МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

Кваліфікаційна робота бакалавра

TELEGRAM-БОТ ДЛЯ ЗАБЕЗПЕЧЕННЯ ЕФЕКТИВНОГО НАВЧАННЯ

Здобувач освіти ІН-82 Богдан ЧАЙКА

Науковий керівник,
кандидат технічних наук, доцент Сергій ПЕТРОВ

Завідувач кафедри
доктор технічних наук, професор. Анатолій ДОВБИШ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедри Довбиш А.С.

"_____" _____2022 р.

ЗАВДАННЯ
до кваліфікаційної роботи бакалавра

Студента четвертого курсу, групи ІН-82 спеціальності 122 -Комп'ютерні науки, денної форми навчання Чайки Б.В.

Тема: Telegram-бот для забезпечення якісного навчання студентів

Затверджена наказом по СумДУ

№_____ від _____2022 р.

Зміст пояснювальної записки: 1) аналіз проблеми та постановка задачі; 2) вибір метода розв'язання задачі; 3) розробка інформаційного і програмного забезпечення системи

Дата видачі завдання "_____"_____2022 р.

Керівник випускної роботи _____ Петров С.О.

Завдання прийняв до виконання _____Чайка Б.В.

РЕФЕРАТ

Записка: 51 стор., 24 рис., 1 додаток, 7 джерел.

Об'єкт дослідження — telegram-бот для забезпечення ефективного навчання

Мета роботи — створення telegram-боту для забезпечення ефективного навчального процесу

Методи дослідження — технології створення telegram-ботів

Результати — розроблено telegram-бот для забезпечення ефективного навчання. Створений telegram-бот дозволить зробити ефективнішим навчальний процес у всіх його учасників та зменшити шлях доступу до потрібної інформації. Розробка проводилась на мові програмування Python з використанням бібліотек Aiogram та GINO.

AIOGRAM, ОСВІТА, PYTHON, REDIS, ПРОЕКТУВАННЯ

ЗМІСТ

ВСТУП

Останні два роки життя - це постійні виклики. Все починалося з пандемії, а тепер ми маємо 3-ій місяць війни з росією, що змусило поглянути на більшість сфер діяльності з іншого боку. І не просто поглянути, а шукати нові способи для роботи та їх розвитку, підтримки життєдіяльності.

Не оминуло це і вищу освіту. Завдяки пандемії було створено безліч умов для дистанційного навчання, до таких реалій звикли всі учасники освітнього процесу: студенти, викладачі, адміністрація університету. Проте, війна також внесла свої корективи, адже одна справа не виходити зайвий раз на вулицю через загрозу вірусу та продовжувати навчання. Інша справа - коли є загроза артилерійського обстрілу або авіаудару. Через це учасники освітнього процесу змушені переміщуватись до укриттів та відволікатись від навчання.

На цій практиці та під час написання дипломної роботи ніяк не винайти систему ППО на кшталт Залізного куполу або щось крутіше за допомогу від іноземних партнерів. Зате є можливість створити інструмент, що допоможе зробити освітній процес ефективнішим. Наприклад, telegram-бот, адже з початку війни месенджер Telegram набрав обертів та майже кожен українець тепер є його користувачем. І це не дивно, бо даний месенджер на голову вищий за конкурентів: пропонує ширший функціонал, зберігає безпеку листувань, а для розробників - відкритий вихідний код і Telegram Bot API, що суттєво спрощує розробку.

Боти - спеціальні акаунти в месенджерах, створені для того, щоб автоматично обробляти і відправляти повідомлення. Користувачі можуть взаємодіяти з ботами за допомогою повідомлень, що відправляються через звичайні або групові чати. Логіка бота контролюється за допомогою HTTPS запитів до нашого API для пошукових роботів.

Метою роботи є розробка Telegram-бота, який стане асистентом для студентів та викладачів. За допомогою нього студенти зможуть оперативно отримувати відповіді на найпоширеніші питання, посилання на лекції, матеріали для підготовки, завдання тощо, а викладачі зможуть все це надавати. Об’єктом дослідження є процеси збору, обробки та подання інформації про навчальний процес. Предметом є розробка боту, який дозволяє реалізувати ці процеси.

1 АНАЛІЗ ВІДОМИХ РІШЕНЬ

Завдяки пандемії COVID-19, не можна сказати, що з’явилося багато рішень для забезпечення ефективного навчального процесу. Вони вже існували до її початку, а пандемія була каталізатором їх розвитку. Так як я є студентом Сумського державного університету, я хочу зробити акцент на сервісах, які є в нашій екосистемі.

Зазвичай при аналізі відомих рішень ми розглядаємо переваги та недоліки для того, щоб якомога краще спроектувати наш майбутній продукт. Проте в даному випадку ми лише ознайомимося з тим, які сервіси вже існують в університеті, без переваг та недоліків. Ознайомлення допоможе тим, що в майбутньому наш telegram-бот матиме можливість по інтеграції з сервісами університету.

1.1 Огляд існуючих рішень

1.1.1 Платформа MIX.СумДУ

Першою ми розглянемо платформу “MIX.СумДУ” [1], яка є фундаментом у навчанні для багатьох студентів різних спеціальностей. Дана платформа має зручний інтерфейс, сучасний дизайн та багатий функціонал.

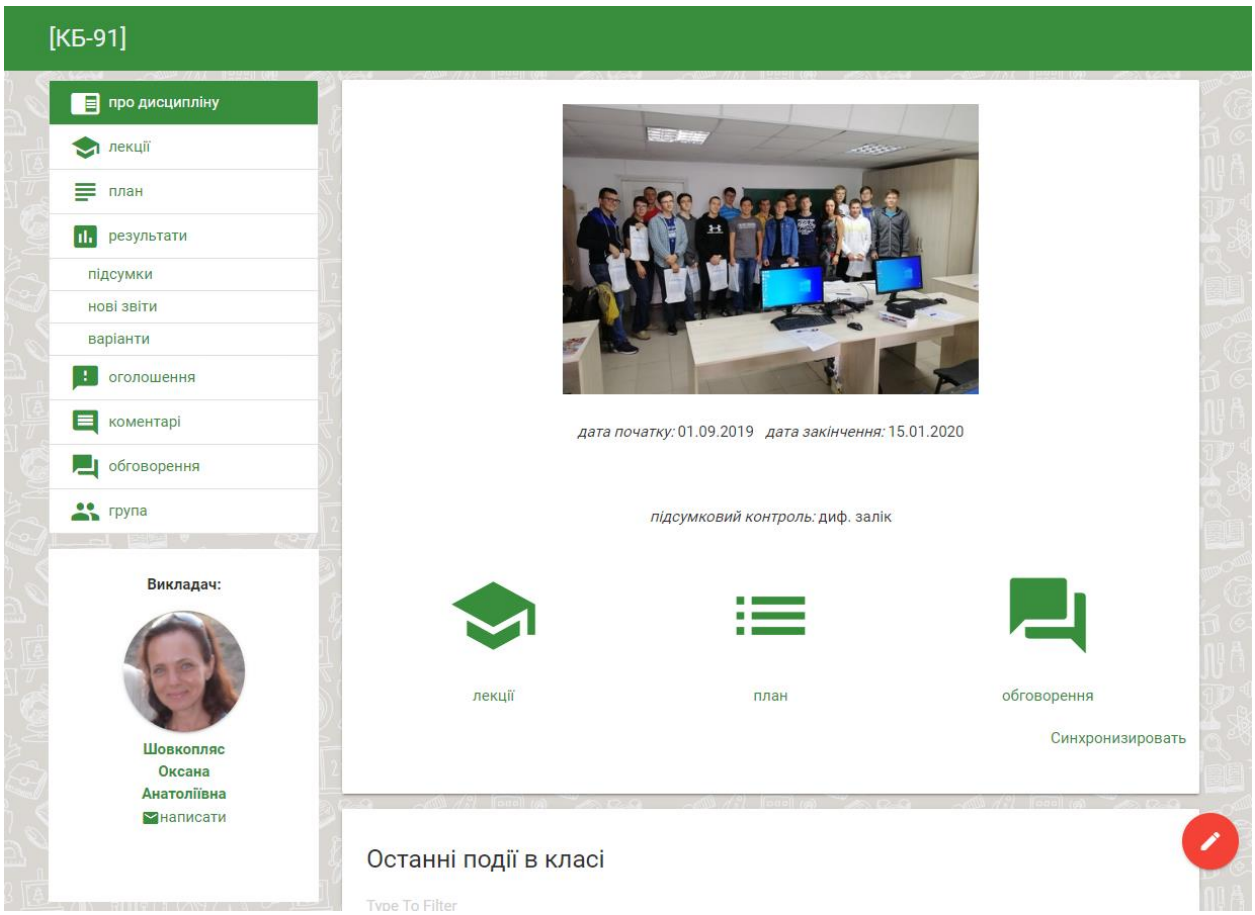

Рис.1.1 Зовнішній вигляд платформи МІХ.СумДУ

Завдяки МІХ.СумДУ спрощена комунікація між викладачем та студентами, адже платформа дозволяє наступне:

- переглядати матеріали та конспекти лекцій;
- ознайомитись з переліком лабораторних та практичних робіт та дедлайнами їх здачі;
- надіслати зроблену лабораторну, практичну роботу;
- переглянути план тестів та контрольних робіт під час дисципліни, а також можливість проходити їх через МІХ.

Варто також згадати про деякі функції для викладачів:

- виставлення оцінок за лабораторні, практичні роботи (в деяких випадках - за пройдені тести);
- створення власних тестувань для студентів;
- запис відео з веб-камери при складанні тесту або іспиту (для ідентифікації того, хто проходить тест);
- розміщення важливих оголошень для студентів.

Як ми бачимо, MIX.СумДУ має багатий функціонал, але на цій платформі ми не зупиняємося.

1.1.2 Особистий кабінет СумДУ

Наступним ми розглянемо особистий кабінет СумДУ [2], який можна порівняти з швейцарським ножем. І це не дивно, якщо ознайомитись з його функціоналом.

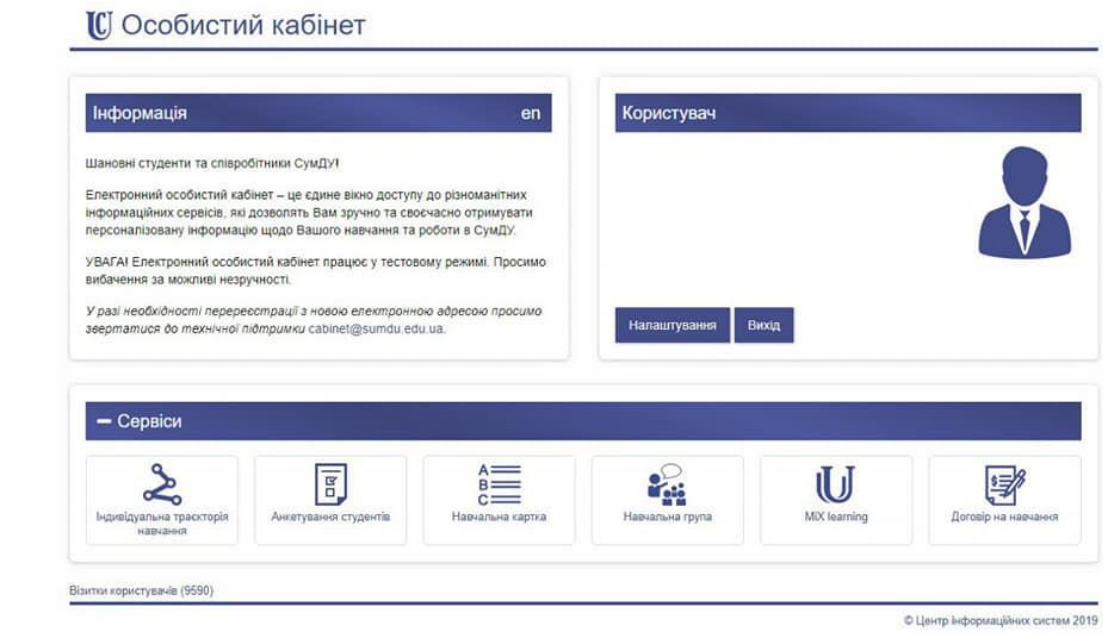

Рис.1.2 Зовнішній вигляд особистого кабінету СумДУ

Кабінет корисний для всіх учасників освітнього процесу - викладачів, студентів, співробітників, адміністрації університету. Для кожного з учасників він має майже однаковий функціонал, але також для кожного є і окремі додаткові функції. Всіх учасників об'єднують такі функції:

- візитівка в системі СумДУ, де вказано всю інформацію про людину;
- інтеграція з MIX.СумДУ, а саме можливість авторизації до платформи за допомогою особистого кабінету;
- отримання наказів, листів, службових записок тощо;
- телефонний довідник співробітників, адміністрації;

Тобто, це як внутрішня поштова скринька, візитівка та інформаційна система в одному.

Студенти мають такі можливості:

- участь в анкетуванні щодо якості навчального процесу;
- електронна відомість з оцінками, що повністю замінила залікові книжки в СумДУ з 2020 року;
- обрання дисципліни за вибором або навчальної траєкторії;
- перегляд стану договору на навчання, у випадку якщо студент вчиться на контрактній основі або за цільовим договором (скільки сплачено, скільки залишилося сплатити);

- перегляд стану договору з гуртожитком, якщо студент в ньому проживає, а також можливість подачі заяви для проживання в гуртожитку;
- участь в електронних виборах органів студентського самоврядування;
- перегляд додатку до диплома.

І це лише частина можливостей для студентів. Для співробітників є такі функції:

- оформлення електронних відомостей за дисциплінами для груп студентів (для викладачів);
- оформлення службових записок, документів та їх підписання в уповноважених осіб;
- перегляд розрахункового листка;
- участь у віртуальних засіданнях Вчених Рад (за допомогою голосування);
- преміювання співробітників (для адміністрації).

Ми розглянули лише вершину айсберга і це не кінець. Особистий кабінет дійсно як швейцарський ніж, який повноцінно поєднує в собі більшість внутрішніх сервісів СумДУ.

1.1.3 Шаблони документів СумДУ

Ще один сервіс для розгляду - “Шаблони документів СумДУ” [3]. У порівнянні з двома першими сервісами, він має більший пріоритет для

співробітників (викладачів, адміністрації), проте є певна дотичність до студентів.

Рис.1.3 Зовнішній вигляд сервісу “Шаблони документів СумДУ”

Під час навчального процесу та роботи в університеті виникає потреба в оформленні певних довідок, службових записок тощо. Для різних цілей відповідно різні формати документів, в яких легко заплутатись та ускладнити життя усім учасникам навчального процесу через невірний формат або недостатню кількість знань з оформлення. Сервіс “Шаблони документів СумДУ” не лише база шаблонів різних типів документів, а також дозволяє

відразу заповнити потрібний документ. На рисунку 1.4 можна побачити приклад такої форми заповнення.

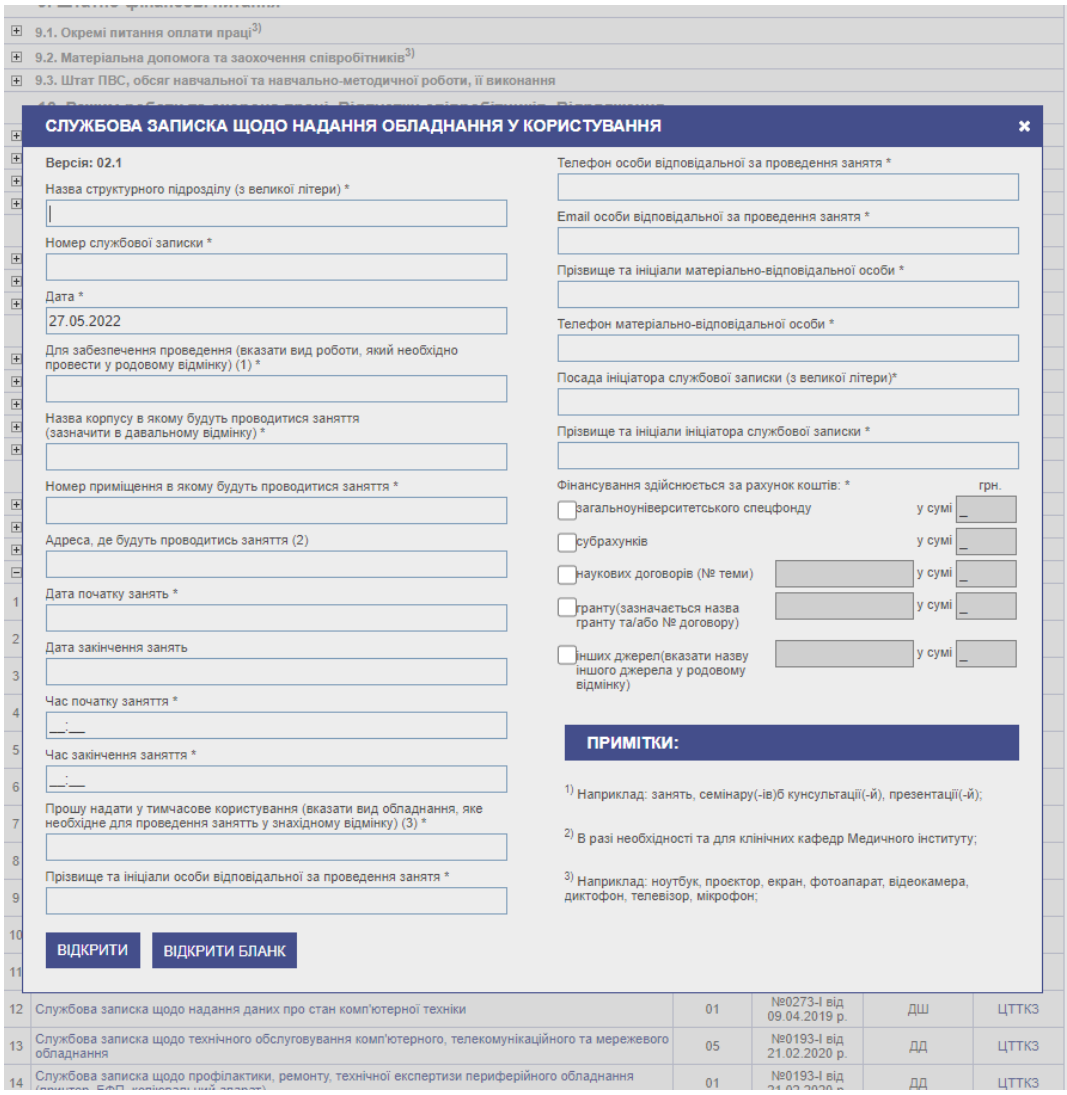

Рис.1.4 Приклад форми для оформлення потрібного документу

Не дивлячись на морально застарілий інтерфейс, він не є перевантаженим, а найголовніше - є зручним для будь-якого учасника навчального процесу.

1.1.4 Нормативна база СумДУ

Останнім ми розглянемо реєстр основної нормативної бази СумДУ [4], який зберігає в собі безліч корисного, а місцями навіть й цікавого.

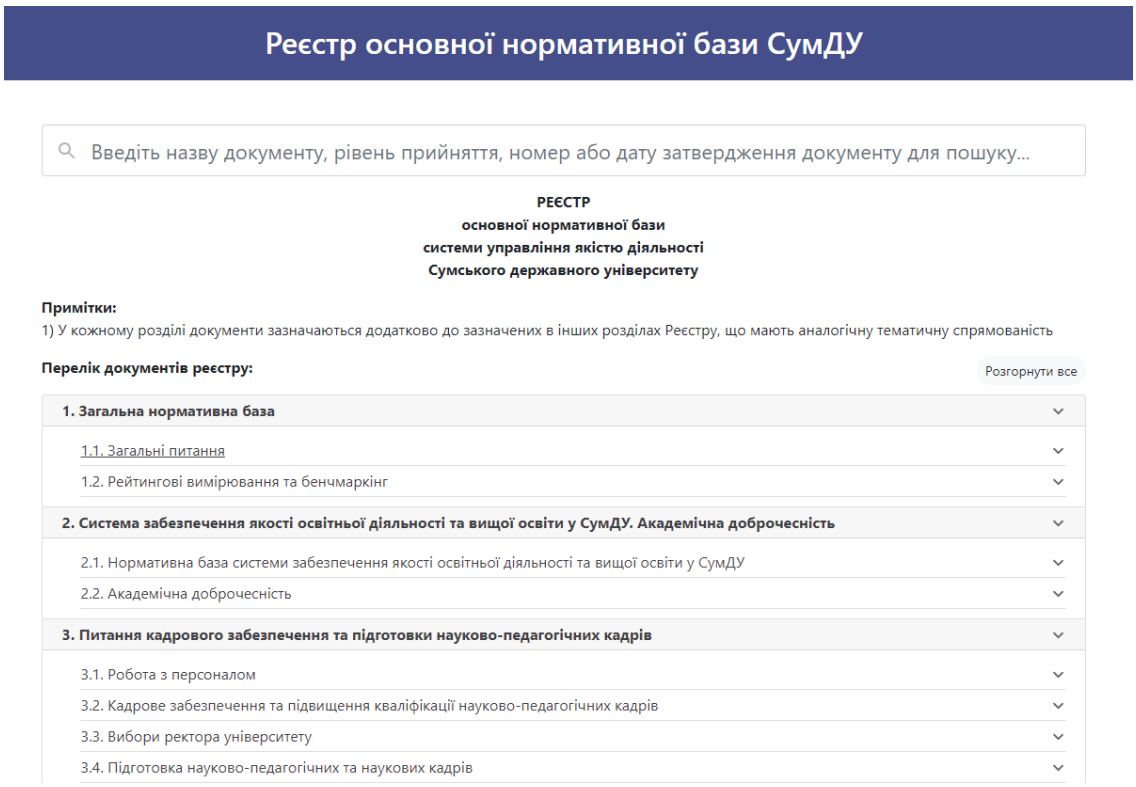

Рис.1.5 Зовнішній вигляд реєстру основної нормативної бази СумДУ

Як і у випадку з сервісом “Шаблони документів СумДУ”, реєстр також орієнтований переважно на співробітників університету, але також є корисним для ознайомлення студентами. Переліком документів, які можуть бути корисними для студентів, є:

- Положення про старосту академічної групи;
- Положення про куратора академічної групи;
- Положення про студента-куратора академічної групи;
- Положення про студентське самоврядування;
- Положення про академічну мобільність здобувачів вищої освіти.

Цей перелік не є вичерпним, адже реєстр містить в собі безліч документів щодо навчального процесу, позанавчальної діяльності, спортивної

діяльності тощо. Не дивлячись на велику кількість інформації, яку надають старости, куратори, органи студентського самоврядування та інші структури, даний реєстр не втрачає свою користь, адже дає можливість детальніше ознайомитися з нормативною частиною університету і знайти відповіді на питання щодо моментів, які турбують безпосередньо під час навчання в університеті.

Отже, ми розглянули основні сервіси в екосистемі СумДУ, якими можуть користуватися студенти. Звичайно, якщо зануритися глибше, то можна знайти ще більше сервісів, проте для аналізу в дипломній роботі було обрано саме ці. На деякі з них ми спираємося для проектування функціоналу майбутнього telegram-бота, а деякі з них будуть присутні у нашому майбутньому продукті.

1.2 Постановка задачі

Під час огляду існуючих рішень в Сумському державному університеті, було встановлено мету нашої роботи – проектування, реєстрація та розробка telegram-бота для забезпечення якісного навчання студентів. Майбутній продукт повинен мати достатній базовий функціонал для студентів та викладачів, зручність підтримки та розширення у майбутньому, а частина

інтерфейсу вже покриватиметься завдяки месенджеру Telegram, що спрощує задачу і нам не потрібно займатися окремо front-end частиною. Для досягнення потрібної мети необхідно виконати такі завдання:

1) Огляд та вибір мови програмування для веб-застосунку;
2) Вибір допоміжних інструментів для розробки поштового сервісу;
3) Проектування взаємодії користувача з сервісом;
4) Проектування та розробка поштового сервісу;
5) Тестування сторонніми користувачами;

2 ПРОЕКТУВАННЯ TELEGRAM-БОТУ

2.1 Текстове та графічне проектування Telegram-боту

Після розгляду сервісів з еко-системи СумДУ, на які ми будемо спиратися під час процесу розробки, ми можемо перейти безпосередньо до проектування Telegram-боту. Під час проектування нам потрібно охопити такі складові:

- функціональна - вибір та перелік функцій, які будуть присутні у нашому продукті;

- логічна - розбір кожної функції на “атоми”, побудова алгоритму дій кожної з них;
- користувацька - визначення ролей користувачів бота, розмежування доступних функцій між ними;

Після проектування згідно цих трьох складових, ми зможемо перейти до вибору компонентів для розробки.

2.1.1 Функціональне проектування

Проаналізувавши схожі сервіси та оглянувши їх функціонал, нам потрібно обрати найпотрібніше з них для студента та викладача. Telegram-бот дозволяє не обмежувати себе у наповненні функціоналом, проте перевантаження ним може негативно вплинути на його роботу та зручність у користуванні. Отже, виділимо основні функції, які будуть присутні у нашому боті. Перелік таких функцій:

- відповіді на найчастіші питання;
- отримання розкладу найближчих пар та посилання на них;
- отримання матеріалів лекцій, лабораторних та практичних робіт;
- сповіщення про тестування та екзамени;
- отримання інформації від викладача.

Даний функціонал дозволить студенту та викладачу не “бігати” між декількома сервісами та студенту мати все потрібне для навчання в кишені, а викладачу - надавати все потрібне, використовуючи одне джерело.

Тут також варто згадати одну з переваг telegram-боту перед веб-застосунком - кросплатформеність. Якби на початку роботи за мету було визначено розробку веб-застосунку, тоді до переліку задач варто було б додавати адаптивність для будь-яких пристроїв, що ускладнило б розробку через різноманітність пристроїв. Наприклад, те що стабільно відпрацьовує у браузері Chrome на ОС Windows, може некоректно працювати у Safari на Mac OS або Chrome на Android-пристрої. Якщо мова йде про telegram-бот, то наш застосунок працюватиме всюди, де встановлена актуальна версія мессенджеру Telegram.

Повернемось до функцій telegram-бота. Це лише вершина айсбергу, проте кожна функція також матиме ще декілька під-функцій, тоді ми маємо розробити такий собі mind map, який допоможе визначити наповнення кожної з функцій. Переходимо до наступного етапу проектування - логічного.

2.1.2 Логічне проектування

Визначивши перелік функцій, ми повинні їх розібрати на “атоми” та побудувати алгоритм дій для кожної з них. Це потрібно для того, щоб під час розробки telegram-боту ми думали над тим, як це втілити у вигляді коду, а не замислювалися над послідовністю дій кожної з функцій.

Першою ми розглянемо функцію “Відповіді на найчастіші питання”. Назва цієї функції вже вказує на її призначення, проте з певним нюансом. Отже, під-функції “Відповідей на найчастіші питання” такі:

1. Користувач при натисканні на “Відповіді на найчастіші питання” отримує список питань, які найчастіше ставлять студенти.
2. а) Якщо користувач знаходить своє питання у списку - він натискає на кнопку та отримує відповідь на своє питання.

   б) Якщо користувач не знаходить своє питання у списку - він натискає на кнопку “Поставити власне запитання”, після чого Telegram-бот попросить користувача написати своє запитання. Він надсилає та отримує повідомлення про те, що адміністратор отримав запитання та незабаром додасть на нього відповідь.

На рисунку 2.1 зображено графічне представлення алгоритму цієї функції

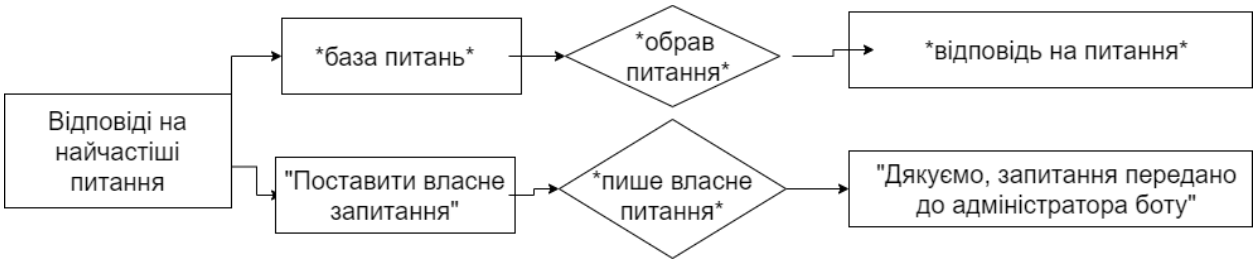

Рис.2.1 Алгоритм функції “Відповіді на найчастіші питання”

Тепер перейдемо до отримання розкладу найближчих пар. Для цієї та подальших функцій потрібно ідентифікувати користувача, щоб умовно студенту групи ІН-82 не надати розклад пар студента ІН-81. У мінімально життєздатному продукті база даних студентів та їх груп буде наповнюватись за допомогою адміністратора, проте в майбутньому можливе розширення функціоналу, а саме додавання можливості реєстрації студента, викладача, курсу тощо. На зараз це зроблено з метою убезпечення бота від можливих

махінацій з боку недобросовісних користувачів, які можуть додати неіснуючі групи або курси.

На рисунку 2.2 зображено графічне представлення функції “Отримання розкладу пар”.

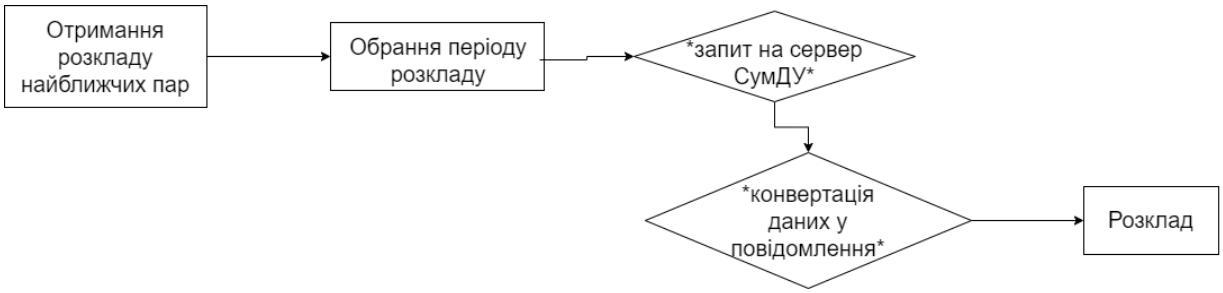

Рис.2.2 Алгоритм функції “Отримання розкладу найближчих пар”

Далі ми повинні спроектувати послідовність дій у функції “Отримання навчальних матеріалів”. Навчальних матеріалів може бути декілька типів: “лекція”, “практичне завдання”, “лабораторна робота”, “методичні рекомендації”, “курсова робота”, “додатковий матеріал”. Отже, при виборі даної функції ми повинні спочатку запитати у користувача, який саме тип навчальних матеріалів йому потрібен, після цього запропонувати тему за якою йому потрібні навчальні матеріали, зробити запит до внутрішньої бази даних та надати посилання на них. Якщо матеріалів за цією темою немає - надати повідомлення про це.

На рисунку 2.3 зображено графічне представлення функції “Отримання навчальних матеріалів”.

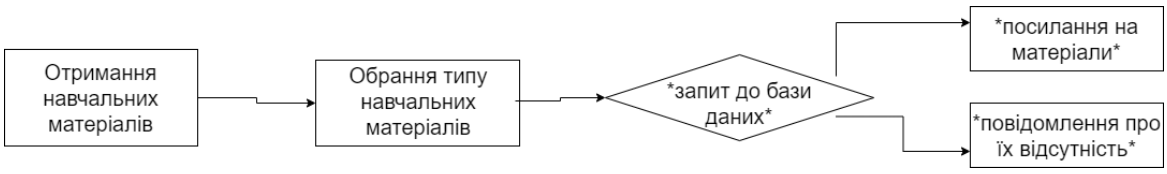

Рис.2.2 Алгоритм функції “Отримання навчальних матеріалів”

Переходимо до проектування функції “Сповіщення про навчальну діяльність”. У потоці навчальних дисциплін, студент з легкістю може забути про екзамен або модульний контроль, що наближається до нього. І це може зіграти злий жарт, адже студент не встигне підготуватися та вдало скласти певний вид контролю. Саме тому існуватиме ця функція, в якій також братиме участь викладач. Алгоритм дій такий:

1. Викладач встановлює у розкладі подію (модульний контроль, екзамен, залік).
2. Подія зберігається у розкладі, а разом з нею - часові рамки, коли потрібно надіслати сповіщення про неї. Вони вираховуються автоматично - за 15 днів, за 10 днів, за 7 днів, за 5 днів, за 3 дні та за день.
3. Студенти отримують сповіщення автоматично, без додаткового втручання викладача.

На рисунку 2.3 зображено графічне представлення функції “Сповіщення про навчальну діяльність”.

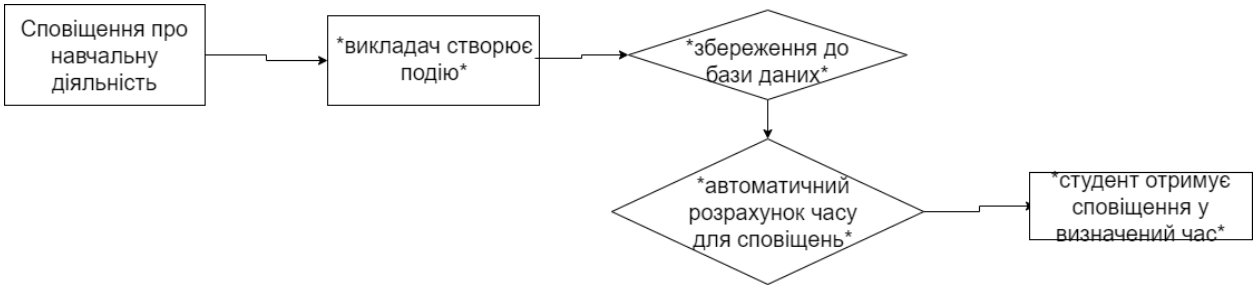

Рис.2.3 Алгоритм функції “Сповіщення про навчальну діяльність”

Останньою функцією для проектування є “Повідомлення від викладача”. Звичайно, що більшість викладачів наразі створюють окремі telegram-чати, користуються електронною поштою, МІХ.СумДУ або альтернативними каналами інформування. Проте всі ці канали інформування об’єднує один недолік - не всі студенти регулярно перевіряють інформацію. Чатів у месенджері може бути чимало, а листи на електронній пошті або МІХ.СумДУ перевіряються не дуже часто. Саме тому цю функцію важливо додати до майбутнього продукту. Викладач підготовлює повідомлення, надсилає до бота, обирає групу студентів якій треба його надіслати та відбувається автоматична розсилка по студентах.

На рисунку 2.4 представлено графічний алгоритм функції “Повідомлення від викладача”.

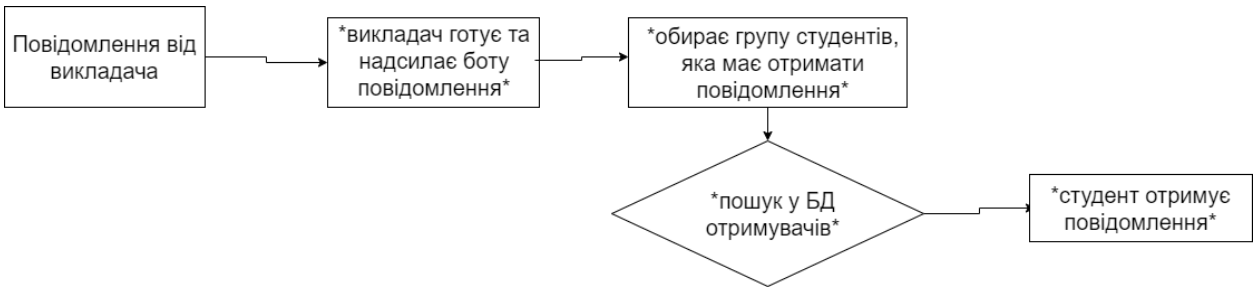

Рис.2.4 Алгоритм функції “Повідомлення від викладача”

Отже, ми завершили логічне проектування. В ньому ми розібрали на “атоми” кожну з функцій, а також спроектували алгоритм дій для кожної з них. Тепер ми можемо перейти до користувацького проектування.

2.1.3 Користувацьке проектування

Визначивши функції нашого майбутнього продукту та розібравши, спроектувавши кожну з них, ми можемо перейти до користувацького проектування. Його сутність полягає в тому, щоб правильно розмежувати ролі користувачів telegram-бота і визначити перелік функцій для кожного з них.

Всього буде 3 основні ролі у майбутнього продукта - студент, викладач та адміністратор. Розглянемо кожну з них окремо.

1. Студент. Він отримує інформацію від викладача за допомогою telegram-бота. Він має доступ до навчальних матеріалів, розкладу пар, отримує сповіщення про модульні та залікові контролі, важливі повідомлення щодо поточного навчального процесу.
2. Викладач. Він займається наповненням інформації для своїх студентів - надсилає до бази даних перелік навчальних матеріалів, встановлює в розкладі події для сповіщення, розміщує важливі повідомлення для студентів.
3. Адміністратор. Він займається реєстрацією студентів та викладачів у системі, створює окремі групи, курси. Тобто, займається наповненням учасниками навчального процесу. Окрім цього, він є відповідальним за відповіді на найчастіші запитання, також займається наповненням цієї бази.

На рисунку 2.5 зображено графічне представлення розмежування ролей та функцій користувачів telegram-бота.

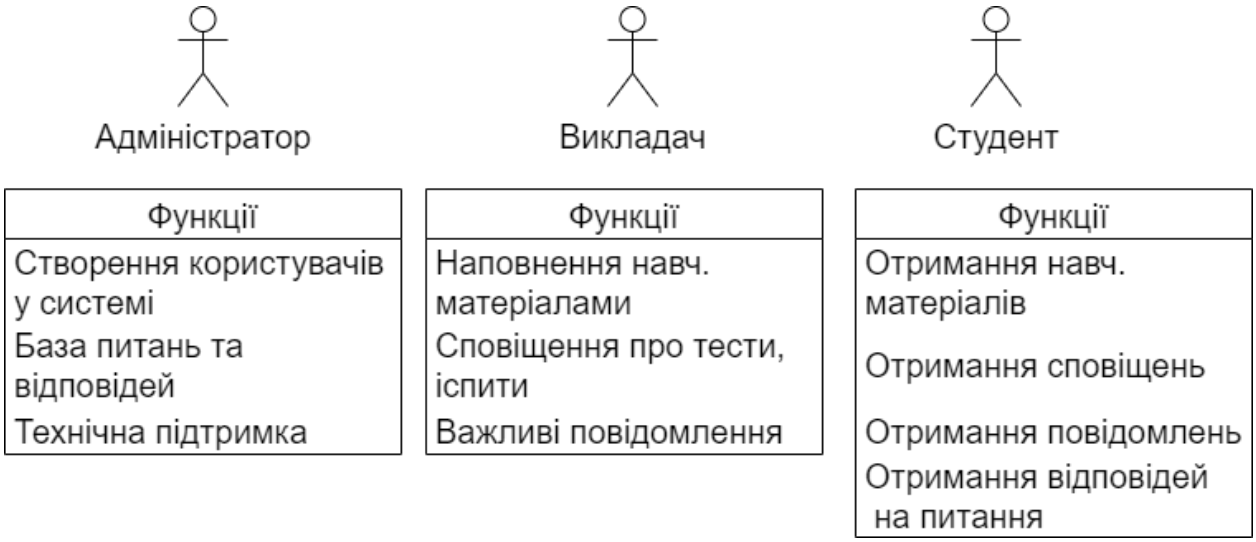

Рис.2.5 Розмежування ролей та функцій telegram-бота

Отже, ми повністю спроектували telegram-бота та можемо перейти безпосередньо до процесу розробки.

3 РОЗРОБКА TELEGRAM-БОТА

3.1 Огляд та вибір мови програмування

Функціональність Telegram-бота можна реалізувати різними шляхами, кожен з яких має свої переваги. Так, створення чат-бота в телеграмі досить просто виконується наступними способами:

- В якості мови програмування можна розглянути C#. При цьому, не потрібно з нуля писати великі масиви коду - набагато простіше скористатися готовим набором шаблонів. На жаль, такий варіант підходить в повній мірі тільки для користувачів Windows.
- Серед інших варіантів можна обрати PHP. Завдяки великій популярності цієї мови програмування, існує безліч бібліотек для розробки ботів в месенджері Telegram. Також це спрощує підтримку в майбутньому, надає можливість знайти відповіді на більшість можливих питань. Проте, популярність PHP приховує за собою велику кількість вразливостей, що дозволять недобросовісним користувачам потрапити, так би мовити, до нутра бота та впровадити шкідливий код, що несе за собою погані наслідки.
- Нарешті, серед мов програмування, що дозволяють швидко і безпроблемно виконати створення бота в телеграмі, є Python. В основному це пов'язано з широкими можливостями, доступними як

при використанні стандартних бібліотек, так і з застосуванням вже готових варіантів, таких як Telebot, Pyrogram, Aiogram, розрахованих на роботу безпосередньо з Telegram. Python має простий та зрозумілий синтаксис, отже проблем у майбутній підтримці не виникне. На відміну від PHP, що також не має проблем з синтаксисом, дана мова програмування є безпечнішою та забезпечує більшу швидкодію та ефективність завдяки можливості асинхронного програмування.

Виходячи з порівняння, для розробки було обрано мову Python через наявність sugar syntax, перевагу в безпеці, крос-платформеність та можливість застосовувати асинхронні підходи в програмуванні.

3.2 Вибір додаткових засобів для розробки

Обравши мову програмування, перейдемо до обрання додаткових засобів для розробки - бібліотек, СУБД та інших.

Мною було обрано такий стек технологій для розробки продукту:

1. Aiogram - асинхронна бібліотека для розробки telegram-ботів на мові Python. Вона має декілька переваг над іншими відомими бібліотеками, а саме:
    - асинхронність - дозволяє ефективно розподіляти ресурси, зменшуючи навантаження на сервіс;

- наявність “машини станів” (state machine) - дозволяє в оперативній пам’яті або Redis, який ми розглянемо пізніше, зберігати усі дії користувачів для правильного розподілення відповідей на запити користувачів;
- велика спільнота користувачів - дозволяє знайти відповіді майже на будь-які питання, а також забезпечує постійну підтримку бібліотеки;

Завдяки цим перевагам, а також наявності унікальних функцій, було обрано саме цю бібліотеку.

2. GINO - легка ORM (Object-Relational Mapping) на базі SQLAlchemy, яка дозволяє працювати з базою даних, не відходячи від синтаксису мови Python. Тобто, якщо розробник не досконало знає SQL, то GINO допоможе сформувати запити мовою Python, спроектувати базу даних тощо.
3. Redis - розподілене сховище пар ключ-значення, яке забезпечить безперебійність використання продукту. За допомогою Redis та state machine у Aiogram, навіть під час технічних робіт або перезавантаження серверу, зберігаються останні запити студентів, тому після відновлення роботи відповідь буде надана на кожен запит.
4. PostgreSQL - реляційна база даних, в якій зберігатимуться
5. У перспективі, можна спростити адміністрування, а саме додавання та видалення інформації за допомогою адміністративної панелі з

використанням веб-фреймворку Django. Даний фреймворк має в собі заготовлені шаблони для адміністративних панелей, а побудова логіки їх роботи та підтримка не є складною для розробників. Тим паче, у даному випадку.

Отже, ми розглянули та обрали інструменти, які потрібні для реалізації нашої мети - розробки працюючого telegram-боту, який допомагатиме студентам та викладачам у навчальному процесі.

3.3 Реєстрація бота в системі

Перш ніж приступати до роботи над кодом бота Telegram, має сенс його зареєструвати. Що добре, сам цей процес дуже простий і повністю автоматизований:

Починаєте чат з аккаунтом @BotFather

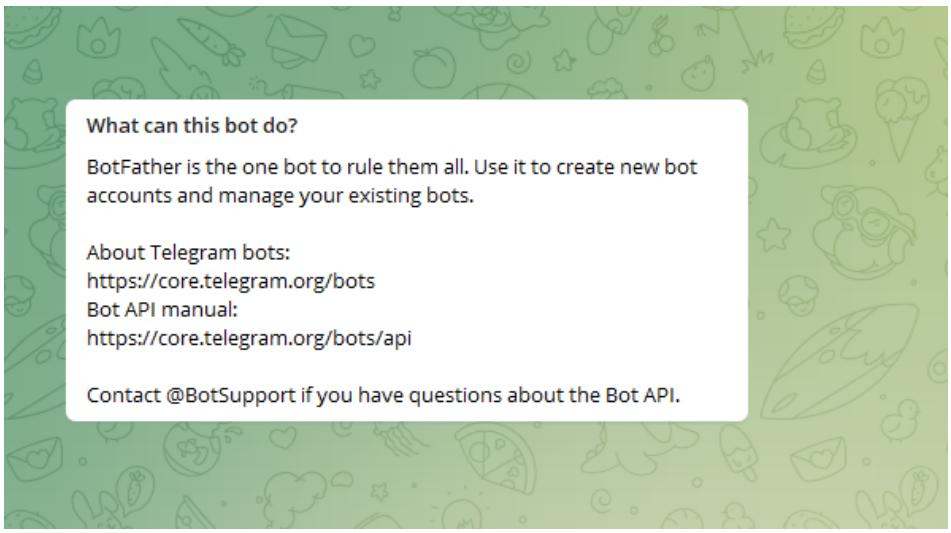

Рис.3.1 Меню BotFather

Вводите базову команду на створення нового бота /newbot. Відправляєте бажане вами ім'я для нового бота і його юзернейм, який обов'язково повинен закінчуватися на "bot". Отримуєте токен, за допомогою якого відбувається взаємодія вашого боту із Telegram API.

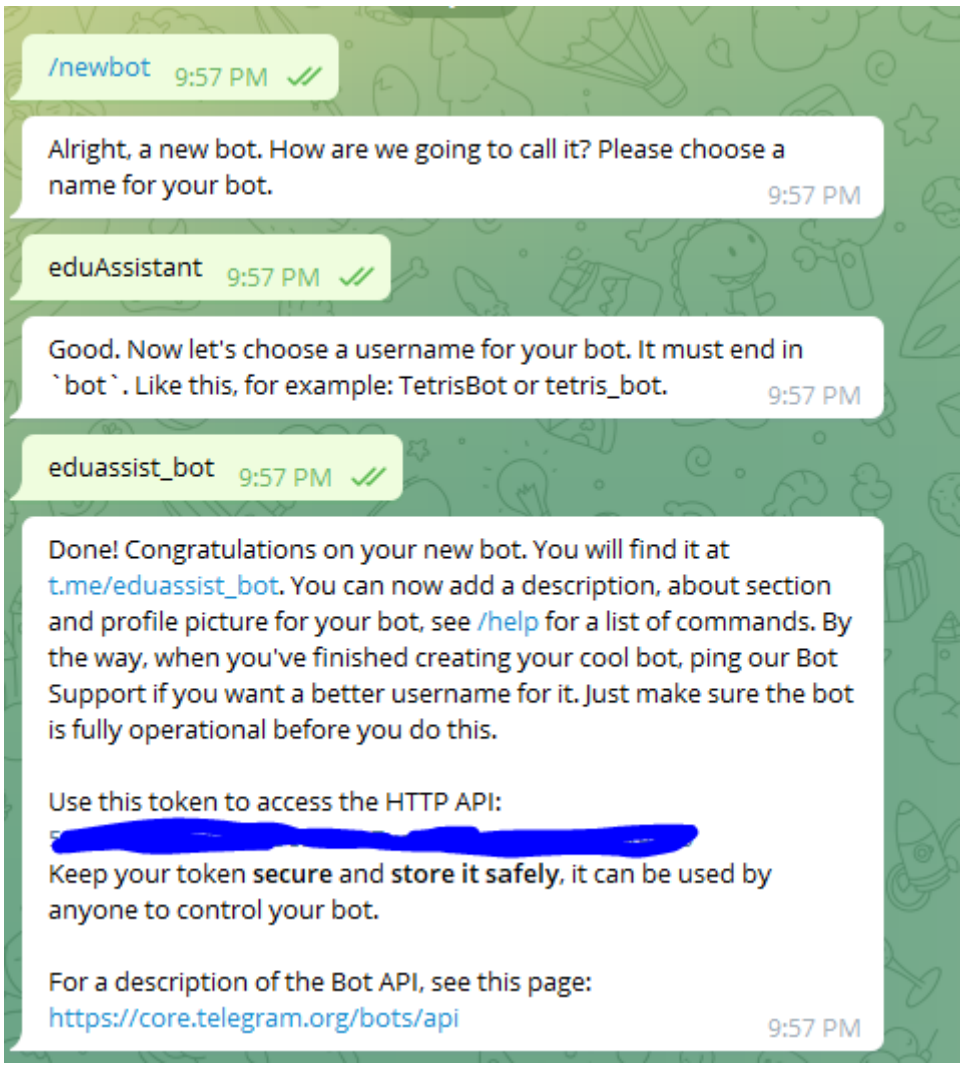

Рис.3.2 Процес реєстрації бота в системі

При бажанні, відразу можна ввести опис для бота, список його команд і відповідне зображення або відкласти ці дії на потім.

3.4 Тестування бота

Після розробки бота, фрагменти коду якого буде продемонстровано у додатку А, переходимо безпосередньо до його тестування. Ми переходимо за його нікнеймом та натискаємо на кнопку /start. Нас зустрічає вітальне повідомлення та клавіатура з діями, які ми можемо обрати (рис.3.3).

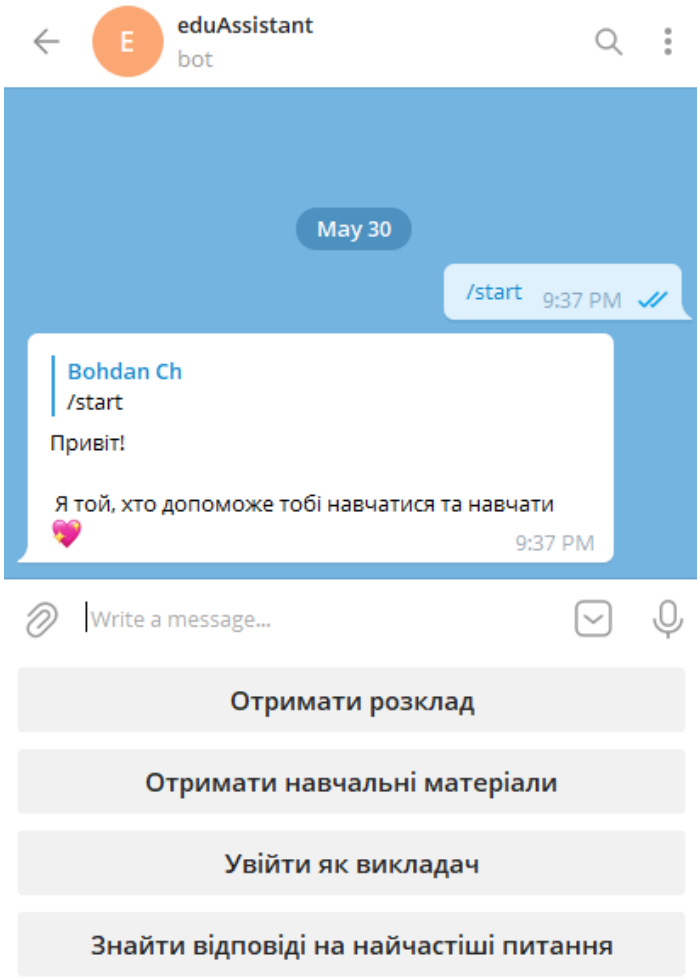

Рис 3.3 Початок роботи з ботом

Почнемо з найпростішої функції - “Знайти відповіді на найчастіші питання”. Натискаємо на кнопку з аналогічною назвою та нас зустрічає клавіатура з переліком питань (рис. 3.4).

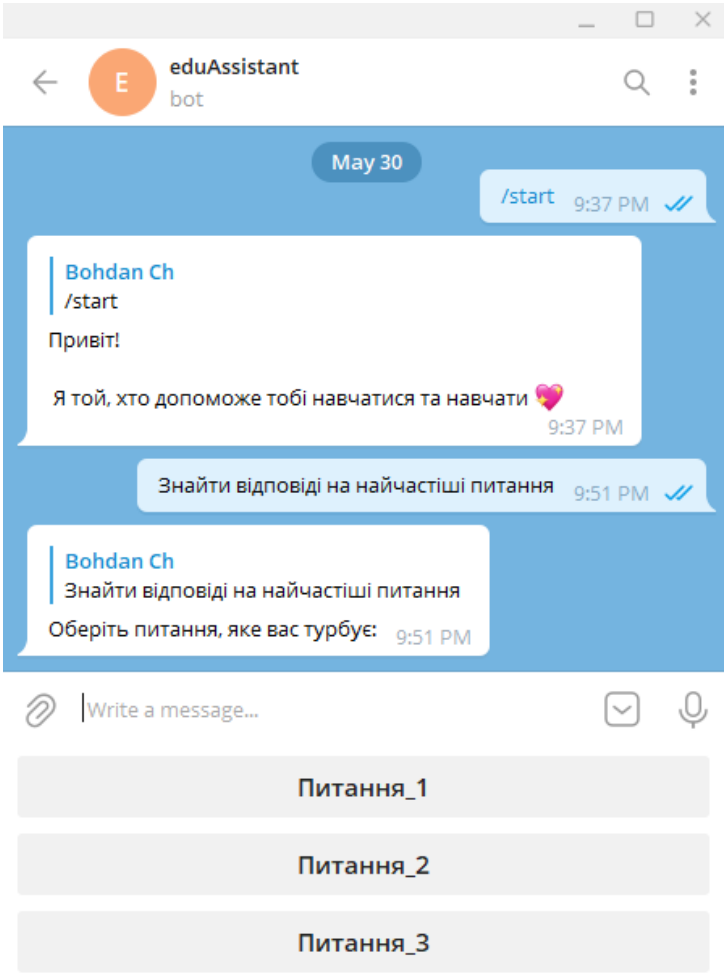

Рис.3.4 Початок роботи з функцією “Найчастіші питання”

Варто зазначити, що наразі нас зустрічатимуть кнопки формату “Питання_1”, Предмет“_1” тощо. Це обумовлено тим, що наповнення бота наразі є тестовим, проте без зайвих проблем можливо замінити на більш корисний текст. Тобто, наразі в більшості випадків у боті використовується так званий текст-риба.

Обираємо перше питання, натиснувши на кнопку “Питання_1” і миттєво отримуємо на нього відповідь (рис.3.5).

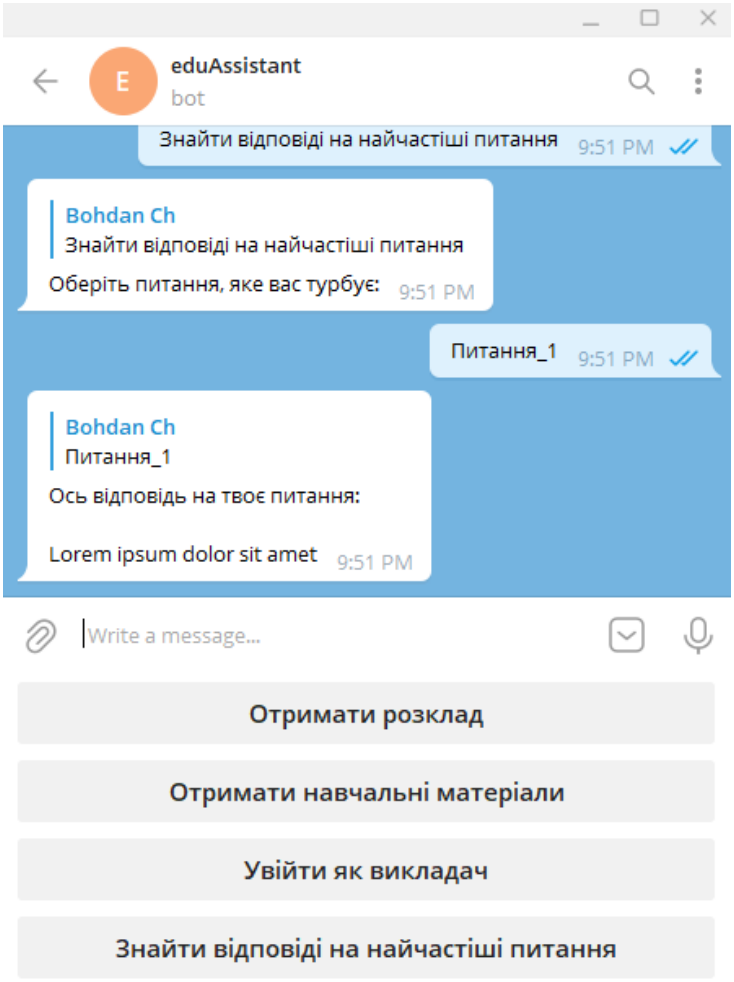

Рис.3.5 Відповідь бота на перше питання

Далі ми протестуємо функцію отримання розкладу. Варто також згадати, що наразі наповнення бази даних студентами та викладачами відбувається без реєстрації, за допомогою адміністратора. Тому бот зможе одразу ідентифікувати студента або викладача без зайвого підтвердження (рис.3.6).

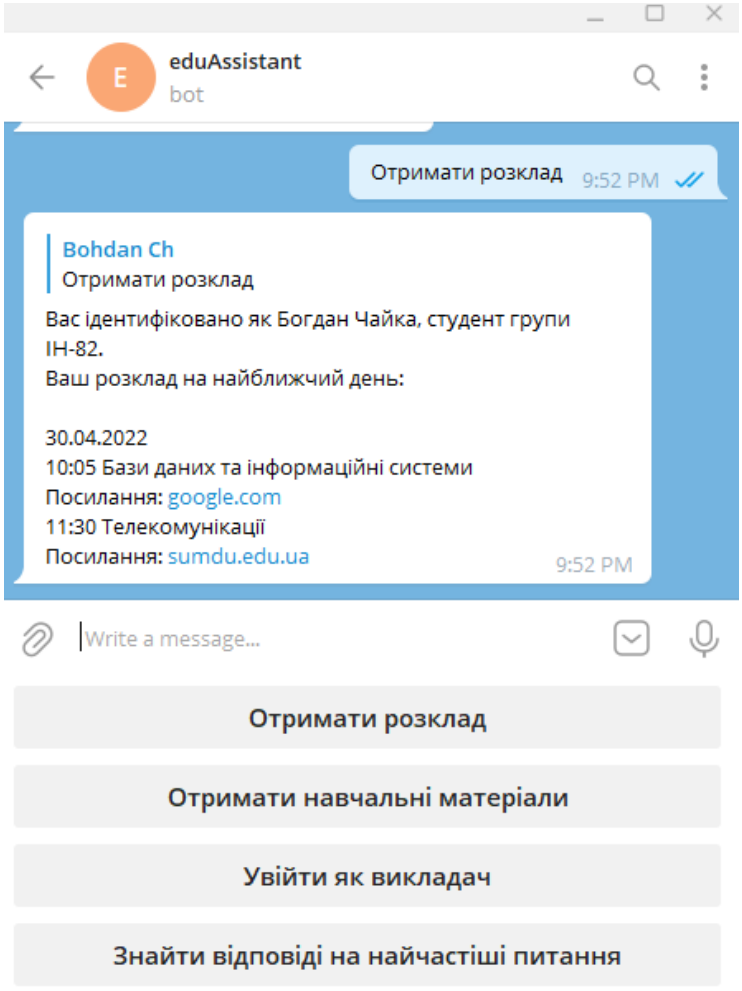

Рис.3.6 Отримання розкладу від бота

Це досягнуто завдяки тому, що кожен користувач у Telegram має унікальний user id, тому даний user id додається до кожного користувача у системі. Бот шукає збіги згідно user id та миттєво ідентифікує, хто з ним має справу.

Тепер перевіримо на працездатність функцію отримання навчальних матеріалів. Натискаємо на кнопку “Отримати навчальні матеріали” та нас зустрічає панель з предметами, за якими ми зараз навчаємося (рис.3.7).

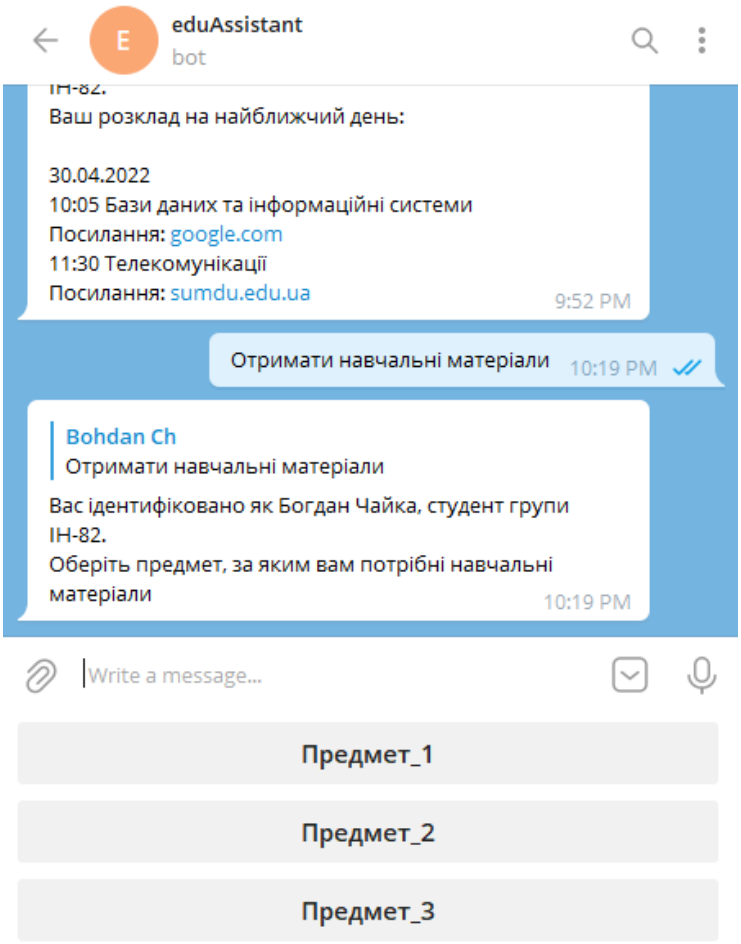

Рис.3.7 Функція “Отримати навчальні матеріали”

Ми обираємо “Предмет_2” та у зручному форматі отримуємо посилання на матеріали для завантаження: лабораторні роботи, лекції, корисні матеріали (рис.3.8).

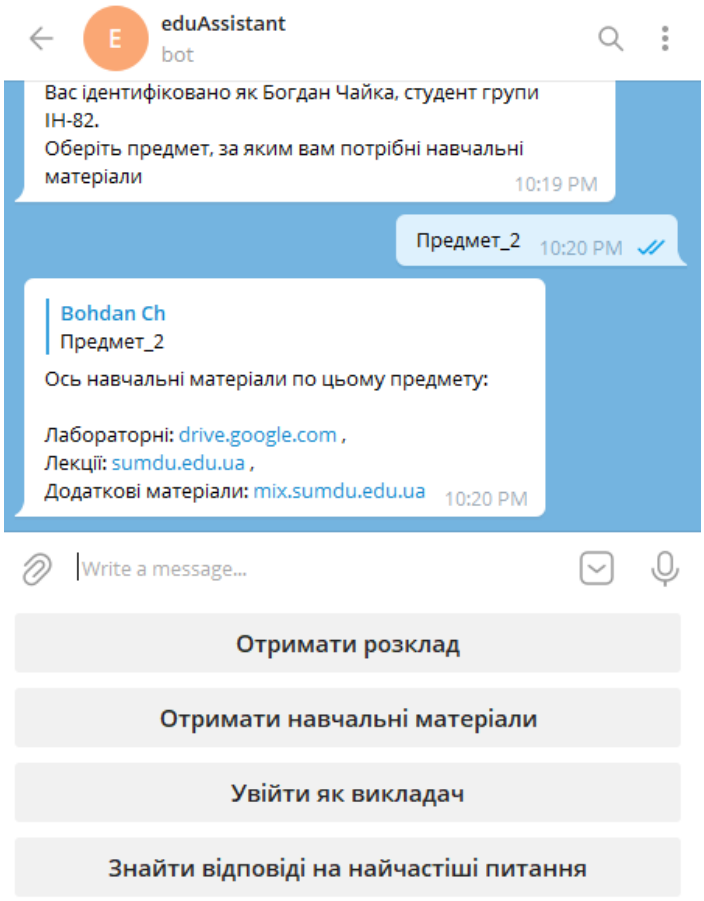

Рис.3.8 Повідомлення від бота з навчальними матеріалами

Викладач має інші функції. Для їх перевірки ми на той же user id створили викладача. Натискаємо на кнопку “Увійти як викладач” і нас також по ньому ідентифікує, але зустрічає вже зовсім інше меню (рис.3.9).

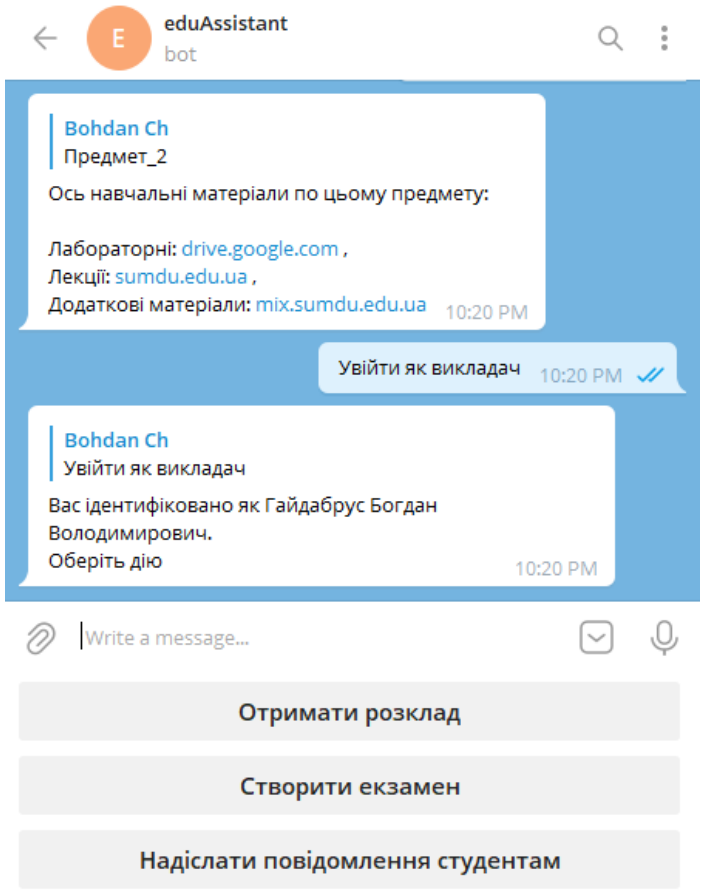

Рис.3.9 Меню викладача в системі

Так, у викладача також є кнопка, але вона виконує трохи іншу функцію, а саме надає викладачу розклад лише його пар. На рис.3.10 ми можемо побачити, що бот відповідає викладачу про те, що завтра у нього пар не буде і він може зайнятися перевіркою лабораторних робіт.

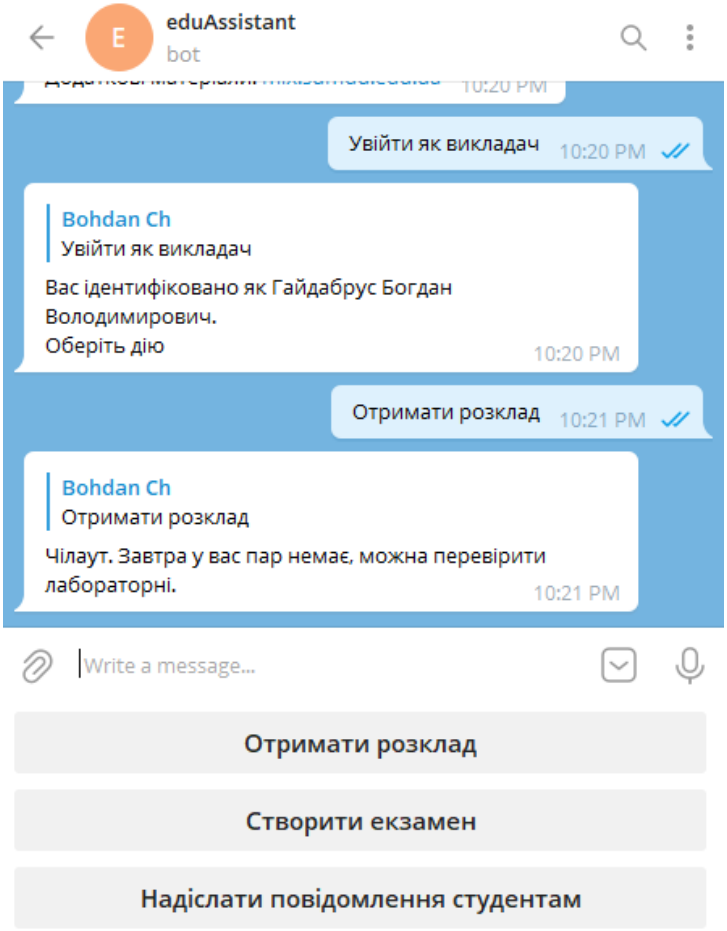

Рис.3.10 Розклад викладача у боті

Перевіримо функцію створення екзамену, про який бот буде сповіщати систему. Натискаємо на кнопку “Створити екзамен” та нас зустрічає панель з предметами викладача (рис.3.11). Так як викладач може викладати декілька предметів, то для зручності створена клавіатура з їх переліком.

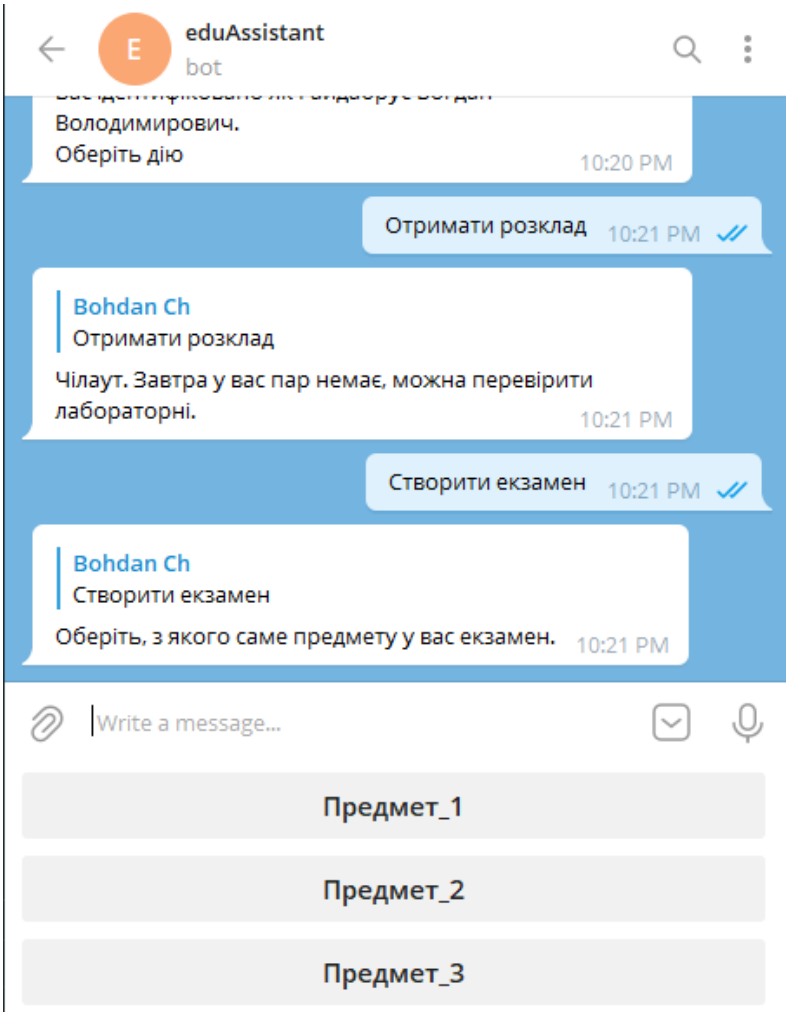

Рис.3.11 Початок роботи з функцією “Створити екзамен”

Ми обираємо “Предмет_3” і бот нас запитує, для якої саме групи або який груп цей екзамен (рис.3.12).

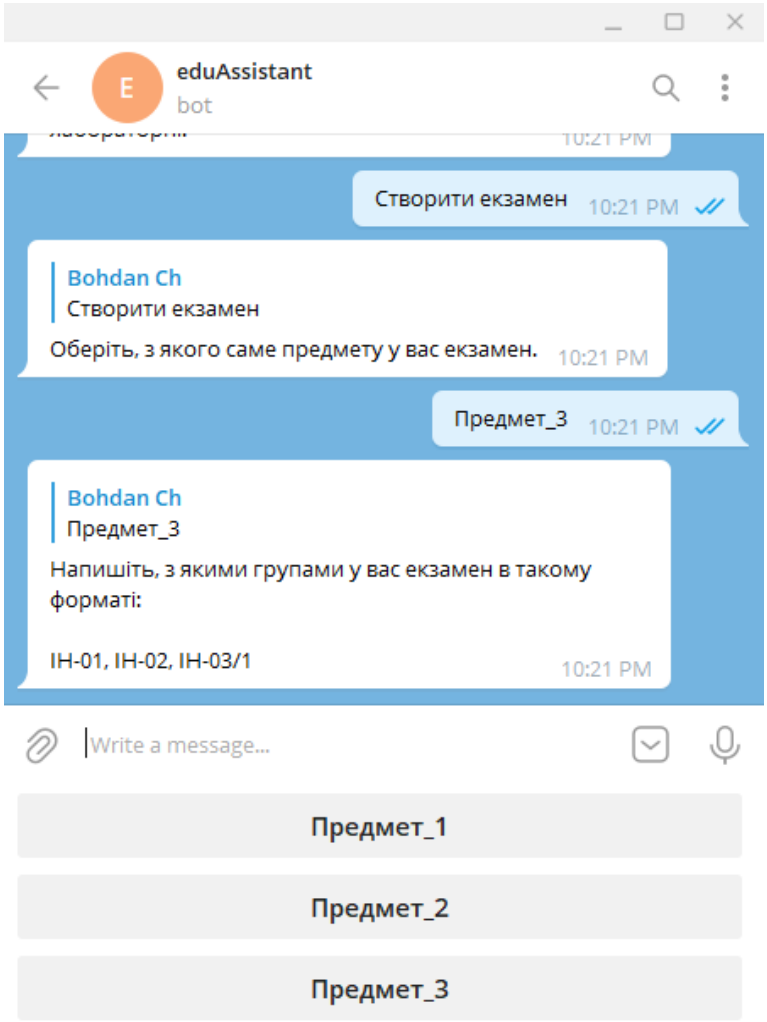

Рис.3.12 Вказання груп(и) при створенні екзамену

Ми вказуємо наприклад ІТ-11, після чого бот в нас запитує, в який день він відбудеться (рис.3.13).

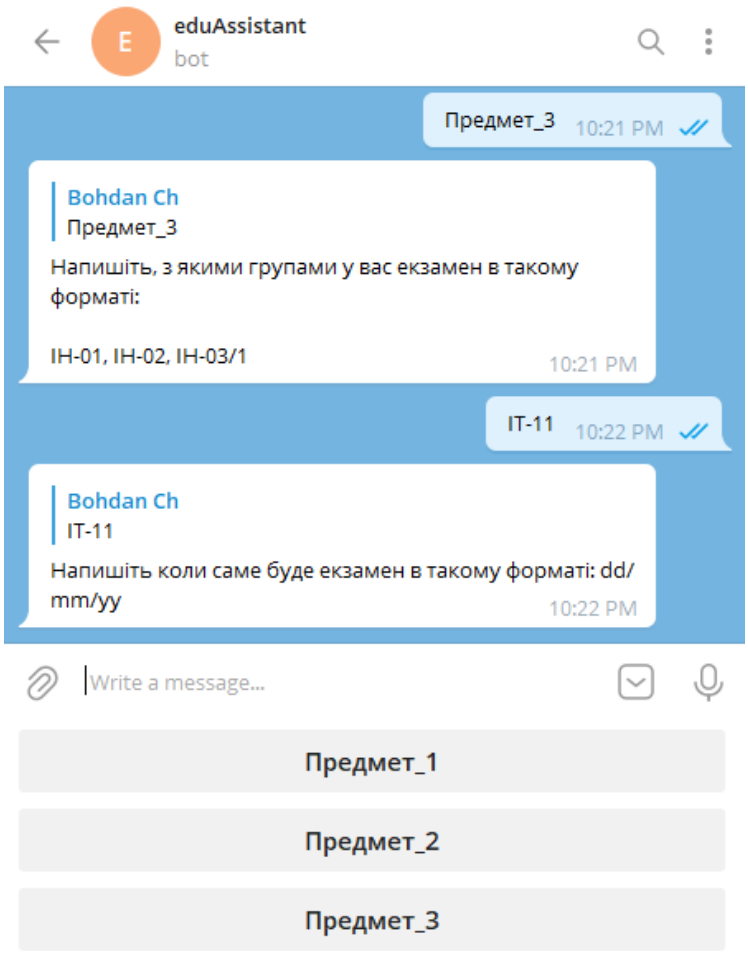

Рис.3.13 Питання бота про дату екзамена

Бот навмисно не запитує час, адже в цьому немає сенсу. Дана інформація також буде в розкладі СумДУ, а дана функція створена для того, щоб сповіщати про день екзамену за певний проміжок часу.

Після виконання всіх дій, викладача зустрічає повідомлення з даними про створений навчальний захід (рис.3.14).

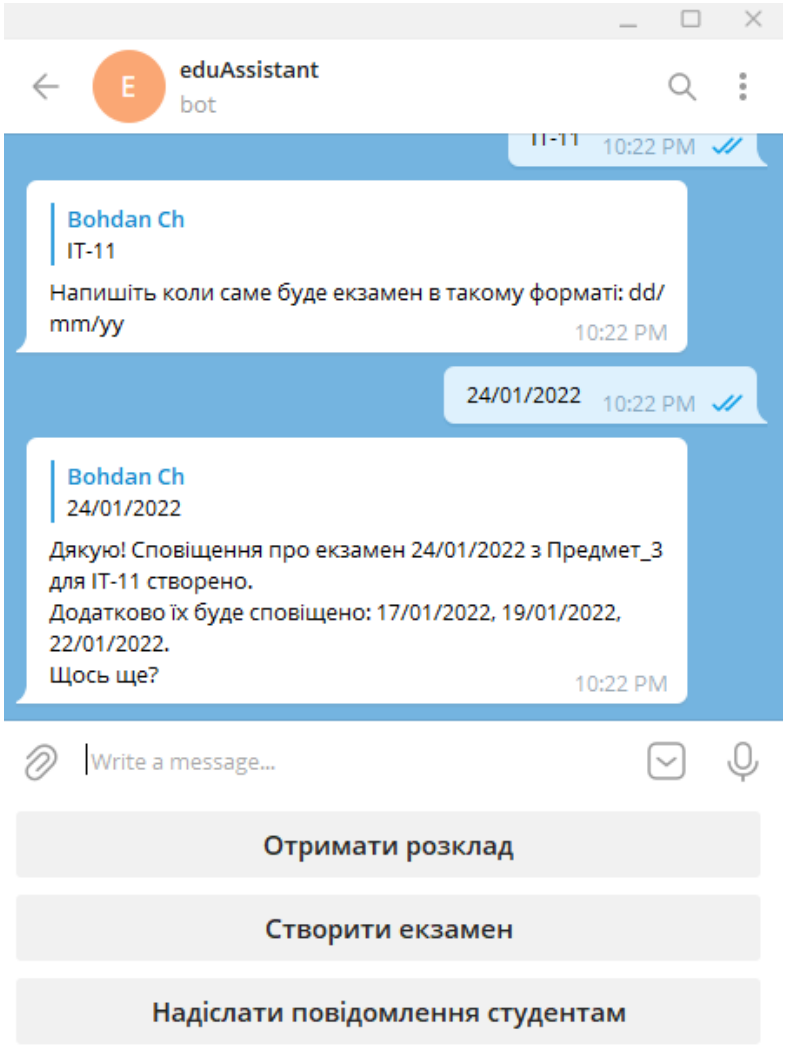

Рис.3.14 Повідомлення про успішне створення сповіщення про навчальну подію

В повідомленні також вказано, коли саме буде сповіщено студентів про екзамен - бот це підраховує автоматично (за 7/5/3 днів).

Отже, бот працює справно для всіх учасників навчального процесу, згідно нашого та стороннього тестування.

ВИСНОВКИ

В ході виконання роботи було розглянуто вже існуючі освітні сервіси в еко-системі СумДУ, проведено проектування telegram-боту згідно запитів учасників навчального процесу, обрано мову програмування та інші програмні засоби для досягнення мети - реалізації проекту, безпосередньо розробка проекту та його тестування.

Результатом виконаної роботи є спроектований, розроблений, зареєстрований та працездатний telegram-бот eduAssistant.

СПИСОК ЛІТЕРАТУРИ

1. MIX.СумДУ [Електронний ресурс] – Режим доступу: https://mix.sumdu.edu.ua/
2. Особистий кабінет СумДУ [Електронний ресурс] – Режим доступу: https://cabinet.sumdu.edu.ua/
3. Документи СумДУ [Електронний ресурс] – Режим доступу: https://document.sumdu.edu.ua/
4. Нормативна база СумДУ [Електронний ресурс] – Режим доступу: https://normative.sumdu.edu.ua/
5. Знайомство з Aiogram [Електронний ресурс] – Режим доступу : https://mastergroosha.github.io/telegram-tutorial-2/quickstart/
6. Redis на практичних прикладах [Електронний ресурс] – Режим доступу : https://habr.com/en/company/manychat/blog/507136/
7. GINO Basics [Електронний ресурс] – Режим доступу: https://python-gino.org/docs/en/1.1b2/tutorials/tutorial.html

ДОДАТОК А

Вихідний код сервісу

Додавання екзамену

```
from aiogram import types
from aiogram.dispatcher import FSMContext

from bot.chains.base.kb import start_kb
from bot.chains.record_event.kb import cancel_record
from bot.chains.record_event.state import RecordExam
from bot.core import dp, bot
from db.models.user import User
from db.models.exams import Exams


@dp.callback_query_handler(lambda x: x.data == "exam_record_cancel", state="*")
async def cancel(c: types.CallbackQuery, state: FSMContext):
    await c.message.delete_reply_markup()
    await state.finish()
    await bot.send_message(
        c.from_user.id, "Скасовано створення навчальної події", reply_markup=start_kb
    )
    await c.answer("Скасовано")


@dp.message_handler(regexp="Захід", state="*")
async def auth(msg: types.Message, state: FSMContext):
    await RecordExam.wait_password.set()
    data_state = await msg.answer(
        "Напишіть пароль, який був наданий адміністратором.", reply_markup=cancel_record
    )
    await state.update_data({"message": data_state})
```

```
@dp.message_handler(state=RecordExam.wait_password)
async def record_exam_start(msg: types.Message, state: FSMContext):
    data = await state.get_data()
    await data.get("message").delete_reply_markup()
    check_password = await User.check_password(msg.text)
    if not check_password:
        await msg.answer("Невірний пароль!", reply_markup=cancel_record)
    else:
        auth_data = await User.auth_data(msg.text)
        await state.update_data(
            {
                "username": auth_data["name"],
                "faculty": auth_data["faculty"],
                "user": auth_data["id"],
            }
        )
        answer = await msg.answer(
            f'Вітаю, {auth_data["name"]}! {auth_data["faculty"]} найкращий! \n\n'
            f"Поділіться зі мною назвою предмету та типом контролю, який Ви плануєте
провести :)",
            reply_markup=cancel_record,
        )
        await RecordExam.wait_event_name.set()
        await state.update_data({"message": answer})


@dp.message_handler(state=RecordExam.wait_event_name)
async def record_event_name(msg: types.Message, state: FSMContext):
    data = await state.get_data()
    await data.get("message").delete_reply_markup()
    await state.update_data({"event_name": msg.text})
    answer = await msg.answer(
```

```
        f"Надішліть мені дату навчальної події у такому форматі: dd/mm/yyyy", re-
ply_markup=cancel_record
    )
    event = await Exams.create(event_name=msg.text, user=data.get("user"))
    await state.update_data({"message": answer, "event_id": event.id})
    await RecordExam.wait_exam_date.set()


@dp.message_handler(state=RecordExam.wait_exam_date, content_types=["photo"])
async def done_recording(c: types.CallbackQuery, state: FSMContext):
    await c.message.delete_reply_markup()
    await state.finish()
    await bot.send_message(
        c.from_user.id,
        "Добре, я обов'язково сповіщу студентів про твій екзамен або тест! Якщо раптом -
звертайся ;)",
        reply_markup=start_kb,
    )
    await c.answer("Готово")
```

Поставити запитання

```
from aiogram import types
from aiogram.dispatcher import FSMContext

from bot import files
from bot.chains.base.kb import start_kb
from bot.chains.ask_question.kb import cancel_kb
from bot.chains.ask_question.state import AskQuestion
from bot.core import dp, bot
```

```
from bot.tree import QuestionAnswerPath

QuestionAnswer = files.loadFile(QuestionAnswerPath)


def mapping_json(nd, p):
    return QuestionAnswer[0][nd][f"{p}"]


@dp.callback_query_handler(lambda x: x.data == "question_cancel", state="*")
async def cancel_action(c: types.CallbackQuery, state: FSMContext):
    await c.message.delete_reply_markup()
    await state.finish()
    await bot.send_message(c.from_user.id, "Дію скасовано", reply_markup=start_kb)
    await c.answer("Скасовано")


@dp.message_handler(regexp="Поставити запитання", state="*")
async def list_question(msg: types.Message, state: FSMContext):
    answer = await msg.answer(
        "Оберіть та напишіть ваше питання:\n\n"
        "1. Питання_1\n"
        "2. Питання_2\n"
        "3. Питання_3",
        reply_markup=cancel_kb,
    )
    await AskQuestion.wait_question.set()
    await state.update_data({"message": answer})


@dp.message_handler(state=AskQuestion.wait_question)
async def answer_on_question(msg: types.Message, state: FSMContext):
    data_state = await state.get_data()
    await data_state.get("message").delete_reply_markup()
```

```
    answer = await msg.answer(
        f"Отже, відповідь на твоє питання: \n\n" f"{mapping_json(msg.text, 'answer')}",
        reply_markup=cancel_kb,
        parse_mode="HTML",
    )
```

Модель бази даних User

```
import hashlib

from db.core import db
from db.models.base import TimedBaseModel



class User(TimedBaseModel):
    __tablename__ = "users"

    id = db.Column(db.Integer, primary_key=True, index=True, unique=True)
    name = db.Column(db.String(255), nullable=False)
    password = db.Column(db.String(255), nullable=False)
    faculty = db.Column(db.String(50), nullable=False)
    group = db.Column(db.String(50), nullable=True)
    is_teacher = db.Column(db.Boolean, default=False)
    is_blocked = db.Column(db.Boolean, default=False)

    @classmethod
    async def check_password(cls, password):
        password = await User.query.where(
            User.password == hashlib.md5(password.encode("utf-8")).hexdigest()
        ).gino.first()
        return bool(password)

    @classmethod
    async def auth_data(cls, password):
        usr = (
```

```
    await User.select("id", "name", "faculty")
    .where(User.password == hashlib.md5(password.encode("utf-8")).hexdigest())
    .gino.first()
)
return usr
```