Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем

РОЗРОБКА БОТА НА ОСНОВІ MICROSOFT .NET ТЕХНОЛОГІЙ

Текстова частина до курсової роботи
за спеціальністю “Інженерія програмного забезпечення” 121

Керівник курсової роботи
с.в. Вовк Н. Є.

_____
(підпис)
“____” _____ 2021 р.
Виконала студентка студентка 4-го
курсу Побережець Б. А.
“____” _____ 2021 р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем,

доцент Жежерун О. П.

“____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентці _____Побережець Богдані_____

__4-го__ курсу факультету інформатики

ТЕМА: Розробка бота на основі Microsoft .Net технологій

Вихідні дані:

- Створення шаблону для реалізації Microsoft боту для системи автоматичного запису

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1. Аналіз предметної області. Постановка завдання курсової роботи
2. Теоретичні відомості.
3. Опис реалізації кроків бота.
4. Висновки

Список джерел

Додатки (за необхідністю)

Дата видачі “___” _____ 2021 р.

Керівник _____ Завдання отримав _____

Календарний план виконання курсової роботи:

Тема: Розробка веб-сайту інтернет-оголошень

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	12.10.2020	
2.	Огляд технічної літератури за темою роботи.	31.12.2020	
3.	Знайомство з базою даних	31.01.2021	
4.	Знайомство з Bot Framework SDK	28.02.2021	
6.	Розробка боту	25.03.2021	
7.	Написання текстової частини.	30.03.2021	
8.	Створення слайдів для доповіді та написання доповіді.	05.04.2021	
9.	Обговорення отриманих результатів з керівником.	08.04.2021	
11.	Остаточне оформлення текстової частини та слайдів.	10.04.2021	
12.	Захист курсової роботи.	19.04.2021	

Студент Побережець Б. А. _____

Керівник Вовк Н. Є. _____

“_____”_____ 2021 р.

Зміст

Анотація

У даній роботі проаналізовано сервіс системи автоматизованого запису та розроблено логіку для боту під цей сайт.

В першій частині роботи досліджується та аналізується предметна область. Береться до уваги аналіз різних технологій для створення ботів та обирається система. В розділі два побудовано логіку застосунку, проаналізовано можливості, які бот міг би мати та частково реалізовано розроблений функціонал. В крайній частині роботи продемонстровано безпосередня робота бота з проведеним тестуванням.

C#, DATABASE, MYSQL, BOT FRAMEWORK SDK

Вступ

Києво-Могилянська академія багата на різноманітні допоміжні сервіси для автоматизації деяких процесів університету. Наші найкращі випускники не один раз долучались до розробки таких веб-сайтів.

Яскравими прикладом таких рішень є візитна картка університету – сайт Києво – Могилянської академії. Його можна знайти відвідавши https://www.ukma.edu.ua.

Іншим представником гордості університету є освітня платформа DistEdu. З допомогою цієї платформи студенти мають змогу отримувати доступ до практичних або лекційних занять, читати деталі самостійних завдань та завантажувати їх, а також дізнаватись крайні новини. Сайт можна відвідати за посиланням https://distedu.ukma.edu.ua/.

Києво-Могилянська академія має особливу систему навчання. Тут студент має можливість самостійно записуватись на обрані дисципліни та створювати індивідуальний план. Спеціально для таких цілей було розроблено “Систему Автоматизованого Запису” (САЗ), що і буде об’єктом мого дослідження. Дана система надає можливість переглянути інформацію по будь-якій дисципліні з кожного факультету. Іншою перевагою сайту є запис на вже обрані дисципліни, а в подальшому групи. Після чого сформований план можна передивитись в розкладах.

Метою даної роботи є розробка бота, який забезпечить альтернативний спосіб використання системи автоматичного запису.

Об’єктом дослідження будуть різні технології, під час аналізу яких буде обрано найкращий варіант.

До методів дослідження відносяться:

- спостереження – детальний аналіз САЗ та всіх його можливостей;
- аналіз – поділ зібраної інформації на різні елементи для створення зв’язків під час розробки боту;

- розробка – пошуки найкращих рішень, що будуть підходити для якісної роботи бота.

Всі необхідні матеріали для спостереження та аналізу були надані. До них відноситься:

1) Доступ до веб-сайту;
2) Код до тестового середовища сайту САЗ;
3) Реалізована та наповнена тестова база даних під сайт САЗ;

В результаті роботи очікується:

- аналіз аналогів та конкурентів;
- розробка функціоналу для неавторизованого користувача;
- використання сховища даних (бази даних) для зберігання та оновлення інформації;
- висновки та подальші рекомендації;

Під час роботи планується використання об’єктно-орієнтованої парадигми інтегрованої з СУБД.

1. Аналіз предметної області. Постановка завдання курсової роботи

1.1. Аналіз сучасного стану питання

На сьогоднішній день, боти як допоміжний сервіс є досить популярним видом розробки. Завдяки ботам є можливість не використовувати веб-аналоги, адже вони імітують живе спілкування.

Також, більшість людей використовують соціальні мережі. Саме тому додаткове встановлення бота не є проблематичним кроком. Зважаючи на те, що більшість проводить багато часу в різних соціальних мережах, багато новин можна донести набагато швидше, аніж через пошту чи на офіційному сайті.

1.2. Огляд існуючих аналогів розробки

Для офіційного сайту САЗ вже існує телеграм-бот:

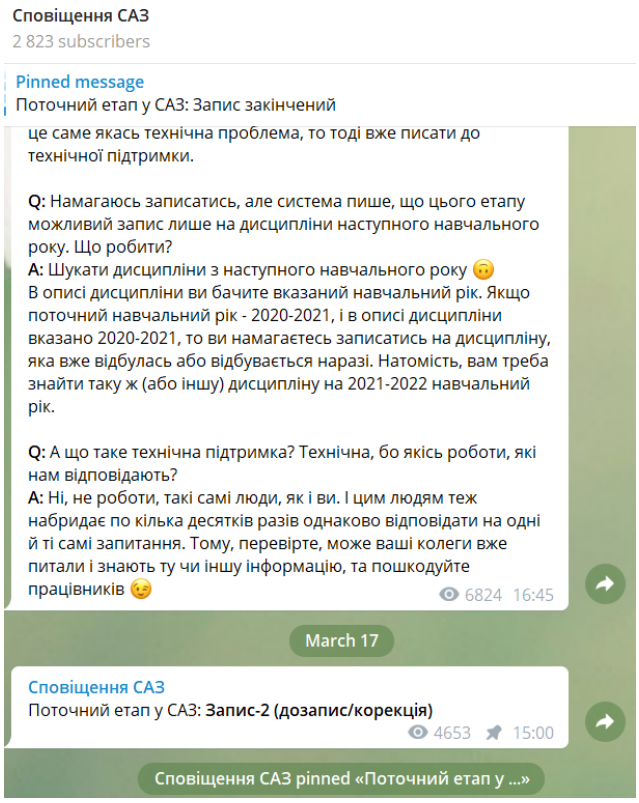

Рисунок 1 - аналог бота для САЗ

Проте, його основна ціль – сповістити студента про початок або закінчення запису на дисципліну та повідомити про поточний стан при створені індивідуального плану.

Проаналізувавши весь доступний функціонал бота, було зауважено, що функціонал можна розширити. Схожий бот - може не лише сповіщавати про певні оновлення, а й також давати детальну інформацію та ознайомлювати студентів з розкладом\дисциплінами.

1.3. Постановка задачі

Під час реалізації логіки бота було вирішено надати наступні функціональні вимоги:

- кожен користувач має мати доступ до головного меню, за яким в результаті певних дій буде надаватись різна інформація;
- інтерфейс має бути достатньо зручним, щоб будь-який користувач міг легко користуватись ботом;
- в боті має бути розмежовування прав доступу відповідно до своєї функціональності, а саме: робота з авторизованим користувачем, та робота з не авторизованим користувачем. В залежності від цього, бот надаватиме доступ до різного функціоналу.

На рис. 2 показано весь доступний функціонал для не авторизованого користувача. Тут він матиме доступ до такої інформації:

- пошук та перегляд всіх доступний дисциплін університету;
- перегляд та завантаження розкладу для своєї спеціальності;
- прочитати всі доступні новини;
- знайти інструкції з певних питань;
- відповіді на найчастіші запитання;

Рисунок 2 - Функціонал не авторизованого користувача

Діаграма на рис. 3 показує можливості студента, після успішної авторизації.

Додатково буде надано такий функціонал:

- можливість авторизуватись за поштою\паролем;
- можливість авторизуватись з допомогою Office 365[1];
- змінити та налаштувати дані профілю;
- змінити та налаштувати пароль;
- переглянути свої групи;

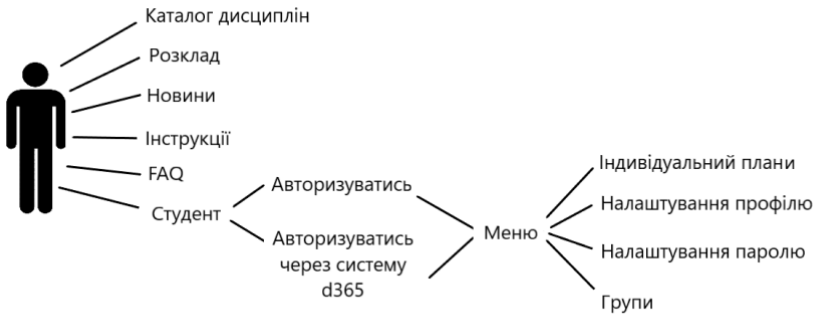

Рисунок 3 - Можливості студента

У даній роботі, буде розробка лише для не авторизованого користувача з демонстрацією повного функціоналу.

2. Теоретичні відомості

2.1. Вибір технічних засобів побудови системи

Аналізуючи предметну область було вирішено, що найкращим рішенням для студентів буде саме бот, адже на даний момент, майже 95% мають різні акаунти в соціальних мережах, в яких ця можливість присутня. Саме тому, при вирішені вибору технологій було взято до уваги такі мови як Java, Python, C#, що підтримують розробку ботів.

Під час аналізу об’єктів мов програмування, першим критерієм було зручність використання та швидкість роботи. Також, важливим фактором було саме об’єктно-орієнтована парадигма. Тим не менш, кожна з обраних мов є

достатньо швидкою та доволі легкою у використанні. Бонусом стала кросплатформеність.

Java – популярна мова програмування, що відкриває широкі можливості перед розробниками[2]. Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи [3]. Говорячи про цю мову програмування, варто зазначити такі переваги: велика кількість вбудованих бібліотек, надійність, швидкодія, вбудований збирач сміття.

Python - проста у використанні, та водночас повноцінна мова програмування, що надає набагато більше засобів для структурування і підтримки великих програм [4]. Завдяки модульності, коди цієї мови набагато коротші та більш зручні для читання. Зазвичай коди, написані мовою Python значно коротші від їх аналогів на C чи C++. Проте, невисока швидкість виконання програми та відсутність статичної типізації виступає в ролі недоліків для цієї мови.

Кращою технологією для ОС Windows є мова C#. Мова C# актуальна, в першу чергу тому, що дозволяє більш раціонально створювати популярні на сьогодні інтернет-додатки. C# тісно інтегрована з мовою XML, різноманітними веб-технологіями [5]. Перевагами цієї мови є підтримка поліморфізму та строга статична типізація.

В результаті аналізу наведених мов програмування, найкращим рішенням було обрано C#, завдяки можливості легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проєктів[6].

2.2. Архітектура системи

За основу було обрано працювати з пакетом SDK для Bot Framework. Такий фреймворк дозволяє будувати боти, що підтримують різні види взаємодій з користувачами. Наприклад, це може бути звичайна розмова – типовими стрінгами, або більш складна , що включає різні типи карток з картинками, кнопками або текстом.

The Bot Framework – це спеціальний пакет SDK для створення програми, що симулює спілкування з живою людиною. Він може використовувати мову, використовувати знання власної мови, виконувати обробку запитів та питань-відповідей.

Перш за все, варто зазначити, що таке Бот? **Роботи або боти** – це спеціальні акаунти, які можуть автоматично обробляти і відправляти повідомлення. Вони створюються програмістами і працюють у них на сервері[7].

До основного функціоналу роботів-ботів можна віднести такі:

- Вхід та виконання відповідних задач;
- розпізнавання вхідних даних від користувача ;
- створення відповідей для відправки користувачу;
- взаємодія через текст та мову, відображення відео або картинок під час діалогу.

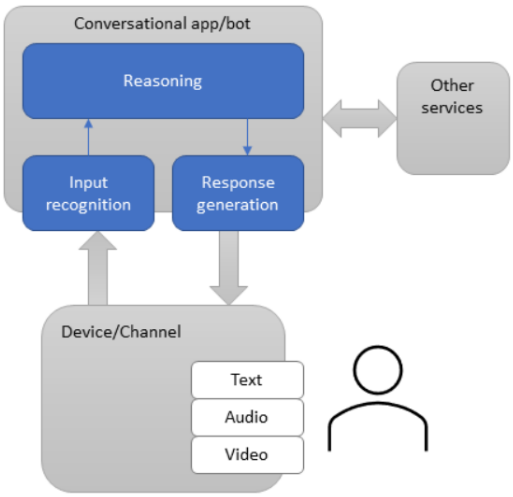

Рисунок 4 - Архітектура бота

На рис. 4 можна побачити просту архітектуру роботи бота. Унікальність цих роботів полягає у тому, що вони можуть як читати так і зберігати файли, а також використовувати різні API інтерфейси в інтеграції з базами даних, виконувати розрахунки.

Проте, перед тим як розпочати розробку бота, розробнику потрібно налаштувати середовище. До попередніх вимог належить:

- Базове знання ASP.NET Core та асинхронного програмування в C#
- Середовище ASP.NET Core 3.1 +;
- Bot Framework Emulator

2.3. СУБД

База даних (БД) — це організована структура, призначена для зберігання, зміни й обробки взаємопов’язаної інформації, переважно великих обсягів.[8] Метою бази даних є задоволення потреби у користувачів.

Система керування базами даних (СКБД) – це програмне забезпечення, яке надає можливість контрольованого доступу до даних, створювати, підтримувати та визначати базу даних.

Основні вимоги та функції для СКБД:

- Можливість створення бази даних;
- додавання\редагування\видалення даних;
- підтримка системи забезпечення захисту та відновлення при доступі до бази даних;
- забезпечення мінімальної надлишковості та несуперечливості даних;
- можливість колективного використання даних;
- підвищений рівень безпеки;
- незалежність прикладних програм від даних[9].

Система САЗ вже розроблена, тому потреби у створенні логіки бази даних боту. Під час розробки використовується MySQL система керування базами даних. MySQL поширюється безкоштовно і встановлюється на сервері, працюючим під операційною системою Windows [10].

3. Опис реалізації програмного продукту

3.1. Результат реалізації бази даних

Зважаючи на те, що система автоматизованого запису – це розроблений веб-застосунок, база даних була попередньо продумана та розроблена.

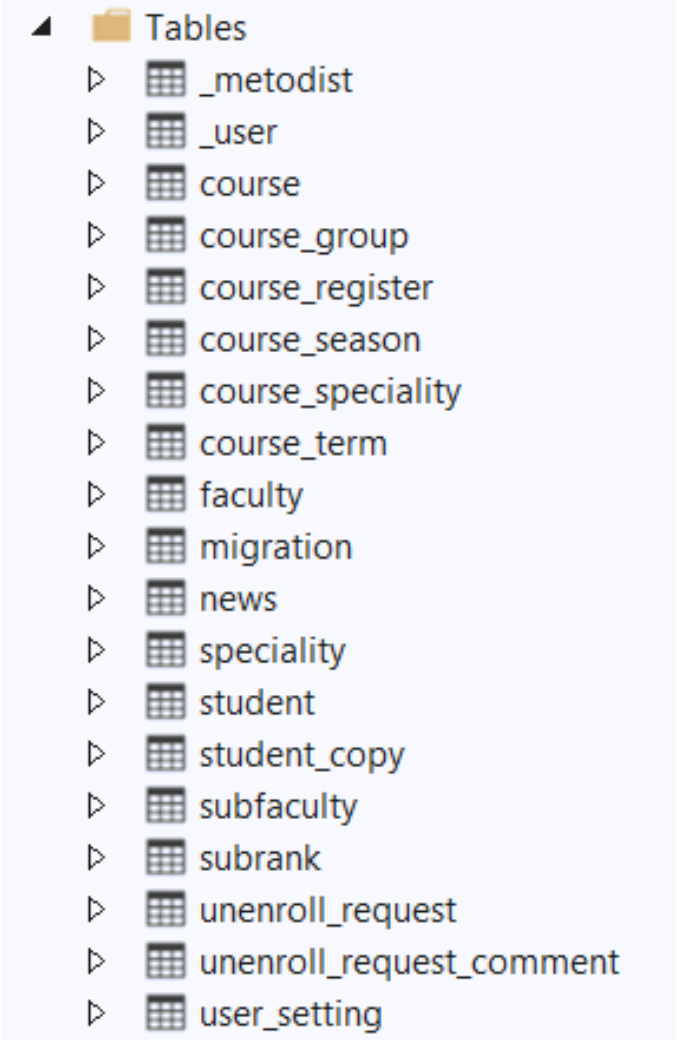

Рисунок 5 - База даних САЗ

На рис. 5 показано всі таблиці з БД. Загалом кажучи, структура бази даних поділена на таку інформацію:

- користувачі системи – в цю групу належить будь-яка особа, що буде користуватись САЗ– методист, студент, адміністратор або викладач;
- студенти – в таблицях, що відносяться до студентів, зберігається інформація по записах на дисциплінах та групах, роки та вся інформація, що належить до студента;

- факультет та спеціальність – детальна інформація для спеціальностей, а саме належність до факультету, список предметів спеціальності, години та кредити для предмету тощо;
- Новини – всі новини САЗ.

Розглянемо таблицю для кожної з груп.

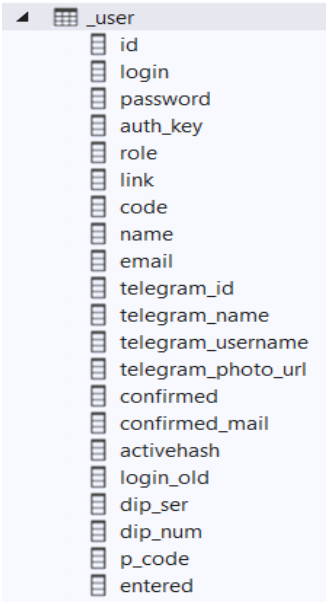

Рисунок 6 - Таблиця користувача системи

Дивлячись на колонки таблиці користувача, видно, що тут зберігаються дані входу, та загальна контактна інформація.

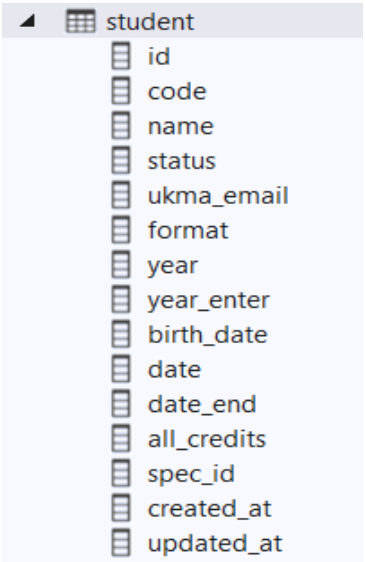

Рисунок 7 - Таблиця для студента

На рис. 7 продемонстровано, що для студента зберігається відповідна інформація, необхідна для запису на дисципліни – кредити, рік навчання тощо.

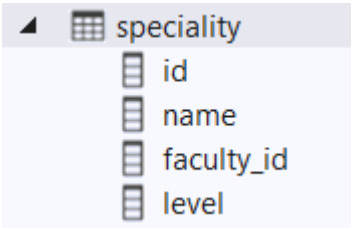

Рисунок 8 - Таблиця спеціальності

Рис. 8 демонструє відповідну інформацію спеціальності. Важливо зазначити, що кожна спеціальність належить якомусь факультету, саме тому в таблиці є відповідна колонка faculty_id.

Рисунок 9 - Таблиця новин

Видно, що таблиця для новин має досить просту структуру, адже тут важливе лише дата та вміст, як показано на рис. 9.

3.2. Опис розробки програми

На рис. 10 можна розглянути схему логіки проекту.

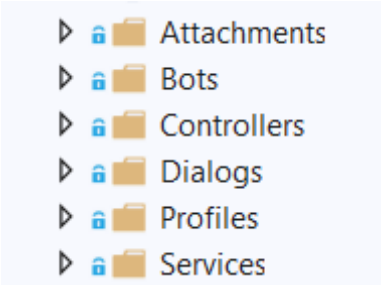

Рисунок 10 - Схема проєкту

В Attachments розташовано підпапки, які поділені відповідно для доступу до файлів з розкладом, інструкціями та FAQ.

Bots – це папка, що створюється автоматично, в ній розташований клас, який розпочинає роботу бота та вітається з користувачем.

Controllers включає відповідний файл-контролер для бота.

Dialogs – в цій папці прописана логіка для кожного з можливих сценаріїв. Саме тут можна переглянути як реалізовано подача інформації. Детальні сценарії відображено на рис. 11.

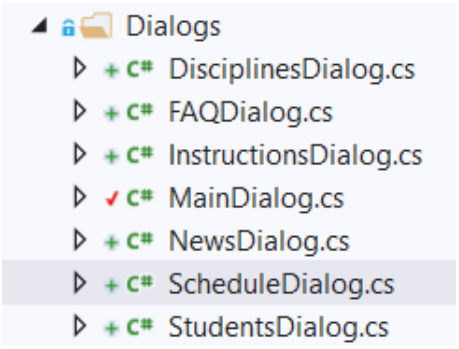

Рисунок 11 - Сценарії роботи бота

На рис. 12 продемонстровано приклад сценарію вибираючи діалог “FAQ”. Як видно бот працює за логікою почергового виклику методів, які в залежності від дій користувача будь надавати різні відповіді.

```
AddDialog(new WaterfallDialog(nameof(WaterfallDialog),
    actions: new WaterfallStep[]
    {
        DisplayAvailableFaQs,
        ShowAnswer,
        ReturnDetails
    }));
```

Рисунок 12 - Водоспад методів для “FAQ”.

Метод “DasplayAvailableFaQ” надає список найбільш поширених запитань студентів. Після дії користувача викликається метод “ShowAnswer”, який надає інструкцію. Деякі відповіді на питання можуть бути надіслані у вигляді окремого файлу. В такому випадку викликається метод “ReturnDetails”.

В папці Profiles – відповідні класи-аналоги до таблиць бази даних. На рис. X видно класи-аналоги, що використовувались у цій роботі.

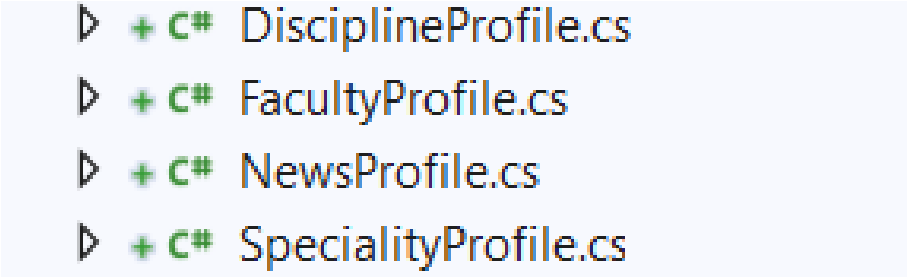

Рисунок 13 – Класи–аналоги таблицям БД

Наприклад, в класі SpecialityProfile можна відслідкувати відповідні атрибути з таблиці Speciality, як на рис. 14.

```
public class SpecialityProfile
{
    ссылка: 1 | 0 исключения
    public int Id { get; set; }
    ссылка: 1 | 0 исключения
    public  string Name { get; set; }
    ссылка: 1 | 0 исключения
    public int FacultyId { get; set; }
    ссылка: 1 | 0 исключения
    public  int Level { get; set; }

    ссылка: 1 | 0 исключения
    public SpecialityProfile(int id, string name, int facultyId, int level)
    {
        Id = id;
        Name = name;
        FacultyId = facultyId;
        Level = level;
    }
}
```

Рисунок 14 - Аналог таблиці Speciality

В Models зібрано всі моделі. Саме в цій папці знаходиться класи-сутності спроектовані для співпраці з базою даних.

Services – допоміжний рівень, тут створено підключення до бази даних, та реалізація всіх запитів.

3.3. Опис файлів даних та інтерфейсу програми

Перш ніж почати роботу з ботом, він вітається з тобою. Відразу після того, як він отримав якусь відповідь – надається меню для вибору подальших дій, як показано на рис. 15.

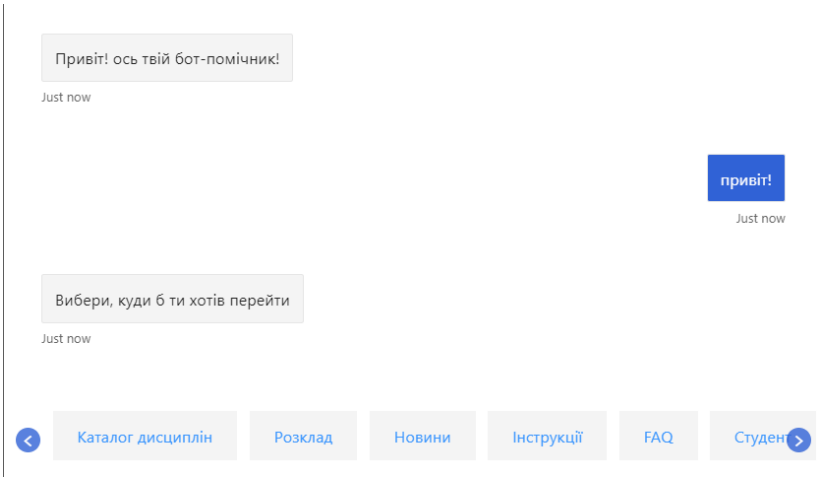

Рисунок 15 - Основне меню бота

На цьому етапі користувач має можливість перейти в кожну з гілок для отримання відповідної інформації.

Якщо користувач натиснув на опцію “Каталог дисциплін” – наступним кроком буде ввести будь яку частинку дисципліни, деталі якої він хотів б переглянути. Це продемонстровано на рис. 16.

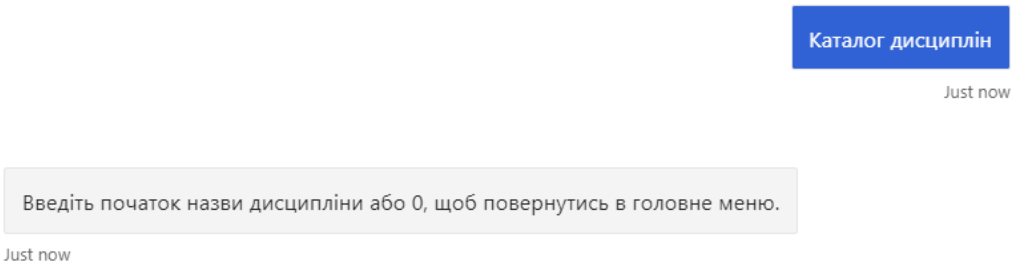

Рисунок 16 - Вибір опції “Каталог дисциплін”

Відразу після того, як будь яка частинка дисципліни була введена, бот відповість з цілий списком дисциплін, що відповідають критерію пошуку. Це показано на рис. 17.

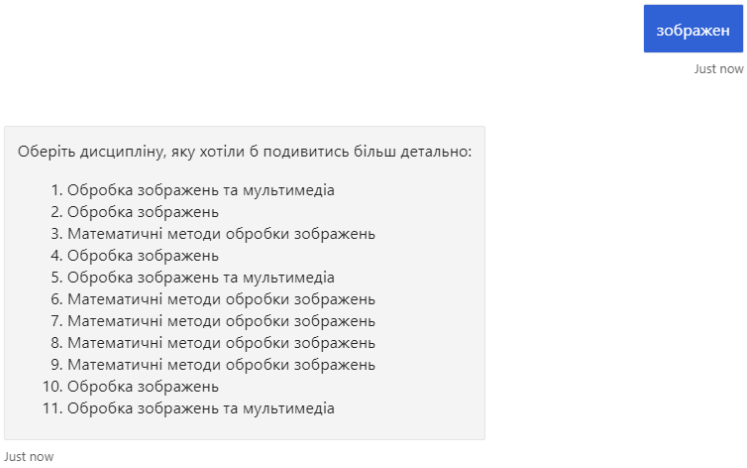

Рисунок 17 - Виведення списку доступних дисциплін.

На рис. 18 показано наступний крок - перегляд анотації та детальної інформації за бажанням.

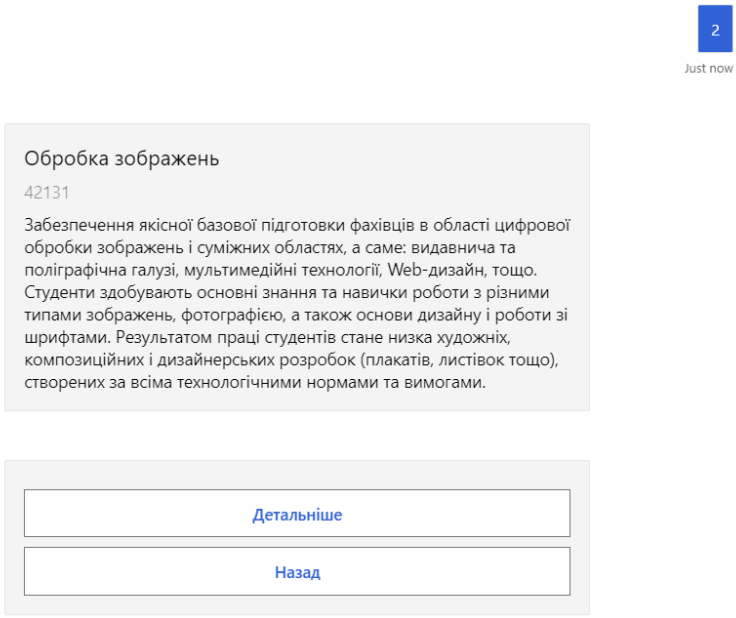

Рисунок 18 - Перегляд анотації

Тут є можливість повернутись в меню пошуку, або дізнатись більше доступних даних за дисципліною.

Натиснувши на “Детальніше”, буде виведено інформація по кредитах та годинах, як на рис. 19.

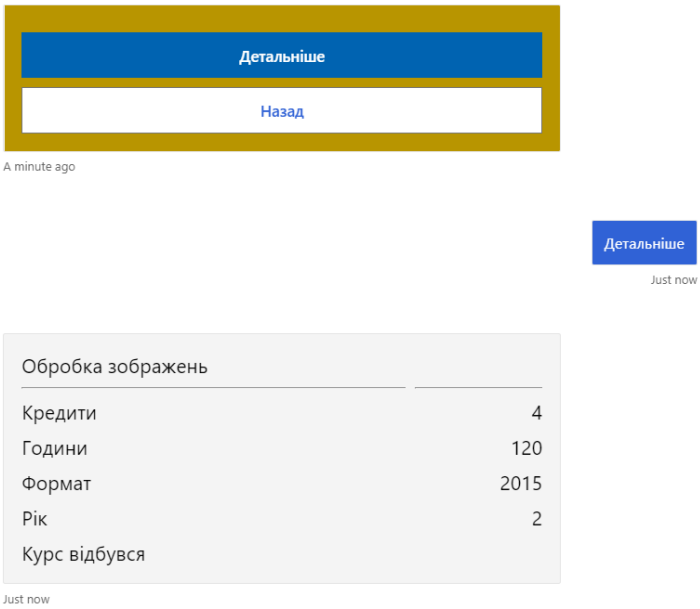

Рисунок 19 - Детальна інформація для дисципліни

Наступна опція меню – це “Розклад”. Обравши факультет -> спеціальність -> рік, бот зможе надіслати відповідний файл, з необхідним розкладом як показано на рис. 20.

Рисунок 20 - Виведення розкладу

Бот має доступ до файлів розкладу всіх років та спеціальностей та може надіслати його у разі необхідності.

Як тільки обрано опцію “Новини”, помічник відправляє список всіх актуальних новин. Їх всі можна прочитати гортаючи вгору або вниз, як на рис. 21.

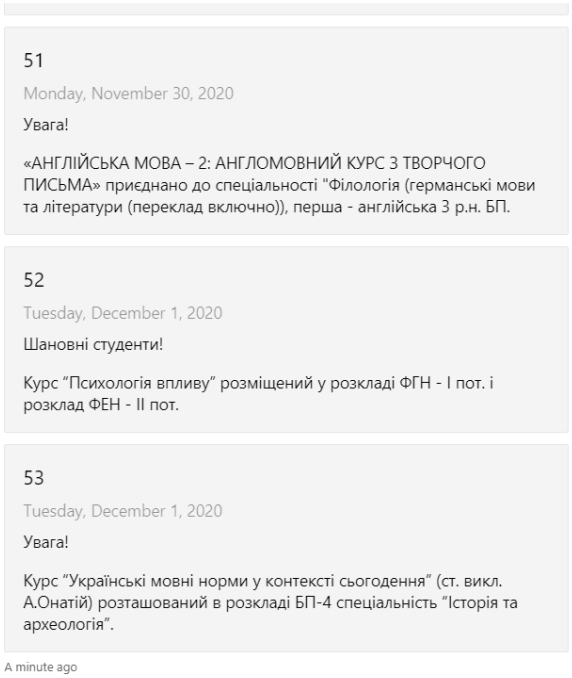

Рисунок 21 - Стрічка новин

В інструкціях, користувач має змогу переглянути інструкцію, щодо входу у систему або користування нею. На рис. 22 це продемонстровано.

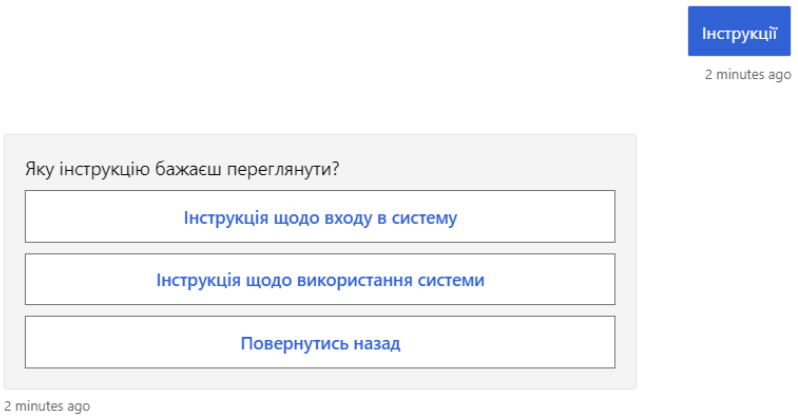

Рисунок 22 - Можливі інструкції

Як тільки, будь-яку з опцій обрано, відповідний .pdf файл буде надіслано, як на рис. 23.

Рисунок 23 - Надісланий файл з інструкцією

Меню “FAQ” надає доступ до відповідей на найчастіші запитання:

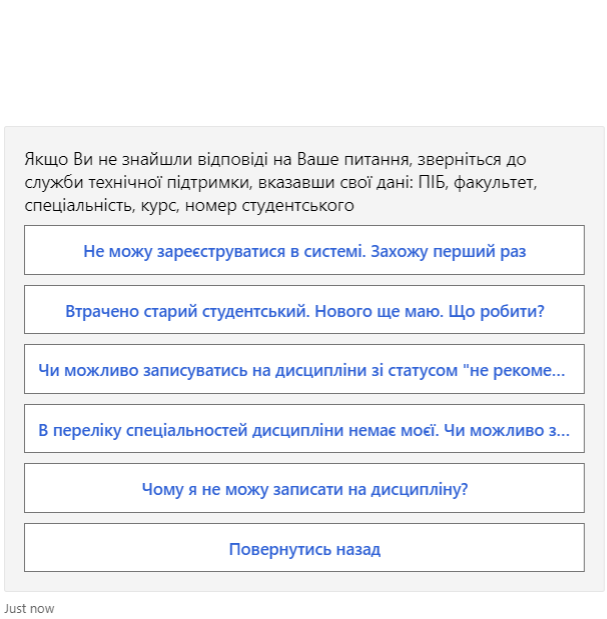

Рисунок 24 - Перелік можливих відповідей

Якщо користувач обирає якусь з опцій, бот надсилає відповідні інструкції, як на рис 25.

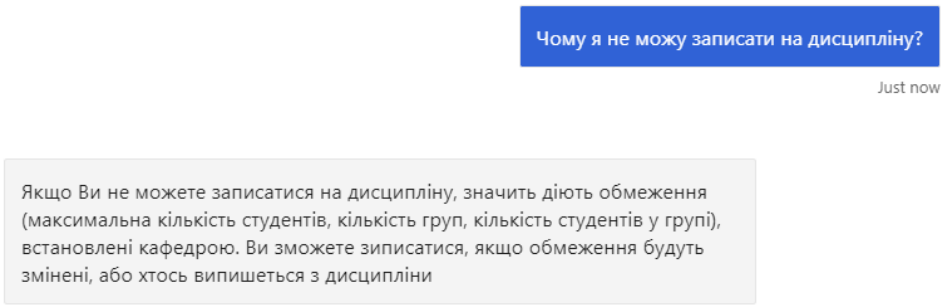

Рисунок 25 - Питання-відповідь в меню FAQ

В кожну з гілок меню можна заходити по декілька разів та повторно просити будь-яку з необхідних інформацій. А також, скрізь є опція повернутись назад, у випадку не запланованого кліку на пункт меню.

3.4. Тестування програми і результати її виконання

В ході тестування програми основну увагу було звернено на зручність користування для людини, що вперше користується системою автоматизовано запису з допомогою бота.

Для цього, я провела аналіз сайту САЗ та виділила основні блоки з якими можна працювати, а саме такі, що показано на рис. 26. Тому, бот може надавати розклад відповідної спеціальності, надавати інструкції та відповідати на часті запитання.

Рисунок 26 - Інформація з системи автоматизованого запису

Також, важливою гілкою є “Дисципліни”. Під час розробки боту, було докладено чимало зусиль, щоб інформація по дисципліні була компактна та зрозуміла.

Для того, щоб зробити інтерфейс зручним, було проаналізовано боти на різні теми, та виявлено, що надсилання вибору для подальшого розвитку подій є найбільш оптимальним рішенням, адже натиснути на кнопку займає набагато менше часу, аніж ввести дані вручну.

Під час тестування, зверталось увагу на те, щоб з будь якої гілки діалогу можна було легко повернутись в основне меню. Часто, у різних ботів немає такої опції.

У фінальному тестуванні відбувалась імітація різних кроків користувача для того, щоб уникнути будь яких помилок. На щастя, детальне тестування допомогло знайти ще декілька сценаріїв, які не були продумані, саме тому бот повертав помилку як на рис. 27.

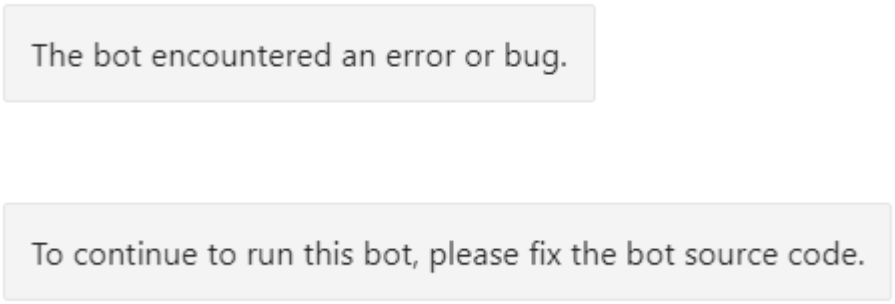

Рисунок 27 - Демонстрація несправності програми

Тим не менш, великий відсоток часу було відведено саме на тестування та відслідковування різних режимів роботи бота.

Висновки

В результаті дослідження та розробки бота було сформовано такі висновки:

1) Створення боту варто розпочинати з плану та продумування всіх можливих опцій.
2) Бот включає розмежування доступу, відповідно з яким ділиться можливість користування функціоналом.
3) Бота можна покращити таким чином:

- допрацювати функціонал для студента (авторизованого користувача);
- розширений пошук за дисциплінами;
- продумати запис на дисципліни та корегування індивідуального плану.

4) Застосування було створено використовуючи мову C#, в середовищі Visual Studio 2019. Використано технології Bot Framework SDK з підключенням до MySQL Server в якості СУБД.

Список джерел

1. Office 365: https://uk.wikipedia.org/wiki/Microsoft_365
2. Java: https://lgs.lviv.ua/shho-potribno-znaty-pered-vyvchennyam-java/
3. Java: https://uk.wikibooks.org/wiki/%D0%9E%D1%81%D0%B2%D0%BE%D1%8E%D1%94%D0%BC%D0%BE_Java
4. Python: http://www.plug.org.ua/documentation/about-python
5. C#: https://www.facebook.com/lgs.lviv.ua/posts/1805342679746716/
6. C# (2): https://uk.wikipedia.org/wiki/C_Sharp
7. Бот: https://techtoday.in.ua/news/shho-take-boti-v-servisi-telegram-ta-yak-nimi-koristuvatisya-64222.html
8. База даних: https://hostiq.ua/wiki/ukr/database/
9. Матеріали з лекцій курсу “Реляційні бази даних” – основні теоретичні відомості: https://distedu.ukma.edu.ua/course/view.php?id=50
10. MySQL: http://ruszura.in.ua/uncategorized/scho-take-mysql-yak-i-de-vykorystovuyut-mysql.html

Додаток А. Список прийнятих скорочень

- ОС – операційна система;
- СКБД - Система управління базами даних;
- СУБД – Система керування базами даних;
- САЗ – система автоматизованого запису.

Додаток Б. Код програми

```
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Schema;
using Microsoft.Extensions.Logging;

namespace ComplexDialogBot.Bots
{
    public class DialogAndWelcomeBot<T> : DialogBot<T> where T : Dialog
    {
        public DialogAndWelcomeBot(ConversationState conversationState, UserState userState,
T dialog, ILogger<DialogBot<T>> logger)
            : base(conversationState, userState, dialog, logger)
        {
        }

        protected override async Task OnMembersAddedAsync(
            IList<ChannelAccount> membersAdded,
            ITurnContext<IConversationUpdateActivity> turnContext,
            CancellationToken cancellationToken)
        {
            foreach (var member in membersAdded)
            {
                if (member.Id != turnContext.Activity.Recipient.Id)
                {
                    var reply = MessageFactory.Text($"Привіт! ось твій бот-помічник для
САЗу! Привітайся, щоб розпочати роботу зі мною");
                    await turnContext.SendActivityAsync(reply, cancellationToken);
                }
            }
        }
    }
}
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Schema;
using Microsoft.Extensions.Logging;

namespace ComplexDialogBot.Bots
{
   public class DialogBot<T> : ActivityHandler where T : Dialog
    {
        protected readonly BotState ConversationState;
        protected readonly Dialog Dialog;
        protected readonly ILogger Logger;
        protected readonly BotState UserState;

        public DialogBot(ConversationState conversationState, UserState userState, T dialog,
ILogger<DialogBot<T>> logger)
        {
            ConversationState = conversationState;
            UserState = userState;
            Dialog = dialog;
            Logger = logger;
        }

        public override async Task OnTurnAsync(ITurnContext turnContext, CancellationToken
cancellationToken = default(CancellationToken))
        {
            await base.OnTurnAsync(turnContext, cancellationToken);
```

```
            await ConversationState.SaveChangesAsync(turnContext, false, cancellationToken);
            await UserState.SaveChangesAsync(turnContext, false, cancellationToken);
        }

        protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity>
turnContext, CancellationToken cancellationToken)
        {
            Logger.LogInformation("Running dialog with Message Activity.");
            await Dialog.RunAsync(turnContext,
ConversationState.CreateProperty<DialogState>(nameof(DialogState)), cancellationToken);
        }
    }
}
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using ComplexDialogBot.Profiles;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Builder.Dialogs.Choices;
using Microsoft.Bot.Schema;
using SAZBot.Database;

namespace ComplexDialogBot.Dialogs
{
    public class DisciplinesDialog : ComponentDialog
    {
        private const string DoneOption = "done";

        private const string DisciplineInfo = "value-disciplineInfo";
        private List<DisciplineProfile> _disciplineProfileList;
        private DisciplineProfile _selectedDiscipline;

        public DisciplinesDialog()
            : base(nameof(DisciplinesDialog))
        {
            AddDialog(new TextPrompt(nameof(TextPrompt)));
            AddDialog(new NumberPrompt<int>(nameof(NumberPrompt<int>)));

            AddDialog(new WaterfallDialog(nameof(WaterfallDialog),
                new WaterfallStep[]
                {
                    SelectionStepAsync,
                    ReturnAvailableDisciplines,
                    ShowDetailedDisciplineInfo,
                    DisplayReceiptCard,
                    LoopStepAsync,
                }));

            InitialDialogId = nameof(WaterfallDialog);
        }

        private async Task<DialogTurnResult> SelectionStepAsync(
            WaterfallStepContext stepContext,
            CancellationToken cancellationToken)
        {
            stepContext.Values[DisciplineInfo] = new DisciplineProfile();
            var promptOptions = new PromptOptions {Prompt = MessageFactory.Text("Введіть
початок назви дисципліни або 0, щоб повернутись в головне меню.")};

            return await stepContext.PromptAsync(nameof(TextPrompt), promptOptions,
cancellationToken);
        }
```

```
        private async Task<DialogTurnResult> ReturnAvailableDisciplines(WaterfallStepContext
stepContext, CancellationToken cancellationToken)
        {

            var disciplineProfile = (DisciplineProfile)stepContext.Values[DisciplineInfo];
            disciplineProfile.Name = (string)stepContext.Result;

            if (disciplineProfile.Name.Equals("0"))
            {
                await stepContext.EndDialogAsync(null, cancellationToken);
                return await stepContext.BeginDialogAsync(nameof(MainDialog), null,
cancellationToken);

            }

            var attachments = new List<Attachment>();
            var worker = new DBWorker();
            _disciplineProfileList = worker.GetDisciplineList(disciplineProfile.Name);

            var list = new List<string>();
            foreach (var disciplineItem in _disciplineProfileList)

                list.Add(disciplineItem.Name);

            if (list.Count == 0)
            {
                await stepContext.Context.SendActivityAsync(
                    MessageFactory.Text("Немає дисциплін, що відповідають критеріям
пошуку."));
                await stepContext.EndDialogAsync(null, cancellationToken);
                return await stepContext.BeginDialogAsync(nameof(DisciplinesDialog), null,
cancellationToken);
            }

            var options = new PromptOptions()
            {
                Prompt = MessageFactory.Text("Оберіть дисципліну, яку хотіли б подивитись
більш детально:"),
                Choices = ChoiceFactory.ToChoices(list),
            };

            return await stepContext.PromptAsync(nameof(ChoicePrompt), options,
cancellationToken);
        }

        private async Task<DialogTurnResult> ShowDetailedDisciplineInfo(WaterfallStepContext
stepContext,
            CancellationToken cancellationToken)
        {

            var choice = (FoundChoice)stepContext.Result;

            var attachments = new List<Attachment>();
            var reply = MessageFactory.Attachment(attachments);

            _selectedDiscipline = _disciplineProfileList.Find(d =>
d.Name.Equals(choice.Value));

reply.Attachments.Add(Cards.GetThumbnailCard(_selectedDiscipline).ToAttachment());


            await stepContext.Context.SendActivityAsync(reply, cancellationToken);


            var options = new PromptOptions()
```

```
            {
                Style = ListStyle.HeroCard,
               Choices = new List<Choice>()
               {
                   new Choice("Детальніше"),
                   new Choice("Назад")
               }
            };

            return await stepContext.PromptAsync(nameof(ChoicePrompt), options,
cancellationToken);
        }

        public async Task<DialogTurnResult> DisplayReceiptCard(WaterfallStepContext
stepContext,
            CancellationToken cancellationToken)
        {

            var choice = (FoundChoice)stepContext.Result;
            if (choice.Value.Equals("Детальніше"))
            {

                var attachments = new List<Attachment>();
                var reply = MessageFactory.Attachment(attachments);


reply.Attachments.Add(Cards.GetReceiptCard(_selectedDiscipline).ToAttachment());


                await stepContext.Context.SendActivityAsync(reply, cancellationToken);
                return await stepContext.EndDialogAsync(null, cancellationToken);
            }

            if (choice.Value.Equals("Назад"))
            {
                await stepContext.EndDialogAsync(null, cancellationToken);
                return await stepContext.BeginDialogAsync(nameof(DisciplinesDialog), null,
cancellationToken);

            }
            return await stepContext.EndDialogAsync(null, cancellationToken);
        }

        private async Task<DialogTurnResult> LoopStepAsync(
            WaterfallStepContext stepContext,
            CancellationToken cancellationToken)
        {
            var disciplineProfile = (DisciplineProfile)stepContext.Values[DisciplineInfo];
            disciplineProfile.Name = (string)stepContext.Result;

            if (disciplineProfile.Name == "0")
            {
                return await stepContext.EndDialogAsync(null, cancellationToken);
            }

            await stepContext.Context.SendActivityAsync(
                MessageFactory.Text(
                    $"Thanks for participating,
{((DisciplineProfile)stepContext.Values[DisciplineInfo]).Name}."),
                cancellationToken);
            return await stepContext.ReplaceDialogAsync(nameof(DisciplinesDialog), null,
cancellationToken);

        }
    }
```

```
}
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Builder.Dialogs.Choices;
using Microsoft.Bot.Schema;

namespace ComplexDialogBot.Dialogs
{
    public class FaqDialog : ComponentDialog
    {

        private readonly string Path = "D:/";
        public FaqDialog()
            : base(nameof(FaqDialog))
        {

            AddDialog(new TextPrompt(nameof(TextPrompt)));
            AddDialog(new NumberPrompt<int>(nameof(NumberPrompt<int>)));
            AddDialog(new WaterfallDialog(nameof(WaterfallDialog),
                new WaterfallStep[]
                {
                    DisplayAvailableFaQs,
                    ShowAnswer,
                    ReturnDetails
                }));

            InitialDialogId = nameof(WaterfallDialog);
        }

        private async Task<DialogTurnResult> DisplayAvailableFaQs(WaterfallStepContext
stepContext,
            CancellationToken cancellationToken)
        {

            var options = new PromptOptions()
            {
                Prompt = MessageFactory.Text("Якщо Ви не знайшли відповіді на Ваше питання,
зверніться до служби технічної підтримки, вказавши свої дані: ПІБ, факультет, спеціальність,
курс, номер студентського"),
                RetryPrompt =
                    MessageFactory.Text("That was not a valid choice, please select a card
or number from 1 to 9."),
                Style = ListStyle.HeroCard,
                Choices = GetChoices()
            };
            return await stepContext.PromptAsync(nameof(ChoicePrompt), options,
cancellationToken);

        }

        private IList<Choice> GetChoices()
        {
            var cardOptions = new List<Choice>()
            {
                new Choice() {Value = "Не можу зареєструватися в системі. Захожу перший
раз"},
                new Choice() {Value = "Втрачено старий студентський. Нового ще маю. Що
робити?"},
                new Choice() {Value = "Чи можливо записуватись на дисципліни зі статусом
\"не рекомендовано\"?"},
                new Choice() {Value = "В переліку спеціальностей дисципліни немає моєї. Чи
можливо записатись на дисципліну?"},
```

```
                new Choice() {Value = "Чому я не можу записати на дисципліну?"},
                new Choice() {Value = "Повернутись назад"},

            };
            return cardOptions;
        }

        private async Task<DialogTurnResult> ShowAnswer(WaterfallStepContext stepContext,
            CancellationToken cancellationToken)
        {
            //TODO: move attachments to the correct folder.
            var attachments = new List<Attachment>();
            var reply = MessageFactory.Attachment(attachments);

            switch (((FoundChoice)stepContext.Result).Value)
            {
                case "Не можу зареєструватися в системі. Захожу перший раз"
                    :
                    await stepContext.Context.SendActivityAsync(
                        MessageFactory.Text(
                            "Ваш тимчасовий логін - номер студентського квитка, тимчасовий
пароль - дата народження у форматі Рік/місяць/число. "));
                    var options = new PromptOptions()
                    {
                        Style = ListStyle.HeroCard,
                        Choices = new List<Choice>()
                        {
                            new Choice("Детальніше"),
                            new Choice("Назад"),
                        }
                    };

                    return await stepContext.PromptAsync(nameof(ChoicePrompt), options,
cancellationToken);

                case "Втрачено старий студентський. Нового ще маю. Що робити?":
                    await stepContext.Context.SendActivityAsync(
                        MessageFactory.Text(
                            "Спробуйте зайти до системи, використовуючи номер старого
студентського. Якщо не виходить, напишіть до служби підтримки, вказавши свої дані."),
cancellationToken);
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
                    return await stepContext.BeginDialogAsync(nameof(FaqDialog), null,
cancellationToken);
                case "Чи можливо записуватись на дисципліни не свого курсу?":
                    await stepContext.Context.SendActivityAsync(
                        MessageFactory.Text(
                            "Якщо дисципліна старшого курсу, переконайтеся, що Ви маєте
базис, необхідний для опанування дисципліни. В разі необхідності, проконсультуйтеся на
кафедрі, яка забезпечує викладання дисципліни."), cancellationToken);
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
                    return await stepContext.BeginDialogAsync(nameof(FaqDialog), null,
cancellationToken);
                case "Чи можливо записуватись на дисципліни зі статусом \"не
рекомендовано\"?":
                    await stepContext.Context.SendActivityAsync(
                        MessageFactory.Text(
                            "Ні, ці дисципліни не відбуваються. Відповідний статус
присвоюється за поданням кафедри у зв'язку із неможливістю проведення дисципліни у
відповідному навчальному році."), cancellationToken);
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
                    return await stepContext.BeginDialogAsync(nameof(FaqDialog), null,
cancellationToken);
                case "В переліку спеціальностей дисципліни немає моєї. Чи можливо записатись
на дисципліну?":
```

```
                    await stepContext.Context.SendActivityAsync(
                        MessageFactory.Text(
                            "В такому разі дисципліна буде для Вас вільного вибору"),
cancellationToken);
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
                    return await stepContext.BeginDialogAsync(nameof(FaqDialog), null,
cancellationToken);

                case "Чому я не можу записати на дисципліну?":
                    await stepContext.Context.SendActivityAsync(
                        MessageFactory.Text(
                            "Якщо Ви не можете записатися на дисципліну, значить діють
обмеження (максимальна кількість студентів, кількість груп, кількість студентів у групі),
встановлені кафедрою. Ви зможете зиписатися, якщо обмеження будуть змінені, або хтось
випишеться з дисципліни"), cancellationToken);

                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
                    return await stepContext.BeginDialogAsync(nameof(FaqDialog), null,
cancellationToken);

                case "Повернутись назад":
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
                    return await stepContext.BeginDialogAsync(nameof(MainDialog), null,
cancellationToken);

            }

            return await stepContext.BeginDialogAsync(nameof(MainDialog), null,
cancellationToken);
        }

        private async Task<DialogTurnResult> ReturnDetails(WaterfallStepContext stepContext,
            CancellationToken cancellationToken)
        {
            //TODO: move attachments to the correct folder.
            var attachments = new List<Attachment>();
            var reply = MessageFactory.Attachment(attachments);

            switch (((FoundChoice)stepContext.Result).Value)
            {
                case "Детальніше"
                    :
                    var attachment = new Attachment
                    {
                        ContentUrl = Path+ "registration-instruction.pdf",
                        ContentType = "application/pdf",
                        Name = "registration-instruction.pdf",
                    };

                    reply.Attachments = new List<Attachment>() {attachment};
                    await stepContext.Context.SendActivityAsync(reply, cancellationToken);
                    break;
                case "Назад"
                    :
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
                    return await stepContext.BeginDialogAsync(nameof(FaqDialog), null,
cancellationToken);
            }
            return await stepContext.EndDialogAsync(cancellationToken: cancellationToken);

        }
    }
}
using System.Collections.Generic;
using System.Threading;
```

```
using System.Threading.Tasks;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Builder.Dialogs.Choices;
using Microsoft.Bot.Schema;

namespace ComplexDialogBot.Dialogs
{
    public class InstructionsDialog : ComponentDialog
    {
        //TODO: make as secret
        private readonly string Path = "D:/";

        public InstructionsDialog()
            : base(nameof(InstructionsDialog))
        {
            AddDialog(new TextPrompt(nameof(TextPrompt)));
            AddDialog(new NumberPrompt<int>(nameof(NumberPrompt<int>)));
            AddDialog(new WaterfallDialog(nameof(WaterfallDialog),
                new WaterfallStep[]
                {
                    DisplayAvailableInstructions,
                    ShowAttachment,
                }));

            InitialDialogId = nameof(WaterfallDialog);
        }

        private async Task<DialogTurnResult>
DisplayAvailableInstructions(WaterfallStepContext stepContext,
            CancellationToken cancellationToken)
        {

            var options = new PromptOptions()
            {
                Prompt = MessageFactory.Text("Яку інструкцію бажаєш переглянути?"),
                RetryPrompt =
                    MessageFactory.Text("That was not a valid choice, please select a card
or number from 1 to 9."),
                Style = ListStyle.HeroCard,
                Choices = GetChoices()
            };
            return await stepContext.PromptAsync(nameof(ChoicePrompt), options,
cancellationToken);

        }


        private IList<Choice> GetChoices()
        {
            var cardOptions = new List<Choice>()
            {
                new Choice() {Value = "Інструкція щодо входу в систему"},
                new Choice() {Value = "Інструкція щодо використання системи"},
                new Choice() {Value = "Повернутись назад"},
            };
            return cardOptions;
        }

        private async Task<DialogTurnResult> ShowAttachment(WaterfallStepContext
stepContext,
            CancellationToken cancellationToken)
        {
            //TODO: move attachments to the correct folder.
            var attachments = new List<Attachment>();
```

```
            var reply = MessageFactory.Attachment(attachments);

            switch (((FoundChoice)stepContext.Result).Value)
            {
                case "Інструкція щодо входу в систему"
                    : //TODO: move attahcments to the correct folder, modify this method to
be beautifull
                    var attachment = new Attachment
                    {
                        ContentUrl = Path+ "o365authorization.pdf",
                        ContentType = "application/pdf",
                        Name = "o365authorization.pdf",
                    };

                    reply.Attachments = new List<Attachment>() {attachment};
                    await stepContext.Context.SendActivityAsync(reply, cancellationToken);

                    break;
                case "Інструкція щодо використання системи":
                    var attachment1 = new Attachment
                    {
                        ContentUrl = Path + "instructions_student.pdf",
                        ContentType = "application/pdf",
                        Name = "instructions_student.pdf",
                    };
                    reply.Attachments = new List<Attachment>() {attachment1};
                    await stepContext.Context.SendActivityAsync(reply, cancellationToken);
                    break;
                case "Повернутись назад":
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
                    return await stepContext.BeginDialogAsync(nameof(MainDialog), null,
cancellationToken);

            }
            await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
            return await stepContext.BeginDialogAsync(nameof(MainDialog), null,
cancellationToken);
        }


    }

}
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using ComplexDialogBot.Profiles;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Builder.Dialogs.Choices;
using Microsoft.BotBuilderSamples;

namespace ComplexDialogBot.Dialogs
{
    public class MainDialog : ComponentDialog
    {
        public readonly UserState _userState;

        public MainDialog(UserState userState)
            : base(nameof(MainDialog))
        {
            _userState = userState;
            AddDialog(new NewsDialog());
            AddDialog(new ScheduleDialog());
            AddDialog(new DisciplinesDialog());
```

```
            AddDialog(new InstructionsDialog());
            AddDialog(new FaqDialog());
            AddDialog(new StudentsDialog());

            AddDialog(new ChoicePrompt(nameof(ChoicePrompt)));
            AddDialog(new WaterfallDialog(nameof(WaterfallDialog), new WaterfallStep[]
            {
                MainChoicesAsync,
                ReturnResultDialog,
            }));

            InitialDialogId = nameof(WaterfallDialog);
        }

        private async Task<DialogTurnResult> MainChoicesAsync(WaterfallStepContext
stepContext, CancellationToken cancellationToken)
        {
            var options = new PromptOptions()
            {
                Prompt = MessageFactory.Text("Вибери, куди б ти хотів перейти"),
                RetryPrompt = MessageFactory.Text("Вибери, куди б ти хотів перейти"),
                Choices = GetChoices(),
            };

            return await stepContext.PromptAsync(nameof(ChoicePrompt), options,
cancellationToken);
        }

        private IList<Choice> GetChoices()
        {
            var cardOptions = new List<Choice>()
            {
                new Choice() { Value = "Каталог дисциплін", Synonyms = new List<string>() {
"disciplines" } },
                new Choice() { Value = "Розклад", Synonyms = new List<string>() { "schedule"
} },
                new Choice() { Value = "Новини", Synonyms = new List<string>() { "news" } },
                new Choice() { Value = "Інструкції", Synonyms = new List<string>() {
"instructions" } },
                new Choice() { Value = "FAQ", Synonyms = new List<string>() { "faq" } },
                new Choice() { Value = "Студент", Synonyms = new List<string>() { "student"
} },

            };
            return cardOptions;
        }
        private async Task<DialogTurnResult> ReturnResultDialog(WaterfallStepContext
stepContext,
            CancellationToken cancellationToken)
        {

            switch (((FoundChoice)stepContext.Result).Value)
            {
                case "Каталог дисциплін":
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);

                    return await stepContext.BeginDialogAsync(nameof(DisciplinesDialog),
null, cancellationToken);
                case "Розклад":
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);

                    return await stepContext.BeginDialogAsync(nameof(ScheduleDialog), null,
cancellationToken);
                case "Новини":
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
```

```
                    return await stepContext.BeginDialogAsync(nameof(NewsDialog), null,
cancellationToken);
                case "Інструкції":
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);

                    return await stepContext.BeginDialogAsync(nameof(InstructionsDialog),
null, cancellationToken);
                case "FAQ":
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);

                    return await stepContext.BeginDialogAsync(nameof(FaqDialog), null,
cancellationToken);
                case "Студент":
                    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);

                    return await stepContext.BeginDialogAsync(nameof(StudentsDialog), null,
cancellationToken);
                default:
                    return await stepContext.EndDialogAsync(cancellationToken:
cancellationToken);


            }
        }

    }
}
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using ComplexDialogBot.Profiles;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Schema;
using SAZBot.Database;

namespace ComplexDialogBot.Dialogs
{
    public class NewsDialog : ComponentDialog
    {
        public NewsDialog()
            : base(nameof(NewsDialog))
        {
            AddDialog(new TextPrompt(nameof(TextPrompt)));
            AddDialog(new NumberPrompt<int>(nameof(NumberPrompt<int>)));
            AddDialog(new WaterfallDialog(nameof(WaterfallDialog), new WaterfallStep[]
            {
                ReturnAllNewsAsync,
                TypeAnythingToOpenMenu
            }));

            InitialDialogId = nameof(WaterfallDialog);
        }

        private async Task<DialogTurnResult> TypeAnythingToOpenMenu(WaterfallStepContext
stepContext, CancellationToken cancellationToken)
        {
            return await stepContext.BeginDialogAsync(nameof(MainDialog), null,
cancellationToken);

        }
        private async Task<DialogTurnResult> ReturnAllNewsAsync(WaterfallStepContext
stepContext, CancellationToken cancellationToken)
        {
```

```
            var attachments = new List<Attachment>();
            DBWorker worker = new DBWorker();
            List<NewsProfile> news = worker.ShowAllnews();

            var reply = MessageFactory.Attachment(attachments);
            foreach (var newsItem in news)
            {

                reply.Attachments.Add(Cards.GetThumbnailCard(newsItem).ToAttachment());
            }
            await stepContext.Context.SendActivityAsync(reply, cancellationToken);

            return await stepContext.EndDialogAsync(reply, cancellationToken);
        }

       }
}
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;

namespace ComplexDialogBot.Dialogs
{
    public class StudentsDialog : ComponentDialog
    {
        public StudentsDialog()
            : base(nameof(StudentsDialog))
        {

            AddDialog(new TextPrompt(nameof(TextPrompt)));
            AddDialog(new NumberPrompt<int>(nameof(NumberPrompt<int>)));
            AddDialog(new WaterfallDialog(nameof(WaterfallDialog),
                new WaterfallStep[] {UpdateUser}));

            InitialDialogId = nameof(WaterfallDialog);
        }

        private async Task<DialogTurnResult> UpdateUser(WaterfallStepContext stepContext,
            CancellationToken cancellationToken)
        {
            await stepContext.Context.SendActivityAsync(
                MessageFactory.Text("Ця гілка ще в розробці"), cancellationToken);
            await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
            return await stepContext.BeginDialogAsync(nameof(MainDialog), null,
cancellationToken);

        }
    }
}
namespace ComplexDialogBot.Profiles
{
    public class DisciplineProfile
    {
        public string Id { get; set; }
        public string Sub_cdoc { get; set; }
        public string Name { get; set; }
        public string Type { get; set; }
        public int Academic_Year { get; set; }
        public int Format { get; set; }
        public int Year { get; set; }
        public int Hours { get; set; }
        public int Credits { get; set; }
        public string Annotation { get; set; }
        public int Chair_Id { get; set; }
```

```
        public int Level { get; set; }
        public string Teacher { get; set; }
        public bool Selected { get; set; }
        public int Stud_Limit { get; set; }
        public string Status { get; set; }
        public string Status_Happened { get; set; }
        public string Status_Enrollment { get; set; }
        public int Max_Stud { get; set; }
        public int Min_Stud { get; set; }
        public int Max_Group { get; set; }
        public int Actutal_Group { get; set; }
        public string Del_Gorup { get; set; }
        public int Stud_Count { get; set; }
        public int Common_Groups_Count { get; set; }
        public int Reverse_Group_Count { get; set; }
        public string Created_At { get; set; }
        public string Updated_At { get; set; }

        public DisciplineProfile()
        {

        }
        public DisciplineProfile(string name)
        {
            Name = name;
        }

        public DisciplineProfile(string id, string sub_cdoc, string name, string type, int
academicYear, int format,
            int year,
            int hours, int credits, string annotation, int chairId, int level, string
teacher, bool selected,
            int studLimit, string status,
            string statusHappened, string statusEnrollment, int maxStud, int minStud, int
maxGroup, int actutalGroup,
            string delGorup, int studCount,
            int commonGroupsCount, int reverseGroupCount, string createdAt, string updated)
        {
            Id = id;
            Sub_cdoc = sub_cdoc;
            Name = name;
            Type = type;
            Stud_Limit = studLimit;
            Status = status;
            Academic_Year = academicYear;
            Format = format;
            Year = year;
            Hours = hours;
            Credits = credits;
            Annotation = annotation;
            Chair_Id = chairId;
            Level = level;
            Teacher = teacher;
            Selected = selected;
            Status_Happened = statusHappened;
            Status_Enrollment = statusEnrollment;
            Max_Stud = maxStud;
            Min_Stud = minStud;
            Max_Group = maxGroup;
            Actutal_Group = actutalGroup;
            Del_Gorup = delGorup;
            Stud_Count = studCount;
            Common_Groups_Count = commonGroupsCount;
            Reverse_Group_Count = reverseGroupCount;
            Created_At = createdAt;
```

```
            Updated_At = updated;

        }
    }
}

namespace ComplexDialogBot.Profiles
{
    public class FacultyProfile
    {
        public  int  Id { get; set; }
        public string Name { get; set; }

        public FacultyProfile(int id, string name)
        {
            Id = id;
            Name = name;
        }

    }
}
using System;

namespace ComplexDialogBot.Profiles
{
    public class NewsProfile
    {
        public string Id { get; set; }
        public DateTime Date { get; set; }

        public string Content { get; set; }

        public NewsProfile(string id, DateTime date, string content)
        {
            Id = id;
            Date = date;
            Content = content;
        }
    }
}
namespace ComplexDialogBot.Profiles
{
    public class SpecialityProfile
    {
        public int Id { get; set; }
        public  string Name { get; set; }
        public int FacultyId { get; set; }
        public  int Level { get; set; }

        public SpecialityProfile(int id, string name, int facultyId, int level)
        {
            Id = id;
            Name = name;
            FacultyId = facultyId;
            Level = level;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace SAZBot.Database
```

```
{
    public class DBConnection
    {
        private DBConnection()
        {
        }

        public string Server { get; set; }
        public string DatabaseName { get; set; }
        public string UserName { get; set; }
        public string Password { get; set; }

        public MySqlConnection Connection { get; set; }

        private static DBConnection _instance = null;
        public static DBConnection Instance()
        {
            if (_instance == null)
                _instance = new DBConnection();
            return _instance;
        }

        public bool IsConnect()
        {
            if (Connection == null)
            {
                if (String.IsNullOrEmpty(DatabaseName))
                    return false;
                string connstring = string.Format("Server={0}; database={1}; UID={2};
password={3}", Server, DatabaseName, UserName, Password);
                Connection = new MySqlConnection(connstring);
                Connection.Open();
            }

            return true;
        }

        public void Close()
        {
            Connection.Close();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Data;
using System.Reflection.Metadata.Ecma335;
using ComplexDialogBot.Profiles;
using MySql.Data.MySqlClient;

namespace SAZBot.Database
{
    public class DBWorker
    {
        private DBConnection _dbCon;
        public DBWorker()
        {

            _dbCon = DBConnection.Instance();
            _dbCon.Server = "localhost";
            _dbCon.DatabaseName = "database";
            _dbCon.UserName = "username";
            _dbCon.Password = "password";

        }
```

```
public List<FacultyProfile> ReturnFacultyList()
{
    var faculties = new List<FacultyProfile>();
    if (_dbCon.IsConnect())
    {
        var query = "select * from faculty;";
        try
        {
            if (_dbCon.Connection.State == ConnectionState.Closed)
                _dbCon.Connection.Open();
            var cmd = new MySqlCommand(query, _dbCon.Connection);
            var reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                int id = reader.GetInt32(0);
                string name = reader.GetString(1);
                faculties.Add(new FacultyProfile(id, name));
            }
        }
        catch (Exception ex)
        {

        }

    }
    _dbCon.Close();
    return faculties;

}

public List<SpecialityProfile> ReturnSpecialityBasedonFaculty(string facultyId)
{
    var speciality = new List<SpecialityProfile>();
    if (_dbCon.IsConnect())
    {
        var query = "select * from speciality where faculty_id = "+facultyId+";";
        try
        {
            if (_dbCon.Connection.State == ConnectionState.Closed)
                _dbCon.Connection.Open();
            var cmd = new MySqlCommand(query, _dbCon.Connection);
            var reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                int id = reader.GetInt32(0);
                string name = reader.GetString(1);
                int faculty_Id = reader.GetInt32(2);
                int level = reader.GetInt32(3);
                speciality.Add(new SpecialityProfile(id, name,faculty_Id, level));
            }
        }
        catch (Exception ex)
        {

        }
    }
    _dbCon.Close();
    return speciality;

}
public List<NewsProfile> ShowAllnews()
{
    var news = new List<NewsProfile>();
    if (_dbCon.IsConnect())
```

```
    {
        var query = "Select * from news ORDER BY date;";
        try
        {
            if (_dbCon.Connection.State == ConnectionState.Closed)
                _dbCon.Connection.Open();
            var cmd = new MySqlCommand(query, _dbCon.Connection);
            var reader = cmd.ExecuteReader();

            while (reader.Read())
            {
                string id = reader.GetString(0);
                DateTime date = reader.GetDateTime(1);
                //TODO: Remove extra characterhs
                string content = reader.GetString(2);
                content = content.Replace("<p>", "");
                content = content.Replace("</p>", "");
                content = content.Replace("<b>", "");
                content = content.Replace("</b>", "");
                content = content.Replace("<br />", "");
                content = content.Replace("<br/>", "");
                content = content.Replace("<span style=“color: #ff0000;”>", "");
                content = content.Replace("</span>", "");
                content = content.Replace("<i>", "");
                content = content.Replace("</i>", "");
                content = content.Replace("</h1>", "");
                content = content.Replace("<h1>", "");
                content = content.Replace("<p align=“center”>", "");
                content = content.Replace("<p align=“left”>", "");
                content = content.Replace("<p align=“right”>", "");
                content = content.Replace("<u>", "");
                content = content.Replace("</u>", "");
                content = content.Replace("<strong>", "");
                content = content.Replace("</strong>", "");
                news.Add(new NewsProfile(id, date, content));

            }
        }
        catch (Exception ex)
        {


        }


    }
    _dbCon.Close();
    return news;

}


public List<DisciplineProfile> GetDisciplineList(string value)
{

    var disciplines = new List<DisciplineProfile>();
    if (_dbCon.IsConnect())
    {
        if (_dbCon.Connection.State == ConnectionState.Closed)
            _dbCon.Connection.Open();
        var query = "select * from course where name like '%"+value+"%';";
        var cmd = new MySqlCommand(query, _dbCon.Connection);
        var reader = cmd.ExecuteReader();
        while (reader.Read())
        {
```

```
                    string id = reader.GetString(0);
                    string sub_cdoc = reader.GetString(1);
                    string name = reader.GetString(2);
                    string type = reader.GetString(3);
                    int academic_year = reader.GetInt32(4);
                    int format = reader.GetInt32(5);
                    int year = reader.GetInt32(6);
                    int hours = reader.GetInt32(7);
                    int credits = reader.GetInt32(8);
                    string annotation = reader.GetString(9);
                    int chair_id = reader.GetInt32(10);
                    int level = reader.GetInt32(11);
                    string teacher = reader.GetString(12);
                    bool selected = reader.GetBoolean(13);
                    int stud_limit = reader.GetInt32(14);
                    string status = reader.GetString(15);
                    string status_happened = reader.GetString(16);
                    string status_enrollement = reader.GetString(17);
                    int max_stud = reader.GetInt32(18);
                    int min_stud = reader.GetInt32(19);
                    int max_group = reader.GetInt32(20);
                    int actual_group = reader.GetInt32(21);
                    //  int del_group = reader.GetInt32(22);
                    int stud_count = reader.GetInt32(23);
                    int common_group_count = reader.GetInt32(24);
                    int reserver_group_count = reader.GetInt32(25);
                    string created_at = reader.GetString(26);
                    string updated_at = reader.GetString(27);

                    var disc = new DisciplineProfile(id, sub_cdoc, name, type,
academic_year, format, year, hours,
                        credits, annotation, chair_id, level, teacher, selected, stud_limit,
status, status_happened,
                        status_enrollement, max_stud, min_stud, max_group, actual_group,
"0", stud_count,
                        common_group_count, reserver_group_count, created_at, updated_at);
                   // var dic1 = new DisciplineProfile(name);
                    disciplines.Add(disc);

                }

                _dbCon.Close();
            }

            return disciplines;
        }


    }
}



using System.Collections.Generic;

using Microsoft.Bot.Builder.Dialogs.Choices;


namespace ComplexDialogBot.Services

{

    public class SpecialityList

    {
```

```
        public SpecialityList()
        {

        }

        public  static IList<Choice> GetChoices(string faculty, string year)
        {
            switch (faculty)
            {
                case "Факультет гуманітарних наук":
                    switch (year)
                    {
                        case "МП, 1 рік навчання":
                            var cardOptions0 = new List<Choice>()
                            {
                                new Choice() {Value = "Історія"},
                                new Choice() {Value = "Філологія. Теорія, історія української
мови та компаративістика"},
                                new Choice()
                                {
                                    Value = "Культурологія"
                                },
                                new Choice() {Value = "Філософія"},
                                new Choice() {Value = "Археологія"},
                                new Choice() {Value = "Філологія. Теорія, історія літератури
та компаративістика"},
                                new Choice() {Value = "Юдаїка"},
                                new  Choice()  {Value  =  "Мови  (англійська  й  українська)  та
комунікація"}
                            };
                            return cardOptions0;

                        case "МП, 2 рік навчання":
                            var cardOptions1 = new List<Choice>()
                            {
                                new Choice() {Value = "Історія"},
                                new Choice() {Value = "Філологія. Теорія, історія української
мови та компаративістика"},
                                new Choice()
```

```
                                {
                                    Value = "Культурологія"
                                },
                                new Choice() {Value = "Філософія"},
                                new Choice() {Value = "Археологія"},
                                new Choice() {Value = "Філологія. Теорія, історія літератури
та компаративістика"},
                                new Choice() {Value = "Юдаїка"}
                            };
                            return cardOptions1;
                        default:
                            var cardOptions = new List<Choice>()
                            {
                                new Choice() {Value = "Історія та археологія"},
                                new Choice() {Value = "Культурологія"},
                                new Choice()
                                {
                                    Value = "Філологія (германські мови та літератури
(переклад включно))"
                                },
                                new Choice() {Value = "Філологія (українська мова та
література)"},
                                new Choice() {Value = "Філософія"},

                            };
                            return cardOptions;

                    }
                case "Факультет інформатики":
                    switch (year)
                    {
                        case "МП, 1 рік навчання":
                            var cardOptions011 = new List<Choice>()
                            {
                                new Choice() {Value = "Інженерія програмного забезпечення"},
                                new Choice() {Value = "Комп`ютерні науки"},
                                new Choice() {Value = "Прикладна математика"},
                                new Choice() {Value = "Системний аналіз"},
                            };
```

```
                    return cardOptions011;

                case "МП, 2 рік навчання":
                    var cardOptions01 = new List<Choice>()
                    {
                        new Choice() {Value = "Інженерія програмного забезпечення"},
                        new Choice() {Value = "Комп`ютерні науки"},
                        new Choice() {Value = "Прикладна математика"},
                        new Choice() {Value = "Системний аналіз"},
                    };
                    return cardOptions01;
                default:
                    var cardOptions01111 = new List<Choice>()
                    {
                        new Choice() {Value = "Інженерія програмного забезпечення"},
                        new Choice() {Value = "Комп`ютерні науки"},
                        new Choice() {Value = "Прикладна математика"},
                    };
                    return cardOptions01111;
            }
        case "Факультет правничих наук":
                    switch (year)
                    {
                        case "МП, 1 рік навчання":
                            var cardOptions0 = new List<Choice>()
                            {
                                new Choice() {Value = "Право"},
                                new Choice() {Value = "Комунікації в демократичному
врядуванні"},

                            };
                            return cardOptions0;

                        case "МП, 2 рік навчання":
                            var cardOptions1 = new List<Choice>()
                            {
                                new Choice() {Value = "Право"},
                                new Choice() {Value = "Публічне управління та
адміністрування"},
```

```
                    };
                    return cardOptions1;
                default:
                    var cardOptions = new List<Choice>()
                    {
                        new Choice() {Value = "Право"},



                    };
                    return cardOptions;

            }
        case "Факультет економічних наук":
            switch (year)
            {
                case "МП, 1 рік навчання":
                    var cardOptions011 = new List<Choice>()
                    {
                        new Choice() {Value = "Економіка"},
                        new Choice() {Value = "Маркетинг"},
                        new Choice() {Value = "Розвиток бізнесу управління та
консалтинг"},
                        new Choice() {Value = "Управління енергоефективністю"},
                        new Choice() {Value = "Фінанси, банківська справа та
страхування"},
                    };
                    return cardOptions011;

                case "МП, 2 рік навчання":
                    var cardOptions01 = new List<Choice>()
                    {
                        new Choice() {Value = "Економіка"},
                        new Choice() {Value = "Маркетинг"},
                        new Choice() {Value = "Розвиток бізнесу управління та
консалтинг"},
                        new Choice() {Value = "Управління енергоефективністю"},
                        new Choice() {Value = "Фінанси, банківська справа та
страхування"},
                    };
```

```
                        return cardOptions01;
                    default:
                        var cardOptions01111 = new List<Choice>()
                        {
                            new Choice() {Value = "Економіка"},
                            new Choice() {Value = "Маркетинг"},
                            new Choice() {Value = "Менеджмент"},
                            new  Choice()  {Value  =  "Фінанси,  банківська  справа  та
страхування"},
                        };
                        return cardOptions01111;
                }
            case "Факультет соціальних наук і соціальних технологій":
                switch (year)
                {
                    case "МП, 1 рік навчання":
                        var cardOptions0121 = new List<Choice>()
                        {
                            new Choice() {Value = "Менеджмент в охороні здоров`я"},
                            new Choice() {Value = "Політологія"},
                            new Choice() {Value = "Соціологія"},
                            new Choice() {Value = "Психологія"},
                            new Choice() {Value = "Зв`язки з громадськістю"},
                            new Choice() {Value = "Соціальна робота"},
                        };
                        return cardOptions0121;

                    case "МП, 2 рік навчання":
                        var cardOptions016 = new List<Choice>()
                        {
                            new Choice() {Value = "Менеджмент в охороні здоров`я"},
                            new Choice() {Value = "Політологія"},
                            new Choice() {Value = "Соціологія"},
                            new Choice() {Value = "Психологія"},
                            new Choice() {Value = "Зв`язки з громадськістю"},
                            new Choice() {Value = "Соціальна робота"},
                        };
                        return cardOptions016;
                    case "БП, 1 рік навчання":
```

```
                            var cardOptions0112 = new List<Choice>()
                            {
                                new Choice() {Value = "Міжнародні відносини, суспільні комунікації та регіональні студії"},
                                new Choice() {Value = "Політологія"},
                                new Choice() {Value = "Психологія"},
                                new Choice() {Value = "Соціальна робота"},
                                new Choice() {Value = "Соціологія"},
                                new Choice() {Value = "Зв`язки з громадськістю"}
                            };
                            return cardOptions0112;
                        default:
                            var cardOptions01111 = new List<Choice>()
                            {
                                new Choice() {Value = "Міжнародні відносини, суспільні комунікації та регіональні студії"},
                                new Choice() {Value = "Політологія"},
                                new Choice() {Value = "Психологія"},
                                new Choice() {Value = "Соціальна робота"},
                                new Choice() {Value = "Соціологія"},

                            };
                            return cardOptions01111;
                    }
                case "Факультет природничих наук":
                    switch (year)
                    {
                        case "МП, 1 рік навчання":
                            var cardOptions011 = new List<Choice>()
                            {
                                new Choice() {Value = "Екологія та охорона навколишнього середовища"},
                                new Choice() {Value = "Молекулярна біологія"},
                                new Choice() {Value = "Хімія"},
                                new Choice() {Value = "Лабораторна діагностика біологічних систем"},
                                new Choice() {Value = "Фізика (Теоретична фізика)"},
                            };
                            return cardOptions011;
```

```
                        case "МП, 2 рік навчання":
                            var cardOptions016 = new List<Choice>()
                            {
                                new Choice() {Value = "Екологія та охорона навколишнього
середовища"},
                                new Choice() {Value = "Молекулярна біологія"},
                                new Choice() {Value = "Хімія"},
                                new Choice() {Value = "Лабораторна діагностика біологічних
систем"},
                                new Choice() {Value = "Фізика та астрономія"},

                            };
                            return cardOptions016;
                        case "БП, 1 рік навчання":
                            var cardOptions0112 = new List<Choice>()
                            {
                                new Choice() {Value = "Екологія"},
                                new Choice() {Value = "Фізика та астрономія"},
                                new Choice() {Value = "Хімія"},
                                new Choice() {Value = "Біологія та біотехнологія"},
                            };
                            return cardOptions0112;
                        default:
                            var cardOptions01111 = new List<Choice>()
                            {
                                new Choice() {Value = "Хімія"},
                                new Choice() {Value = "Фізика та астрономія"},
                                new Choice() {Value = "Екологія"},
                                new Choice() {Value = "Біологія"},

                            };
                            return cardOptions01111;
                    }
            }
            return null;
        }

    }
}
```