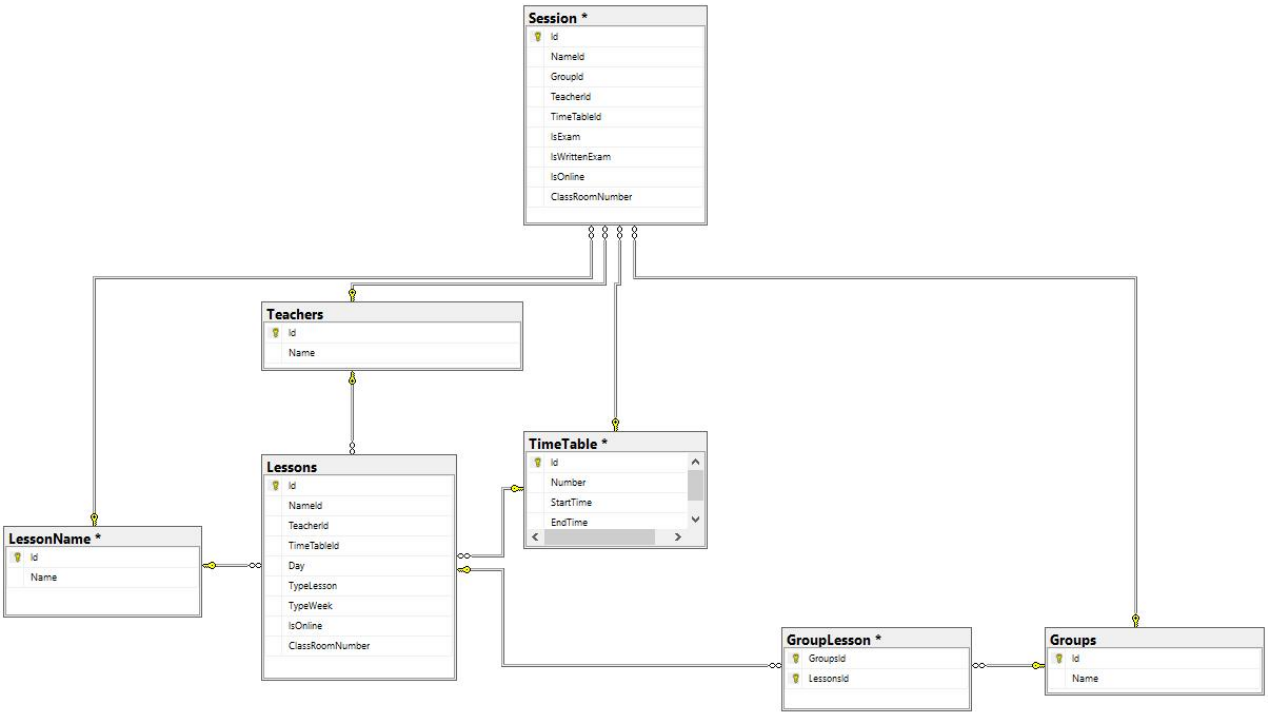


Рисунок Б.1 – Схема бази даних

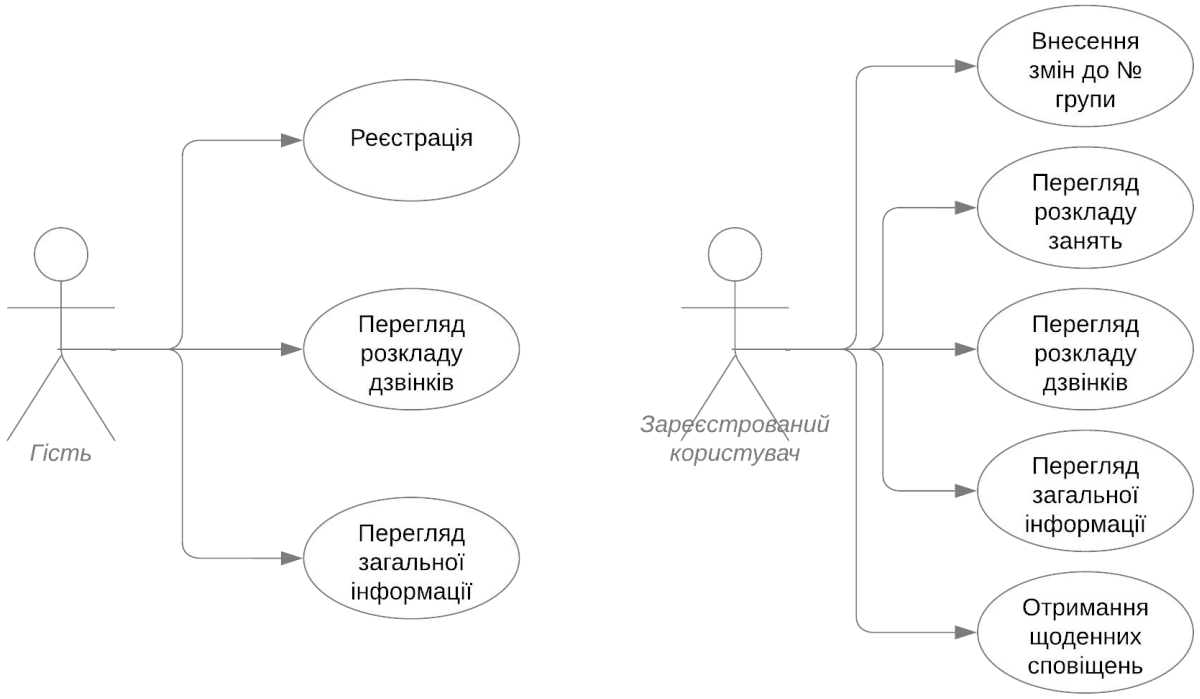


Рисунок Б.2 – Діаграма варіантів використання

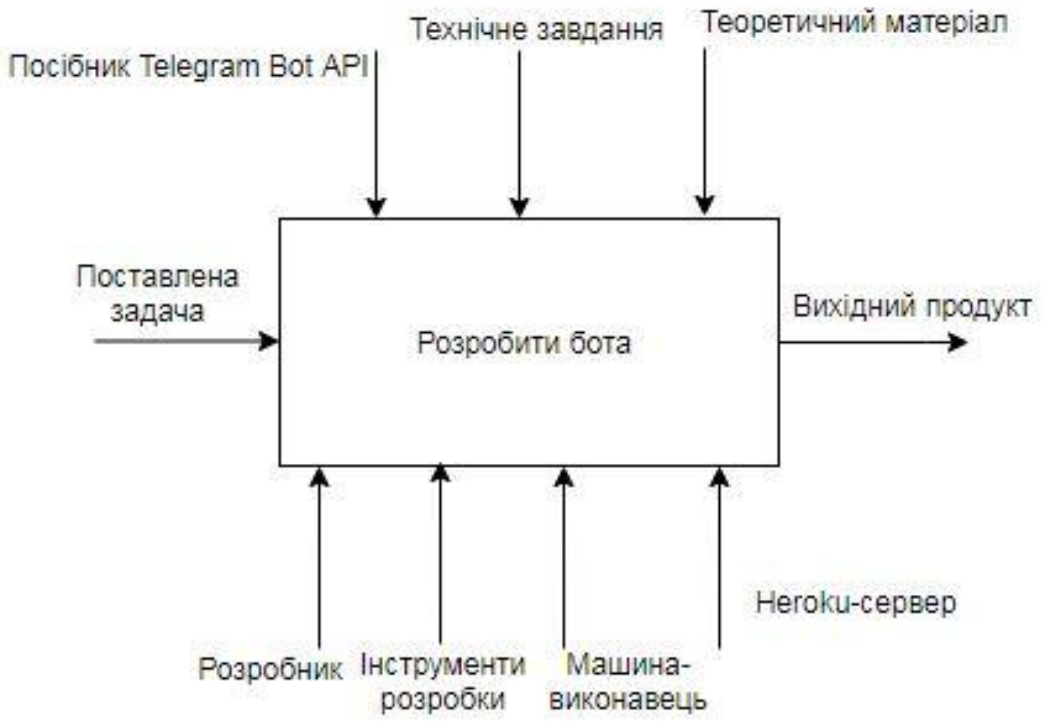


Рисунок Б.3 – Контекстна діаграма процесу розробки бота

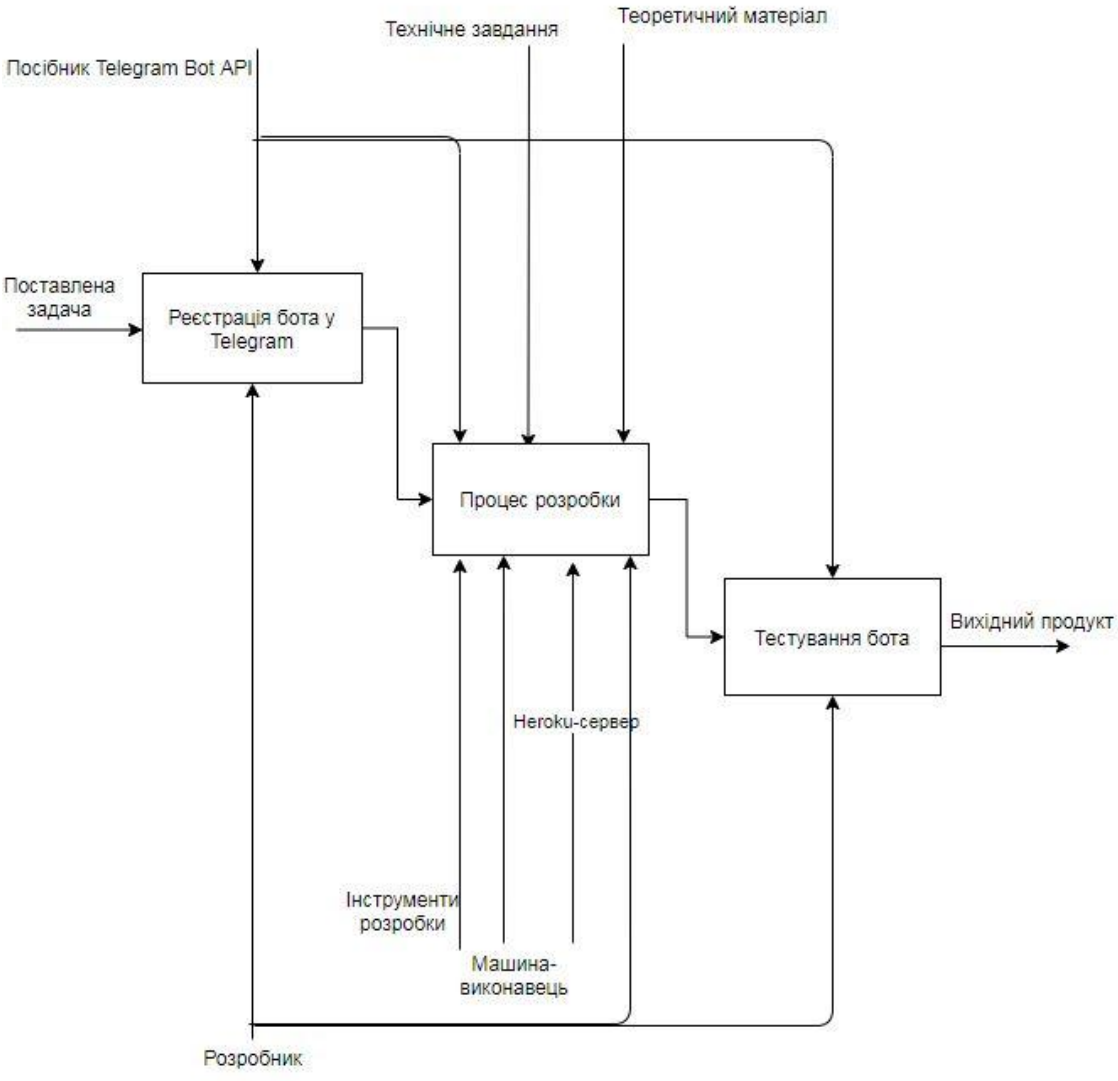


Рисунок Б.4 – Діаграма декомпозиції розробки бота

ДОДАТОК В

(Обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

POWERPOINT NINJA

Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

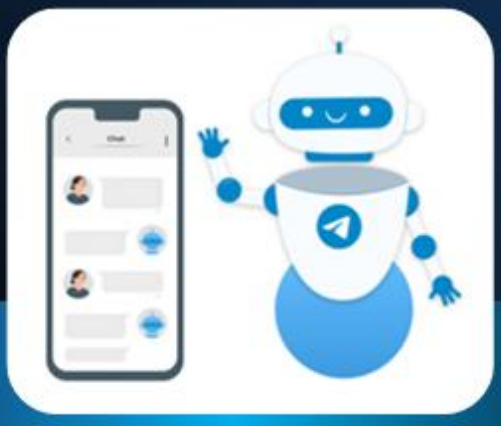
На тему:

“Телеграм-бот для розкладу занять в університеті”

Підготував:
Студент групи ІПЗ-**20-1**
Остапчук Микита Сергійович

Керівник кваліфікаційної роботи
канд. техн. наук, доцент
Яшина Оксана Миколаївна

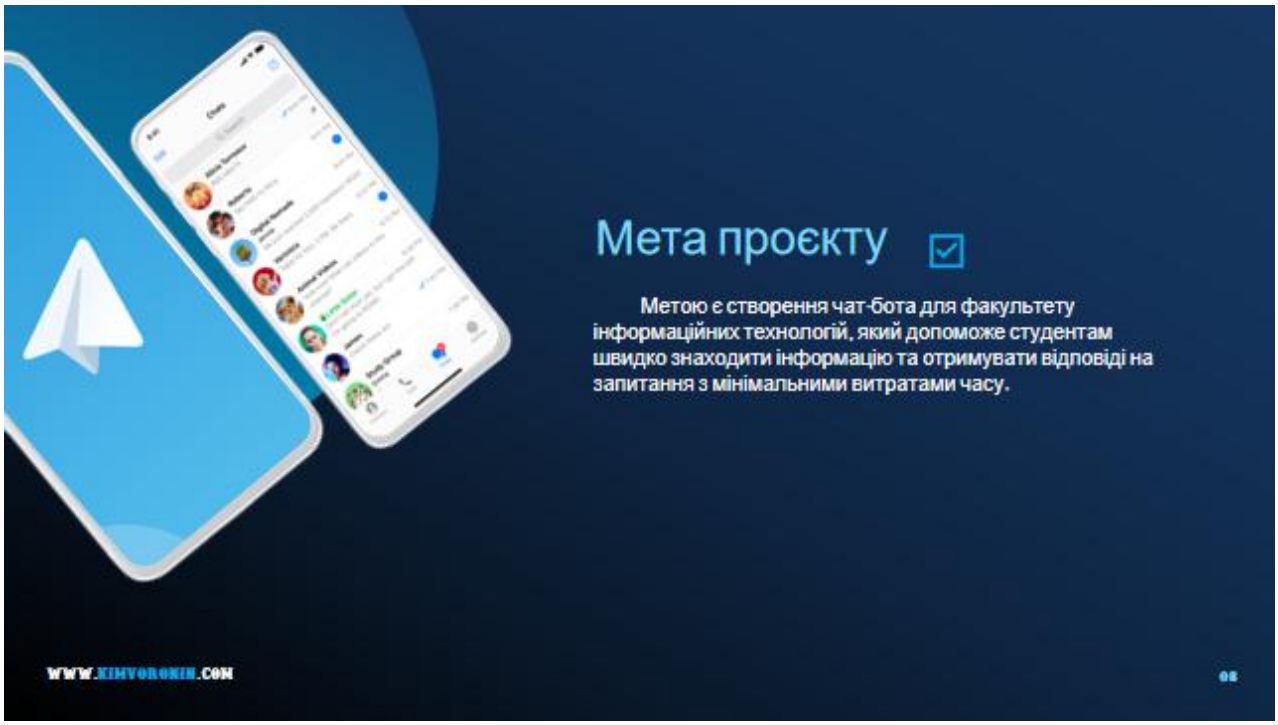
POWERPOINT NINJA

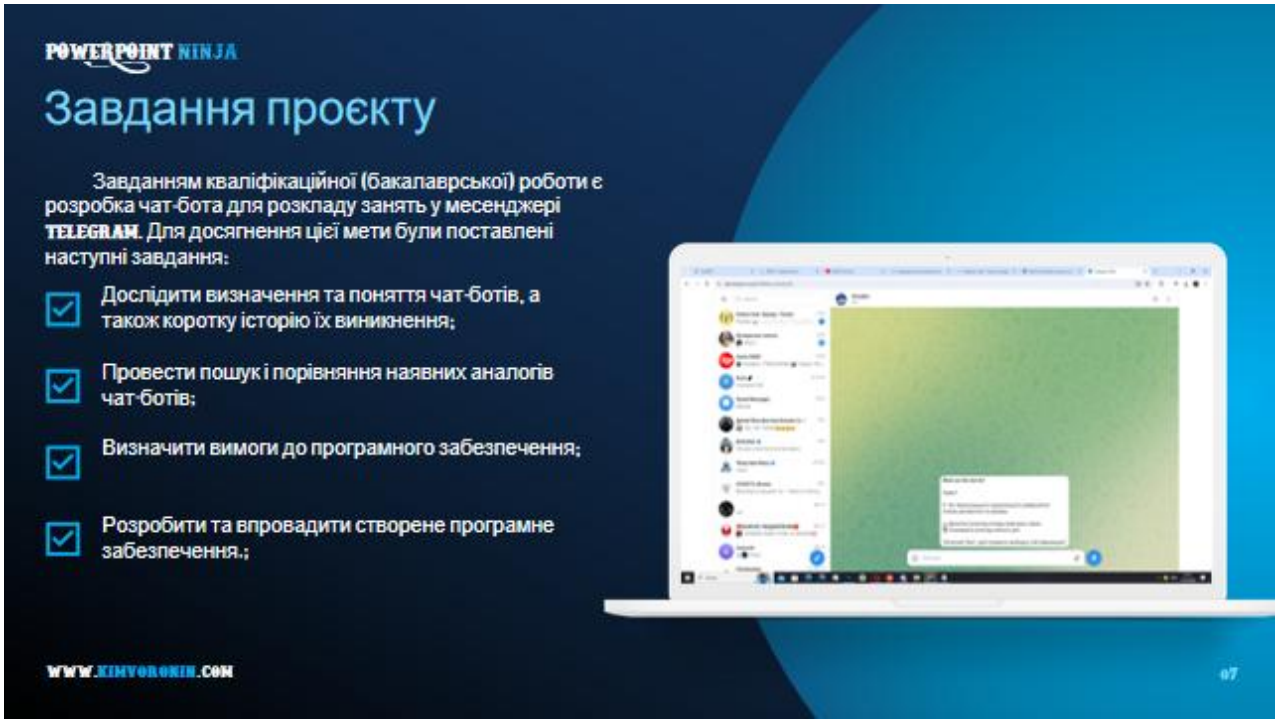


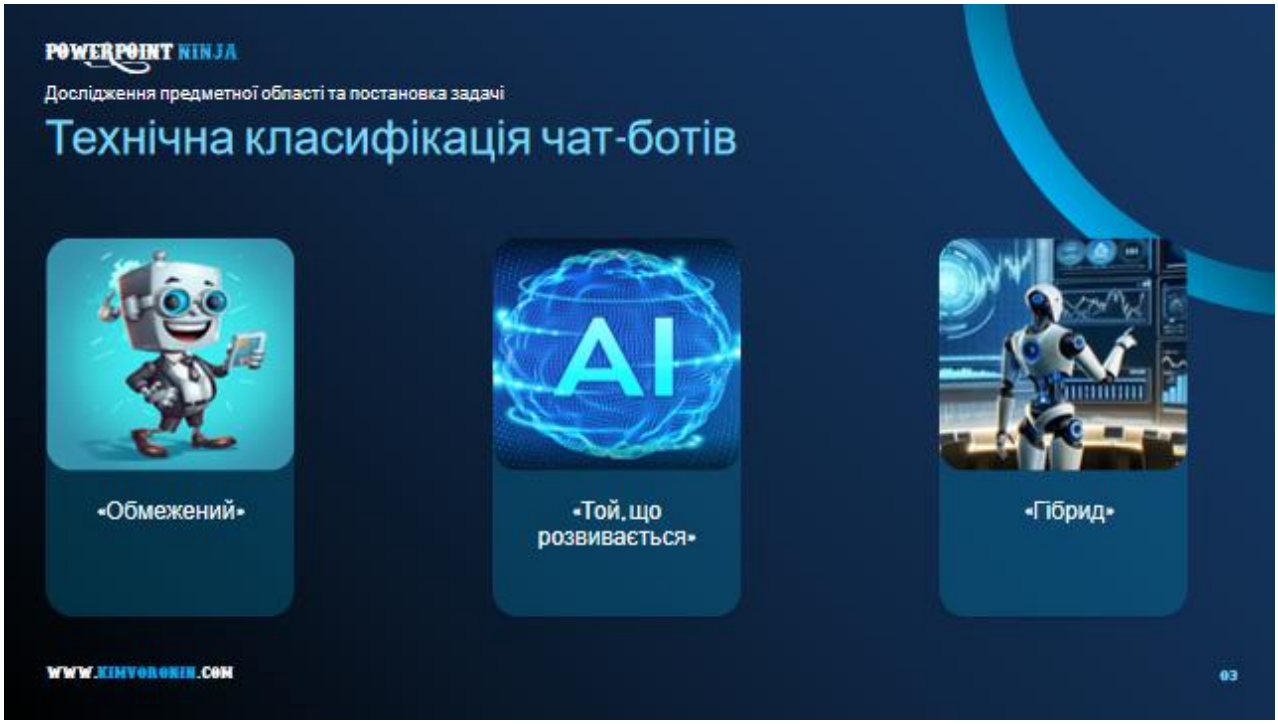
Актуальність теми проєкту:

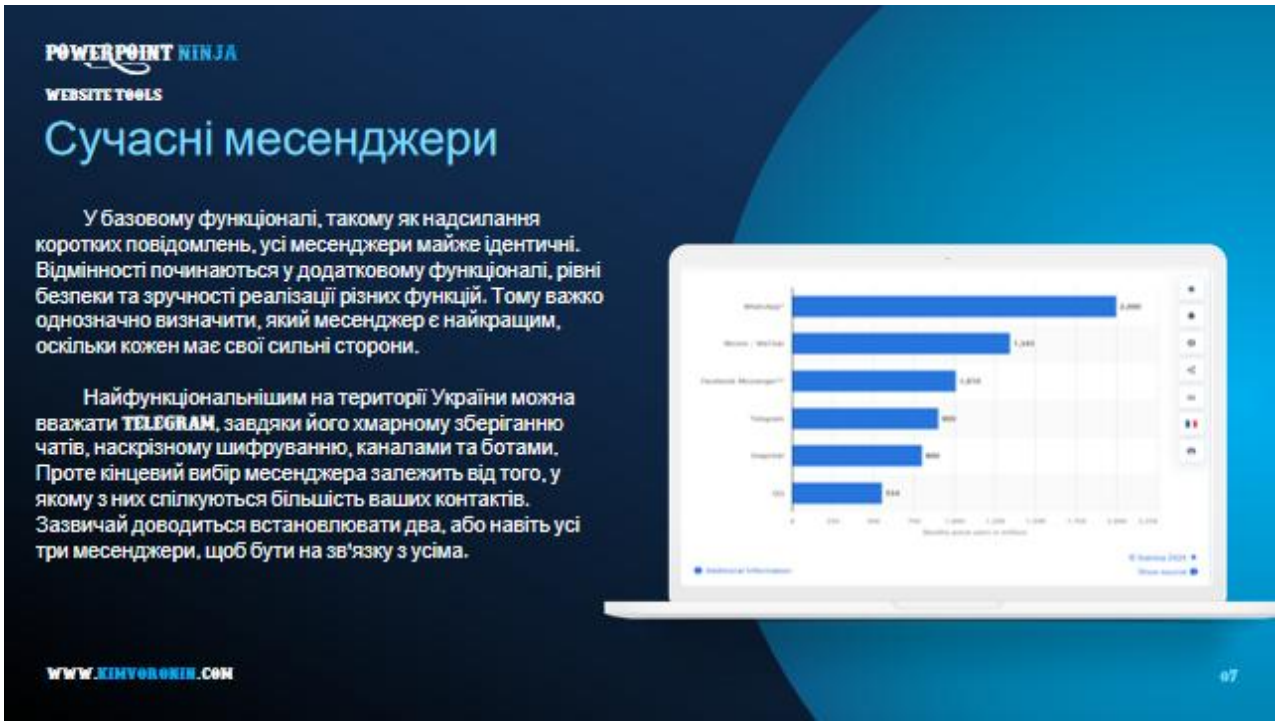
Зростання використання чат-ботів у якості сучасного інструменту комунікації пояснюється тим, що вони все частіше використовуються в різних галузях для встановлення спілкування з інтернет-користувачами. Особливу популярність чат-боти здобули, коли їх інтегрували у мессенджери та соціальні мережі, зокрема, у **TELEGRAM** та **FACEBOOK**, де з'явилися цілі вітрини та магазини чат-ботів.

Чат-бот спрощує рутинні завдання студентів, такі як отримання інформації про розклад занять, дзвінки, зміни в розкладі та інше. Головна перевага чат-бота полягає в тому, що всі ці можливості об'єднані на платформі єдиного месенджера.


Мета проєкту
Метою є створення чат-бота для факультету інформаційних технологій, який допоможе студентам швидко знаходити інформацію та отримувати відповіді на запитання з мінімальними витратами часу.
WWW.KIMVOROKIN.COM
08


POWERPOINT NINJA
Завдання проєкту
Завданням кваліфікаційної (бакалаврської) роботи є розробка чат-бота для розкладу занять у месенджері TELEGRAM. Для досягнення цієї мети були поставлені наступні завдання:
Дослідити визначення та поняття чат-ботів, а також коротку історію їх виникнення;
Провести пошук і порівняння наявних аналогів чат-ботів;
Визначити вимоги до програмного забезпечення;
Розробити та впровадити створене програмне забезпечення.;
WWW.KIMVOROKIN.COM
07


POWERPOINT NINJA
Дослідження предметної області та постановка задачі
Технічна класифікація чат-ботів
AI
«Обмежений»
«Той, що розвивається»
«Гібрид»
WWW.KIMVOROKIN.COM
03

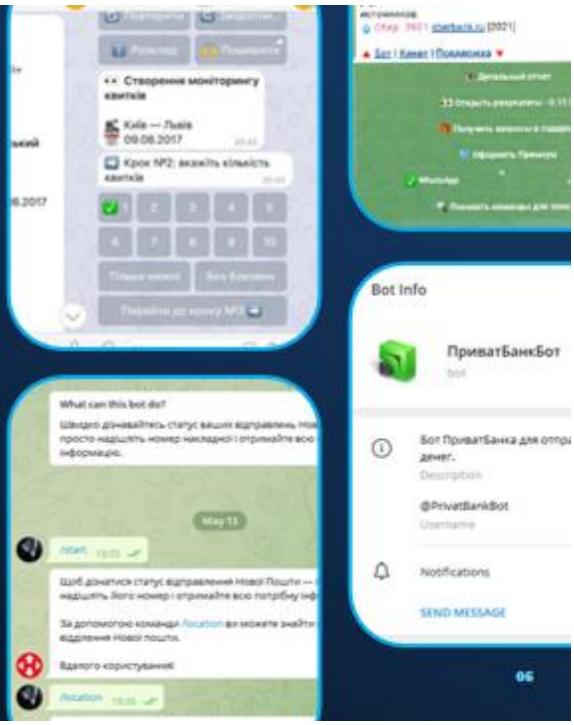

POWERPOINT NINJA
WEBSITE TOOLS
Сучасні месенджери
У базовому функціоналі, такому як надсилання коротких повідомлень, усі месенджери майже ідентичні. Відмінності починаються у додатковому функціоналі, рівні безпеки та зручності реалізації різних функцій. Тому важко однозначно визначити, який месенджер є найкращим, оскільки кожен має свої сильні сторони.
Найфункціональнішим на території України можна вважати TELEGRAM, завдяки його хмарному зберіганню чатів, наскрізному шифруванню, каналами та ботами. Проте кінцевий вибір месенджера залежить від того, у якому з них спілкуються більшість ваших контактів. Зазвичай доводиться встановлювати два, або навіть усі три месенджери, щоб бути на зв'язку з усіма.
WWW.KIMVOROKIN.COM
07

POWERPOINT NINJA

Дослідження предметної області та постановка задачі

Огляд існуючих рішень

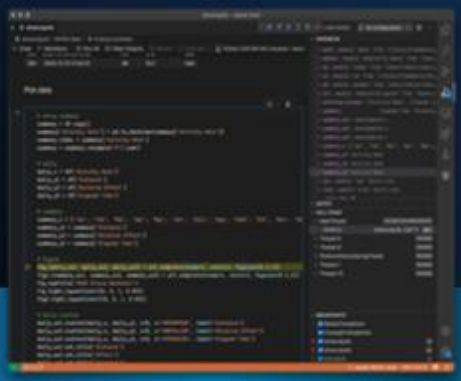
Оцінивши наявні рішення, було враховано, що швидкість бота має вирішальне значення. Якщо очікування триває кілька секунд, бот втрачає свою перевагу перед пошуком у **GOOGLE** чи відкриттям додатків. Штучно-інтелектні боти більше нагадують реальну людську комунікацію, тому мають перевагу над скриптовими ботами. Водночас, гібридні боти є більш зручними у використанні. Представлення готових варіантів спрощує комунікацію та зменшує кількість дій, що потрібно виконувати користувачу.



WWW.KIMVOROKIN.COM

06

POWERPOINT NINJA



Вибір технологій для розробки

На початку розробки будь-якої системи або програмного забезпечення кожен розробник аналізуючи висунуті вимоги до проекту має обрати стек технологій та засобів для реалізації продукту.

PYTHON — це відмінна мова програмування як для навчання, так і для реальної розробки. Він допомагає вирішити величезний спектр завдань!

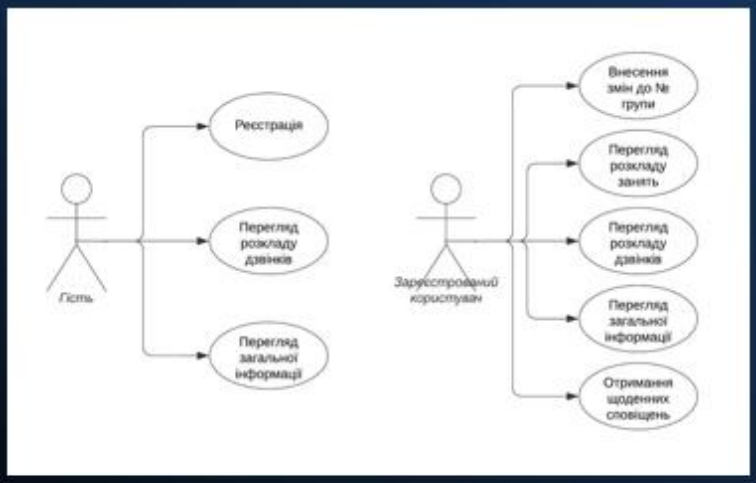
VISUAL STUDIO CODE є одним з найпопулярніших та найбільш широко використовуваних середовищ розробки на сьогодні. Це безкоштовне інтегроване середовище розробки, яке підтримує багато мов програмування. Він має розширення, що дозволяють розширити його функціональність, що забезпечує широкі можливості для розробки.

WWW.KIMVOROKIN.COM

02

POWERPOINT NINJA

Діаграма варіантів використання

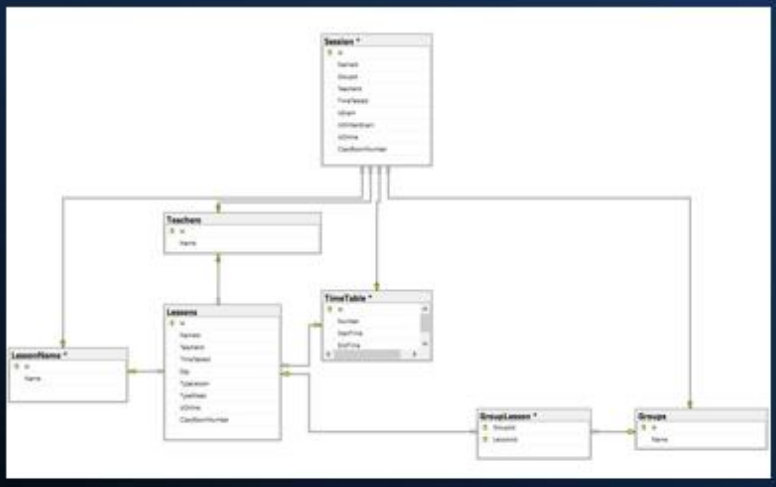


WWW.KIMVORONIN.COM

16

POWERPOINT NINJA

Схема бази даних

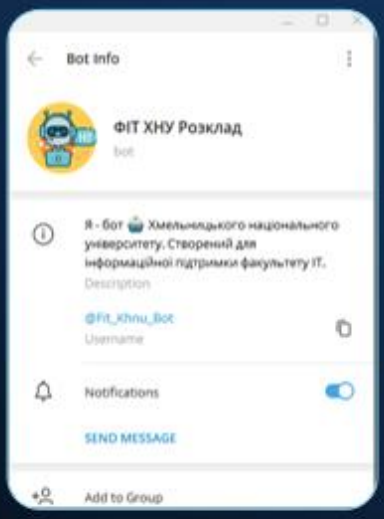


WWW.KIMVORONIN.COM

16

POWERPOINT NINJA

Опис чат-бота

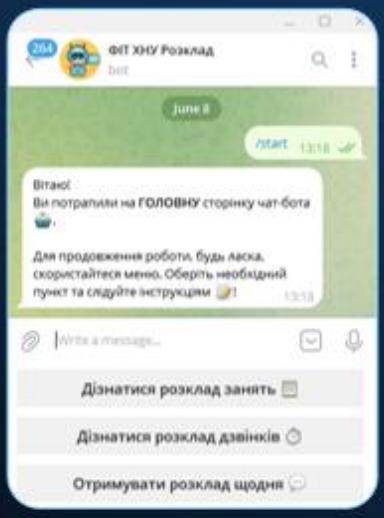


WWW.KIMVORONIN.COM

12

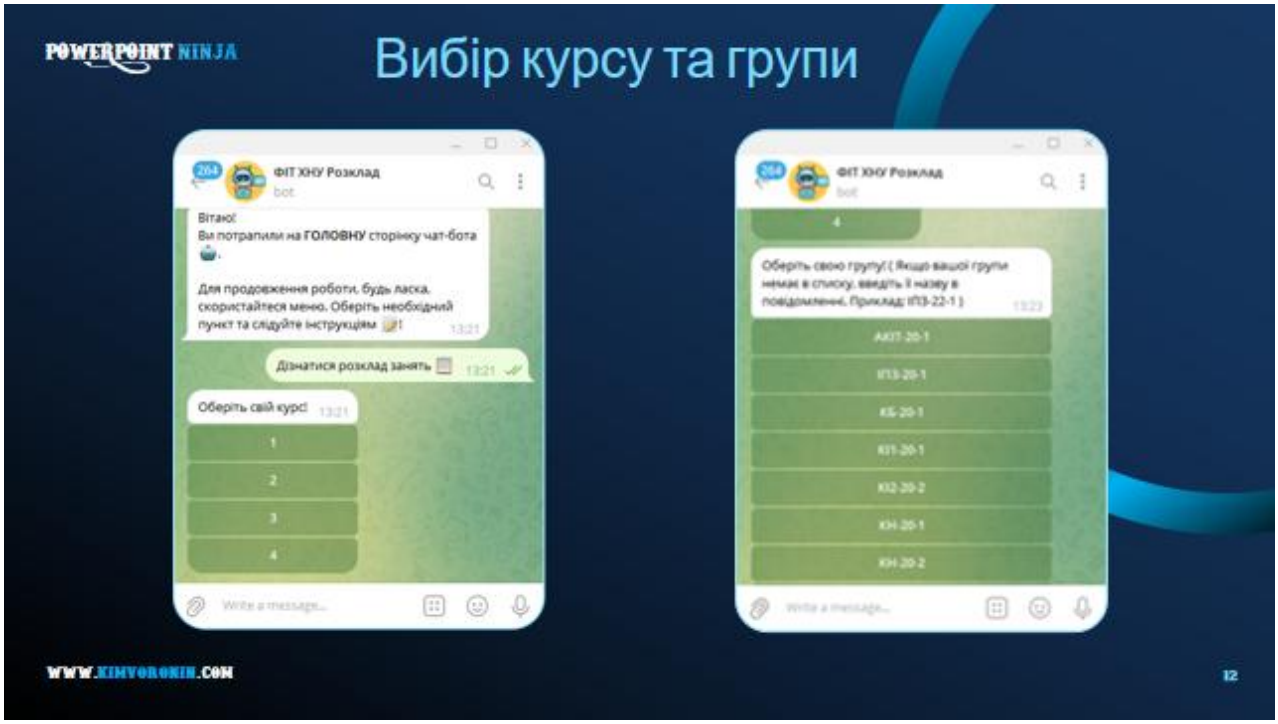
POWERPOINT NINJA

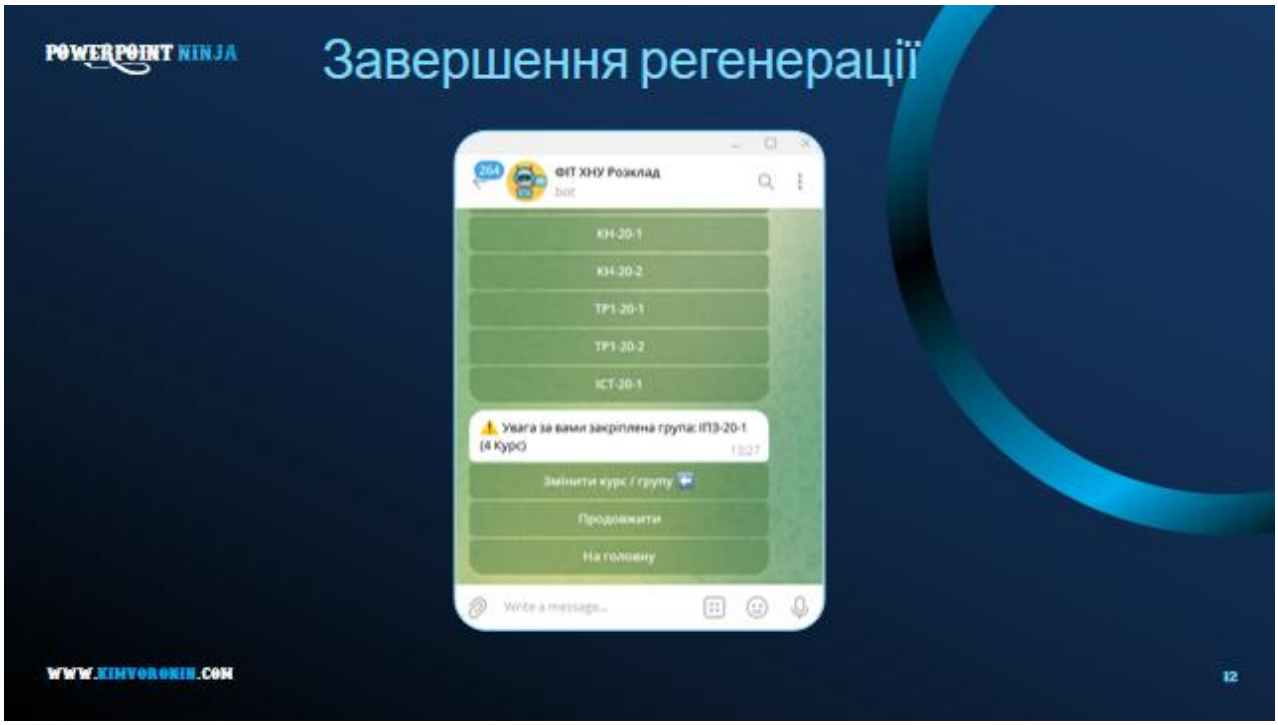
Головна сторінка



WWW.KIMVORONIN.COM

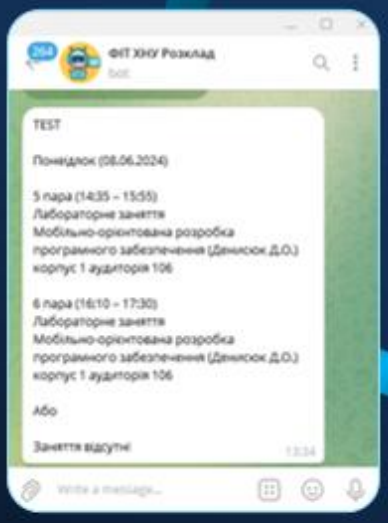
12


POWERPOINT NINJA
Вибір курсу та групи
ФІТ ХНУ Розклад
bot
Вітаю!
Ви потрапили на ГОЛОВНУ сторінку чат-бота
Для продовження роботи, будь ласка, скористайтеся меню. Оберіть необхідний пункт та слідуйте інструкціям!
13:21
Дізнатися розклад занять
13:21
Оберіть свій курс!
13:21
1
2
3
4
Write a message...
ФІТ ХНУ Розклад
bot
4
Оберіть свою групу! (Якщо вашої групи немає в списку, введіть її назву в повідомленні. Приклад: ІПЗ-22-1)
13:23
АКІТ-20-1
ІПЗ-20-1
КБ-20-1
КІ1-20-1
КІ2-20-2
КН-20-1
КН-20-2
Write a message...
WWW.KIMVORONIN.COM
12


POWERPOINT NINJA
Завершення регенерації
ФІТ ХНУ Розклад
bot
КН-20-1
КН-20-2
ТР1-20-1
ТР1-20-2
ІСТ-20-1
⚠ Увага за вами закріплена група: ІПЗ-20-1 (4 Курс)
13:27
Змінити курс / групу
Продовжити
На головну
Write a message...
WWW.KIMVORONIN.COM
12

POWERPOINT NINJA

Розклад занять

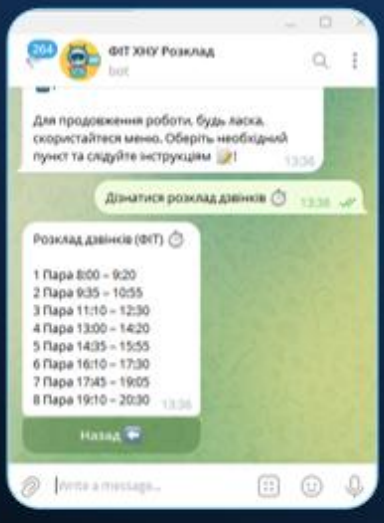


WWW.KIMVORONIN.COM

12

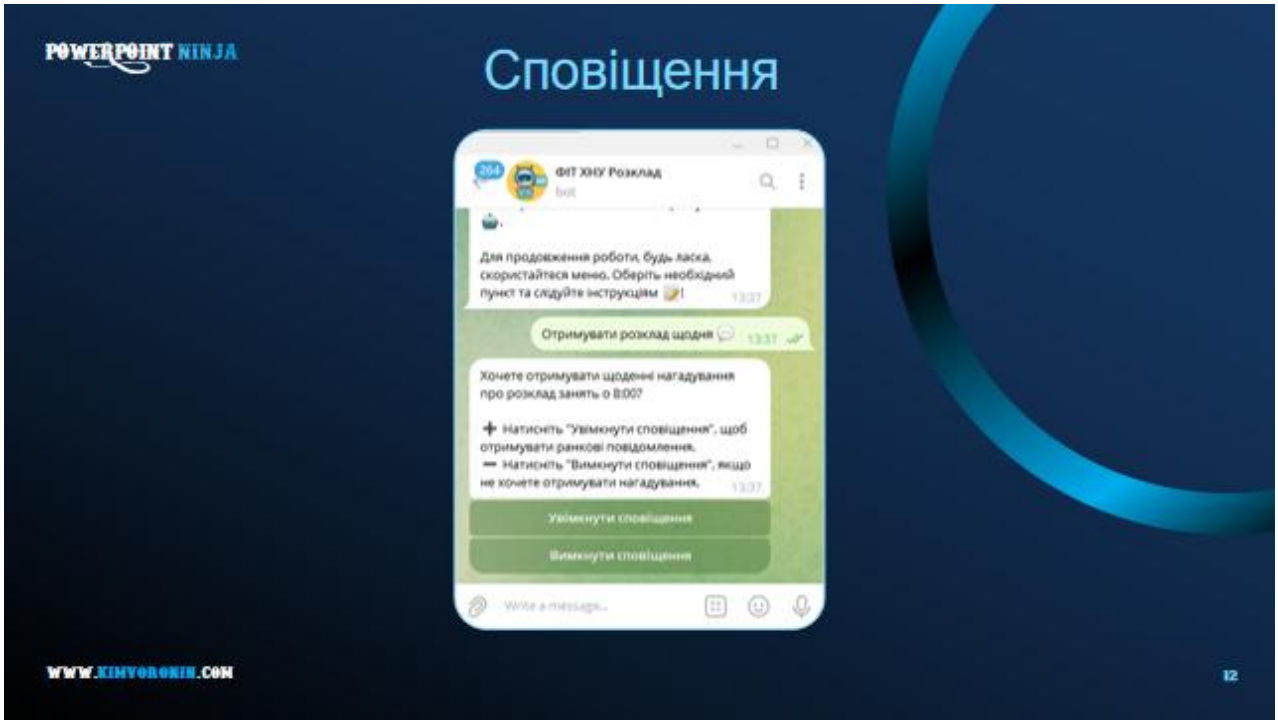
POWERPOINT NINJA

Розклад дзвінків



WWW.KIMVORONIN.COM

12


POWERPOINT NINJA
Сповіщення
ФІТ ХНУ Розклад
bot
Для продовження роботи, будь ласка, скористайтеся меню. Оберіть необхідний пункт та слідуйте інструкціям
Отримувати розклад щодня
Хочете отримувати щоденні нагадування про розклад занять о 8:00?
Натисніть "Увімкнути сповіщення", щоб отримувати ранкові повідомлення.
Натисніть "Вимкнути сповіщення", якщо не хочете отримувати нагадування.
Увімкнути сповіщення
Вимкнути сповіщення
Write a message...
WWW.KIMVORONIN.COM
12


POWERPOINT NINJA
ВИСНОВКИ
В ході виконання цієї кваліфікаційної роботи було створено телеграм бот для відображення розкладу занять факультету ІТ ХНУ, що вирішує важливу задачу автоматизації та спрощення доступу до інформації про заняття для студентів.
На етапі програмної реалізації та тестування було здійснено реєстрацію телеграм бота та розроблено необхідні програмні модулі. Підготовлено керівництво користувача, визначено вимоги до технічних та програмних засобів. Заключним етапом стало тестування програмної системи, яке підтвердило її працездатність та відповідність поставленим вимогам.
Результати розробки чітко демонструють, що створений телеграм бот успішно впорався з основною метою - надати студентам зручний та ефективний доступ до розкладу занять. Система виявилася надійною, зручною у використанні та легкою в адмініструванні, що підтверджує її доцільність.
WWW.KIMVORONIN.COM

ДОДАТОК Г

(Обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

autoposting/auto_posting_thread.py

```
import logging
import threading
from datetime import datetime, timedelta
from time import sleep

import telebot

from config import config
from helpers import daysOfWeek, get_date_keyboard, get_week_type
from scheduleCreator import create_schedule_text
from scheduledb import ScheduleDB
from statistic import track

bot = telebot.AsyncTeleBot(config["Тут необхідно ввести токен вашого чат-бота"])


logging.basicConfig(format='%(asctime)-15s [ %(levelname)s ] uid=%(userid)s %(message)s',
                    filemode='a',
                    filename=config["LOG_DIR_PATH"] + "log-
{0}.log".format(datetime.now().strftime("%Y-%m-%d")),
                    level="INFO")
logger = logging.getLogger('bot-logger')


def auto_posting(current_time, day, week_type, is_today=True):
    # Вибірка користувачів із бази
    with ScheduleDB(config) as db:
        users = db.find_users_where(auto_posting_time=current_time, is_today=is_today)

    if users is None:
        return None
    try:
        for user in users:
            cid = user[0]
            tag = user[1]

            schedule = create_schedule_text(tag, day[0], week_type)
            if len(schedule[0]) <= 14:
                continue
            bot.send_message(cid, schedule, reply_markup=get_date_keyboard())

            # Статистика
            if config['STATISTIC_TOKEN'] != '':
                track(config['STATISTIC_TOKEN'], cid, current_time, 'auto_posting')
            else:
                logger.info('auto_posting. Time: {0}'.format(current_time), extra={'userid':
cid})
```

```
    except BaseException as e:
        logger.warning('auto_posting: {0}'.format(str(e)), extra={'userid': 0})


def today_schedule(current_time):
    today = datetime.now()
    week_type = get_week_type(today)

    if datetime.weekday(today) == 6:
        today += timedelta(days=1)
        week_type = (week_type + 1) % 2

    day = [daysOfWeek[datetime.weekday(today)]]

    auto_posting(current_time, day, week_type)


def tomorrow_schedule(current_time):
    tomorrow = datetime.now()
    tomorrow += timedelta(days=1)
    week_type = get_week_type(tomorrow)

    # Вибірка користувачів із бази, у яких встановлено відправлення розкладу на завтрашній день

        if datetime.weekday(tomorrow) == 6:
        tomorrow += timedelta(days=1)
        week_type = (week_type + 1) % 2

    day = [daysOfWeek[datetime.weekday(tomorrow)]]

    auto_posting(current_time, day, week_type, is_today=False)


if __name__ == "__main__":
    while True:
        # Надсилання розкладу на сьогодні

threading.Thread(target=today_schedule(datetime.now().time().strftime("%H:%M:00"))).start()

        # Обчислюємо різницю в секундах, між початком хвилини і часом завершення потоку
        time_delta = datetime.now() - datetime.now().replace(second=0, microsecond=0)
        # Потік засинає на час, що дорівнює кількості секунд до наступної хвилини
        sleep(60 - time_delta.seconds)
```

autoposting/config.py

```
import configparser
import os

config_file = configparser.ConfigParser()
```

```
config = config_file['DEFAULT']

current_path = os.path.abspath(os.path.dirname(__file__))
config_file.read(current_path + '/' + "config.ini")
```

autoposting/helpers.py

```
from telebot import types
from datetime import datetime
from config import config


daysOfWeek = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

ScheduleType = {
    "Понеділок": daysOfWeek[0],
    "Вівторок": daysOfWeek[1],
    "Середа": daysOfWeek[2],
    "Четвер": daysOfWeek[3],
    "Пятниця": daysOfWeek[4],
    "Субота": daysOfWeek[5],
    "Неділя": daysOfWeek[6],
    "Сьогодні": "Today",
    "Завтра": "Tomorrow",
    "Весь тиждень": daysOfWeek
}

daysOfWeek_ua = {
    daysOfWeek[0]: "Понеділок",
    daysOfWeek[1]: "Вівторок",
    daysOfWeek[2]: "Середа",
    daysOfWeek[3]: "Четвер",
    daysOfWeek[4]: "Пятниця",
    daysOfWeek[5]: "Субота",
    daysOfWeek[6]: "Неділя",
}


def get_date_keyboard():
    now = datetime.now()

    date_select = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True,
one_time_keyboard=False)

    if now.month == 1 or now.month == 12 or now.month == 5 or now.month == 6:
        date_select.row('Іспити')

    date_select.row("Сьогодні")
    date_select.row("Завтра")
    date_select.row("Весь тиждень")
```

```
    date_select.row("Понеділок", "Вівторок")
    date_select.row("Середа", "Четвер")
    date_select.row("Пятниця", "Субота")

    return date_select


def get_week_type(day):
    return (day.isocalendar()[1] + int(config["WEEK_TYPE"])) % 2
```

autoposting/scheduleCreator.py

```
from functools import lru_cache

import scheduledb
from config import config
from helpers import daysOfWeek_rus


def print_type(raw_type, week_type=-1):
    # Якщо задано тип тижня (чисельник/знаменник), тобто week_type не дорівнює значенню за
замовчуванням,
    # то додаткова інформація про тип тижня не виводиться
    if week_type != -1:
        return ""

    raw_type = int(raw_type)
    if raw_type == 0:
        return "чисельник"
    elif raw_type == 1:
        return "знаменник"
    elif raw_type == 2:
        return ""


@lru_cache(maxsize=128)
def create_schedule_text(tag, day, week_type=-1):
    result = []
    schedule = ""
    try:
        with scheduledb.ScheduleDB(config) as db:
            data = db.get_schedule(tag, day, week_type)

        schedule += ">{0}:\n".format(daysOfWeek_rus[day])
        index = 0
        while index < len(data):
            row = data[index]

            title = ' '.join(str(row[1]).split())
            classroom = ' '.join(str(row[2]).split())
```

```
            schedule += str(row[0]) + " пара:\n"
            # Цей блок потрібен для виведення тих занять, де заняття по чисельнику та знаменнику
різняться
            if index != len(data) - 1:
                if data[index + 1][0] == row[0]:
                    schedule += '{0} {1} {2}\n'.format(title, classroom, print_type(row[3],
week_type))

                    index += 1
                    row = data[index]
                    title = ' '.join(str(row[1]).split())
                    classroom = ' '.join(str(row[2]).split())

                    schedule += '{0} {1} {2}\n'.format(title, classroom, print_type(row[3],
week_type))
                else:
                    schedule += '{0} {1} {2}\n'.format(title, classroom, print_type(row[3],
week_type))
            else:
                schedule += '{0} {1} {2}\n'.format(title, classroom, print_type(row[3],
week_type))

            schedule += "------------\n"
            index += 1
        result.append(schedule)
    except:
        pass
    finally:
        return result


def create_schedule_xls(tag, day):
    pass


def create_schedule_pdf(tag, day):
    Pass
```

autoposting/scheduledb.py

```
import hashlib
import logging
import psycopg2
from datetime import datetime

class ScheduleDB:
    def __init__(self, config):
        self.con = psycopg2.connect(
```

```
            dbname=config["DB_NAME"],
            user=config["DB_USER"],
            password=config["DB_PASSWORD"],
            host=config["DB_HOST"])
        self.cur = self.con.cursor()

        logging.basicConfig(format='%(asctime)-15s [ %(levelname)s ] %(message)s',
                            filemode='a',
                            filename=config["LOG_DIR_PATH"] + "log-
{0}.log".format(datetime.now().strftime("%Y-%m")))
        self.logger = logging.getLogger('db-logger')

    def __enter__(self):
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        self.con.commit()
        self.con.close()

    @staticmethod
    def create_tag(organization, faculty, group):
        org_hash = hashlib.sha256(organization.encode('utf-8')).hexdigest()
        faculty_hash = hashlib.sha256(faculty.encode('utf-8')).hexdigest()
        group_hash = hashlib.sha256(group.encode('utf-8')).hexdigest()
        Return faculty_hash[:довжина_поля_факультету] + \
               group_hash[:довжина_поля_групи]

    def add_lesson(self, tag, day, number, week_type, time_start, time_end, title, classroom,
lecturer):
        try:
            self.cur.execute('INSERT INTO schedule(tag, day, "number", type, "startTime",
"endTime", \
                             title, classroom, lecturer) VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s);',
                             (tag, day, number, week_type, time_start, time_end, title,
classroom, lecturer))
            self.con.commit()
            return True
        except BaseException as e:
            self.logger.warning('Add to schedule failed. Error: {0}. Data:\
                            tag={1},\
                            day={2},\
                            number={3},\
                            week_type={4},\
                            time_start={5},\
                            time_end={6},\
                            title={7},\
                            classroom={8},\
                            lecturer={9}'.format(
```

```
                str(e), tag, day, number, week_type, time_start, time_end, title, classroom,
lecturer))
            return False

    def add_exam(self, tag, title, classroom, lecturer, day):
        try:
            self.cur.execute("INSERT INTO examinations(tag, title, classroom, lecturer, day)
VALUES(%s,%s,%s,%s,%s);",
                             (tag, title, classroom, lecturer, day))
            self.con.commit()
            return True
        except BaseException as e:
            self.logger.warning("Add exam failed. Error: {0}. Data:\
                            tag={1},\
                            title={2},\
                            classroom={3},\
                            lecturer={4},\
                            day={5}".format(str(e), tag, title, classroom, lecturer, day))
            return False

    def add_organization(self, organization, faculty, group):
        tag = self.create_tag(organization, faculty, group)
        try:
            self.cur.execute("INSERT INTO organizations(organization, faculty, studgroup, tag)
VALUES(%s,%s,%s,%s);",
                             (organization, faculty, group, tag))
            self.con.commit()
            return tag
        except BaseException as e:
            self.logger.warning("Add organization failed. Error: {0}. Data:\
                            organization={1},\
                            faculty={2},\
                            group={3},\
                            tag={4}".format(str(e), organization, faculty, group, tag))
            return None

    def add_report(self, cid, report):
        try:
            self.cur.execute('INSERT INTO reports (type, user_id, report, date)
VALUES(%s, %s, %s, %s)',
                             ('tg', cid, report, datetime.now().strftime("%Y-%m-%d %H:%M:%S")))
            self.con.commit()
            return True
        except BaseException as e:
            self.logger.warning('Add report failed. Error: {0}. Data: cid={1},
report={2}'.format(str(e), cid, report))
            return False

    def add_user(self, cid, name, username, tag):
```

```
        try:
            self.cur.execute('INSERT INTO users VALUES(%s,%s,%s,%s,%s,null,null)', ('tg', cid,
name, username, tag))
            self.con.commit()
            return True
        except BaseException as e:
            self.logger.warning('Add user failed. Error: {0}. Data: cid={1}, name={2},
username={3}, tag={4}'.format(
                str(e), cid, name, username, tag))
            raise e

    def update_user(self, cid, name, username, tag):
        try:
            self.cur.execute('UPDATE users SET "scheduleTag" = (%s) WHERE id = (%s) AND type =
(%s)', (tag, cid, 'tg'))
            self.con.commit()
            return True
        except BaseException as e:
            self.logger.warning('Update user failed. Error: {0}. Data: cid={1}, name={2},
username={3}, tag={4}'.format(
                str(e), cid, name, username, tag))
            raise e

    def find_user(self, cid):
        try:
            self.cur.execute('SELECT "scheduleTag" FROM users WHERE id = (%s) AND type = (%s)',
(cid, 'tg'))
            return self.cur.fetchone()
        except BaseException as e:
            self.logger.warning('Select user failed. Error: {0}. Data: cid={1}'.format(str(e),
cid))
            raise e

    def find_users_where(self, auto_posting_time=None, is_today=None):
        try:
            if auto_posting_time is not None and is_today is not None:
                self.cur.execute('SELECT id, "scheduleTag" FROM users \
                                  WHERE auto_posting_time = %s AND is_today = %s  AND type =
(%s)',
                                 (auto_posting_time, is_today, 'tg'))
                return self.cur.fetchall()
            elif auto_posting_time is not None:
                self.cur.execute('SELECT id, "scheduleTag" FROM users \
                                  WHERE auto_posting_time = %s AND type = (%s)',
                                 (auto_posting_time, 'tg'))
                return self.cur.fetchall()
            elif is_today is not None:
                self.cur.execute('SELECT id, "scheduleTag" FROM users WHERE is_today = %s AND
type = (%s)',
```

```
                                (is_today, 'tg'))
                return self.cur.fetchall()
            else:
                self.cur.execute('SELECT id, "scheduleTag" FROM users WHERE type = (%s)', ['tg'])
                return self.cur.fetchall()
        except BaseException as e:
            self.logger.warning('Select users failed. Error: {0}. auto_posting_time={1}'.format(
                str(e), auto_posting_time))
            raise e

    def get_exams(self, tag):
        exams = []
        try:
            self.cur.execute("SELECT day, title, classroom, lecturer FROM examinations \
                              WHERE tag = (%s) ORDER BY day", [str(tag)])
            exams = self.cur.fetchall()
        except BaseException as e:
            self.logger.warning('Select exams failed. Error: {0}. Data: tag={1}'.format(str(e),
tag))
            raise e
        finally:
            return exams

    def get_schedule(self, tag, day, week_type=-1):
        data = []
        try:
            if week_type != -1:
                self.cur.execute('SELECT number,title,classroom,type FROM schedule \
                            WHERE tag = (%s) AND day = (%s) AND (type = 2 OR type = %s) \
                            ORDER BY number, type ASC', (tag, day, week_type))
            else:
                self.cur.execute('SELECT number,title,classroom,type FROM schedule \
                            WHERE tag = (%s) AND day = (%s) ORDER BY number, type ASC', (tag,
day))
            data = self.cur.fetchall()
        except BaseException as e:
            self.logger.warning('Select schedule failed. Error: {0}. Data: tag={1}, day={2},
week_type={3}'.format(
                str(e), tag, day, week_type))
            raise Exception
        finally:
            return data

    def get_organizations(self, tag=""):
        organizations = []
        try:
            self.cur.execute("SELECT DISTINCT ON (organization) organization, tag \
            FROM organizations WHERE tag LIKE %s ORDER BY organization;", [tag + '%'])
            organizations = self.cur.fetchall()
```

```
        except BaseException as e:
            self.logger.warning('Select schedule failed. Error: {0}. Data:
tag={1}'.format(str(e), tag))
            raise e
        finally:
            return organizations

    def get_faculty(self, tag=""):
        faculties = []
        try:
            self.cur.execute("SELECT DISTINCT ON (faculty) faculty, tag \
            FROM organizations WHERE tag LIKE %s ORDER BY faculty;", [tag + '%'])
            faculties = self.cur.fetchall()
        except BaseException as e:
            self.logger.warning('Select schedule failed. Error: {0}. Data:
tag={1}'.format(str(e), tag))
            raise e
        finally:
            return faculties

    def get_group(self, tag=""):
        group = []
        try:
            self.cur.execute("SELECT DISTINCT ON (studGroup) studGroup, tag \
            FROM organizations WHERE tag LIKE %s ORDER BY studGroup;",
                             [tag + '%'])
            group = self.cur.fetchall()
        except BaseException as e:
            self.logger.warning('Select group failed. Error: {0}. Data: tag={1}'.format(str(e),
[tag]))
            raise e
        finally:
            return group

    def set_auto_post_time(self, cid, time, is_today):
        try:
            self.cur.execute('UPDATE users SET auto_posting_time = %s, is_today = %s \
            WHERE id = %s AND type = (%s)',
                             (time, is_today, cid, 'tg'))
            self.con.commit()
            return True
        except BaseException as e:
            self.logger.warning('Set auto post time failed. Error: {0}. Data: cid={1},
auto_posting_time={2}'.format(
                str(e), cid, time))
            raise e

    def clear_tables(self):
        try:
```

```
            self.cur.execute('TRUNCATE users;')
            self.cur.execute('TRUNCATE organizations CASCADE;')
            self.cur.execute('TRUNCATE reports;')
            self.con.commit()

            old_isolation_level = self.con.isolation_level
            self.con.set_isolation_level(0)

            self.cur.execute('VACUUM')
            self.con.commit()

            self.con.set_isolation_level(old_isolation_level)

            return True
        except BaseException as e:
            self.logger.warning('clear tables failed. Error: {0}.'.format(
                str(e)))
            raise e
```

autoposting/statistic.py

```
from multiprocessing import Process


def send_statistic(token, uid, message, intent, user_type=None):
    try:
        if user_type is not None:
            pass
        else:
            if intent != 'unknown':
                pass
            else:
                pass

        return ''
    except:
        pass


def track(token, uid, message, intent, user_type=None):
    Process(target=send_statistic, args=(token, uid, message, intent, user_type)).start()
```

bot/UniversityScheduleBot.py

```
import logging
import re
from datetime import datetime, time, timedelta

import flask
from flask import request, jsonify

import telebot
```

```
from telebot import types

from config import config
from helpers import daysOfWeek, ScheduleType, get_date_keyboard, get_week_type
from scheduleCreator import create_schedule_text
from scheduledb import ScheduleDB, organization_field_length, faculty_field_length

# Статистика
from statistic import track

WEBHOOK_URL_BASE = "https://{}:{}".format(config["WEBHOOK_HOST"], config["WEBHOOK_PORT"])
WEBHOOK_URL_PATH = "/{}/".format(config["TOKEN"])

bot = telebot.AsyncTeleBot(config["TOKEN"])
app = flask.Flask(__name__)

logger = telebot.logger
telebot.logger.setLevel(logging.INFO)

commands = {  # Опис команд, що використовується в команді "help"
    'start': 'Стартове повідомлення та пропозиція зареєструватися',
    'help': 'Інформація про бота та список доступних команд',
    'registration': 'Вибір курсу та групи для виведення розкладу',
    'auto_posting_on': 'Увімкнення та вибір часу для автоматичної відправки розкладу в діалог',
    'auto_posting_off': 'Вимкнення автоматичної відправки розкладу'
}



# обробник команди "/registration"
@bot.message_handler(commands=['registration'])
def command_registration(m):
    cid = m.chat.id

    # Процедура реєстрації проходить в чотири етапи:
    # 1 етап: вибір навчального курсу <--
    # 2 етап: вибір групи
    # 3 етап: додавання даних про належність користувача до групи в БД
    try:
        # Статистика
        if config['STATISTIC_TOKEN'] != '':
            track(config['STATISTIC_TOKEN'], cid, 'stage 1', 'registration-stage-1')

        keyboard = types.InlineKeyboardMarkup()

        with ScheduleDB(config) as db:
            result = db.get_organizations()
        for row in result:
            callback_button = types.InlineKeyboardButton(
```

```
                text=str(row[0]),
                callback_data="reg:stage 2:{0}".format(str(row[1])[:organization_field_length]))
            keyboard.add(callback_button)

        bot.send_message(cid, "Оберіть ваш курс:", reply_markup=keyboard)
    except BaseException as e:
        logger.warning('Registration problem: {0}'.format(str(e)))
        bot.send_message(cid, "Сталося щось дивне, спробуйте почати спочатку, ввівши команду
/registration")


# обробник команди "/start"
@bot.message_handler(commands=['start'])
def command_start(m):
    # Статистика
    if config['STATISTIC_TOKEN'] != '':
        track(config['STATISTIC_TOKEN'], m.chat.id, m.text, 'start')
    else:
        logger.info('start')

    cid = m.chat.id
    command_help(m)

    try:
        with ScheduleDB(config) as db:
            user = db.find_user(cid)
        if user and user[0] is not None:
            bot.send_message(cid, "Ви вже додані до бази даних",
       reply_markup=get_date_keyboard())
        else:
            bot.send_message(cid, "Вас ще немає в базі даних, тому пройдіть просту процедуру
реєстрації")
            command_registration(m)
    except BaseException as e:
        logger.warning('command start: {0}'.format(str(e)))
        bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново",
                         reply_markup=get_date_keyboard())
# help page
@bot.message_handler(commands=['help'])
def command_help(m):
    # Статистика
    if config['STATISTIC_TOKEN'] != '':
        track(config['STATISTIC_TOKEN'], m.chat.id, m.text, 'help')
    else:
        logger.info('help')

    cid = m.chat.id
    help_text = "Доступні наступні команди: \n"
    for key in commands:
```

```
        help_text += "/" + key + ": "
        help_text += commands[key] + "\n"
    bot.send_message(cid, help_text, reply_markup=get_date_keyboard())

    help_text = ('Опис кнопок:\nКнопка "Сьогодні", як це не дивно, виводить розклад на
сьогоднішній день, '
                 'причому з урахуванням типу тижня (чисельник/знаменник), але є один нюанс: якщо
сьогодні неділя '
                 'або час більше ніж 21:30, то виводиться розклад на наступний день\n')
    bot.send_message(cid, help_text, reply_markup=get_date_keyboard())

@bot.message_handler(commands=['send_report'])
def command_send_report(m):
    # Статистика
    if config['STATISTIC_TOKEN'] != '':
        track(config['STATISTIC_TOKEN'], m.chat.id, m.text, 'report')
    else:
        logger.info('report')

    cid = m.chat.id
    data = m.text.split("/send_report")

    if data[1] != '':
        report = data[1]
        with ScheduleDB(config) as db:
            if db.add_report(cid, report):
                bot.send_message(cid, "")
            else:
                bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново",
                                 reply_markup=get_date_keyboard())
    else:
        bot.send_message(
            cid,
            "Ви відправили порожній рядок. Приклад: /send_report <повідомлення>",
            reply_markup=get_date_keyboard())


# handle the "/auto_posting_on" command
@bot.message_handler(commands=['auto_posting_on'])
def command_auto_posting_on(m):
    # Статистика
    if config['STATISTIC_TOKEN'] != '':
        track(config['STATISTIC_TOKEN'], m.chat.id, m.text, 'auto_posting_on')
    else:
        logger.info('auto_posting_on')

    cid = m.chat.id

    try:
```

```
        data = m.text.split("/auto_posting_on")[1].strip()
        if re.match(data, r'\d{1,2}:\d\d'):
            raise BaseException
    except:
        bot.send_message(cid, "Ви відправили порожній рядок або рядок неправильного формату.
Правильний формат ГГ:ХХ",
                         reply_markup=get_date_keyboard())
        return None

    try:
        db = ScheduleDB(config)
        user = db.find_user(cid)
        if user and user[0] is not None:
            keyboard = types.InlineKeyboardMarkup()
            callback_button = types.InlineKeyboardButton(
                text="На Сьогодні",
                callback_data="ap:{0}:1".format(data))
            keyboard.add(callback_button)
            callback_button = types.InlineKeyboardButton(
                text="На Завтра",
                callback_data="ap:{0}:0".format(data))
            keyboard.add(callback_button)

            bot.send_message(cid, "Виберіть день на який буде приходити розклад:",
reply_markup=keyboard)
        else:
            bot.send_message(cid, "Вас ще немає в базі даних, тому пройдіть просту процедуру
реєстрації")
            command_registration(m)
    except BaseException as e:
        logger.warning('command auto_posting_on: {0}'.format(str(e)))
        bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново")
@bot.message_handler(commands=['auto_posting_off'])
def command_auto_posting_off(m):
    # Статистика
    if config['STATISTIC_TOKEN'] != '':
        track(config['STATISTIC_TOKEN'], m.chat.id, m.text, 'auto_posting_off')
    else:
        logger.info('auto_posting_off')

    cid = m.chat.id

    try:
        db = ScheduleDB(config)
        user = db.find_user(cid)
        if not user:
            bot.send_message(cid, "Вас ще немає в базі даних, тому пройдіть просту процедуру
реєстрації")
            command_registration(m)
```

```
            return

        if db.set_auto_post_time(cid, None, None):
            bot.send_message(cid, "Автоматична відправка розкладу успішно відключена")
        else:
            bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново",
                             reply_markup=get_date_keyboard())

    except BaseException as e:
        logger.warning('command auto_posting_off: {0}'.format(str(e)))
        bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново")


# exams message handler
@bot.message_handler(func=lambda message: 'Екзамени' in (message.text if message.text is not
None else ''), content_types=['text'])
def exams(m):
    cid = m.chat.id

    # Статистика
    track(config['STATISTIC_TOKEN'], cid, m.text, 'exams')

    # Якщо користувача немає в базі, то йому виведе пропозицію зареєструватися
    try:
        with ScheduleDB(config) as db:
            user = db.find_user(cid)
        if not user or user[0] is None:
            message = "Вас ще немає в базі даних, тому пройдіть просту процедуру реєстрації:\n"
            message += 'Введіть команду(без лапок):\n\nреєстрація "назва ВНЗ" "факультет"
"група"\n\n'
            message += 'Якщо ви допустите помилку, то просто наберіть команду заново.\n'

            bot.send_message(cid, message, reply_markup=get_date_keyboard())
    except BaseException as e:
        bot.send_message(cid, 'Сталося щось дивне, спробуйте ввести команду заново',
                         reply_markup=get_date_keyboard())

    try:
        with ScheduleDB(config) as db:
            exams_list = db.get_exams(user[0])

        message = ''
        for exam in exams_list:
            message += exam[0].strftime('%d.%m.%Y') + ":\n"

            title = ' '.join(str(exam[1]).split())
            lecturer = ' '.join(str(exam[2]).split())
            classroom = ' '.join(str(exam[3]).split())
```

```
            message += title + ' | ' + lecturer + ' | ' + classroom + "\n"
            message += "------------\n"
        if len(message) == 0:
            message = 'Схоже розкладу екзаменів для вашої групи немає в базі'

    except BaseException as e:
        message = "Сталося щось дивне, спробуйте ввести команду заново"

    bot.send_message(cid, message, reply_markup=get_date_keyboard())
@bot.message_handler(commands=['auto_posting_off'])
def command_auto_posting_off(m):
    # Статистика
    if config['STATISTIC_TOKEN'] != '':
        track(config['STATISTIC_TOKEN'], m.chat.id, m.text, 'auto_posting_off')
    else:
        logger.info('auto_posting_off')

    cid = m.chat.id

    try:
        db = ScheduleDB(config)
        user = db.find_user(cid)
        if not user:
            bot.send_message(cid, "Вас ще немає в базі даних, тому пройдіть просту процедуру
реєстрації")
            command_registration(m)
            return

        if db.set_auto_post_time(cid, None, None):
            bot.send_message(cid, "Автоматична відправка розкладу успішно відключена")
        else:
            bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново",
                             reply_markup=get_date_keyboard())

    except BaseException as e:
        logger.warning('command auto_posting_off: {0}'.format(str(e)))
        bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново")


# exams message handler
@bot.message_handler(func=lambda message: 'Екзамени' in (message.text if message.text is not
None else ''), content_types=['text'])
def exams(m):
    cid = m.chat.id

    # Статистика
    track(config['STATISTIC_TOKEN'], cid, m.text, 'exams')

    # Якщо користувача немає в базі, то йому виведе пропозицію зареєструватися
```

```
    try:
        with ScheduleDB(config) as db:
            user = db.find_user(cid)
        if not user or user[0] is None:
            message = "Вас ще немає в базі даних, тому пройдіть просту процедуру реєстрації:\n"
            message += 'Введіть команду(без лапок):\n\nреєстрація "назва ВНЗ" "факультет"
"група"\n\n'
            message += 'Якщо ви допустите помилку, то просто наберіть команду заново.\n'

            bot.send_message(cid, message, reply_markup=get_date_keyboard())
    except BaseException as e:
        bot.send_message(cid, 'Сталося щось дивне, спробуйте ввести команду заново',
                         reply_markup=get_date_keyboard())

    try:
        with ScheduleDB(config) as db:
            exams_list = db.get_exams(user[0])

        message = ''
        for exam in exams_list:
            message += exam[0].strftime('%d.%m.%Y') + ":\n"

            title = ' '.join(str(exam[1]).split())
            lecturer = ' '.join(str(exam[2]).split())
            classroom = ' '.join(str(exam[3]).split())

            message += title + ' | ' + lecturer + ' | ' + classroom + "\n"
            message += "------------\n"
        if len(message) == 0:
            message = 'Схоже розкладу екзаменів для вашої групи немає в базі'

    except BaseException as e:
        message = "Сталося щось дивне, спробуйте ввести команду заново"

    bot.send_message(cid, message, reply_markup=get_date_keyboard())
# text message handler
@bot.message_handler(func=lambda message: True, content_types=['text'])
def response_msg(m):
    cid = m.chat.id
    if m.text in ScheduleType:
        # Статистика
        if config['STATISTIC_TOKEN'] != '':
            track(config['STATISTIC_TOKEN'], m.chat.id, m.text, 'schedule')
        else:
            logger.info('message: {0}'.format(m.text))

        week_type = -1

        if m.text == "Весь тиждень":
```

```
            days = ScheduleType[m.text]
        elif m.text == "Сьогодні":
            today = datetime.now()

            week_type = get_week_type(today)

            if datetime.weekday(today) == 6:
                today += timedelta(days=1)
                week_type = (week_type + 1) % 2

            days = [daysOfWeek[datetime.weekday(today)]]
        elif m.text == 'Завтра':
            tomorrow = datetime.now()
            tomorrow += timedelta(days=1)
            week_type = get_week_type(tomorrow)
            if datetime.weekday(tomorrow) == 6:
                tomorrow += timedelta(days=1)
                week_type = (week_type + 1) % 2

            days = [daysOfWeek[datetime.weekday(tomorrow)]]
        else:
            days = [ScheduleType[m.text]]

        for day in days:
            try:
                with ScheduleDB(config) as db:
                    user = db.find_user(cid)
                if user and user[0] is not None:
                    result = create_schedule_text(user[0], day, week_type)
                    for schedule in result:
                        bot.send_message(cid, schedule, reply_markup=get_date_keyboard())
                else:
                    bot.send_message(cid, "Вас ще немає в базі даних, тому пройдіть просту
процедуру реєстрації")
                    command_registration(m)
            except BaseException as e:
                logger.warning('response_msg: {0}'.format(str(e)))
                bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново")
    else:
        # Статистика
        if config['STATISTIC_TOKEN'] != '':
            track(config['STATISTIC_TOKEN'], m.chat.id, m.text, 'unknown')
        else:
            logger.info('unknown message: {0}'.format(m.text))

        bot.send_message(cid, "Невідома команда", reply_markup=get_date_keyboard())

@bot.callback_query_handler(func=lambda call: "reg:stage 2:" in call.data)
def callback_registration(call):
```

```
    cid = call.message.chat.id
    callback_data = re.split(r':', call.data)

    # Процедура реєстрації проходить у чотири етапи:
    # 1 етап: вибір курсу
    # 3 етап: вибір групи <--
    # 4 етап: додавання даних про належність користувача до групи в БД
    try:
        # Статистика
        if config['STATISTIC_TOKEN'] != '':
            track(config['STATISTIC_TOKEN'], cid, 'stage 3', 'registration-stage-3')

        keyboard = types.InlineKeyboardMarkup()

        faculty_id = callback_data[2]

        with ScheduleDB(config) as db:
            result = db.get_group(faculty_id)

        for row in result:
            callback_button = types.InlineKeyboardButton(
                text=str(row[0]),
                callback_data="reg:stage 4:{0}".format(str(row[1])))
            keyboard.add(callback_button)

        bot.send_message(cid, "Виберіть групу:", reply_markup=keyboard)
    except BaseException as e:
        logger.warning('Registration problem: {0}'.format(str(e)))
        bot.send_message(cid, "Сталося щось дивне, спробуйте почати спочатку, ввівши команду
/registration")


@bot.callback_query_handler(func=lambda call: "reg:stage 4:" in call.data)
def callback_registration(call):
    cid = call.message.chat.id
# Парсинг повідомлення, яке вказує стадію реєстрації
# reg : stage : tag
callback_data = re.split(r':', call.data)

    # Процедура реєстрації проходить у чотири етапи:
    # 1 етап: вибір курсу
    # 3 етап: вибір групи
    # 3 етап: додавання даних про належність користувача до групи в БД try <--

    # Статистика
    if config['STATISTIC_TOKEN'] != '':
        track(config['STATISTIC_TOKEN'], cid, 'stage 4', 'registration-stage-4')

    group_id = callback_data[2]
```

```
    db = ScheduleDB(config)

    row = db.get_group(group_id)
    user = db.find_user(cid)

    if user:
        db.update_user(cid, call.message.chat.first_name, call.message.chat.username,
str(row[0][1]))
    else:
        db.add_user(cid, call.message.chat.first_name, call.message.chat.username,
str(row[0][1]))

    bot.send_message(cid, "Чудово, ви зареєструвалися, ваша група: " + row[0][0] +
                     "\nЯкщо ви помилилися, то просто введіть команду /registration і змініть
дані",
                     reply_markup=get_date_keyboard())
    bot.send_message(cid, "Тепер ви можете налаштувати автоматичну відправку розкладу у заданий
вами час,"
                           " ввівши команду /auto_posting_on <час>, "
                           "де <час> повинен мати формат ГГ:ХХ")

except BaseException as e:
    logger.warning('Registration problem: {0}'.format(str(e)))
    bot.send_message(cid, "Сталося щось дивне, спробуйте почати спочатку, ввівши команду
/registration")


@bot.callback_query_handler(func=lambda call: "ap:" in call.data)
def callback_auto_posting(call):
    cid = call.message.chat.id

    try:
        callback_data = re.split(r':', call.data)

        # Перевірка на відповідність введених користувачем даних прийнятому формату
        if len(callback_data) != 4:
            bot.send_message(cid,
                             "Ви відправили порожній рядок або рядок неправильного формату.
Правильний формат ГГ:ХХ",
                             reply_markup=get_date_keyboard())
            return

        hour = ''.join(filter(lambda x: x.isdigit(), callback_data[1]))
        minutes = ''.join(filter(lambda x: x.isdigit(), callback_data[2]))
        is_today = callback_data[3]

        # Перевірка на відповідність введених користувачем даних прийнятому формату
        if not hour.isdigit() or not minutes.isdigit():
```

```
            bot.send_message(cid,
                             "Ви відправили порожній рядок або рядок неправильного формату.
Правильний формат ГГ:ХХ",
                             reply_markup=get_date_keyboard())
            return

        with ScheduleDB(config) as db:
            if db.set_auto_post_time(cid, (hour + ":" + minutes + ":" + "00").rjust(8, '0'),
is_today):
                bot.send_message(cid, "Час встановлено", reply_markup=get_date_keyboard())
            else:
                bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново",
                                 reply_markup=get_date_keyboard())
    except BaseException as e:
        logger.warning('callback_auto_posting: {0}'.format(str(e)))
        bot.send_message(cid, "Сталося щось дивне, спробуйте ввести команду заново")


# Порожній індекс вебсервера, не повертає нічого, тільки http 200
@app.route('/', methods=['GET', 'HEAD'])
def index():
    return ''


# Видаляє вебхук
@app.route("/remove_webhook", methods=["GET", "HEAD"])
def remove_webhook():
    bot.remove_webhook()
    return "ok", 200


# Скидає вебхук
@app.route("/reset_webhook", methods=["GET", "HEAD"])
def reset_webhook():
    bot.remove_webhook()
    bot.set_webhook(url=WEBHOOK_URL_BASE+WEBHOOK_URL_PATH,
certificate=open(config["WEBHOOK_SSL_CERT"], 'r'))
    return "ok", 200


# Обробка запитів до вебхуку
@app.route(WEBHOOK_URL_PATH, methods=['POST'])
def webhook():
    if flask.request.headers.get('content-type') == 'application/json':
        json_string = flask.request.get_data().decode('utf-8')
        update = telebot.types.Update.de_json(json_string)
        bot.process_new_updates([update])
        return ''
    else:
```

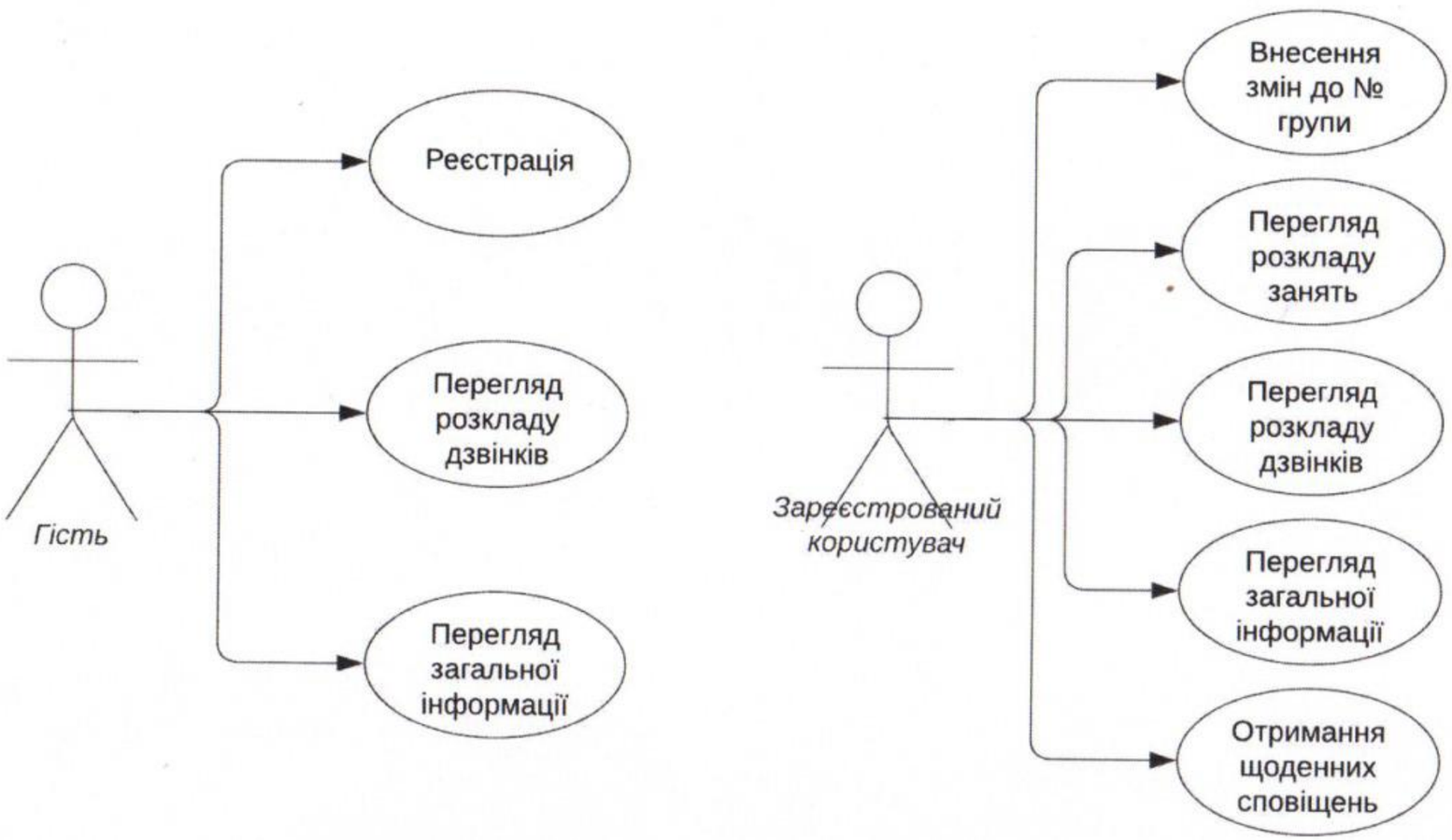
```
        flask.abort(403)


# Додавання організації до бази даних
@app.route("/api/organization", methods=["POST"])
def add_organization():
    if not request.is_json:
        return "incorrect request", 403

    try:
        content = request.get_json()

        if content['key'] != config["TOKEN"]:
            return "invalid api key", 403

        data = content['data']

        answer = {
            'ok': [],
            'failed': []}

        with ScheduleDB(config) as db:
            for org_data in data:
                # Обов'язкові параметри запиту
                organization = org_data['organization']
                group = org_data['group']

                # Необов'язкові параметри
                faculty = org_data['faculty'] if 'faculty' in org_data else ''

                tag = db.add_organization(organization, faculty, group)
                json_data = {
                    'tag': tag,
                    'data': org_data
                }

                if tag is не None:
                    answer['ok'].append(json_data)
                else:
                    answer['failed'].append(json_data)

        return jsonify(answer), 200
    except KeyError as e:
        return 'key not found: {}'.format(str(e)), 403


# Додавання розкладу до бази даних
@app.route("/api/schedule", methods=["POST"])
def add_schedule():
```

```
    if not request.is_json:
        return "incorrect request", 403

    try:
        content = request.get_json()

        if content['key'] != config["TOKEN"]:
            return "invalid api key", 403

        data = content['data']

        answer = {'failed': []}
        with ScheduleDB(config) as db:
            for lecture in data:
                tag = lecture['tag']
                day = lecture['day']
                number = lecture['number']
                week_type = lecture['week_type']
                title = lecture['title']
                classroom = lecture['classroom']

                time_start = lecture['time_start'] if 'time_start' in lecture else None
                time_end = lecture['time_end'] if 'time_end' in lecture else None
                lecturer = lecture['lecturer'] if 'lecturer' in lecture else None

                if not db.add_lesson(tag, day, number, week_type, time_start, time_end, title,
classroom, lecturer):
                    answer['failed'].append(lecture)
        return jsonify(answer), 200
    except KeyError as e:
        return 'key not found: {}'.format(str(e)), 403


if __name__ == '__main__':
    # Запуск сервера flask
    app.run(
        host=config["WEBHOOK_LISTEN"],
        port=config["WEBHOOK_PORT"],
        ssl_context=(config['WEBHOOK_SSL_CERT'], config['WEBHOOK_SSL_PRIV']),
        debug=True)
```

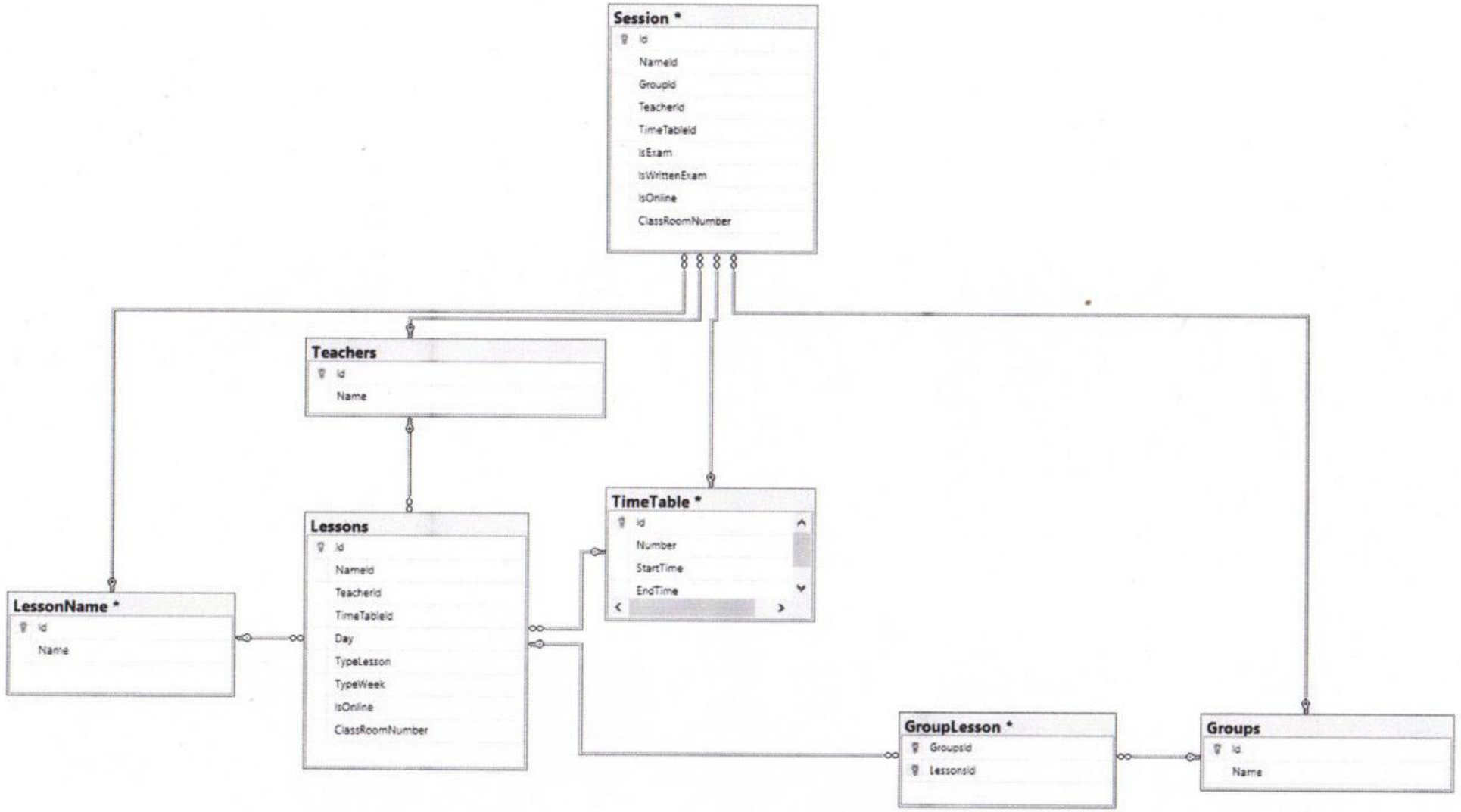
ДОДАТОК Д

(Обов’язковий)

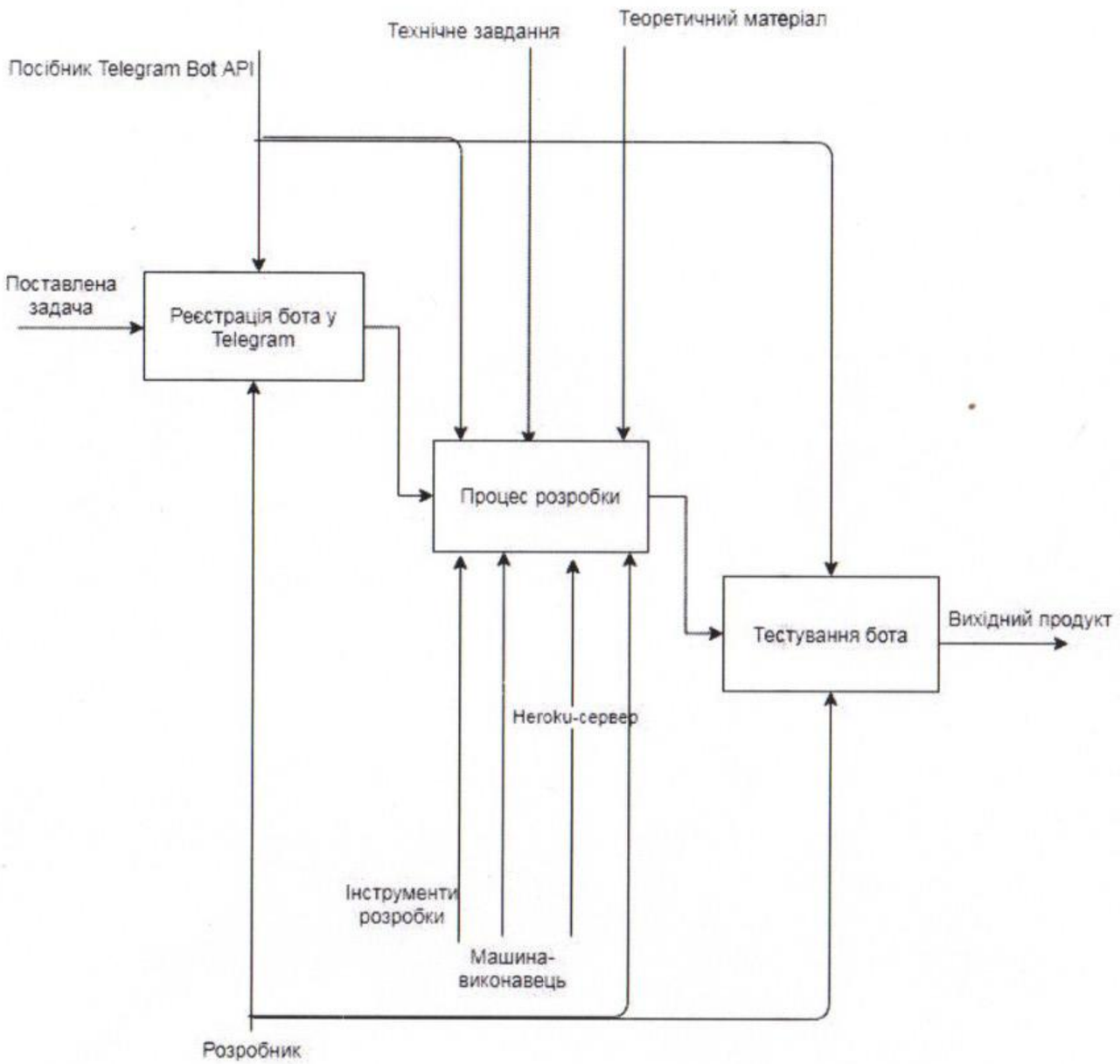
ГРАФІЧНІ МАТЕРІАЛИ



					КвРІПЗ.200252.01.18.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	Діаграма варіантів використання	Літера / Маса / Масштаб
Розробив		Остапчук М.С.		12.06		
Керівник		Яшина О.М.		12.06		Аркуш 1 / Аркушів 3
Консульт.						
Н. Контр.		Форкун Ю.В.		12.06		ХНУ, ІПЗ-20-1
Зав. каф.		Бедратюк Л.П.		12.06		



					КвРІПЗ.200252.01.18.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	Схема бази даних	Літера / Маса / Масштаб
Розробив		Остапчук М.С.		12.06		
Керівник		Яшина О.М.		12.06		Аркуш 2 / Аркушів 3
Консульт.						
Н. Контр.		Форкун Ю.В.		12.06		ХНУ, ІПЗ-20-1
Зав. каф.		Бедратюк Л.П.		12.06		



					КвРІПЗ.200252.01.18.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	Діаграма декомпозиції розробки бота	Літера / Маса / Масштаб
Розробив		Остапчук М.С.		12.06		
Керівник		Яшина О.М.		12.06		Аркуш 3 / Аркушів 3
Консульт.						
Н. Контр.		Форкун Ю.В.		12.06		ХНУ, ІПЗ-20-1
Зав. каф		Бедратюк Л.П.		12.06		

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Остапчука М. С.

Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті», згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

05.06.2024

дата

підпис

UNICHECK
by Turnitin

Ім'я користувача:
ІПЗ

ID перевірки:
1016342785

Дата перевірки:
11.06.2024 01:17:46 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2024 09:54:08 EEST

ID користувача:
100012953

Назва документа: БКР_Телеграм-бот для розкладу занять в університеті_Остапчук_Яшина

Кількість сторінок: 68 Кількість слів: 9056 Кількість символів: 70238 Розмір файлу: 2.47 MB ID файлу: 1016144176

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

25.7%
Схожість

Найбільша схожість: 8.38% з Інтернет-джерелом (http://reposit.nupp.edu.ua/bitstream/PoltNTU/9885/1/401-%d0%a2%d..

24.8% Джерела з Інтернету 428 ..... Сторінка 70

3.96% Джерела з Бібліотеки 176 ..... Сторінка 73

0.39% Цитат

Цитати 1 ..... Сторінка 74

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи 3

Підозріле форматування 24 сторінки

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 7%**

	Документ		Сумарний збіг по Базі Даних	
ID: 129571 Назва: БКР_Телеграм-бот для розкладу занять в університеті_Остапчук_Яшина Додано в БД: 2024-06-10 Автора: Остапчук М. Керівники: Яшина О.М., канд. техн. наук, доцент Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	50485	802	2141 (4%)	26 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Остапчук Микита Сергійович

Тема Телеграм-бот для розкладу занять в університеті

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 115.
1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі проведено дослідження та аналіз предметної області. Було визначено всі функціональні та нефункціональні вимоги. Проаналізовано наявні рішення на ринку, висвітлено їх переваги та недоліки, що обґрунтувало потребу в розробці нового програмного продукту. Розглянуто інструменти для реалізації проєкту, та на їх основі створено програмне забезпечення. Після тестування було підтверджено, що програмне забезпечення працює цілком коректно і повністю готове до використання.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі була висвітлена актуальність теми та сформульовано мету та завдання проєкту, що визначило напрямок подальшої роботи. У першому розділі здійснено аналіз предметної області та оцінку наявного програмно-технічного забезпечення з метою визначення функціональних та нефункціональних вимог. Другий розділ присвячений проєктуванню програмного забезпечення, включаючи аналіз архітектури системи, проєктування бота з використанням стандартів IDEF0 та UML, а також вибір технологій і методів реалізації системи. У третьому розділі виконано реалізацію програмної частини проєкту та проведено тестування чат-бота. Здійснено реєстрацію телеграм бота, розроблено програмні модулі та сформовано керівництво користувача.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є важливою, оскільки на сьогоднішній день в університетському житті Telegram-боти є недостатньо розвинутими та не забезпечують учасників освітнього процесу необхідною інформацією та функціональними можливостями. Під час розробки були використані передові технології програмування, а також розроблені актуальні архітектурні рішення. Також слід зазначити зручність розробленого інтерфейсу користувача, який надає можливість швидко та легко отримувати необхідну інформацію про розклад занять та дзвінків через вже знайомий інструмент комунікації - Telegram.

5. Негативні сторони роботи Потреба в ручному заповненні бази даних розкладом занять, що може призвести до зайвих часових та трудових затрат для адміністраторів системи. Кращим підходом було б реалізувати автоматичний парсинг даних з розкладу занять ФІТ ХНУ, що дозволило б уникнути помилок під час внесення інформації в базу даних та забезпечило б швидке та ефективне оновлення інформації для користувачів.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики проектування. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Гнатишук Є.Г., доцент кафедри АІТС (КНЕУ)

"12" 06 2024 р.

(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів:

Назва кваліфікаційної роботи: «Телеграм-бот для розкладу занять в університеті»
Автор: Остапчук Микита Сергійович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Яшина Оксана Миколаївна, кандидат технічних наук, доцент
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	**відповідає**
3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв`язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат Unicheck виявлено схожість з деякими документами у частині загальновживаних обов`язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів, у рамках основних написів, у назвах та URL-адресах публікацій переліку джерел посилання;

2) в якості запозичень системою Unicheck було зафіксовано деякі послідовності вихідного коду і посилання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення, а також загальновживані стандартні поняття, що стосуються розробки програмного забезпечення;

3) запозичення, виявлені в тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 2.0%. Обсяг запозичень, визначений системою Unicheck виявлення збігів ідентичності/схожості, складає 25.7% і адресується до 428 джерел з Інтернету і 176 джерела з бібліотеки, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 10.06.2024 р.

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Оксана ЯШИНА