НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.584+519.816

До захисту допущено
Завідувач кафедри ММСА
_____Оксана ТИМОЩУК
«___» _____ 2024 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз фінансового ринку»
зі спеціальності 124 «Системний аналіз»
на тему: «Чат-бот як середовище розгортання системи підтримки прийняття рішень: приклад телеграм-боту по наданню рекомендацій щодо вибіркових дисциплін»

Виконав:
Студент 2 курсу, групи КА-22мп
Харабара Денис Вікторович _____

Науковий керівник:
к.ф.-м.н., доцент кафедри ММСА,
Статкевич Віталій Михайлович _____

Рецензент:
Професор кафедри штучного інтелекту НН ІПСА,
д.т.н., професор,
Данилов Валерій Якович _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань
Студент (*підпис*): _____

Київ – 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 124 «Системний аналіз»
Освітньо-професійною програмою «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
_____Оксана ТИМОЩУК
«___» _____ 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Харабарі Денису Вікторовичу

1. Тема дисертації: «Чат-бот як середовище розгортання системи підтримки прийняття рішень: приклад телеграм-боту по наданню рекомендацій щодо вибіркових дисциплін», науковий керівник дисертації доцент кафедри ММСА Статкевич Віталій Михайлович, затверджені наказом по університету від «08» листопада 2023 р. № 5200-с

2. Строк подання студентом дисертації: _____

3. Об’єкт дослідження: процес створення рекомендаційних систем.

4. Предмет дослідження: рекомендаційні системи що базуються на колаборативній фільтрації, їх поєднання із чат-бот платформами.

5. Перелік завдань, які потрібно розробити:
1) провести аналіз предметної області та актуальності теми;
2) провести огляд основних підходів до побудови чат-ботів;
3) визначитись із середовищем розгортання боту та програмним апаратом;
4) провести аналіз наявної літератури в сфері рекомендаційних систем;
5) обрати та детальніше дослідити один із підходів до побудови рекомендаційних систем;
6) розробити концепцію чат-боту;
7) розробити логічну схему чат-боту;
8) реалізувати систему підтримки прийняття рішень згідно вимог;
9) провести тестування програмного продукту;

10) проаналізувати отримані резутьтати та оцінити перспективи подальших досліджень.

6. Перелік графічного (ілюстративного) матеріалу:

1) схема функціонування чат-ботів;
2) приклади інтерсейсу чат-боту в телеграмі;
3) приклади інтерфейсу середовища розробки Corezoid;
4) результати створення та роботи тестових процесів;
5) інформаційні відомості про BotFather;
6) описові відомості про створений чат-бот;
7) результати створення частин чат-боту;
8) логічна схема флоу бота;
9) візуальне представлення формату даних в сховищах;
10) структурна схема модулю отримання рекомендацій;
11) результати тестування рекомендаційного боту.

7. Орієнтовний перелік публікацій:

1. Харабара Д.В., Статкевич В.М. Чат-бот як середовище розгортання системи підтримки прийняття рішень: приклад телеграм-боту по наданню рекомендацій щодо вибіркових дисциплін: збірник доповідей II науково-практичної конференції «Системні науки та інформатика», 4–8 грудня 2023 року, Київ. – К., НН ІПСА КПІ ім. Ігоря Сікорського, 2023. С. 229-236.

8. Дата видачі завдання: 1 вересня 2023 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1.	Формування теми магістерської дисертації	01.01.2023 – 05.09.2023	Виконано
2.	Аналіз предметної області	06.09.2023 – 11.09.2023	Виконано
3.	Огляд основних підходів до побудови чат-ботів	12.09.2023 – 23.09.2023	Виконано
4.	Аналіз існуючих месенджерів та їх функціоналу	24.09.2023 – 28.09.2023	Виконано
5.	Знайомство із підходами розробки чат-ботів	29.09.2023 – 04.10.2023	Виконано
6.	Вивчення документації Corezoid	05.10.2023 – 16.10.2023	Виконано
7.	Аналіз літератури в сфері розробки рекомендаційних сиситем	17.10.2023 – 24.10.2023	Виконано
8.	Порівняльний-аналіз існуючих підходів до побудови систем підтримки прийняття рішень	25.10.2023 – 03.11.2023	Виконано
9.	Поглиблене вивчення рекомендаційних систем, що базуються на колаборативній фільтрації	03.11.2023 – 10.11.2023	Виконано
10.	Розробка концепції чат-боту	11.11.2023 – 13.11.2023	Виконано
11.	Розробка логічної схеми проекту	14.11.2023 – 17.11.2023	Виконано
12.	Розробка логічного ядра чат-боту	18.11.2023 – 23.12.2023	Виконано
13.	Наповнення сховищ тестовими даними	24.11.2023 – 27.11.2023	Виконано
14.	Розробка модулю занесення оцінок	28.11.2023 – 02.12.2023	Виконано
15.	Розробка модулю надання рекомендацій	03.12.2023 – 10.12.2023	Виконано
16.	Збір компонентів сиситеми	11.12.2023 – 14.12.2023	Виконано
17.	Тестування роботи рекомендаційного чат-боту	15.12.2023 – 19.12.2023	Виконано
18.	Опис стартап-проекту	20.12.2023 – 29.12.2023	Виконано
19.	Оформлення магістерської дисертації	30.12.2023 – 04.01.2024	Виконано

Студент _____ Денис ХАРАБАРА

Науковий керівник дисертації _____ Віталій СТАТКЕВИЧ

РЕФЕРАТ

Магістерська дисертація: 99 с., 42 рис., 26 табл., 3 додатки, 28 джерел.

Ключові слова: ЧАТ-БОТ, МЕСЕНДЖЕР, TELEGRAM BOT API, ХМАРНЕ СЕРЕДОВИЩЕ РОЗРОБКИ, COREZOID, АВТОМАТНЕ ПРОГРАМУВАННЯ, НОДА, СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, РЕКОМЕНДАЦІЙНА СИСТЕМА, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, АЛГОРИТМ ЗАСНОВАНИЙ НА МОДЕЛІ, АЛГОРИТМ ЗАСНОВАНИЙ НА ПАМ'ЯТІ, ВИБІРКОВІ ДИСЦИПЛІНИ.

Актуальність: чат-боти, як канали вирішення проблем уже не перший рік широко використовуються компаніями задля пришвидшення та спрощення комунікації із клієнтами. Тому їх використання, в поєднанні із рекомендаційними системами, є чудовим підходом до вирішення буденних дилем, зокрема дилеми отримання рекомендацій щодо вибіркових дисциплін.

Об'єкт дослідження: процес створення рекомендаційних систем.

Предмет дослідження: рекомендаційні системи що базуються на колаборативній фільтрації, їх поєднання із чат-бот платформами.

Мета дослідження: дослідити системи підтримки прийняття рішень, зокрема рекомендаційні системи, що базуються на колаборативній фільтрації, знайти шляхи їх поєднання із чат-бот платформами, задля створення комплексного підходу по вирішенню рутинних дилем.

Результат дослідження: демонстрація зручності та швидкості використання рекомендаційної системи, що інтегрована в чат-бот.

Наукова новизна дослідження: спроектовано, розроблено та протестовано програмну реалізацію рекомендаційної системи на базі телеграм-боту, що ґрунтуються на колаборативній фільтрації, із використанням хмарного середовища розробки Corezoid.

ABSTRACT

Master’s thesis: 99 p., 42 fig., 26 tabl., 3 appendices, 28 ref.

Keywords: CHAT BOT, MESSENGER, TELEGRAM BOT API, CLOUD DEVELOPMENT ENVIRONMENT, COREZOID, AUTOMATIC PROGRAMMING, NODA, DECISION SUPPORT SYSTEM, RECOMMENDER SYSTEM, COLLABORATIVE FILTERING, MODEL-BASED ALGORITHM, MEMORY-BASED ALGORITHM, ELECTIVE DISCIPLINES.

The relevance of the study: chatbots, as channels for solving problems, have been widely used by companies to speed up and simplify communication with customers for many years. Therefore, their use, in combination with recommender systems, is an excellent approach to solving any dilemma, in particular the dilemma of obtaining recommendations for selective disciplines.

The object of research: the process of creating recommender systems.

The subject of research: recommender systems based on collaborative filtering, their combination with chat-bot platforms.

The purpose of the research: to investigate decision support systems, in particular recommender systems based on collaborative filtering, to find ways to combine them with chatbot platforms, in order to create a comprehensive approach to solving routine dilemmas.

The research result: demonstration of the convenience and speed of using the recommender system integrated into the chatbot.

The scientific novelty: the software implementation of a Telegram bot-based recommender system based on collaborative filtering was designed, developed and tested using the Corezoid cloud development environment.

ЗМІСТ

ВСТУП

Звичайни день сучасної людини повний рутинних дилем, від вибору одягу в залежності від погоди до вибору кінотеатру чи перевірки курсу валюти. Багато з цих рішень ми приймаємо на основі власного досвіду, порад друзів або власних переконань. Але якщо досвід відсутній або немає можливості отримати рекомендацію від знайомих, пошук необхідної інформації перетворюється в багатогодине перебирання масивів даних у всесвітній павутині, яке не завжди надає чіткої рекомендації.

Зокрема, із подібними труднощами неодноразово зіштовхуються студенти, при підборі вибіркових дисциплін на наступний семестр. Цей процес потребує врахування багатьох аспектів, таких як рекомендації попередників, програму предмету та багато інших факторі. Зважаючи на це, очевидним рішенням, задля спрощення даної процедури, є створення комплексного підходу з аналізу досвіду попередників, без їх безпосереднього залучення до процесу, який міг би швидко та без зайвих дій надавати актуальні рекомендації.

Ідеальним варіантом, який задовольнив би вимоги вище, може стати створення системи підтримки прийняття рішень на базі чат-бота, з інтуїтивно зрозумілим інтерфейсом та логічним механізмом отримання вхідних даних для визначення рекомендацій, проектування, розробка та тестування якого є основним завданням даної роботи.

Магістерська дисертація складається з вступу, чотирьох розділів, висновків, переліку джерел посилання та трьох додатків. В першому розділі розглядається основні визначення та характеристики чат-ботів, їх класифікація та підходи до побудови. Також даний розділ вміщує в собі інформацію щодо хмарного середовища розробки Corezoid та принципів побудови процесів у ньому. В другому розділі описані підходи до побудови

систем підтримки прийняття рішень, розглянуто їх класифікацію. Більш детальна увага приділена саме системам, які засновані на колаборативній фільтрації, їх перевагам та недолікам. Третій розділ – це практична частина роботи, яка вміщує в собі ввесь процес конструювання, розробки та тестування рекомендаційного чат-боту на базі месенджеру Telegram. В четвертому розділі було сформовано стратегію виходу проекту на ринок, визначено його сильні та слабкі сторони, проведено порівняння із наявними наразі традиційними методами визначення вибіркових дисциплін.

Перелік джерел посилання містить 28 джерел. Додаток А містить в собі програмний код, який написаний на мові програмування JavaScript та використовувався для генерації тестових даних, в додатку Б продемонстрований повний код визначення коефіцієнтів подібності студентів, що знаходиться в ноді коду модулю отримання рекомендацій, в додатку В – код із ноди, що прогнозує оцінки із вибіркових дисциплін, та сортує їх.

РОЗДІЛ 1 ОСНОВНІ ПОНЯТТЯ ТА ПРИНЦИПИ РОЗРОБКИ ЧАТ-БОТІВ. COREZOID ЯК СЕРЕДОВИЩЕ РОЗРОБКИ ПРОЦЕСІВ РІЗНОЇ СКЛАДНОСТІ

Чат-бот – це програмне забезпечення, призначене для імітації спілкування з людьми. Воно виконує роль співрозмовника або віртуального помічника, користуючись заздалегідь написаними сценаріями для надання швидких та коректних відповідей [1].

В наш час, чат-боти, як технології, дуже стрімко розвиваються та поширюються, що пов’язане із широким використанням різних месенджерів та можливістю переносу функціоналу мобільних додатків в них. Користувачам більше не потрібно встановлювати нові програми на свої пристрої, а достатньо знайти необхідний канал із інформацією, в середині звичних для них месенджерах.

Варто відмітити, що величезний внесок в глобалізацію чат-ботів роблять світові компанії, адже саме вони дають можливість вбудовувати в свої продукти функціонал ботів, і фактично є платформами для їх створення.

1.1 Класифікація чат-ботів

В цілому існує безліч підходів до класифікації чат-ботів, пов’язано це із їх різновекторним використанням: для зворотного зв’язку клієнта із компанією, споживання контенту, створення інтернет магазинів, проведення фінансових операцій і т.д. Даній темі присвячено чимало робіт як вітчизняних так і зарубіжних вчених [2-8].

Також розмаїття підходів пов’язане із великим спектром існуючого програмного інструментарію для розробки чат-ботів, який постійно розширюється.

Серед доступних інструментів є навіть такі, що дають можливість створювати чат-боти без знань в програмуванні [7]:

- Chatfuel;
- SendPulse;
- Botsify;
- Dialigflow;
- ManyChat.

Якщо говорити про мету сферу застосування чат-ботів, то їх умовно можна поділити на персональні та комерційні (для ділового використання).

Перші зазвичай створюються для допомоги користувачу в буденних справах: створення нотатків, керування календарем, керуванням викликів. В більш рідших кейсах – задля керування системою “розумного будинку”.

Інші ж використовуються для спрощення комунікації компаній із клієнтами, оптимізації внутрішніх та зовнішніх бізнес-процесів, організації продажів та інших корпоративних завдань.

Зважаючи на інформацію вище, та уже існуючі напрацювання в даному напрямку, можна сформувати наступну узагальнену класифікацію чат-ботів.

1. За призначенням:

- комунікаційний (основною задачею є комунікація);
- функціональний (кінцевою метою є отримання певних результатів, виконання певних дій).

2. За цільовою аудиторією:

- персональний (створений для власного використання);
- для бізнесу (створений для потреб бізнесу).

3. За наявним інтерфейсом:

- кнопковий (керування відбувається шляхом натискання кнопок);
- текстовий (керування відбувається шляхом обміну повідомленнями);
- голосовий (голосове керування).

4. За принципом роботи:

- шаблонований (має в своїй структурі певний набір інструкції, та слідує ним);
- який навчається (постійно навчається та змінює свій алгоритм дій).

Зрозуміло, що наведена вище класифікація є ідеалізованою, в більшості випадків ботів складно віднести до якоїсь однієї групи за певними ознаками, як наприклад за структурою інтерфейсу, тому їх прийнято називати гібридними.

1.2 Структура чат-боту

Для створення програмної частини чат-боту можна використовувати будь-яку мову програмування: JavaScript, Python, C++ тощо. Проте структурна модель функціонування боту завжди однакова (рис 1.1) [6].

Рисунок 1.1 – Структурна модель функціонування чат-боту

Front-End – інтерфейс користувача, в ролі якого можуть бути як уже існуючі продукти: Facebook, Telegram, Viber, Skype, Instagram, так і власні реалізації (веб-сторінки, мобільні застосунки і т.д.).

Back-End – серверна частина, яка відповідає за обробку вхідних подій, що надходять від front-end сторони, маршрутизацію по процесу та відправку результату.

Webhook – принцип взаємодії front-end та back-end частин, згідно якого front-end при отриманні будь-якої дії від користувача направляє її на back-end

в вигляді HTTP POST запиту із заздалегіть визначеним форматом даних, зазвичай це JSON чи XML [9].

Polling – принцип взаємодії беку із фронтом, при якому back-end надсилає все той ж HTTP POST запит та стає в очікування на реакцію front-end сторони [10].

1.3 Месенджер як платформа для створення чат-ботів

Месенджер – це програма, мобільний додаток або веб-сервіс для миттєвого обміну повідомленнями, що може включати в себе додаткові сервіси (голосові та відео виклики, чат-боти, магазин покупок, ігри), та одночасно використовується великою кількістю користувачів [11].

До найпопулярніших месенджерів сьогодення відносяться Facebook, Telegram та Viber, тому проведемо їх порівняльний аналіз (таб. 1.1).

Для порівняння виділимо наступні критерії:

- ціна;
- інтерфейс;
- механізм зв’язку;
- фунціонал;
- цільова аудиторія.

Таблиця 1.1 – Порівняльна характеристика месенджерів

	Telegram	Viber	Facebook
Ціна	Повністю безкоштовно	Безкоштовна тільки частина функціоналу	Безкоштовна тільки частина функціоналу
Інтерфейс	Тест, кнопки, голосові повідомлення, веб застосунки	Тест, кнопки, голосові повідомлення	Тест, кнопки, голосові повідомлення

Продовження таблиці 1.1

Механізм зв’язку	Webhook, Polling	Webhook	Webhook
Функціонал	Листування, опитування, відеодзвінки, передача файлів, веб-застосунки, аудіоповідомлення, шеринг геолокації, шеринг номеру телефону.	Листування, опитування, відеодзвінки, передача файлів, аудіоповідомлення, шеринг геолокації, шеринг номеру телефону.	Листування, опитування, передача файлів, аудіоповідомлення
Цільова аудиторія	14 – 45 років	28 – 65 років	28 – 65 років

Із таблиці вище можна чітко побачити, що найоптимальнішим для створення чат-боту є месенджер Telegram, оскільки він має більш ширший функціонал, в порівнянні із попередниками, і цільовою аудиторією даного застосунку є саме та вікова група, яку можуть зацікавити чат-боти.

Якщо говорити про принципи зв’язку, то телеграм також лідирує, оскільки оскільки має два механізми, а не один.

Тому в даній роботі ми будемо розробляти чат-бот саме в Телеграмі.

1.4 Особливості розроби ботів в Telegram

В попередньому пункті визначено, що найоптимальнішим для розробки чат-боту є саме месенджер Telegram, тому варто відзначити, що чат-бот, в рамках телеграму, це не окремий застосунок, а вбудований в екосистему

телеграму додаток, який фактично спеціальним окремим аккаунтом в месенджері, взаємодіяти із яким користувачі можуть декількома методами:

- пряме спілкування, при якому користувач напряму пише боту, як звичайному користувачу телеграму;
- інтеграція в групи – коли бот додається в уже існуючу групу, де є декілька акаунтів реальних людей, і користувачі взаємодіють із ним через спеціальні команди, які заздалегідь прописані в структурі боту.

Варто відмітити, що в цілях запобігання спаму, ініціювати контакт із чат-ботом може тільки користувач.

Якщо ж говорити про програмну частину, то керування ботом відбувається за допомогою спеціального API, із використанням протоколу HTTPS. А саме його функціонування складається із двох частин [12]:

- отримання оновлень;
- виклик методів.

Перша складова відповідає за сповіщення back-end частини про певні дії користувача, а друга – за надсилання даних користувачу.

В двох цих кейсах інформація передається в форматі JSON (текстовий формат обміну інформації між системами), основним завданням якого є опис об’єктів, а структура даних представлена двома варіантами [12].

1. Пара “ключ-значення”.
2. Впорядкований масив значень.

Отримання оновлень – це передача подій взаємодії користувача із ботом, при якій в кінцевий процес приходять дані в уніфікованому форматі. Зокрема обов’язкові поля для відправки оновлень наведені нижче (табл. 1.2) [12].

Таблиця 1.2 – Обов’язкові поля для відправки оновлень

Поле	Опис	Тип даних
update_id	Ідентифікатор оновлення, який є унікальним	Цілочисельний
message	Нове повідомлення, яке може представлятися як текст, фото, відео, аудіо	Об’єкт
edited_message	Відредаговане повідомлення	Об’єкт
callback_query	Результат взаємодії із інтерактивними частинами боту	Текстовий

При цьому наявність всіх трьох полів (message, edited_message, callback_query) не є обов’язковим, достатньо наявності тільки одного із них.

За своєю структурою поля message та edited_message однакові, саме ці об’єкти відповідають за передачу основної інформації про дії, які робить користувач. Основні поля об’єктів наведено нижче (табл. 1.3) [12].

Таблиця 1.3 – Поля об’єктів message та edited_message

Поле	**Опис**	**Тип даних**
message_id	Ідентифікатор повідомлення	Цілочисельний
date/edit_date	Час відправки/редагування повідомлення	Цілочисельний
text	Текст, що надіслав користувач	Текстовий
photo	Масив із посиланнями на фото, що відправив користувач, та їх описом	Масив об’єктів

Продовження таблиці 1.3

document	Посилання на документ, що відправив користувач, та його опис	Об'єкт
audio	Посилання на аудіофайл, що відправив користувач, та його опис	Об’єкт
reply_markup	Додаткові параметри інтерфейсу. Спеціалізований JSON-об’єкт для вбудованої клавіатури, спеціальної клавіатури відповіді, інструкцій щодо видалення клавіатури відповіді або примусової відповіді від користувача.	Масив об’єктів
chat	Інформація про канал комунікації, його тип, унікальний ідентифікатор, ім’я та ідентифікатор користувача, що відправив повідомлення	Об’єкт

Із таблиці вище можна помітити, що телеграм самостійно зберігає аудіо та медіа файли в своєму сховищі, та дає можливість їх вивантажувати звідти, проте варто зважати на те, що файли зберігаються не вічно, тому за необхідності потрібно їх додатково вивантажувати в локальну базу даних.

Наведені вище поля (табл. 1.2 та табл. 1.3) використовуються не тільки для відправки оновлень зі сторони телеграму, а й для відправки конкретному користувачу, за допомогою методів.

Якщо перейти на документацію до Telegram Bot API [12], то можна помітити, що там уже наявний величезний список доступних до використання методів, із детальним описом кожного із полів. Це і /getMe для тестування дієздатності боту, і /sendPhoto, /sendAudio, /sendDocument, /sendContact - для відправки різних медіафайлів, і /banChatMember,/unbanChatMember – для керування користувачами в групі, в якій є бот, і навіть /setChatPhoto, /setChatTitle – для редагування загальної інформації про самого боту.

Більш детально розглянемо один із них, який використовується практично у всіх чат-ботах і відповідає за безпосередню відправку повідомлень - /sendMessage.

Для використання даного методу, достать передавати всього два поля – chat_id та text, проте в більшості випадків даний запит також доповнюють полем reply_markup, щоб відправляти користувачу інтерективні елементи, такі як клавіатурки (рис. 1.2).

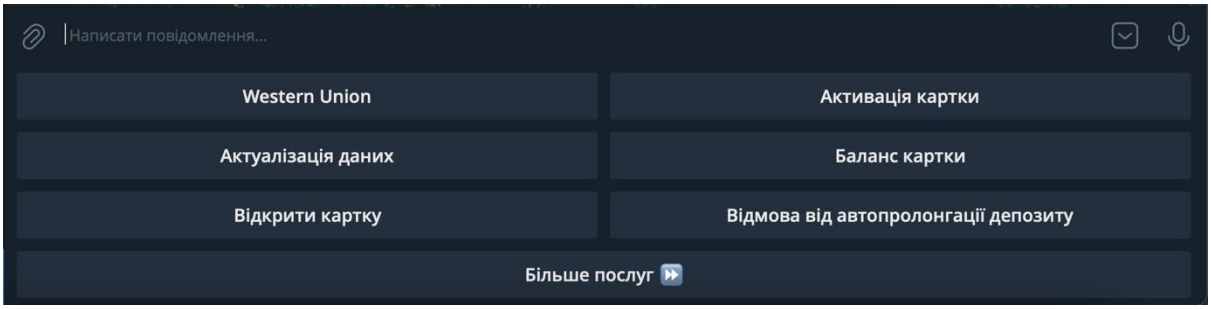


Рисунок 1.2 – Приклад клавіатури в Telegram

Отже ознайомившись із функціоналом мессенджеру, перейдемо до варіантів розробки back-end частини боту.

1.5 Corezoid як середовище розробки

Як зазначалось раніше, розробляти бек частину чат-боту можна на будь-якій серверній мові, будь це JavaScript, Python чи C++, не принципово.

Проте в контексті даної роботи ми розглянемо менш традиційний підхід, а саме розгортання back-end складових в хмарних середовищах розробки на прикладі Corezoid.

Corezoid - це нова хмарна операційна система для створення IT-рішень з використанням методів автоматного програмування з явним виділенням станів [13].

Простими ж словами, Corezoid – це інноваційне середовище програмування, яке спеціалізується на створенні, автоматизації та управлінні бізнес-процесами в хмарі. Ця платформа дозволяє створювати складні бізнес-логіку та автоматизовані рішення без необхідності в глибокому програмуванні.

Варто відмітити, що дана система є саме українською розробкою, яка на перших етапах була створена для зручності керування розсилками клієнтам в “Приват Банку”, а з часом отримала визнання та статус партнера в сфері Компетентності в фінансових послугах від Amazon Web Services (AWS), що дало чимали поштовх до відокремлення системи від інфраструктури банку, та позиціювання себе, як окремої хмарної платформи.

Як будь-яка система, із нетрадиційним підходом, Корезоїд має свої особливості [14].

1. Графічний інтерфейс та конструктор процесів: Corezoid пропонує інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам створювати, візуалізувати та моделювати бізнес-процеси у вигляді графів.

2. Централізоване керування: платформа дозволяє централізовано керувати усіма процесами, що спрощує моніторинг та управління бізнес-логікою.

3. Швидкість розгортання та інтеграція: Corezoid дозволяє швидко створювати та впроваджувати бізнес-процеси. Також, вона легко інтегрується з різноманітними системами, такими як CRM, ERP, банківські системи тощо.

4. Скейлінг та гнучкість: Платформа масштабується від малих бізнесів до великих корпорацій, дозволяючи змінювати та адаптувати процеси відповідно до потреб компанії.

5. Аналітика та звітність: Corezoid надає можливість аналізувати ефективність бізнес-процесів через зручний інструментарій збору та візуалізації даних.

Ця платформа стала популярною серед бізнес-користувачів через свою простоту використання, швидкість впровадження та гнучкість в адаптації до різних потреб підприємств. Вона дозволяє створювати і управляти бізнес-процесами у хмарі без значних витрат на програмування та підтримку інфраструктури.

Якщо говорити про процес розробки, то в Corezoid він базується на роботі з сутностями (заявками), нодами та логікою процесу. Ось ключові аспекти цього процесу [15].

1. Сутності (Entities): у Corezoid сутності - це дані або об'єкти, які представляють будь-яку інформацію, з якою працює користувач. Це може бути, наприклад, клієнт, замовлення, транзакція, документ тощо. Сутності можна створювати, оновлювати та використовувати для подальшої обробки в процесах.

2. Ноди (Nodes): ноди - це базові будівельні блоки процесу. Кожен вузол виконує конкретну дію або операцію. Наприклад, ноди можуть включати створення сутностей, виконання умовних операцій, взаємодію з зовнішніми системами через API, відправку повідомлень тощо.

3. Моделювання процесу: розробка процесу в Corezoid відбувається через створення графічного представлення процесу. Можна перетягувати та з'єднувати різні ноди для створення логіки процесу, встановлювати умови переходів між ними та управляти потоком даних.

4. Правила маршрутизації: Corezoid дозволяє налаштовувати правила маршрутизації, що визначають, який набір дій виконується в залежності від умов чи даних, отриманих в процесі виконання.

5. Інтеграція з іншими системами: однією з ключових можливостей Corezoid є здатність легко інтегруватися з різними зовнішніми системами та сервісами через вбудовані API.

6. Тестування та впровадження: після створення процесу його можна протестувати на різних тестових сценаріях. Після успішного тестування процес готовий до впровадження в реальне виробниче середовище.

Якщо ж говорити про робочий простір, візуальний вигляд якого продемонстрований на рисунку 1.3, то він складається із чотирьох простих та зрозумілих компонентів [15].

1. Процеси (Processes): процеси у Corezoid представляють собою послідовність дій або операцій, що виконуються для обробки даних або керування бізнес-процесами. Кожен процес може містити різні ноди, що виконують конкретні завдання, умови переходів між ними та маршрутизацію даних.

2. Діаграми станів (State Diagrams): Corezoid дозволяє моделювати діаграми станів для керування процесами на основі поточних умов чи даних. Це дозволяє визначати, як процес повинен змінювати свій стан в залежності від різних умов або подій, що відбуваються в системі. Також діаграми станів часто використовуються як сховища даних, якщо для повноцінного функціонування не потрібно зберігати великі масиви даних, таким чином вони замінюють звичайні бази даних.

3. Дашборди (Dashboards): Corezoid надає можливість створення дашбордів для візуалізації ключової інформації та аналізу ефективності процесів. Дашборди можуть відображати дані про виконання процесів, статуси сутностей, аналітику продуктивності тощо, що допомагає командам зрозуміти та вдосконалювати їхню бізнес-логіку.

4. Папки (Folders): звичайні директорії, які можуть містити в собі всі перераховані компоненти і допомагають зробити архітектуру проектів чистішою та зрозумілішою.

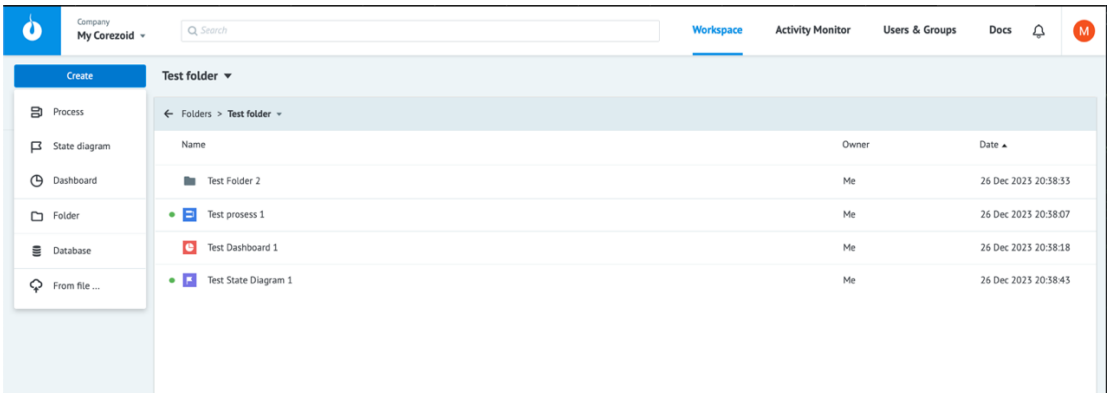


Рисунок 1.3 – Робочий простір в Corezoid

1.6 Ноди як основний інструмент програмування

Як зазначалося раніше, в Корезоїді основним “будівельним матеріалом” для створення процесів є ноди, або як їх називають в класичній теорії автоматів – автомати, які видозмінюють інформацію, яка проходить через них. Зокрема набір даних вузлів в Corezoid є обмеженим, і варіюється в залежності від того, яка структура створюється – процес чи діаграма станів.

Зокрема опис всіх доступних нод, та їх властивостей продемонстрований в таблиці нижче (табл. 1.4) [15].

Таблиця 1.4 – Опис доступних вузлів (нод)

Назва ноди	**Де доступна**	**Опис**
Start	Diagrams, Processes	Обов’язковий вузол, який може бути тільки в одному екземплярі в кожному процесі чи діаграмі станів, помічає початок процесу, та місце з якого заявка починає свій рух по ньому.
Condition	Diagrams, Processes	Вузол «Condition» функціонує як оператор «if» і дає змогу оцінювати параметри заявки та спрямовувати їх потік на основі конкретних вимог, встановлених користувачем. Вузол Condition може складатися з одного або кількох блоків Condition; кожен блок містить одну або більше умов і одне вихідне ребро, яке веде до кінцевого вузла.

Продовження таблиці 1.4

Code	Diagrams, Processes	Вузол «Code» забезпечує зручний спосіб додавання користувацьких функцій до процесу, дозволяючи отримувати доступ, змінювати та створювати параметри заявки за допомогою JavaScript або Erlang.
Copy Task	Diagrams, Processes	Вузол «Copy Task» дозволяє надсилати копію завдання іншому процесу. Коли використовується цей вузол, завдання фактично дублюється: одна копія виконується в поточному процесі, а нова копія передається іншому процесу.
Sum	Processes	Вузол «Sum» виконує кілька функцій: розраховує сумарне значення заданих параметрів проходження завдань і зберігає накопичені результати; формує статистику та відстежує певні параметри;
Modify Task	Diagrams, Processes	Вузол «Modify Task» дозволяє змінювати заявку, що розташована в іншому процесі, додаючи або змінюючи її параметри, не впливаючи на робочий процес вихідного процесу.
Delay	Diagrams, Processes	Вузол «Delay» дозволяє ввести часову затримку у робочий процес, вказуючи, як довго заявка має чекати перед переходом до наступного вузла. Таким чином, можна запланувати відкладення виконання завдання на певний час, який можна вказати в секундах, хвилинах, годинах і днях.
API Call	Processes	Вузол «API Call» — це потужний інструмент, який дозволяє надсилати запити до будь-якого доступного API. Вузол забезпечує гнучкий і ефективний спосіб взаємодії із зовнішніми системами та інтегрувати їх у процеси.
GIT Call	Processes	За допомогою вузла «Git Call» можна легко отримати та виконати код із будь-якого сховища Git, будь то GitHub чи будь-яка інша платформа.

Продовження таблиці 1.4

Database Call	Processes	Вузол «Database Call» дозволяє писати та виконувати запити SQL безпосередньо в Corezoid, що спрощує процеси керування та доступу до інформації у базах даних.
Waiting for Callback	Processes	Вузол «Waiting for Callback» призупиняє обробку завдання до отримання відповіді. Вузол зазвичай використовується, коли процес повинен чекати дії від користувача. Наприклад, чат-бот очікує на введення користувача, перш ніж згенерувати відповідь. Вузол «Waiting for Callback» подібний до вузла «Set State» у тому, що коли завдання потрапляє у вузол, воно не може рухатися далі, доки воно не буде змінено якимось чином або не закінчиться. Однак, на відміну від вузла «Set State», для вузла «Waiting for Callback» немає умов. Щойно завдання буде змінено, воно вийде з вузла, незважаючи на зміни.
Call a Process	Processes	Вузол «Call a Process» пропонує потужні засоби для спрощення та оптимізації процесів, вибравши потрібний процес із каталогу або вказавши його ідентифікатор, можна надіслати завдання цьому процесу та отримати результат назад до вузла «Call a Process».
Reply to Process	Processes	Вузол «Reply to Process» служить для надсилання даних назад від процесу, викликаного вузлом «Call a Process».
Queue	Diagrams, Processes	Вузол «Queue» дозволяє тимчасово утримувати завдання, доки система не буде готова до його обробки. Він зазвичай використовується, коли завдання вимагають ручного втручання, наприклад, коли працівнику потрібно виконати певні дії для обробки замовлення, наприклад зв’язатися з клієнтом або підготувати замовлення до відправлення.

Продовження таблиці 1.4

Get from Queue	Processes	Вузол «Get from Queue» дає змогу отримувати завдання з будь-якого вузла «Queue».
Set Params	Diagrams, Processes	Вузол «Set Params» пропонує простий і ефективний спосіб роботи зі змінними заявки. Використовуючи цей вузол, можна: - створити нові змінні; - очистити імена змінних; - виконувати операції зі змінними.
Set State	Diagrams	Вузол «Set State» призначений для зберігання заявок і керуванням їх потоком в діаграму стану. Це дозволяє зберігати завдання в межах вузла, доки не буде виконано певну умову, а потім направляти його для переходу до наступного вузла, пов'язаного з цією умовою. У певному сенсі вузол «Set State» в діаграмі стану поєднує в собі функціональні можливості вузлів «Conditions» та «Waiting for Callback», які є в процесах.
End	Diagrams, Processes	Вузол «End» є кінцевою точкою процесу чи діаграми стану для всіх заявок, створених вручну та надісланих із зовнішніх джерел. Кінцевий вузол не має вихідних ребер, оскільки він є кінцевим пунктом призначення для завдань. Існує візуальне розділення кінцевого вузла на Error Success, яке не несе ніякого програмного змісту.

1.7 Побудова логіки та інтерфейс розробки

Для наглядності принципів побудови процесів в Corezoid та візуальної демонстрації самого інтерфейсу розробки було створено невеличкий процес (рис. 1.4), який складається із наступних нод:

1. Start.

2. Set Parameters, із створенням змінної end зі значенням false (рис. 1.5.а).

3. Condition, яка перевіряє чи є значення створеного поля true, і відправляє заявку в кінцеву ноду успіху (6), якщо це так, або маршрутизує дані далі.

4. Copy Task, копіює дані заявки до іншого процесу, налаштування ноди зображені на рисунку 1.5.б.

5. Code, яка перезаписує значення поля end на true (рис. 1.5.в) та повторно відправляє до вузлу перевірки.

6. End, із візуальним представленням успіху.

7. Condition, в згорнутому стані, яка створена в цілях обробки помилок в ноді коду, якщо такі будуть.

8. End, із візуальним представленням помилки.

9. Delay, у звернутому стані.

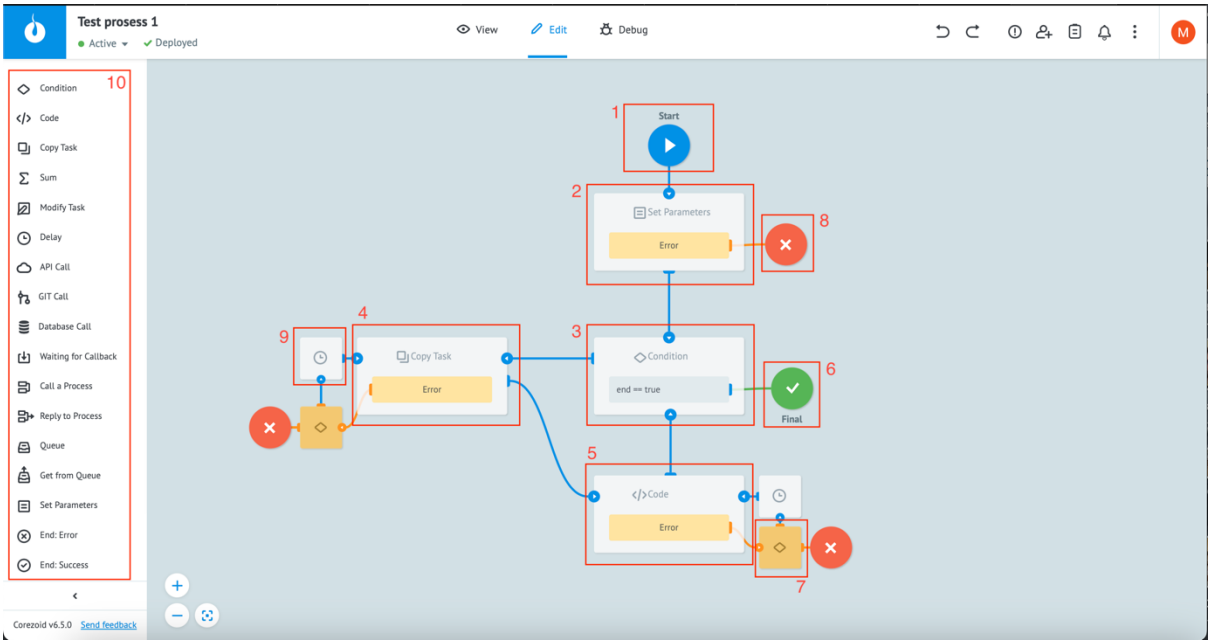
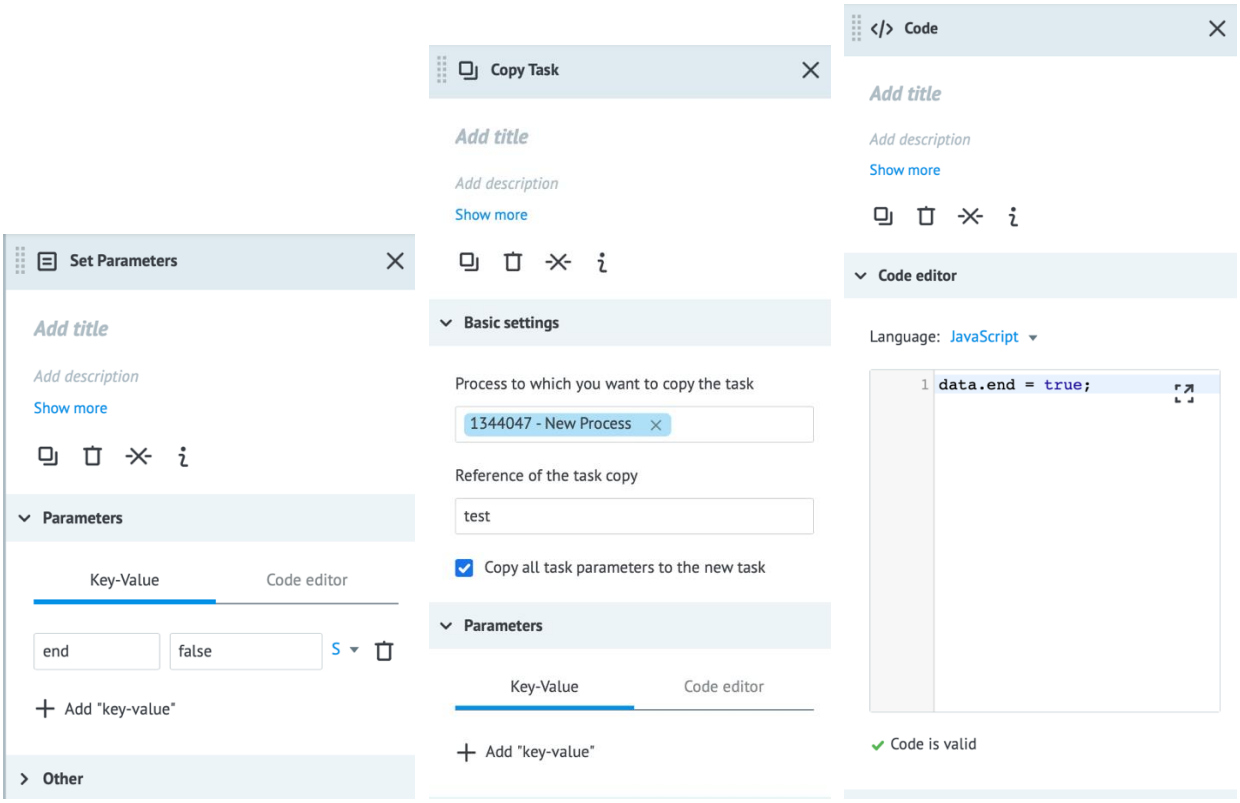


Рисунок 1.4 – Інтерфейс створення процесу

Також на рисунку 1.4 під цифрою 10 можна помітити палітру всіх нод, звідки власне вони й перетягуються на робочий простір.



а) Set Parameters б) Copy Task в) Code

Рисунок 1.5 – Конфігурації вузлів

Із логіки, яку описує процес вище (рис. 1.4), виходить наступне – коли в процес попадає заявка в ній створюється нове поле зі значенням false, після чого ця заявка рухається до ноди перевірки створеного поля і через не виконання умов потрапляє на копіювання даних в інший процес, після чого значення поля змінюється на true і повторно відправляється на перевірку, успішність якої перенаправляє заявку в кінцеву успішну ноду.

Перевіримо коректність роботи процесу, для цього створимо заявку вручну через режим View із пустим тілом даних (рис. 1.6).

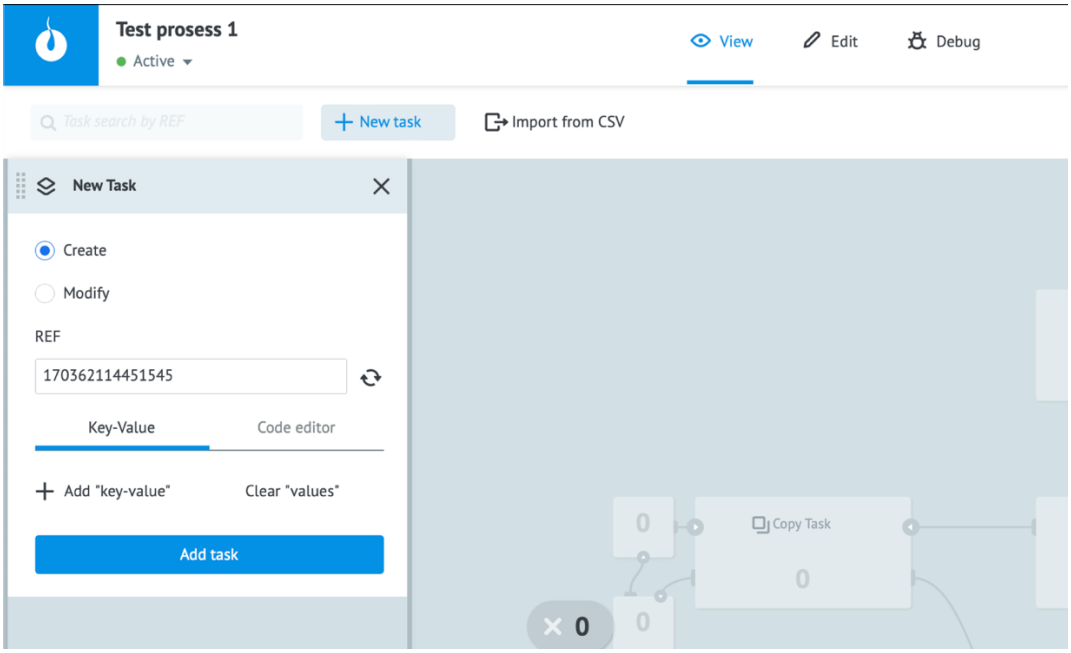


Рисунок 1.6 – Створення заявку вручну

Після створення заявки можна помітити, що в кінцевій успішній ноді збільшився лічильник успішних заявок, а розгорнувши заявку можна помітити, що її дані успішно були змінені (рис 1.7). При цьому старі дані були успішно скопійовані в інший процес (рис 1.8), це свідчить про те, що все працює коректно.

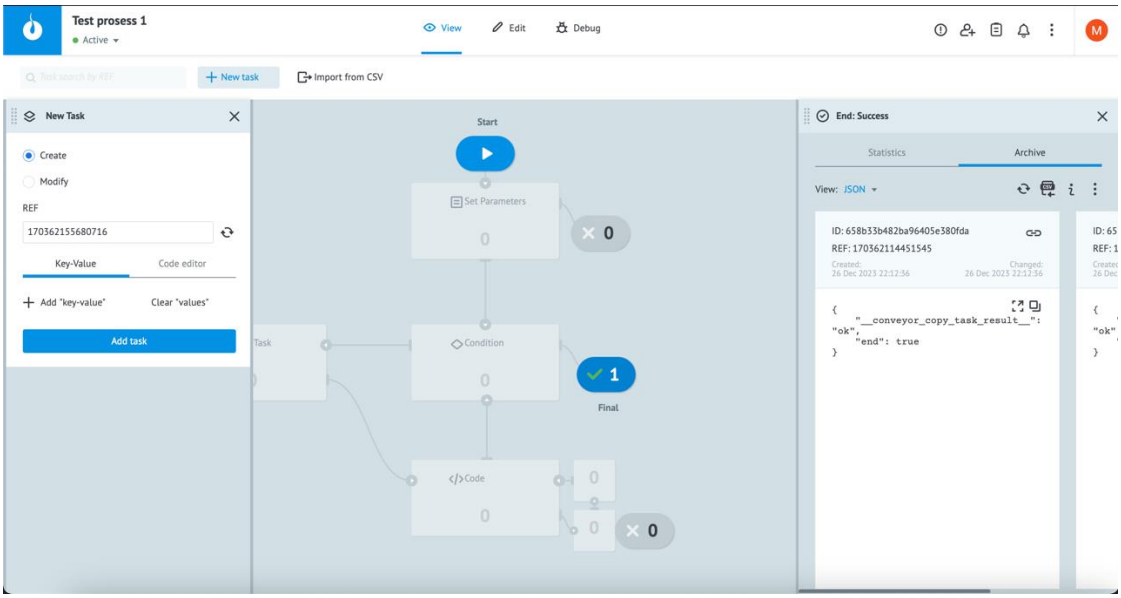


Рисунок 1.7 – Результат роботи основного процесу

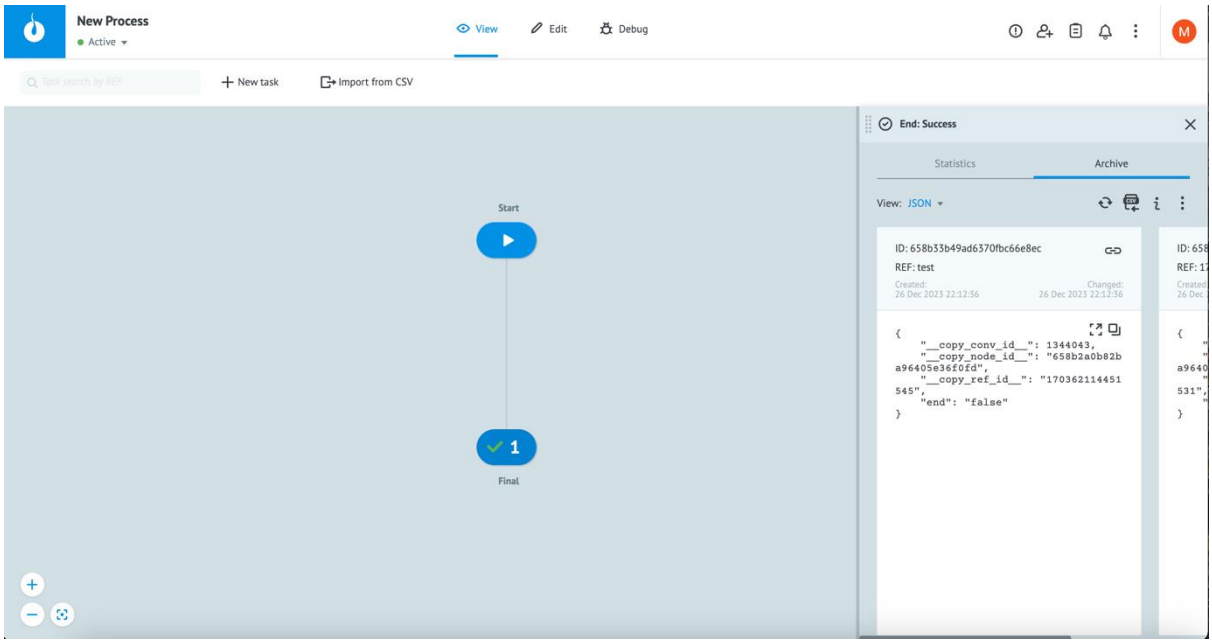


Рисунок 1.8 – Результат копіювання даних в побічний процес

Із проведеного аналізу видно що Корезоїд це непогане рішення для створення невеличких процесів, в яких немає необхідності в громіздкій back-end частині, а сама розробка значно швидша та простіша, в порівнянні із традиційними підходами.

1.8 Висновки до розділу 1

В даному розділі було розглянуто поняття “чат-бот” та принципи їх роботи, проведено аналіз існуючих підходів до класифікацій та створено узагальнену класифікацію, згідно якої ботів можна систематизувати за призначенням, цільовою аудиторією, наявним інтерфейсом та принципом роботи.

Було проведено аналіз найвідоміших месенджерів, які дають можливість розгортати на своїй платформі чат-ботів, згідно якого найоптимальнішим месенджером для інтеграції є Telegram, оскільки він є повністю безкоштовним, має широкий спектр інтерактивного функціоналу, і детально описану документацію щодо імплементації із використанням Telegram Bot API.

Також було розглянуто Corezoid, як хмарне середовище розробки та підтримки бізнес-процесів різної складності, наведено його особливості та складові. Окрема увага була приділена набору доступних нод та власне створенню процесів, в ході чого було на прикладі продемонстровано інтерфейс та процедуру розробки.

РОЗДІЛ 2 ОПИС ТА ПРИНЦИПИ ПОБУДОВИ СИСТЕМ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

2.1 Поняття «Системи підтримки прийняття рішень»

Система підтримки прийняття рішень (СППР) - це інтелектуальна платформа, яка шляхом аналізу важливої інформації, зібраної з віртуального сховища даних, сприяє формуванню рекомендацій [16]. Вона оперує на основі аналізу користувацької поведінки в мережі чи експертних знань у конкретній області. Основне завдання цієї системи полягає в тому, щоб надати користувачу інформацію про товар або послугу, яка може зацікавити саме його. Часто рекомендаційні системи використовуються в сфері торгівлі.

СППР застосовуються в різних сферах людської діяльності. Ми зустрічаємося з ними у соціальних мережах (Facebook, Twitter), музичних сервісах (Spotify, Apple Music), онлайн кінотеатрах (Netflix, MEGOGO), інтернет-магазинах (Rozetka.ua, Amazon, eBay) і багатьох інших.

Компанії використовують СППР або рекомендаційні системи, щоб збільшити свій прибуток через поліпшення співпраці з клієнтами, надаючи їм відповідні рекомендації. Рекомендації сприяють прискоренню пошуку та допомагають користувачам отримати пропозиції щодо товарів та послуг, які вони, можливо, не помітили б самі. Іншими словами, системи підтримки прийняття рішень допомагають у навігації через великий обсяг контенту (наприклад, за даними 2019 року, на YouTube щохвилинно завантажується 500 годин відео [17]). Також компанії можуть надсилати персоналізовані пропозиції по електронній пошті, що сприяє збереженню клієнтської бази. Ці дії стимулюють клієнтів частіше користуватися послугами компанії, роблячи її більш конкурентоспроможною на ринку.

Різні означення та конструкції систем підтримки прийняття рішень описані в [18], в [19] загальна конструкція пристосовується до ситуації, що

розглядається і має наступний вигляд: маючи набір користувачів *U*, набір предметів *I* та відомі оцінки (*R*), які відображають уподобання користувачів до певних предметів, задачею є прогнозування оцінок для нової пари (*u', j'*), де *u'* належить до користувачів зі множини *U*, *j'* - до предметів з множини *I*, при умові, що оцінка *ui'*, *j'* для цієї пари ще не відома та відсутня серед вже існуючих оцінок R.

2.2 Класифікація та принципи побудови рекомендаційних систем

В ході дослідження рекомендаційних систем було виявлено певний набір харектеристик, які їм притаманні [16]:

- тип рекомендованих об'єктів: системи можуть рекомендувати різні види об'єктів, такі як товари, послуги, контент тощо;
- ціль рекомендації: вони мають різні цілі - можуть стимулювати покупки, підвищувати задоволення користувачів або надавати інформацію;
- контекст рекомендації: рекомендації можуть базуватися на конкретних обставинах, оточенні або певних умовах;
- джерело рекомендації: це вказує на походження інформації, на основі якої система робить рекомендації;
- рівень персоналізації: різні системи можуть мати високий чи низький рівень персоналізації залежно від того, наскільки вони враховують індивідуальні уподобання користувачів;
- прозорість: це вказує на ступінь, до якої система може пояснити чи виправдати свої рекомендації;
- використовувані алгоритми: різні алгоритми та методи можуть бути використані для формування рекомендацій.

Ці риси вказують на різноманітність підходів та функцій цих систем.

Якщо ж говорити про побудову систем підтримки прийняття рішень, то тут також є безліч підходів, зокрема варто виділити наступні [18]:

- системи, що базуються на колаборативній фільтрації, використовують схожість між користувачами для передбачення їхніх виборів;
- системи, що базуються на змісті, порівнюють властивості об'єктів з вподобаннями користувачів;
- системи, що використовують знання, працюють на основі визначених правил і вимог;
- демографічні системи групують користувачів та надають спільні рекомендації;
- системи, які оцінюють корисність, розраховують вигоду об'єктів для користувачів за певними формулами;
- гібридні системи поєднують кілька різних підходів для надання рекомендацій.

2.3 Рекомендаційні системи, що базуються на колаборативній фільтрації

В ході аналізу підходів до побудови рекомендаційних систем, початковою ідеєю було створення гібридної системи, яка б включала в себе колаборативну фільтрацію та фільтрацію на основі вмісту. Проте, при більш детальному розгляді кінцевої мети роботи, ідею із використанням фільтрації на основі вмісту було відкинуто, через її непрактичність в певних кейсах (коли вибіркові дисципліни за один семестр неможливо сегментувати принаймі на дві групи) роботи систему, і основний акцент було прийнято зробити саме на колаборативній фільтрації. Основні принципи якої описані нижче.

Рекомендаційні системи, що ґрунтуються на колаборативній фільтрації, прогнозують користувачеві вибір предметів у системі, використовуючи його попередню взаємодію з цими об'єктами. Це може включати оцінки за певною шкалою чи просто дані про використання конкретних предметів. При цьому система робить рекомендації, враховуючи також дії інших користувачів у цій системі.

Основу колаборативної фільтрації становить матриця, де користувачі розташовані вздовж однієї вісі, а предмети - вздовж іншої. В цій матриці знаходиться інформація про те, як користувач оцінив певні предмети. Якщо немає оцінки, це означає, що користувач ще не взаємодіяв з даним предметом. Основна мета такої системи - заповнити ці пусті комірки.

Існують два підходи до створення таких систем [20]:

- на основі користувачів;
- на основі предметів.

Наведемо принцип їх роботи.

Метод колаборативної фільтрації на основі користувачів полягає у наступному процесі: спочатку необхідно відібрати n користувачів, найбільш схожих на основного користувача, і на основі їхніх оцінок здійснювати прогноз. Цей підхід можна наглядно проілюструвати за допомогою таблиці 2.1.

Таблиця 2.1 – Приклад колаборативної фільтрації на основі користувачів

	Предмет 1	Предмет 2	Предмет 3	Предмет 4
Користувач 1	**76**	**75**	**95**	**74**
Користувач 2	62	93	70	96
Користувач 3	94	88	70	70
Користувач 4	**76**	**?**	**95**	**70**

У таблиці 2.1 представлена матриця оцінок між чотирма користувачами та чотирма предметами. На цьому фоні ми маємо завдання: здійснити прогноз оцінки, яку Користувач 1 міг би отримати із Предмету 2. З огляду на отримані оцінки, видно, що Користувач 4 найбільше схожий на Користувача 1. Тому за допомогою цієї схожості можна припустити, що шукане значення, ймовірно, дорівнює приблизно 75.

Метод колаборативної фільтрації на основі предметів діє наступним чином: спочатку потрібно відібрати n предметів, які найбільше схожі на ключовий предмет, і на основі їхніх оцінок робити прогноз. Цей підхід можна наглядно показати через таблицю 2.2.

Таблиця 2.2 – Приклад колаборативної фільтрації на основі предметів

	Предмет 1	Предмет 2	Предмет 3	Предмет 4
Користувач 1	76	**75**	95	**74**
Користувач 2	62	**93**	70	**96**
Користувач 3	94	**88**	70	**85**
Користувач 4	76	**?**	95	**70**

З таблиці 2.2 бачимо, що на Предмет 2 найбільше схожий Предмет 4, тому можна зробити прогноз, що шукане значення приблизно дорівнює 70.

Наступним питання, що постає – як точно виміряти подібність між користувачами чи предметами, і як в подальшому виконати передбачення. Для вирішення цього завдання існують дві групи алгоритмів [20]:

– алгоритми, що базуються на пам'яті;
– алгоритми, що базуються на моделі.

2.4 Колаборативна фільтрація із використанням алгоритмів, що базуються на пам’яті

Алгоритми, що ґрунтуються на пам'яті, функціонують за допомогою обчислення певної міри на основі наявних даних у системі (наприклад, у вигляді матриці оцінок). Оцінки кожного користувача (або оцінки для кожного предмета) можна подати у вигляді векторів, до яких можна застосувати різні метрики подібності. Після визначення відповідної міри схожості можна розрахувати передбачення.

Серед найбільш використовуваних метрик варто зазначити [21, 22]:

- косинусну схожість (формула (2.1)):

$$cosineSimilarity(u_i, u_j) = cos(u_i, u_j) = \frac{\sum_{k=1}^{n} u_{i,k} u_{j,k}}{\sqrt{\sum_{k=1}^{n} u_{i,k}^2} \sqrt{\sum_{k=1}^{n} u_{j,k}^2}}; \quad (2.1)$$

- критерій кореляції Пірсона (формула (2.2)):

$$PearsonSimilarity(u_i, u_j) = \frac{\sum_{k=1}^{n} (u_{i,k_k} - \overline{u_\iota})\,(u_{j,k_k} - \overline{u_j})}{\sqrt{\sum_{k=1}^{n} (u_{i,k} - \overline{u_\iota})^2} \sqrt{\sum_{k=1}^{n} (u_{j,k} - \overline{u_j})^2}}; \quad (2.2)$$

- скориговану косинусну схожість (формула (2.3)):

$$adjustedCosineSimilarity(u_i, u_j) = \frac{\sum_{k=1}^{n} (u_{i_k} - \overline{u_k})\,(u_{j_k} - \overline{u_k})}{\sqrt{\sum_{k=1}^{n} (u_{i,k} - \overline{u_k})^2} \sqrt{\sum_{k=1}^{n} (u_{j,k} - \overline{u_k})^2}}; \quad (2.3)$$

- коефіцієнт Жаккара (формула (2.4)):

$$JaccardSimilarity(u_i, u_j) = \frac{|u_i \cap u_j|}{|u_i \cup u_j|}. \quad (2.4)$$

В наведених формулах (2.1)-(2.4):

- u_i, u_j– вектори оцінок користувачів i та j;
- n – кількість предметів;
- $\overline{u_\iota}, \overline{u_j}$ – середні оцінки користувачів i та j;
- $\overline{u_k}$ – середня оцінка предмета k.

Після отримання міри схожості виникає необхідність розрахувати передбачене значення оцінки. Для цього потрібно вибрати найбільш схожих користувачів серед усієї групи. Це можна здійснити, наприклад, шляхом відбору верхніх n користувачів за ступенем подібності або шляхом встановлення певного порогового значення схожості.

У процесі передбачення оцінки використовуються агрегуючі функції, серед яких найчастіше використовуються [23]:

- середнє значення (формула (2.5)):

$$u_{i,k} = \frac{\sum_{u_j \in U'} u_{j,k}}{|U'|}; \quad (2.5)$$

- середнє зважене (формула (2.6)):

$$u_{i,k} = \frac{\sum_{u_j \in U'} similarity(u_i, u_j) u_{j,k}}{\sum_{u_j \in U'} \left| similarity(u_i, u_j) \right|}; \quad (2.6)$$

- відносне середнє зважене (формула (2.7)):

$$u_{i,k} = \bar{u_i} + \frac{\sum_{u_j \in U'} similarity(u_i, u_j)(u_{j,k} - \overline{u_{j,k}})}{\sum_{u_j \in U'} \left| similarity(u_i, u_j) \right|}. \quad (2.7)$$

В формулах вище (2.5)-(2.7):

- $u_{i,k}$ - оцінка користувача і предмету k;
- U' – множина користувачів, які найбільше схожі на даного;
- u_j – вектор оцінок користувача з U;
- u_i – середня оцінка користувача і серед оцінених ним предметів.

Отримане у результаті значення $u_{i,k}$ і є шуканим прогнозом.

Також до алгоритмів колаборативної фільтрації на основі пам'яті можна віднести алгоритми з родини Slope One. Вони вирізняються своєю простотою та ефективністю [24]. Основу цього методу складає предиктор, який представляє собою середню різницю між оцінками двох предметів [25, 26] (формула (2.8)). Надалі цей предиктор використовується для прогнозування оцінки. Передбачуване значення оцінки обчислюється за формулою [26] (2.9). Для отримання більш точного передбачення також може використовуватися середнє зважене значення [26] (формула (2.10)).

$$dev_{k,m} = \frac{\sum_{u_i \in U_{k,m}} (u_{k,j} - u_{j,m})}{\left| U_{k,m} \right|}, \quad (2.8)$$

$$u_{i,k} = \frac{\sum_{m \in S_i} (dev_{k,m} + u_{i,m})}{|S_i|}, \quad (2.9)$$

$$u_{i,k} = \frac{\sum_{m \in u_i} (dev_{k,m} + u_{i,m}) \left| U_{k,m} \right|}{\sum_{m \in u_i} \left| U_{k,m} \right|}. \quad (2.10)$$

В наведених формулах (2.8)-(2.10):

- $dev_{k,m}$ – предиктор для предмета k на основі предмета m;

- $U_{k,m}$ – множина користувачів, які оцінили як предмет k, так і предмет m;
- u_i – оцінки користувача i;
- S_i – множина предметів, які були оцінені користувачем i та були оцінені будь-яким іншим користувачем з множини спільно з предметом k.

2.5 Колаборативна фільтрація із використанням алгоритмів, що засновані на моделі

У колаборативній фільтрації, заснованій на моделі, використовуються різноманітні алгоритми добування даних та машинного навчання для передбачення оцінок користувачів. Ці алгоритми можна поділити на три основні групи: непараметричні методи (наприклад, метод k-найближчих сусідів), алгоритми факторизації матриць (такі як SVD) та методи глибинного навчання (наприклад, багатошарові нейронні мережі) [27].

Факторизація матриць є одним із найпоширеніших підходів у побудові рекомендаційних систем, заснованих на моделі. Ця техніка полягає у розкладі об'єкта на його базові складові [28].

Початково потрібно виокремити конкретний набір прихованих факторів у предметній області. Нехай цей набір факторів складається з r характеристик предметів. Ці характеристики представляють собою певні властивості предметів. Після цього застосовується один із алгоритмів для розкладання матриці оцінок – X – на дві матриці: матрицю відношень користувачів до цих прихованих факторів (матриця представлення користувачів – U) та матрицю відношень предметів до цих прихованих факторів (матриця представлення предметів – V).

Серед найвідоміших алгоритмів факторизації матриць можна віднести SVD, TruncatedSVD, PCA та NMF. Ці алгоритми дозволяють отримати

наближену рівність X приблизно рівну добутку матриць U на V, що можна виразити формулою (2.11) [26].

$$\begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} \approx \begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mr} \end{pmatrix} \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{r1} & \cdots & x_{rn} \end{pmatrix} \quad (2.11)$$

Після цього, для отримання рекомендацій, тобто заповнення пропусків у початковій матриці X, потрібно перемножити матрицю U на матрицю V. Це призведе до отримання матриці X', яка наблизиться до матриці X і буде містити числа замість пропусків. Однак ці числа не є точними передбаченнями оцінок; це лише певні значення, які тепер можна використовувати для рекомендацій. Наприклад, можна провести інтерполяцію оцінок предметів у певному діапазоні, а потім рекомендувати користувачам ті предмети, де отримане значення перевищує заданий поріг [28].

2.6 Переваги та недоліки рекомендаційних систем, заснованих на колаборативній фільтрації

Проаналізувавши підходи та принципи побудови рекомендаційних систем що ґрунтуються на колаборативній фільтарції, можемо виділити переваги та недоліки використання даного підходу.

1. Переваги:

- забезпечує можливість крос-жанрових рекомендацій;
- не вимагає спеціалізованих знань у конкретній галузі;
- точність рекомендацій підвищується з часом використання;
- для роботи достатньо лише інформації про оцінки;
- можливість використання неявної інформації з взаємодії користувача з системою.

2. Недоліки:

- проблеми при старті для нових користувачів;
- проблеми при старті для нових предметів;

- проблема виділення унікальних користувачів або предметів;
- вплив популярності на рекомендації;
- потреба у великому обсязі даних для точних рекомендацій;
- баланс між стабільністю та гнучкістю системи.

2.7 Висновки до розділу 2

В даному розділі розглянуто поняття рекомендаційної системи, проведено їх класифікацію, та досліджено різновиди. Зокрема було визначено основні характеристики рекомендаційних систем та підходи до їх побудови.

Більш детальна увага була приділена рекомендаційним системам що базуються на колаборативній фільтрації. В рамках їх дослідження було виділено два підходи до побудови таких систем, та на прикладі розглянуто принципи їх роботи.

Також було досліджено дві групи алгоритмів, які використовуються в колаборативній фільтрації, наведено математичний аппарат для кожної із них та виділено переваги та недоліки даного підходу до побудови систем підтримки прийняття рішень.

РОЗДІЛ 3 РОЗГОРТАННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ПО ВИЗНАЧЕННЮ ВИБІРКОВИХ ДИСЦИПЛІН НА БАЗІ ЧАТ-БОТУ

Метою даного розділу є розробка та дослідження функціональної системи підтримки прийняття рішень, на основі колаборативної фільтрації, на базі чат-бота із використанням хмарного середовища розробки Corezoid та технологій Telegram Bot API.

3.1 Постановка задачі

Перш ніж приступати до розробки продукту, варто визначити основні його принципи роботи, та структуру.

Так як основною задачею боту є отримання рекомендацій студентом, а побічною – внесення оцінок із дисциплін викладачем, то це означає, що бот має мати певну рольову модель із різними режимами доступу (таблиця 3.1)

Таблиця 3.1 – Рольова модель

Роль	Доступний функціонал	Режим доступу
Студент	Отримання рекомендацій стосовно вибіркових дисциплін	Вільний доступ
Викладач	Виставлення та модифікація оцінок із доступних йому предметів	Авторизація шляхом введення логіну та паролю

Наступним кроком визначимо які саме модулі потрібні в рекомендаційній системі, та які сховища даних потрібно розгорнути.

Так як Інтерфейсом для боту буде месенджер, то першочергово потрібно розробити Систему обробки повідомлень, за допомогою якої Користувач

зможе спілкуватися із ботом. Також важливим є Логічне Ядро, яке буде відповідати за роутинг користувача по логіці боту.

Якщо ж говорити Сховища, то тут, із зазначених вище ролей можна помітити, що однозначно потрібне Сховище Викладачів, де будуть зберігатися із особисті та авторизаційні дані. Також, оскільки рекомендаційна система, яку побудуємо, буде працювати на базі колаборативної фільтрації із алгоритмом основаним на пам'яті, то потрібно десь зберігати оцінки студентів, для подальшого їх використання.

Також доцільним буде розгортання невеличкого сховища для зберігання дисциплін, які будуть поділені в залежності від семестру, на якому вони викладаються.

І найцікавіше – модулі обробки даних. Тут потрібні два модулі:

- модуль отримання рекомендацій;
- модуль занесення оцінок.

В результаті отримаємо наступну схему рекомендаційного чат-боту (рис. 3.1).

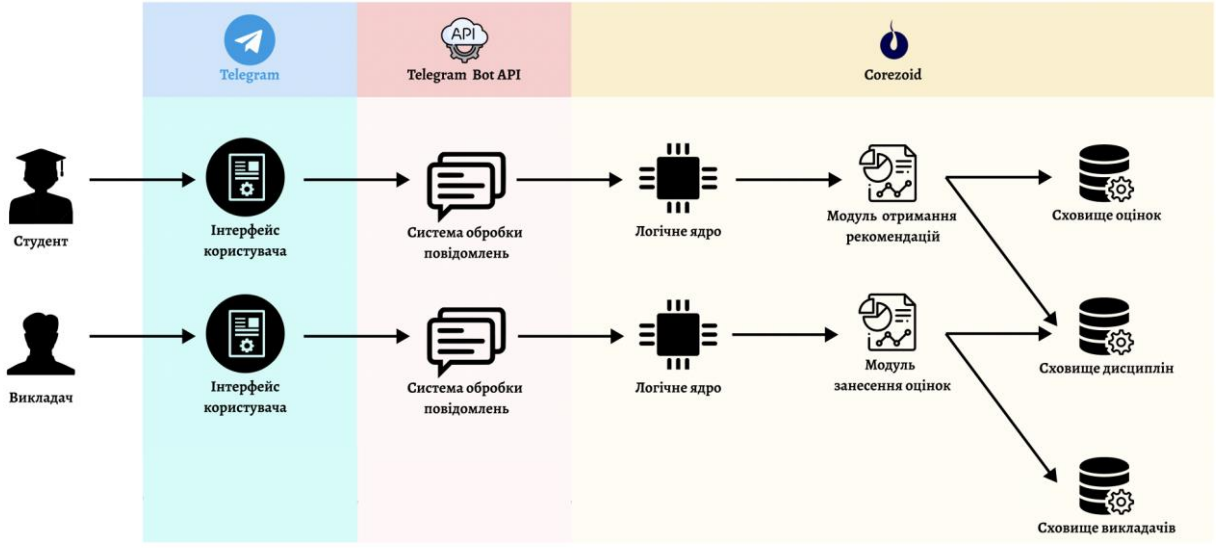


Рисунок 3.1 – Структурна схема чат-боту

Що ж, перейдемо до розробки.

3.2 Розгортання чат-боту

Для створення каналу комунікацій в Telegram скористаємось готовим програмним апаратом від Telegram, а саме телеграм ботом BotFather (рис. 3.2).

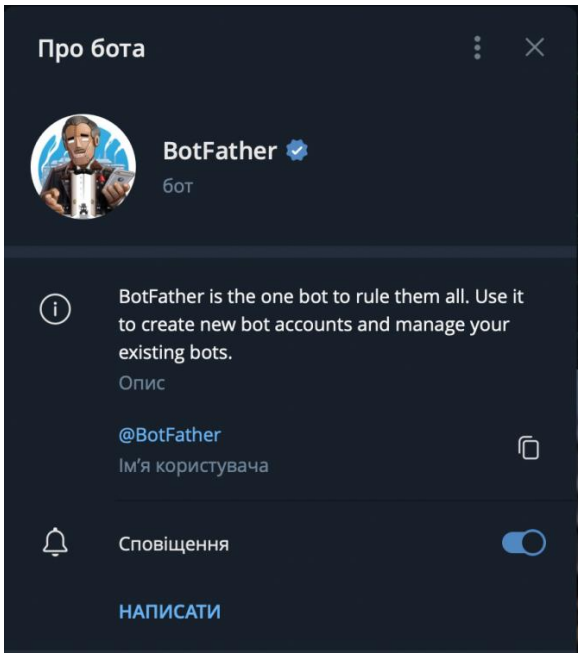


Рисунок 3.2 – Офіційний бот платформи Telegram

BotFather – це офіційний бот платформи Telegram, який дозволяє користувачам створювати, керувати та настроювати власних ботів. За допомогою BotFather користувачі можуть легко створювати ботів з різними функціями, такими як розсилка повідомлень, обробка команд, управління групами та багато іншого.

Для створення достатньо написати боту команду /newbot та слідувати інструкціям, згідно яких потрібно буде вказати ім'я бота, його опис та додатково налаштувати косметичні властивості.

Після створення отримаємо унікальний токен боту (рис. 3.3), який в подальшому будемо використовувати для створення Системи обробки повідомлень.

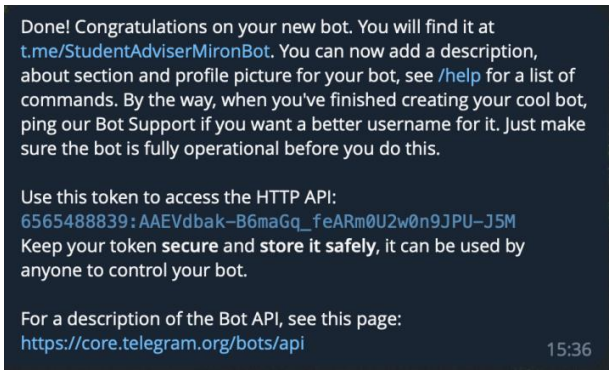


Рисунок 3.4 – Підтвердження успішного створення боту

Тобто на даному етапі отримано унікальний ідентифікатор каналу обміну повідомлень, та контентно продуманий бот (рис. 3.5).

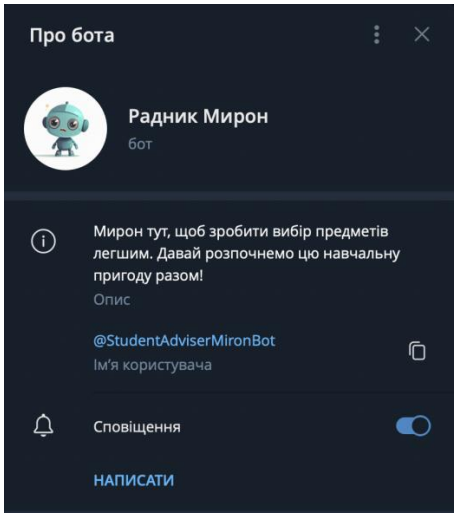


Рисунок 3.5 – Інформація про створений бот

Проте якщо користувач напише зараз боту, то у відповідь нічого не отримає, адже немає ніякої логіки обробки вхідних повідомлень та відправки вихідних, тому перейдемо до неї.

3.3 Розробка системи обробки повідомлень

Після ознайомлення із технічною документацією Telegram Bot API [12] з'ясувалось, що для створення каналу отримання всіх повідомлень,

відправлених в бот, потрібно скористатися методом /setWebhook, який зв'язує бота із кінцевим URL для відловлення повідомлень.

Тому першим етапом розробки системи обробки повідомлень буде створення процесу в Corezoid та прив'язка його до боту.

В результаті чого буде отримано процес, в який приходять всі повідомлення, які надсилаються боту (рис. 3.6). В отриманій інформації також містяться дані про користувача який надіслав повідомлення, які в подальшому будуть в нагоді, при створенні сесії чат-бота.

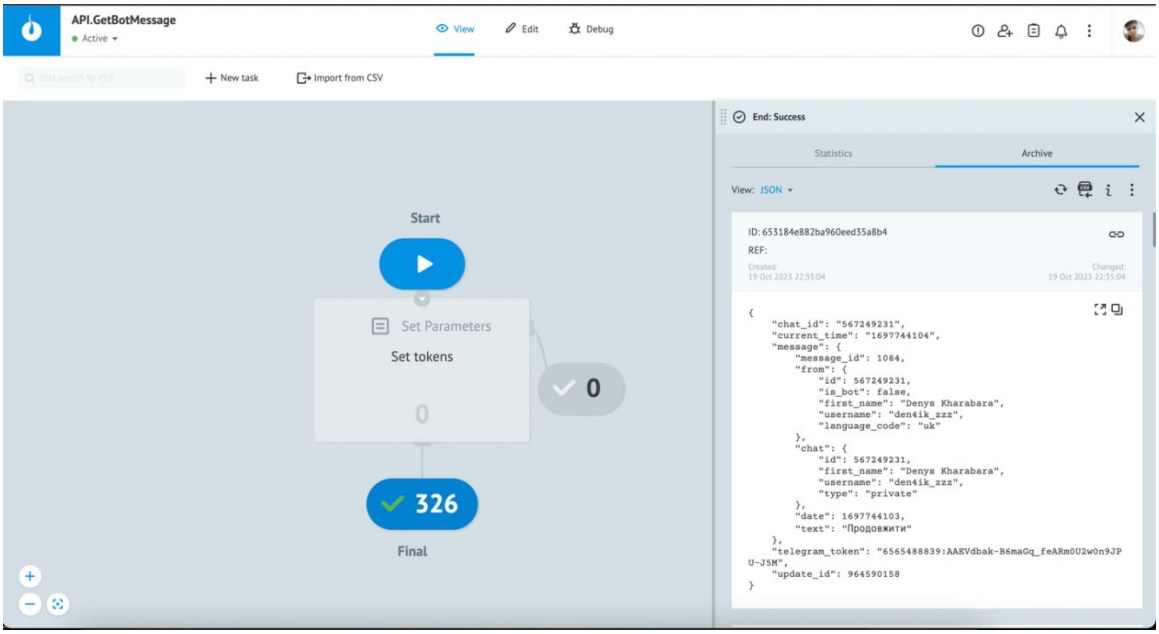


Рисунок 3.6 – Corezoid процес, що відловлює повідомлення із прикладом вхідної заявки

Для створення каналу комунікацій не достатньо одного процесу по отриманню повідомлень, потрібно ще якось їх відправляти, тому для цього створимо ще один процес який буде використовувати метод /sendMessage для відправки POST запитів по відправці повідомлень із наступними параметрами (рис. 3.7):

- chat_id – ід каналу, який є унікальним для кожного користувача, що написав боту;
- text – текст повідомлення;

- parse_mode – режим обробки тексту (HTML\Markdown);
- reply_markup – додатковий медіа інструментарій в форматі JSON, який будемо використовувати для відмалювання клавіатури.

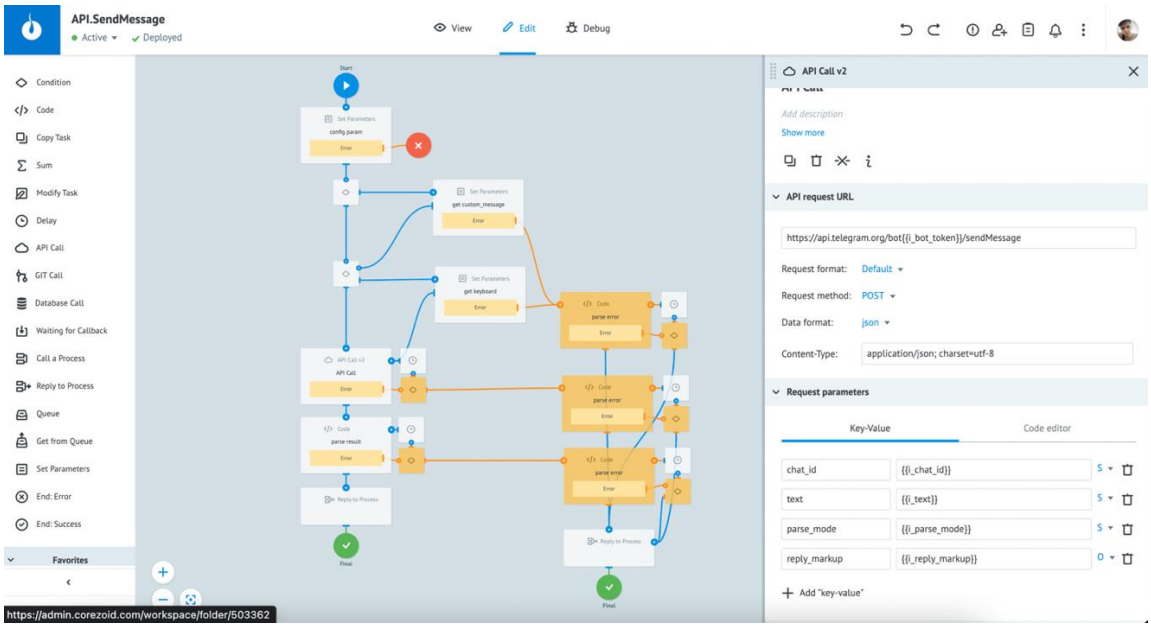


Рисунок 3.7 – Процес відправки повідомлень

Для зручності також було розгорнуто два невеличких сховища із повідомленнями та клавіатурами, які містять в собі заявки із назвами клавіатур\повідомлень та JSONом\текстом які потрібно відправити в запиті, для їх відображення користувачу, це робить процес внесення змін значно простішим та зрозумілішим.

Окрім нод отримання повідомлень та клавіатур, в процесі відправки повідомлень можна помітити й інші, конфігураційні ноди (*parse error*) – які відповідають за обробку помилок при виклику API.

Ну і останнім кроком в розробці системи обробки повідомлень буде створення Сховища сесій (рис. 3.8), яке буде відповідати за збереження поточних сесій користувачів та їх очистку після 15 хв.

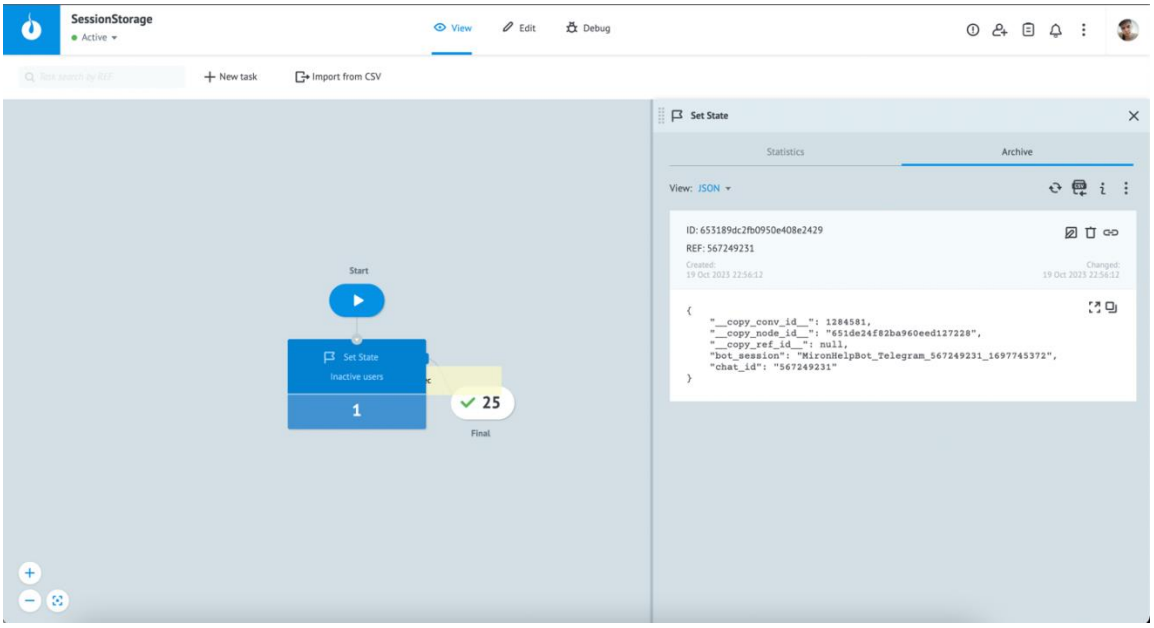


Рисунок 3.8 – Сховища сесій

Зроблено це в цілях безпеки, оскільки телеграм у більшості користувачів встановлений на різних гаджетах, і досить часто ці гаджети лежать без нагляду.

Сесії зберігаються в Сховищі за референсом унікального каналу комунікацій користувача з ботом, про який попередньо уже говорилося та формуються за наступним алгоритмом: `MironHelpBot_(назва месенджера)_(ід каналу комунікації)_(час створення сесії в Unixtime)`.

Така логіка формування і збереження сесії дозволяє надійно та стабільно маршрутизувати користувача по флоу боту, та відправляти його на самий початок після закінчення 15-ти хвилинного інтервалу існування сесії.

3.4 Розробка флоу чат-бота та реалізація Логічного ядра

Перш ніж перейти до розробки Логічного ядра потрібно продумати все флоу, по якому може рухатись користувач, із урахуванням його прав доступу (рис. 3.9).

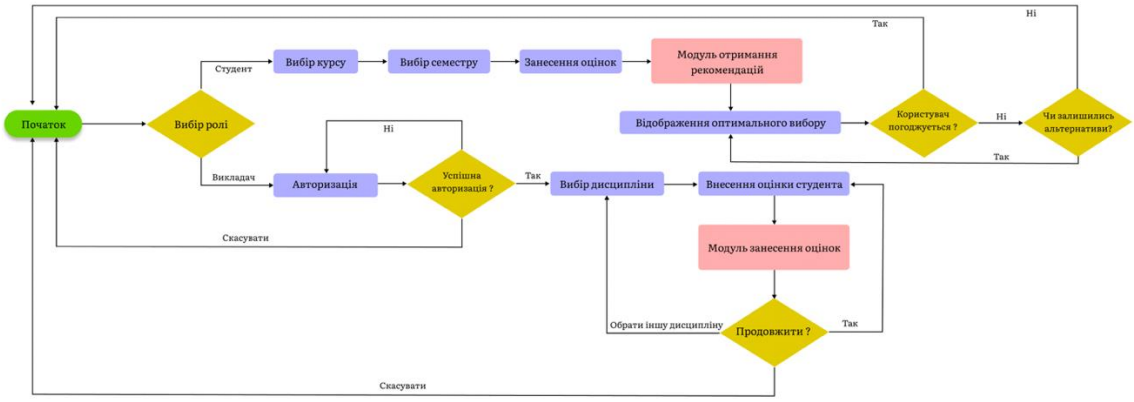


Рисунок 3.9 – Логічна схема флоу бота

На базі сформованої схеми розробимо Логічне ядро, для цього:

1. *Створимо новий процес в Corezoid (рис. 3.10).*

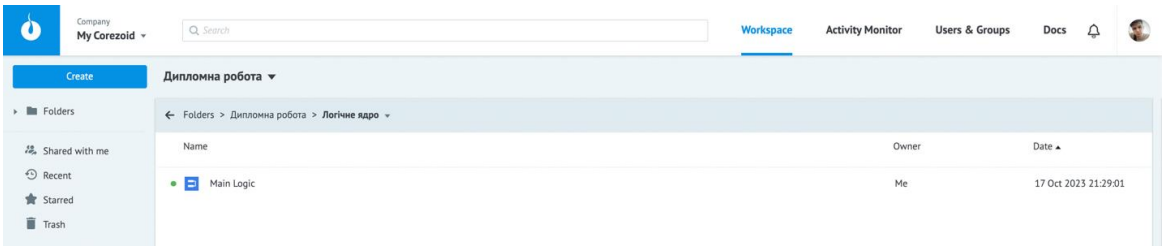


Рисунок 3.10 – Створений процес, в якому буде розміщене логічне ядро

2. *Наповнимо сховища Повідомлень та Клавіатур необхідними даними.*

Так як попередньо продуманий концепт боту передбачає створення його унікального тону спілкування, то повідомлення були сформовані згідно єдиного, не формального стилю.

3. *Створимо кластер маршрутизації.*

Під кластером маршрутизації розуміється набір нод, які відповідатимуть за відправку повідомлення користувачу, очікування його реакції та подальшу маршрутизацію. Фактично це уніфікований набір нод, які будуть повторятися

впродовж всього вказаного флоу, і єдине що у них буде відмінне, це контент для відправки та умови маршрутизації.

Для створення даного кластеру використаємо ноду CallProcess, яка буде викликати процес по відправці повідомлень, який був створений в пункті 3.3. Після неї буде нода Waiting for Callback, в ній заявка буде зависати до моменту надходження повідомлення від користувача в процес API.GetBotMessage, який також був створений в пункті 3.3. Зокрема в даному процесі зробимо доопрацювання, в рамках якого, після отримання повідомлення будемо перевіряти чи є у даного користувача активна сесія в боті, тоді:

1) якщо сесії немає, то створимо її і направимо користувача на початок флоу;

2) якщо сесія є, то відправимо в ноду Waiting for Callback, яка знаходиться в Логічному ядрі, інформацію, яку вказав користувач боту.

І останньою, третьою нодою в кластері маршрутизації буде Condition Node, яка фактично є візуальним представленням звичного в програмуванні оператора if…else.

В результаті отримаємо процес із логічних зв’язків повідомлень та реакцій на них, да додаткових нод збереження даних, що вводяться користувачем, для подальшого їх використання. Спрощену версію даного процесу продемонстровано на рисунку 3.11.

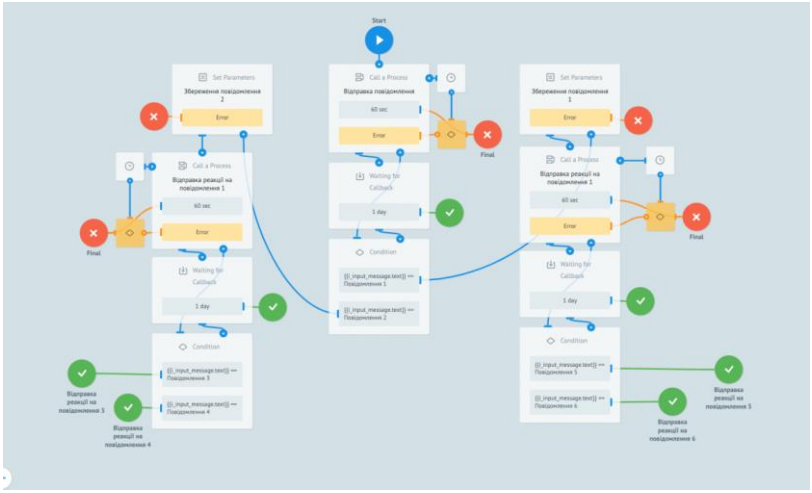


Рисунок 3.12 – Спрощена версія логічного ядра із трьома кластерами маршрутизації

3.5 Наповнення сховищ

Як зазначалось раніше, для функціонування бота та рекомендаційної системи потрібно розгорнути три сховища, тому для цього створимо три State diagram в Corezoid, які й будуть нашими сховищами:

1. *Сховище дисциплін.*

В сховищі дисциплін будуть зберігатися всі дисципліни які викладаються, поділені по семестрах та курсах. Зберігатися вони будуть в форматі заявок, як продемонстровано на рисунку 3.13, де референс це курс та семестр викладання. А дані, які містяться в заявках в форматі JSON, це масив предметів на даний курс та семестр із вказанням назви дисципліни(*name*), її внутрішнього коду (*code*) та ознаки вибірковості (*is_selective*).

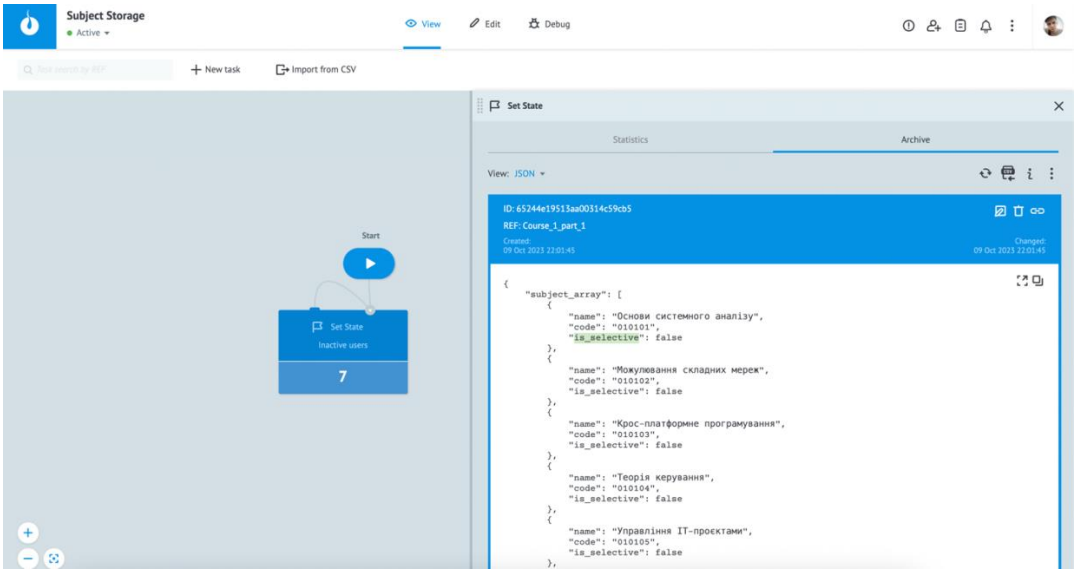


Рисунок 3.13 – Сховища дисциплін

Варто зазначити що код дисципліни має наступний формат – ККССДД, де КК- це порядковий номер курсу, СС- номер семестру, ДД- порядковий номер дисципліни.

Даний алгоритм дає змогу формувати унікальні та зрозумілі для визначення коди дисциплін.

2. *Сховище викладачів.*

В сховищі дисциплін принцип збереження даних такий ж як і в попереднього, тільки тут за референс заявки береться логін викладача, а дані, які там зберігаються:

- login – логін викладача;
- name – ім’я викладача;
- password – пароль для авторизації;
- subject – масив доступних для виставлення оцінок дисциплін, кожний об'єкт якого містить назву дисципліни та її код (рис. 3.14).

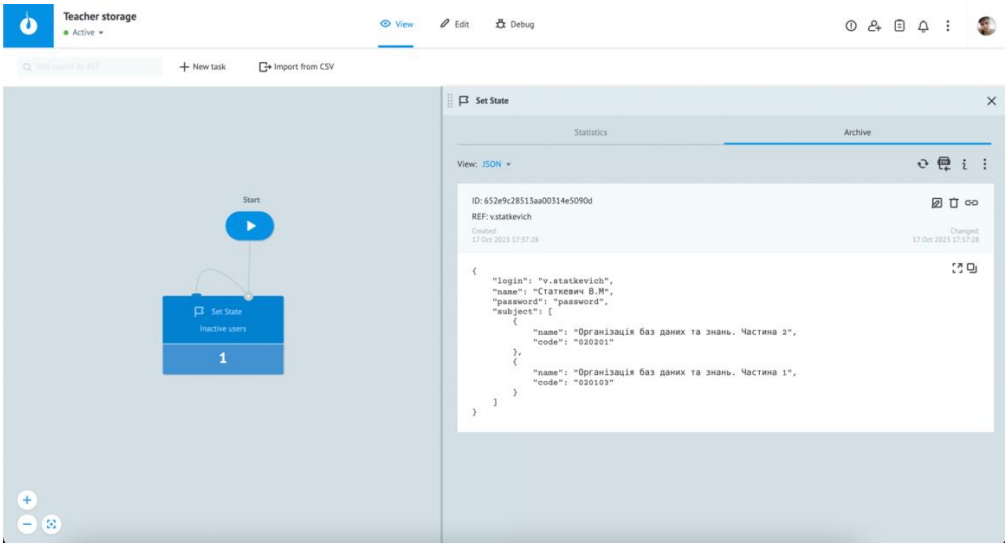


Рисунок 3.14 – Сховище викладачів

3. Сховище оцінок.

В сховищі оцінок зберігаються заявки із оцінками всіх студентів в розрізі курсу та семестру. Всі оцінки зберігаються в масиві *marks_arr*, кожний елемент якого містить номер залікової книжки студента (*id*) та масив оцінок із дисциплін (marks), кожна дисципліна в свою чергу має наступні атрибути:

- subject – код дисципліни;
- mark – оцінка;
- additional – ознака вибірковості дисципліни.

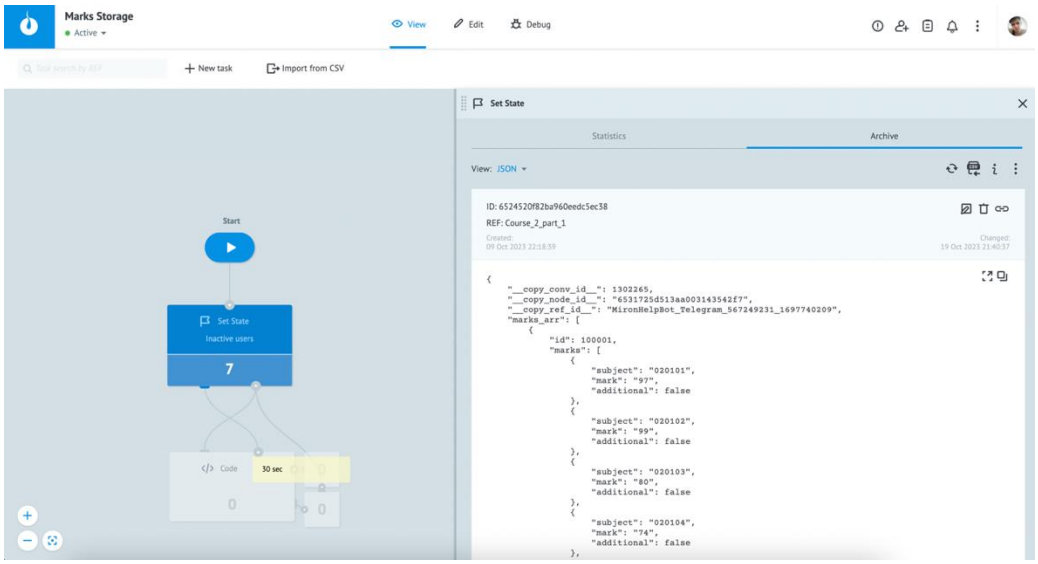


Рисунок 3.15 – Сховище оцінок

Візуальне представлення сховища продемонстроване на рисунку 3.15, а приклад даних, які в ньому зберігаються, наведений нижче.


```
"marks_arr": [
        {
            "id": 100001,
            "marks": [
                {
                    "subject": "020101",
                    "mark": "97",
                    "additional": false
                },
                {
                    "subject": "020102",
                    "mark": "99",
                    "additional": false
                },
                {
                    "subject": "020108",
                    "mark": "76",
                    "additional": true
                }
            ]
        },
        {
            "id": 100002,
            "marks": [
                {
```

```
                    "subject": "020101",
                    "mark": "71",
                    "additional": false
                },
                {
                    "subject": "020102",
                    "mark": "87",
                    "additional": false
                },
                {
                    "subject": "020108",
                    "mark": "77",
                    "additional": true
                }
            ]
        },
]
```


3.6 Розробка модуля внесення оцінок

Для початку розгорнемо новий процес в Corezoid із зазначенням обов’язкових вхідних даних:

- i_student – номер залікової книжки студента;
- i_subject – дисципліна;
- i_mark – оцінка.

В рамках логіки даного модуля можуть зустрічатися дві ситуації:

1) внесення оцінки по уже існуючому студенту;
2) внесення оцінки по новому студенту.

Тому задля задоволення повноцінних потреб був розроблений наступний процес (рис. 3.16).

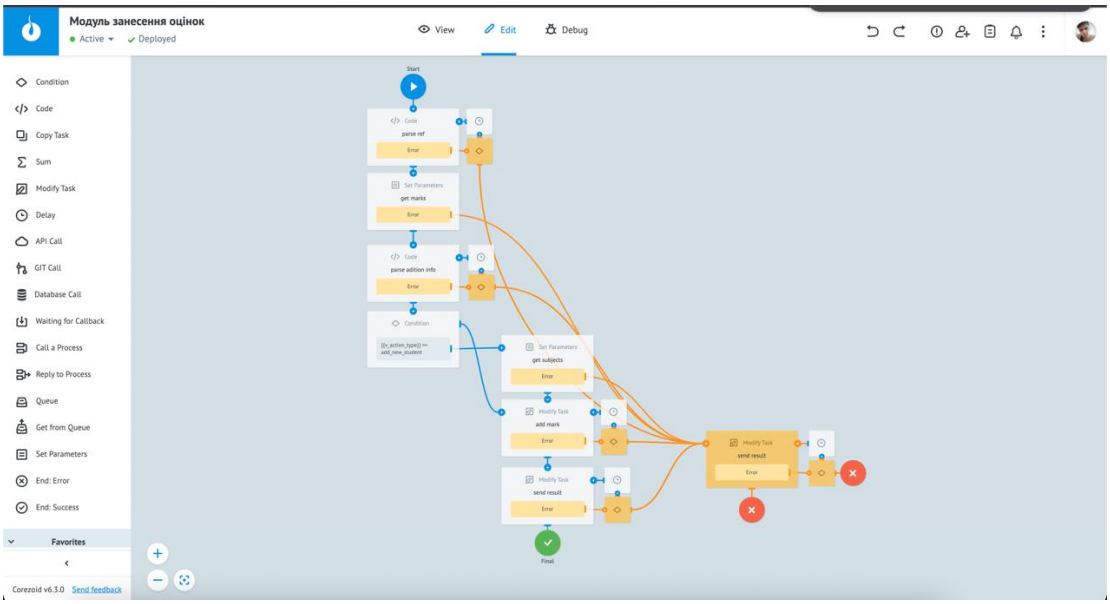


Рисунок 3.16 – Модуль занесення оцінок

В рамках процесу:

1) перевіряємо чи є дані по поточному семестрі в сховищі за певний семестр;

2.1) якщо дані є, то змінюємо оцінку із дисципліни на актуальну;

2.2) якщо даних немає, то в масив оцінок додаємо поточного студента із усіма дисциплінами, після чого проставляємо оцінку із поточної предмету.

3.7 Розробка модуля надання рекомендацій

Перш ніж приступати до розробки, розглянемо структурну схему, за якою має працювати модуль (рис. 3.17).

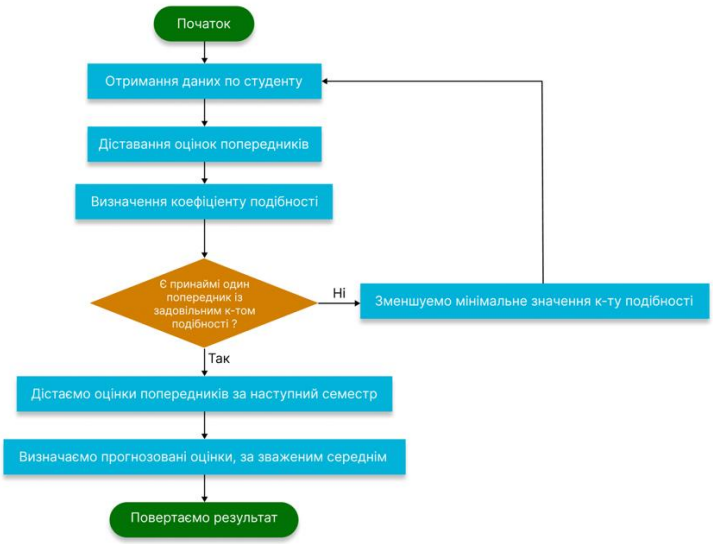


Рисунок 3.17 – Структурна схема процесу отримання рекомендацій

Дивлячись на схему можна помітити, що регулюючим фактором точності надання рекомендацій в системі є мінімальний коефіцієнт подібності, який набуває початкового значення 0.995, і при кожному наступному кроці зменшується на 0.005. Експериментальним методом було виявлено, що саме такий алгоритм регулювання даного коефіцієнту дає оптимальну точність та швидкість обчислення.

Якщо говорити про обчислення то головну роль у них будуть виконувати дві функції, написані на JavaScript та імплементовані в ноди коду в Corezoid:

1. *Функція визначення коефіцієнтів подібності.*

В основі даної функції лежить формула косинусу подібності, згідно якої потрібно знайти суму всіх добутків оцінок поточного студента та студента-попередника, після чого розділити її на добуток коренів квадратних сумованих оцінок студентів, піднесених до квадрату.

Повний код функції наведено нижче.

```
function calc_coef_similar (std1_marks, std2_marks){
    let main_std_marks = std1_marks.filter(el => el.additional == false);
```

```
        let upper = 0;
        let lower_1 = 0;
        let lower_2 = 0;
        main_std_marks.forEach(m => {
            upper = upper + parseInt(m.mark) *
parseInt(std2_marks.find(el => el.subject == m.subject).mark);
            lower_1 = lower_1 + (parseInt(m.mark) *
parseInt(m.mark));
            lower_2 = lower_2 + (parseInt(std2_marks.find(el =>
el.subject == m.subject).mark) * parseInt(std2_marks.find(el =>
el.subject == m.subject).mark));
        });
        let result =
upper/(Math.sqrt(lower_1)*Math.sqrt(lower_2));
        return result;
    }
```

Вхідними даними в даній функції є масиви оцінок обох студентів, а результуючими – коефіцієнт подібності.

Для обчислення всіх коефіцієнтів використовується скрипт, наведений нижче, який перебирає всіх студентів попередників через функцію calc_coef_similar та масив коефіцієнтів подібності із вказанням попередників, яким вони належать.

```
    data.prev_stud_marks.forEach(stud => {
        similarity_coefficients.push({
            'student': stud.id,
            'coef':
calc_coef_similar(data.student_data.marks,stud.marks)
        });
    });
```

2. *Функція обчислення прогнозованої оцінки.*

Дана функція базується на обчисленні середнього зваженого значення, шляхом поділу суми добутків коефіцієнтів подібності попередників із їхніми оцінками за поточний предмет на суму всіх коефіцієнтів подібності із поточної дисципліни по модулю.

Скрипт, який виконує дані дії, наведено нижче.

```
addition_subj.forEach(sbj =>{
        let upper = 0;
        let lower = 0;
        v_pred_stud_marks.forEach(std =>{
            if(std.marks.find(m => m.subject == sbj.code).mark !=
""){
                upper = upper + (parseInt(std.marks.find(m =>
m.subject == sbj.code).mark) *
data.similarity_coefficients.find(cof => cof.student ==
std.id).coef);
                lower = lower +
data.similarity_coefficients.find(cof => cof.student ==
std.id).coef;
            }
        });
        let result = Math.round(upper/lower);
        predict_marks.push({
            'pred_mark': result,
            'subject_code': sbj.code,
            'subject_name': sbj.name
        });
    });
```

Варто зауважити, що обчислюються тільки оцінки із вибіркових дисциплін, а самі обчислення відбуваються на базі оцінок студентів, коефіцієнт подібності яких більше граничного значення.

Взявши до уваги схему модуля та розроблені функції, створимо новий процес в Corezoid та побудуємо необхідну логіку (рис. 3.18).

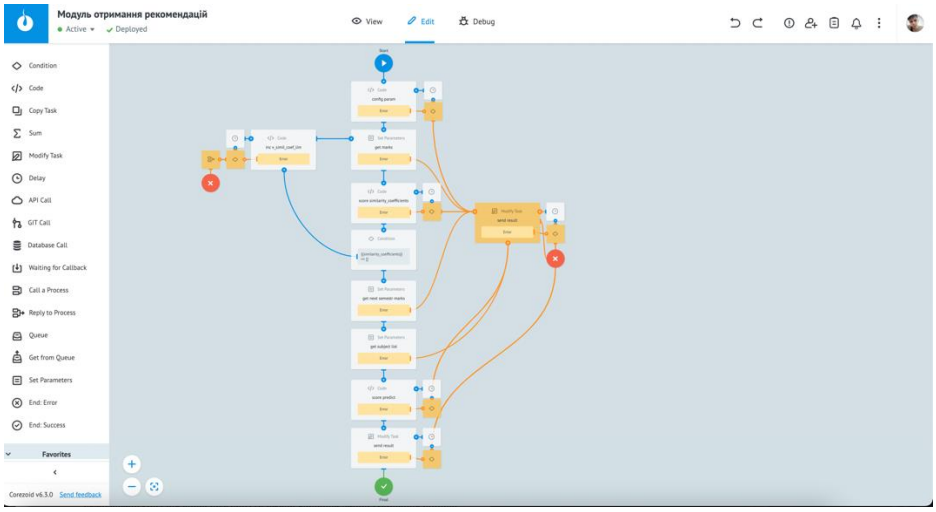


Рисунок 3.18 – Процес системи надання рекомендацій

Із рисунку 3.18 можна помітити, що процес має наступну структуру:

1. отримання оцінок попередників;
2. обчислення коефіцієнтів подібності;
3. перевірка наявності принаймні одного попередника, коефіцієнт подібності якого більше лімітного значення, зменшення коефіцієнту за необхідності;
4. отримання оцінок попередників за наступний семестр;
5. обчислення прогнозованих оцінок;
6. повернення результату в Логічне ядро.

Також в цілях зручності було створено ноду відловки помилок.

3.8 Тестування

В рамках тестування розглянемо два флоу:

1) флоу студента;

2) флоу викладача.

Почнемо із флоу користувача, для цього, після написання боту, потрібно вибрати роль Студент (рис. 3.19)

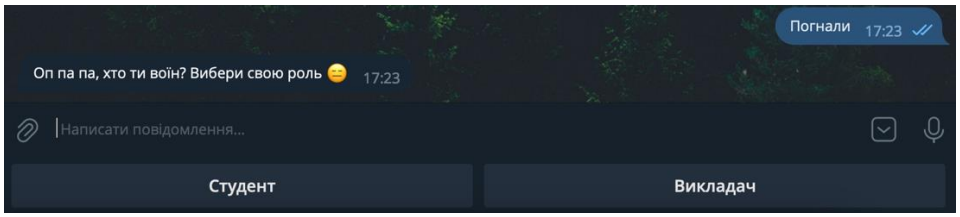


Рисунок 3.19 – Вибір ролі

На наступному кроці потрібно вибрати курс (рис. 3.20) та семестр (рис. 3.21)

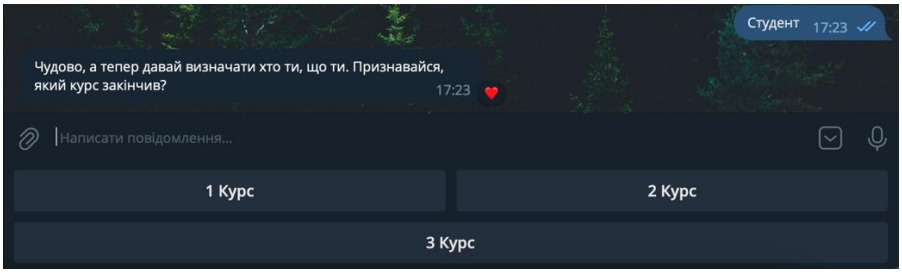


Рисунок 3.20 – Вибір курсу

Рисунок 3.21 – Вибір семестру

Наступним етапом є внесення оцінок із дисциплін (рис. 3.22), варто підмітити, що якщо користувач вводить некоректні значення, то бот його не пускає далі, як це відображено на рисунку 3.23.

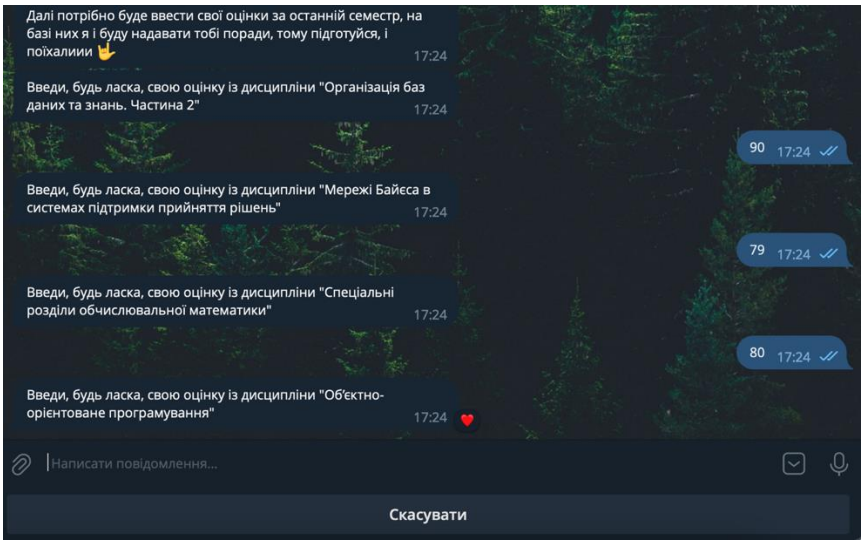


Рисунок 3.22 – Внесення оцінок

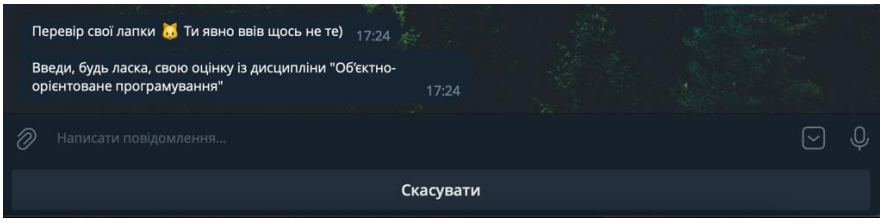


Рисунок 3.23 – Повідомлення про некоректні дані

Після заповнення всіх оцінок студент отримає повідомлення про запуск процесу аналізу, а через декілька секунд і саму рекомендацію (рис. 3.24)

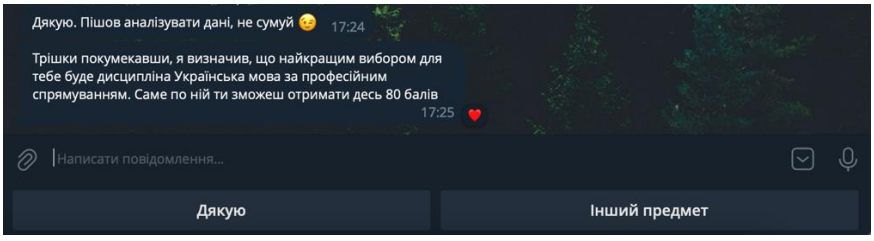


Рисунок 3.24 – Повідомлення із рекомендацією

Якщо користувач натисне на кнопку “Інший предмет”, то він отримає наступну альтернативу, по спаданню прогнозованої оцінки (рис. 3.25). Отримання альтернатив можливе до тих пір, поки не закінчяться вибіркові предмети на наступний семестр (рис. 3.26).

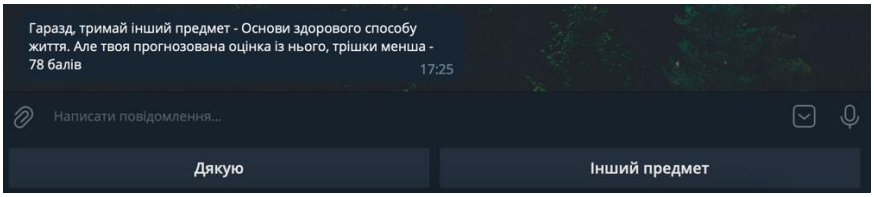


Рисунок 3.25 – Повідомлення із іншою альтернативою

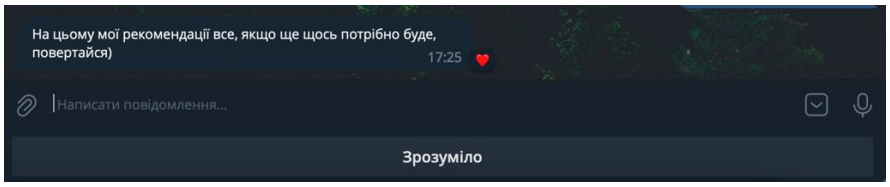


Рисунок 3.26 – Повідомлення про закінчення альтернатив

Перейдемо до тестування флоу викладача, для цього було заведено одного викладача в Сховища викладачів із доступом до двох дисциплін (рис. 3.27).

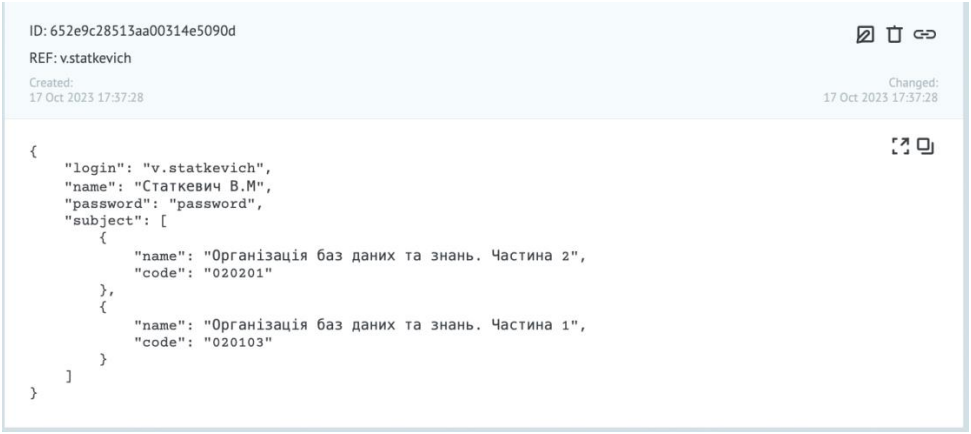


Рисунок 3.27 – Заведений викладач

Для запуску флоу потрібно вибрати роль Викладач (рис. 3.28), після чого ввести логін викладача (рис. 3.29).

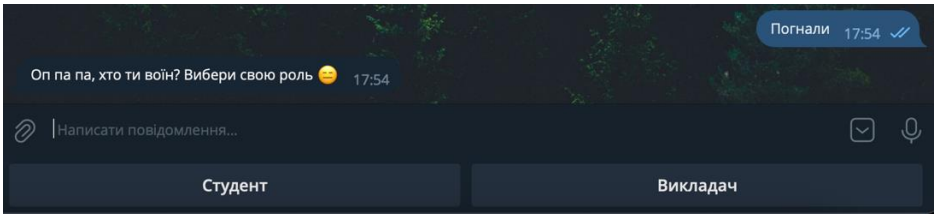


Рисунок 3.28 – Вибір ролі

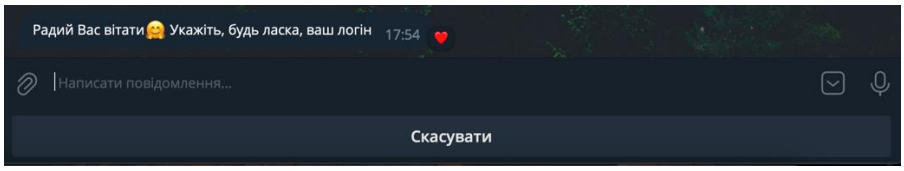


Рисунок 3.29 – Введення логіну

Після коректного вводу логіну (рис. 3.30) та паролю (рис. 3.31), користувача буде направлено на вибір дисципліни, якщо ж дані були введені некоректно, то бот далі не пропустить (рис. 3.32).

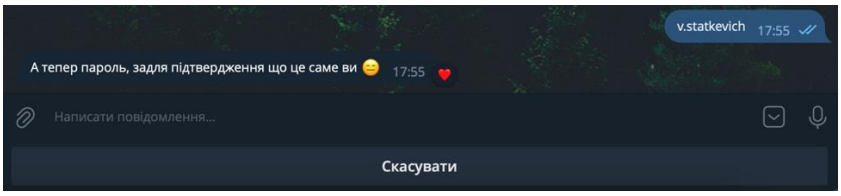


Рисунок 3.30 – Введення доступного логіну

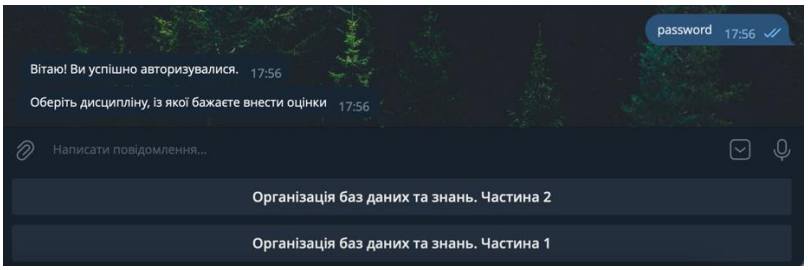


Рисунок 3.31 – Введення коректного паролю

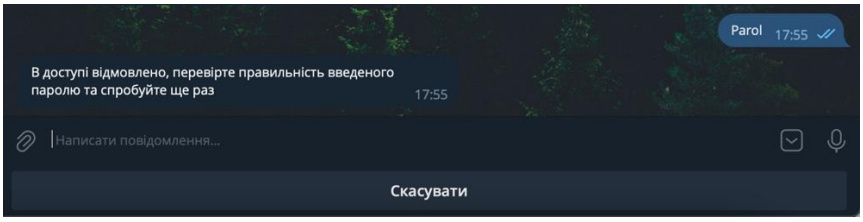


Рисунок 3.32 – Введення некоректного паролю

На наступному етапі оберемо одного студента, по якому будемо змінювати оцінку, для зручності візьмемо студента із номером залікової книжки – 10000. Поточні оцінки студента за 1 семестр другого курсу (саме в ньому викладається одна із доступних дисциплін) наведені нижче.

{

```
"id": 100001,
"marks": [
    {
        "subject": "020101",
        "mark": "97",
        "additional": false
    },
    {
        "subject": "020102",
        "mark": "99",
        "additional": false
    },
    {
        "subject": "020103",
        "mark": "80",
        "additional": false
    },
    {
        "subject": "020104",
        "mark": "74",
        "additional": false
    },
    {
        "subject": "020105",
        "mark": "89",
        "additional": false
    },
    {
        "subject": "020106",
        "mark": "91",
        "additional": false
    },
    {
        "subject": "020107",
        "mark": "84",
        "additional": true
    },
    {
        "subject": "020108",
        "mark": "76",
```

```
                "additional": true
            },
            {
                "subject": "020109",
                "mark": "84",
                "additional": false
            },
            {
                "subject": "020110",
                "mark": "88",
                "additional": false
            },
            {
                "subject": "020111",
                "mark": "",
                "additional": true
            }
        ]
    },
```


Нас цікавить дисципліна “Організація баз даних та знань. Частина 1”, код якої 020103, а поточна оцінка студента 80 балів.

Оберемо цільову дисципліну, введемо номер заліковки студента та його нову оцінку (рис. 3.33).

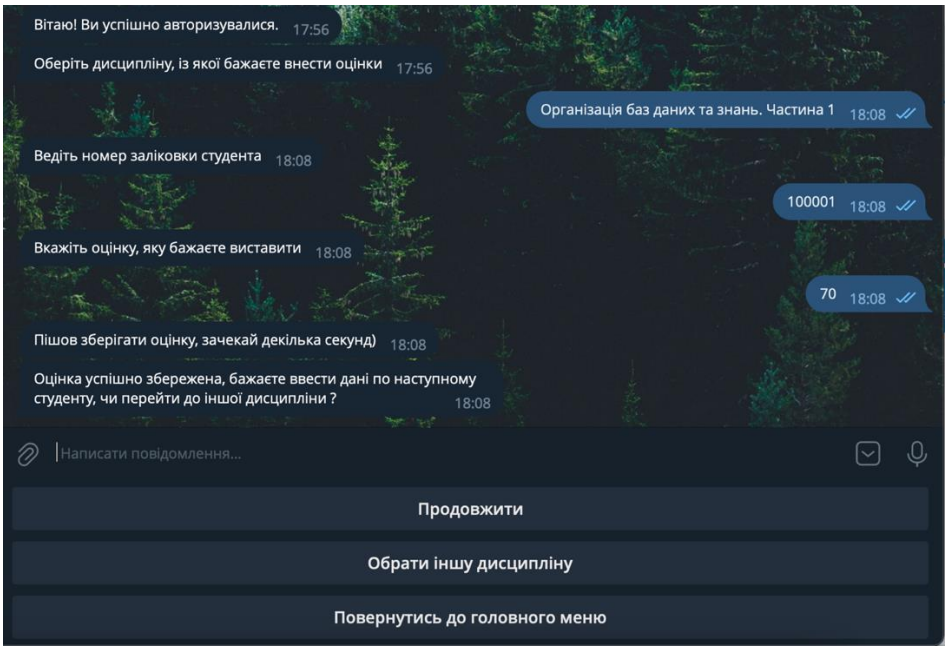


Рисунок 3.33 – Занесення нової оцінки

Перевіривши дані в сховищі оцінок бачимо, що оцінка студента із даної дисципліни змінилася (рис 3.34), а це свідчить тому, що модуль працює.

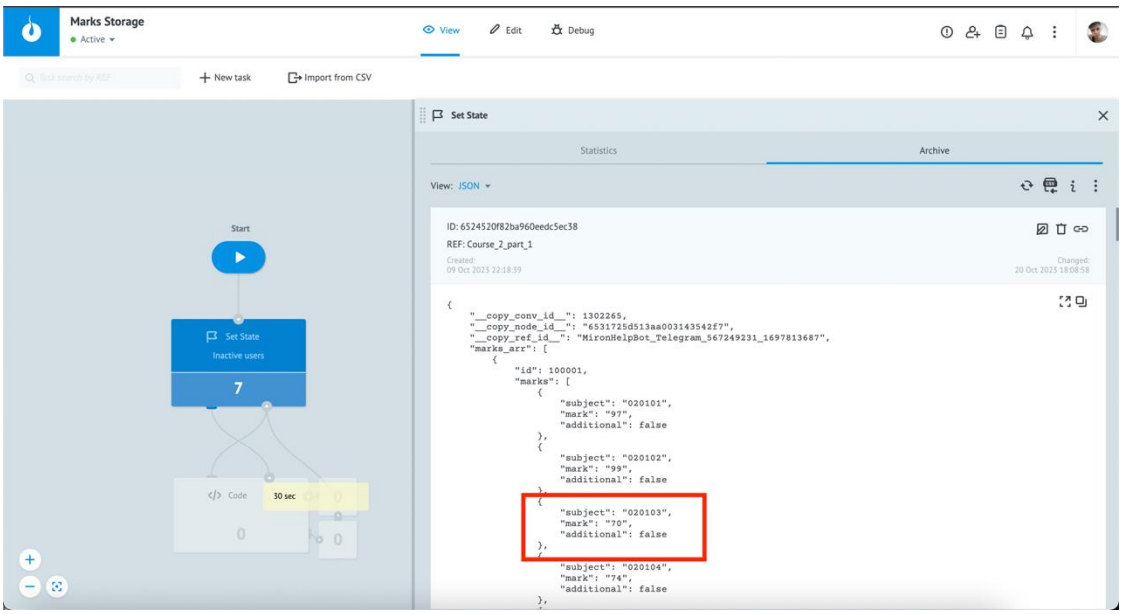


Рисунок 3.34 – Оновлена оцінка студента

Після успішного занесення оцінки у викладача є можливість продовжити виставлення оцінок із даної дисципліни, обрати іншу дисципліну, чи закінчити роботу. Тобто всі гілки, продемонстровані на логічній схемі флоу (рис. 3.9), працюють.

3.9 Висновки до розділу 3

В рамках даного розділу розроблено та протестувано рекомендаційний чат-бот, що працює на базі колаборативної фільтрації за алгоритмом базованим на пам’яті.

Зокрема за допомогою хмарного середовища розробки Corezoid та Telegram Bot API було створено систему обробки повідомлень, спроектовано і реалізовано Логічне ядро, розгорнуто базові сховища даних (Сховище оцінок, Сховище дисциплін, Сховище викладачів) та розроблено Модулі отримання рекомендацій та Занесення оцінок.

Результати тестування показали, що чат-бот працює, і виконує всі закладені в нього функції без помилок.

РОЗДІЛ 4 РОЗРОБКА ВЛАСНОГО СТАРТАП ПРОЕКТУ

4.1 План розробки стартапу та масштабування його на ринку

Для початку розглянемо план розробки стартапу та виведення його на ринок.

Першим кроком буде проведення маркетингового аналізу, що буде включати наступні аспекти:

- аналіз існуючих рішень для вирішення поточної проблеми;
- формування основної ідеї проекту та визначення цільової аудиторії;
- аналіз ринкового середовища;
- розробка стратегії розгортання товару на ринку.

Другим кроком є організація стартапу, в рамках якої потрібно:

- скласти план та графік розробки, запуску продукту;
- визначити обсяги виробництва;
- обчислити витрати, що необхідні для реалізації та запуску проекту.

Наступним кроком буде фінансово-економічний аналіз та оцінка ризиків стартапу, в рамках якого потрібно:

- визначити обсяг інвестиційних витрат;
- обчислити основні фінансово-економічні показники проекту та визначити показники інвестиційної привабливості проекту;
- визначити ризики та способи їх усунення.

Останнім, але не менш важливим для масштабування продукту, кроком є розробка заходів з комерціалізації та залучення інвесторів, в межах якого необхідно:

- провести дослідження щодо зацікавленості потенційних інвесторів;

- скласти інвестиційну пропозицію із описом продукту та шляхами його розширення;
- визначити канали комунікації із цільовою аудиторією.

4.2 Опис ідеї стартап-проекту

Основна ідея стартапу це вирішення буденних дилем визначення вибіркових дисциплін студентом. Суть стартапу – збір даних щото студента, який бажає отримати рекомендації через чат-бот, їх обробка та визначення рекомендацій із використанням системи підтримки прийняття рішень що базується на колаборативній фільтрації, та повернення результатів користувачу.

Інформаційна карта стартапу наведена в таблиці 4.1.

Таблиця 4.1 – Інформаційна карта стартап-проекту

Назва проекту	«Радник Мирон»
Автори проекту	Харабара Денис Вікторович
Коротка анотація	Чат-бот буде обробляти оцінки студента за попередній семестр та надавати рекомендації щодо вибіркових дисциплін, що базуватимуться на рекомендаційній системі із використанням колаборативної фільтрації.
Термін реалізації проекту	3 місяці

Продовження таблиці 4.1

Необхідні ресурси	Комп’ютер із доступом до Інтернет мережі та електромережі. Доступ то хмарного середовища розробки Corezoid та сервісу обміну повідомлень Telegram. Кошти для оплати оренди хмарного середовища та супутніх витрат.
Опис проблеми, яку вирішує проект	Продукт вирішує задачу вибору не обов’язкових дисциплін на наступний семестр для студента
Головні цілі та завдання проекту	Метою стартапу є створення зручної рекомендаційної системи, яка буде загально доступною та зрозумілою для використання студентами, та зможе самовдосконалюватись
Очікувані результати	Залучення вищих навчальних закладів до використання програмного продукту, задля полегшення студентських виборів

4.3 Технологічний аудит ідеї проекту

Наступним кроком сформуємо ідею стартапу (таблиця 4.2) та проведемо його конкурентний аналіз (таблиця 4.3).

У таблиці 4.2 наведений опис ідеї стартапу.

Таблиця 4.2 – Опис ідеї стартапу

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Основною ідеєю стартапу є формування комплексної системи підтримки прийняття рішень, яка буде зручною для використання та вирішуватиме проблему визначення оптимальних вибіркових предметів	Обробка вхідних даних, в форматі оцінок за попередній семестр та надання рекомендації щодо вибіркових дисциплін	Користувач отримає рекомендації стосовно оптимальних вибіркових дисциплін на базі інформації про попередників
	Система внесення нових відомостей стосовно оцінок.	Система буде оновлюватись і доповнюватись, що збільшить точність прогнозування оцінок

Далі проведемо порівняльний аналіз конкурентів проекту. Так як наразі на ринку немає прямих конкурентів то порівнювати будемо із традиційними шляхами підбору вибіркових предметів: на базі рекомендацій попередників; на базі інформації про програми дисциплін. Результати наведені в таблиці 4.3.

Таблиця 4.3 – Порівняльний аналіз конкурентів проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W	N	S
		Власний проект	Вибір на базі рекомендацій попередників	Вибір на базі програм дисциплін			

Продовження таблиці 4.3

1	Якість отриманих рекомендацій	Рекомендації надаються тільки по доступним вибірковим дисциплінам для поточного семестру та на базі досвіду великого числа результатів попередників	Рекомендації надаються тільки по доступним вибірковим дисциплінам для поточного семестру на базі невеликого числа результатів попередників	Рекомендації надаються тільки по доступним вибірковим дисциплінам для поточного семестру на базі власного враження від програми дисципліни		+	
2	Зручність використання	Досить зручно, оскільки не потрібно нічого додатково шукати	Потребує чималих зусиль для пошуку та аналізу рекомендацій попередників	Потребує чималих зусиль для пошуку та аналізу програм дисциплін			+
3	Швидкість використання	Повний процес отримання рекомендацій займає до 5 хв.	Повний процес отримання рекомендацій може зайняти від 1 години до декількох тижнів	Повний процес отримання рекомендацій може зайняти від 1 години до декількох тижнів			+

Продовження таблиці 4.3

4	Доступність по ціні	Для користувачів ПЗ безкоштовне, проте для освітньому закладу потрібно платити за хостинг процесу щомісячно.	Безкоштовно	Безкоштовно	+		
5	Точність прогнозованих оцінок	Досить висока точність із подальшим її збільшенням	Точність прогнозування оцінки невисока, оскільки формується на невеликій вибірці	Прогнозування оцінок відсутнє			
6	Персоналізація	Існує персоналізація для студентів за курсом та семестром, а також можливість налаштування манери спілкування бота.	Відсутня	Відсутня			

Наступним кроком стане аналіз технічної реалізації продукту (таблиця 4.4).

Таблиця 4.4 – Технологічна реалізація продукту

№ п/п	Ідея проекту	Технології і реалізації	Наявність технологій	Доступність технологій	Новизна
1	Створення комплексної системи підтримки прийняття рішення, яка на базі попередніх оцінок буде надавати рекомендації щодо вибіркових дисциплін	Використання мови програмування JavaScript із розгортанням окремої бази даних в MySql	Наявні	Доступні	Використовувались в подібних проектах
2		Використання мови програмування Python із розгортанням окремої бази даних в MySql	Наявні	Доступні	Використовувались в подібних проектах
3		Використання мови програмування JavaScript із використанням хмарного середовища розробки Corezoid	Наявні	Доступні	Раніше не використовувались
Обрана технологія реалізації ідеї проекту: Хмарне середовище розробки Corezoid із використанням мови програмування JavaScript					

4.4 Аналіз ринкових можливостей запуску стартап-проекту

Наступним етапом буде аналіз ринку для запуску стартап-проекту (таблиця 4.5).

Таблиця 4.5 – Попередня характеристика потенційного ринку стартапу

№ п/п	Показники ринку (найменування)	Характеристика
1	Кількість головних гравців, од	0
2	Загальний обсяг продаж, грн/ум.од	3000
3	Динаміка ринку (якісна оцінка)	Позитивна, зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	12%

Тепер сформуємо основні характеристики цільової аудиторії стартапу (таблиця 4.6).

Таблиця 4.6 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреби, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Персоналізований підбір вибіркових дисциплін із подальшим підвищенням середньостатистичної оцінки успішності	Вищі освітні заклади	Зацікавленість в покращенні процесу підбору вибіркових дисциплін та збільшенні середніх показників успішності	Можливість аналізу популярності дисциплін та успішності студентів задля зацікавлення нових абітурієнтів та підвищення середнього показника успішності
2	Статистичні дані стосовно успішності студентів в різних напрямках	Державні органи освіти	Зацікавленість в аналізі сьогоденних тенденцій в сфері освіти	Можливість аналізу популярності дисциплін та успішності студентів задля подальшого визначення пріоритетних дисциплін
3	Простота в підборі вибіркових дисциплін	Студентські ради та самоврядування	Зацікавленість в спрощенні процесу підбору вибіркових дисциплін	Можливість швидкого отримання рекомендацій без пошуку зайвої інформації

Визначимо фактори загроз (таблиця 4.7) та можливостей (таблиця 4.8). Проаналізуємо загрози, щоб зрозуміти можливі перешкоди при запуску продукт на ринок. Фактори можливостей же треба обрахувати, щоб знати усі сприятливі умови та по можливості ними скористатися.

Таблиця 4.7 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Поки що на ринку немає конкуруючих продуктів, проте, оскільки він відкритий, є великий ризик появи нових проектів із подібними властивостями	Успішно вийти на ринок та постійно вдосконалюватись, задля задання тенденцій розвитку сфери
2	Ціна збуту	Ціна за розгортання системи досить таки низька, проте існує також щомісячна плата за хостинг продукту	Продумати маркетингову стратегію, яка буде зосередження на якості та зручності системи в противагу її ціні
3	Якість аналізу	На початкових етапах функціонування продукту є ризик низької якості пошуку найкращих альтернатив та прогнозування оцінки, через малу початкову вибірку	За необхідності впровадити не одну модель прогнозування, а декілька, які будуть надавати різні рекомендації, на базі яких буде формуватися кінцева пропозиція

Таблиця 4.8 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Універсальність	Продукт розгортається в хмарному середовищі, що робить його універсальним, а процес розгортання, без необхідності внесення додаткових змін, та із наявними стартовими вибірками займає до 1 години	В маркетинговій компанії зробити акцент на самостійність на швидкість розгортання продукту
2	Простота у використанні	Система працює на базі широко розповсюдженого месенджера серед молоді – Telegram, що робить її використання зрозумілим та простим	Зробити акцент в маркетингу на простоті да доступності продукту
3	Якість та гарантії	Надання найоптимальніших моделей аналізу та побудови рекомендацій	Пропонувати різні моделі аналізу, зокрема їх поєднання
4	Швидкість отримання рекомендацій	Проходження повного флоу по отриманню рекомендацій займає до 5 хв.	Зробити акцент в маркетингу на швидкості роботи системи
5	Самовдосконалення моделей аналізу	Організувати постійне вдосконалення моделей аналізу та точності прогнозування	Розгорнути систему доповнення початкової бази оцінок, збільшення якої позитивно впливає на точність прогнозування та аналізу

На наступному кроці визначимо тип конкуренції та її рівень (таблиця 4.9).

Таблиця 4.9 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип бізнесу: національний	Основний ринок, на якому розповсюджується продукт – ринок освітніх застосунків України	Продукт має відповідати державним стандартам України.
2. За рівнем конкурентної боротьби: національний	Конкуренція між продуктами, що розповсюджуються по всій території України.	Розширити цільову аудиторію, розробити просту та зрозумілу систему модифікації баз даних та комунікаційної моделі
3. За галузевою ознакою: внутрішньогалузева	Боротьба між конкурентами ведеться в межах галузі розробки освітніх програм	Постійний аналіз галузевого ринку та впровадження інновацій
4. Конкуренція за видами товарів: товарно-родова	Конкуренція ведеться між різноманітними видами товарів, що виконують схожі функції	Акцентувати увагу на простоті та швидкості роботи продукту при створенні маркетингової компанії
5. За характером конкурентних переваг: нецінова	Головним методом конкурентної боротьби є покращення нецінових характеристик продукції	Надання спектру можливих моделей аналізу для вибору, їх консолідації та пошук нових моделей

Продовження таблиці 4.9

6. За інтенсивністю: не марочна	Відсутні відомі компанії	Брати активну участь в обробці зворотного зв'язку та усуненні негативного враження від продукту на ранніх етапах впровадження, задля створення позитивних асоціацій із продуктом

На наступному етапи виконаємо аналіз конкуренції за моделлю 5 сил Майкла Портера (таблиця 4.10).

Таблиця 4.10 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товарозамінники
Складові аналізу	Наразі прямі конкуренти відсутні,	Ринок відкритий та постійно зростаючий	Відсутні	Контроль якості, порівняння цін	Традиційні методи пошуку оптимального вибору: на базі рекомендацій попередників; на базі програм дисциплін
Висновки	Наразі пряма конкуренція відсутня, проте можлива в майбутньому	Можливості входження на ринок, нові потенційні конкуренти	Постачальники не мають впливу	Клієнти диктують напрямок розвитку продукту на ринку	Товарозамінники є безкоштовними, проте незручними та часозатратними

Маючи результати аналізу конкуренції (таблиця 4.10), характеристики ідеї стартап-проекту (таблиця 4.5), характеристики потенційних клієнтів і їх вимоги до продукту (таблиця 4.6) та фактори ринкового середовища (таблиці 4.7 і 4.8), було сформульовано та обґрунтовано перелік факторів конкурентоспроможності (таблиця 4.11).

Таблиця 4.11 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Універсальність	Продукт простий в розгортанні та не залежить від апаратної складової, так як розміщується в хмарному середовищі
2	Простота у використанні	Продукт має зрозумілий інтерфейс та доступну комунікаційну модель
3	Якість та гарантії	Продукт надає можливість зміни моделей аналізу, зокрема створення їх симбіозів
4	Швидкість отримання рекомендацій	Повне флоу отримання рекомендацій займає до 5 хвилин
5	Самовдосконалення моделей аналізу	Сховище оцінок постійно збільшується, що позитивно впливає на точність прогнозів

Наступним кроком є аналіз сильних та слабких сторін продукту (таблиця 4.12).

Таблиця 4.12 – Порівняльний аналіз сильних та слабких сторін системи

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	1	2	3
1	Універсальність	20			+				
2	Простота у використанні	15				+			
3	Якість та гарантії	13			+				
4	Швидкість отримання рекомендацій	17		+					
5	Самовдосконалення моделей аналізу	10	+						

Наступним кроком є SWOT-аналіз стартапу (таблиця 4.13).

Таблиця 4.13 – SWOT-аналіз стартап-проекту

Сильні сторони	Слабкі сторони
- універсальність; - простота в використанні; - якість та гарантії; - швидкість отримання рекомендацій; - самовдосконалення моделей аналізу.	- відсутність відомого імені компанії; - початкова зацікавленість в продукті на ринку; - відсутність універсальних каналів для просування стартапу.
Можливості	**Загрози**
- персоналізація (можливість зміни стилю комунікації в боті); - інтеграція із електронними журналами (можливість обновлювати сховища оцінок без залучення викладачів); - масштабування (розширення бот-платформи на інші месенджери (Viber, Instagram, Facebook)).	- поява конкурентів; - малий ринок збуту; - відключення основного каналу функціонування чат-боту.

Після проведення SWOT-аналіз визначено сильні та слабкі сторони стартапу, а також можливості та загрози проекту, що пов’язані із конкуренцією та плануванням стартапу.

На наступному кроці визначимо альтернативи впровадження проекту на ринку та приблизний час реалізації стартап-проекту.

Таблиця 4.14 – Альтернативи ринкового впровадження стартап проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Первинна реалізація проекту та подальший пошук цільової аудиторії шляхом реклами	60%	3 місяці
2	Реалізація проекту після отримання замовлення на нього шляхом реклами	50%	5 місяців
3	Пошук інвестицій, подальша реалізація проекту та пошук цільової	90%	8 місяців
4	Пошук інвестицій, реклама продукту, пошук ринку збуту, реалізація	100%	12 місяців

За наведеною таблицею вище, можна помітити, що оптимальною є перша альтернатива (Первинна реалізація проекту та подальший пошук цільової аудиторії шляхом реклами), оскільки доцільніше спочатку розробити продукт, щоб потім було що презентувати цільовій аудиторії та інвесторам.

4.5 Розроблення ринкової стратегії стартап-проекту

Для розробки ефективної ринкової стратегії стартапу, перш за все потрібно проаналізувати цільову аудиторію (таблиця 4.15).

Таблиця 4.15 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит у межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Студенти	Висока	30%	Низька	Середня
2	Студентські ради, самоорганізації	Висока	40%	Низька	Середня
3	Вищі освітні заклади	Середня	20%	Низька	Середня
4	Держава органи освіти	Середня	10%	Низька	Висока
Які цільові групи обрано: 2, 3, 4					

Отже було обрано 2, 3 та 4 цільові групи, оскільки перша, хоч і є зацікавленою, проте є вузько направленою, і розробка ПЗ для неї недоцільна.

Студенти, фактично, і є тим самим кінцевим користувачем, для будь якої із цільових груп, проте якщо брати кожного студента, як окремого цільового

користувача, то, в розрізі витрат-прибутків, побудова маркетингової стратегії для цієї аудиторії є збитковою, адже для кожного студента потрібно розробляти окремий, унікальний чат-бот, саме тому дану групу ми розглядати не будемо.

Визначившись із цільовою аудиторією, сформуємо базову стратегію розвитку продукту (таблиця 4.16).

Таблиця 4.16 – Базова стратегія розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	2 та 3	Диференційованого маркетингу	Простота та швидкість отримання рекомендацій	Оптимальних витрат
2	4	Диференційованого маркетингу	Простота, швидкість та автоматизація абсолютно нової рекомендаційної системи на ринку зі збором статистичних даних	Оптимальних витрат

Наступним етапом є вибір стратегії конкурентної поведінки (таблиці 4.17, 4.18).

Таблиця 4.17 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Так	Так	Ні	Стратегія лідера

Таблиця 4.18 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
- простота у використанні; - якість результатів; - швидкість отримання рекомендацій.	Оптимальних витрат	- універсальність; - простота у використанні; - якість та гарантії; - швидкість отримання рекомендацій; - самовдосконалення моделей аналізу.	Система яка швидко надає рекомендації щодо вибіркових дисциплін, має простий інтерфейс та не складна в розгортанні та налаштуванні

4.6 Розроблення маркетингової програми стартап-проекту

Попередньо провівши комплексний аналіз, можемо сформувати ключові переваги концепції продукту (таблиця 4.19) та побудувати план маркетингових комунікацій (таблиця 4.20).

Таблиця 4.19 – Ключові переваги концепції продукту

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Простий інтерфейс	Система має простий та зрозумілий інтерфейс побудований на базі Telegram	Робить поріг входження для користувачів дуже низьким
2	Простота розгортання	Для чат-боту достатньо інсталювати проект в хмарне середовище Corezoid та завантажити початкові вибірки в сховища	Спрощує та пришвидшує процес кінцевого запуску платформи
3	Швидкість отримання рекомендацій	Повне флоу отримання рекомендацій триває не більше 5 хвилин	Дає можливість користувачам отримувати бажаний результат тут і зараз, буз затрат часу
4	Адаптивність	Модуль занесення оцінок дає змогу розширювати початкове сховище, що збільшує точність прогнозованих оцінок	Створює можливості для самовдосконалення моделей аналізу
5	Гнучкість	В системі є функціонал для гнучкого налаштування моделі спілкування, та прав доступу	Дає можливість налаштувати систему під будь які потреби замовника

Таблиця 4.20 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламно о повідомлення	Концепція рекламного звернення
1	Зацікавленість в простому та швидкому отриманні рекомендацій	Соціальні мережі, інформативні тематичні форуми, інтернет сторінки університетів	Швидкість, Простота, Прогнозованість	Закцентувати увагу на простоті та швидкості отримання оптимальних рекомендацій із прогнозовано ю оцінкою	Реклама на тематичних форумах, інтернет сторінках університетів
2	Зацікавленість в спрощенні та оптимізації процесів побудови календарних планів	Технічні форуми, Наукові конференції	Адаптивність, Гнучкість, Простота	Закцентувати увагу на новизні, простоті, та невибагливост і до ресурсів. Також виділити вплив на середній показник успішності та можливий збір статистики	Виступи на наукових конференціях, технічних форумах. Гарячі контакти із керівництвом

4.7 Висновки до розділу 4

В даному розділі досліджено стартап “Чат-боту із вбудованою рекомендаційною системою, для вибору вибіркових дисциплін”.

В рамках дослідження було сформовано ідею стартапу та розроблено стратегію його виходу на ринок, на якому, як виявилось в процесі аналізу, наразі немає подібних продуктів, що позитивно впливає на перспективи проекту.

Також було визначено сильні та слабкі сторони стартапу та проведено його порівняння із наявними наразі традиційними методами визначення вибіркових дисциплін. Зокрема, було проведено SWOT-аналіз та аналіз конкуренції в галузі за М. Портером, на базі яких було сформовано основну концепції конкурентоспроможності.

Останнім, але не менш важливим, кроком було визначення цільової аудиторії проекту, формування маркетингової стратегії та виділення каналів комунікацій із потенційними користувачами.

ВИСНОВКИ

Дана магістерська дисертація присвячена вивченню чат-бот платформ, як осередків розгортання рекомендаційних систем.

Перший розділ роботи присвячений вивченню поняття чат-боту, підходів до їх класифікацій та основним аспектам розробки. Зокрема було розглянуто найпопулярніші месенджери, які надають можливість вбудовувати в свою інфраструктуру різноманітних ботів, та обрано для подальшого використання найоптимальніший – Telegram. Основними перевагами даного месенджеру є широкий спектр інтерактивних функцій та детально описана документація щодо принципів інтеграції ботів. Також даний розділ вміщує детальний опис структури та компонентів хмарного середовища розробки Corezoid, його основних “будівельних блоків” – нод.

В другому розділі дисертації проаналізовано широкий спектр напрацювань щодо систем підтримки прийняття рішень, їх класифікацій, складових та підходів до конструювання. Більш детально розглянуто принципи та математичні апарати, які використовують в розробці рекомендаційних систем на базі колаборативної фільтрації, виділено переваги та недоліки даного підходу.

Практична частина роботи зосереджена на розробці рекомендаційного боту, який базується на алгоритмі надання рекомендацій на основі пам’яті. В її рамках було розроблено унікальний чат-бот із використанням хмарного середовища програмування Corezoid та Telegram Bot API, структурна схема якого включає Систему обробки повідомлень, Логічне ядро, Модуль отримання рекомендацій, Модуль занесення оцінок та Сховища даних.

Зокрема були розроблені та інтегровані в Corezoid скрипти, на мові програмування JavaScript, по отриманню масиву коефіцієнтів подібності та обчисленні прогнозованих оцінок, які базуються на формулах косинуса подібності та середнього зваженого значення відповідно.

Розроблений чат-бот був протестований від лиця двох користувачів – Студента та Викладача. Було пройдено повне успішне флоу для кожної ролі та перевірено коректність відпрацювання модулів, згідно логічної схеми бота.

Також в рамках даної роботи було створено та проаналізовано стартап-проект “Чат-боту із вбудованою рекомендаційною системою, для вибору вибіркових дисциплін”, під назвою “Радник Мирон”.

В рамках дослідження було сформовано стратегію виходу проекту на ринок, визначено його сильні та слабкі сторони, проведено порівняння із наявними наразі традиційними методами визначення вибіркових дисциплін.

Зокрема було проведено SWOT-аналіз та аналіз конкуренції в галузі, на базі яких було сформовано основну концепції конкурентоспроможності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мельниченко О. Що таке чат-бот: секрети використання та основні переваги для бізнесу // helpcrunch.com. 2020. URL: https://helpcrunch.com/blog/uk/shcho-take-chat-bot/ (дата звернення: 15.05.2023).

2. Провотар О.І., Клочко Х.А. Особливості та проблеми віртуального спілкування за допомогою чат-ботів // praci.vntu.edu.ua. 2013. URL: https://praci.vntu.edu.ua/index.php/praci/article/view/375/373 (дата звернення: 15.05.2023).

3. Сухас У. Oracle Intelligent Bots: Чат-боты с искусственным интеллектом // Oracle. 2017. URL: https://www.oracle.com/a/ocom/docs/chatbots.pdf (дата звернення: 17.05.2023).

4. Michiels E. Modelling Chatbots with a Cognitive System Allows for a Differentiating User Experience // ceur-ws.org. URL: http://ceur-ws.org/Vol-2027/paper24.pdf (дата звернення: 18.05.2023).

5. Sheth B. The Bot Lifecycle. What to know before you make your chatbot // chatbotsmagazine.com. 2016. URL: https://chatbotsmagazine.com/the-bot-lifecycle-1ff357430db7 (дата звернення: 25.05.2023).

6. McElvaney P. 10 Reasons You Need To Use Chatbots For Learning Support // elearningindustry.com. 2018. URL: https://elearningindustry.com/chatbots-for-learning-support-10-reasons (дата звернення: 30.05.2023).

7. Kumar C. 12 AI Platform to Help you in Creating Facebook Chatbot // geekflare.com. URL: https://geekflare.com/create-facebook-chatbot/ (дата звернення: 26.05.2023).

8. Messaging apps are now bigger than social networks // businessinsider.com. 2016. URL: https://www.businessinsider.com/the-messaging-app-report-2015-11 (дата звернення: 02.06.2023).

9. Вебхуки для відстеження активності // esputnik.com. URL: https://esputnik.com/uk/support/vebhuki-webhooks (дата звернення: 15.06.2023).

10. Polling и long polling // dvmn.org. URL: https://dvmn.org/encyclopedia/about-chatbots/long-polling/ (дата звернення: 18.06.2023).

11. Поляков А., Пилипів І. Telegram, Viber, WhatsApp, Signal – яким месенджерам можна довіряти // epravda.com.ua. 2017. URL: https://www.epravda.com.ua/publications/2017/12/15/632183/ (дата звернення: 19.06.2023).

12. Technical documentation. Telegram Bot API // telegram.org. URL: https://core.telegram.org/bots/api (дата звернення: 17.06.2023).

13. Як створити чат-бот для бізнесу: Kwizbot vs. Corezoid // evergreens.com.ua. 2020. URL: https://evergreens.com.ua/ua/articles/kwizbot-vs-corezoid-comparison.html (дата звернення: 20.06.2023).

14. Даниленко С. Обзор новой облачной ОС Corezoid: бизнес-процессы, роботы и умные города // blogmiddleware.wordpress.com. 2015. URL: https://blogmiddleware.wordpress.com/2015/06/08/обзор-новой-облачной-ос-corezoid-бизнес-проц/ (дата звернення: 21.06.2023).

15. Technical documentation. Corezoid actor engine // corezoid.com. URL: https://doc.corezoid.com/docs (дата звернення: 21.06.2023).

16. Ржеуський А.В., Кунанець Н.Е., Стахів М. Рекомендаційна система інформаційного обслуговування користувачів бібліотек. *Інформаційні моделі, системи та технології*: матеріали V наук.-техн. конф. (Секція 2. Інформаційні системи, м. Тернопіль, 1–2 лютого 2018 р.). Тернопіль : ТНТУ, 2018. С. 37.

17. Пфанштиль И. Статистика YouTube 2019. Инфографика // rusability.ru. 2019. URL: https://rusability.ru/pfanshtil/statistika-youtube-2019-infografika/5fd296262dda593c3483e887 (дата звернення: 03.08.2023).

18. Бідюк П.І., Коршевнюк Л.О. Проектування комп'ютерних інформаційних систем підтримки прийняття рішень : навч. посіб. / Київ: НТК

«Інститут прикладного системного аналізу» Національний технічний університет України «Київський політехнічний інститут», 2010. 340 с.

19. Харабара Д.В., Статкевич В.М. Чат-бот як середовище розгортання системи підтримки прийняття рішень: приклад телеграм-боту по наданню рекомендацій щодо вибіркових дисциплін. *Системні науки та інформатика*: матеріали II наук.-практ. конф. (Секція 2. Системний аналіз фінансового ринку, м. Київ, 4–8 грудня 2023 р.). Київ : НН ІПСА КПІ ім. Ігоря Сікорського, 2023. С. 229–236 .

20. Бідюк П.І., Тимощук О.Л., Коваленко А.Є., Коршевнюк Л.О. Системи і методи підтримки прийняття рішень : підручник / Київ: Національний технічний університет України «Київський політехнічний інститут», 2020. 619 с.

21. Agarwal A., Chauhan M. Similarity Measures used in Recommender Systems: A Study // International Journal of Engineering Technology Science and Research IJETSR. Volume 4, Issue 6, 2017. P. 2394–3386.

22. Item-based collaborative filtering // cs.carleton.edu. URL: https://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/itembased.html (дата звернення: 18.08.2023).

23. Garcin F., Faltings B., Jurca R., Joswig N. Rating Aggregation in Collaborative Filtering Systems. *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, ACM, New York, USA, 2009. P. 349–352.

24. Ghisloti R. Slope One // girlincomputerscience. 2010. URL: http://girlincomputerscience.blogspot.com/2010/11/slope-one.html (дата звернення: 23.08.2023).

25. Espinosa A. Slope One Predictors for Online Rating-Based Collaborative Filtering // @andresespinosapc. 2017. URL: https://medium.com/@andresespinosapc/today-im-writing-about-slope-one-predictors-for-online-rating-based-collaborative-filtering-by-3c06677f75c6 (дата звернення: 27.08.2023).

26. Lemire D., Maclachlan A. Slope One Predictors for Online Rating-Based Collaborative Filtering *Proceedings of the SIAM Data Mining*, Newport Beach, California, USA, 2005. P. 471–475.

27. Grover P. Various Implementations of Collaborative Filtering // towardsdatascience.com. 2017. URL: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0 (дата звернення: 10.09.2023).

28. Bondarenko K. Факторизация в рекомендательных системах. Что такое и с чем её едят? // @bond.kirill.alexandrovich. 2019. URL: https://link.medium.com/tG9rbxNbJ6 (дата звернення: 27.10.2023).

ДОДАТОК А КОД ГЕНЕРАЦІЇ ТЕСТОВОЇ БАЗИ ОЦІНОК

```
function getRandomNumber() {
    const numbers = [208, 207, 206];
    const randomIndex = Math.floor(Math.random() * numbers.length);
    return numbers[randomIndex];
}
function generateRandomMarks() {
    const marks = [];
    let random_addition = getRandomNumber();
    for (let subject_id = 101; subject_id <= 106; subject_id++) {
      const mark = Math.floor(Math.random() * (100 - 61 + 1)) + 61;
      const additional =
        subject_id === 208 || subject_id === 207 || subject_id ===
206;
      let markValue = "";
      if (!additional) {
        markValue = mark.toString();
      } else if(subject_id != random_addition){
        markValue = mark.toString();
      }
      marks.push({
        subject: `010${subject_id}`,
        mark: markValue,
        additional: additional,
      });
    }
    return marks;
  }
  function generateArray() {
    const result = [];
    let current_id = 100000;
    for (let i = 0; i < 150; i++) {
      current_id++;
```

```
    const marks = generateRandomMarks();
    result.push({
      id: current_id,
      marks: marks,
    });
  }
  return result;
}
const generatedArray = generateArray();
```

ДОДАТОК Б КОД НОДИ ПОШУКУ КОЕФІЦІЄНТІВ ПОДІБНОСТІ

```
let similarity_coefficients = [];
data.prev_stud_marks.forEach(stud => {
    similarity_coefficients.push({
        'student': stud.id,
        'coef': calc_coef_similar(data.student_data.marks,stud.marks)
    });
});
function calc_coef_similar (std1_marks, std2_marks){
    let main_std_marks = std1_marks.filter(el => el.additional ==
false);
    let upper = 0;
    let lower_1 = 0;
    let lower_2 = 0;
    main_std_marks.forEach(m => {
        upper = upper + parseInt(m.mark) * parseInt(std2_marks.find(el
=> el.subject == m.subject).mark);
        lower_1 = lower_1 + (parseInt(m.mark) * parseInt(m.mark));
        lower_2 = lower_2 + (parseInt(std2_marks.find(el => el.subject
== m.subject).mark) * parseInt(std2_marks.find(el => el.subject ==
m.subject).mark));
    });
    let result = upper/(Math.sqrt(lower_1)*Math.sqrt(lower_2));
    return result;
}
data.similarity_coefficients = similarity_coefficients.filter(el =>
el.coef > data.v_simil_coef_lim);
delete data.prev_stud_marks;
```

ДОДАТОК В КОД НОДИ ПРОГНОЗУВАННЯ ОЦІНОК

```
var v_pred_stud_marks = data.pred_stud_marks.filter(m => data.similarity_coefficients.find(elem => elem.student == m.id)!== undefined);
var addition_subj = data.v_subject_array.filter(sbj => sbj.is_selective == true);
var predict_marks = [];
//similarity_coefficients
addition_subj.forEach(sbj =>{
    let upper = 0;
    let lower = 0;
    v_pred_stud_marks.forEach(std =>{
        if(std.marks.find(m => m.subject == sbj.code).mark != ""){
            upper = upper + (parseInt(std.marks.find(m => m.subject == sbj.code).mark) * data.similarity_coefficients.find(cof => cof.student == std.id).coef);
           lower = lower + data.similarity_coefficients.find(cof => cof.student == std.id).coef;
        }
    });
    let result = Math.round(upper/lower);
    predict_marks.push({
        'pred_mark': result,
        'subject_code': sbj.code,
        'subject_name': sbj.name
    });
});
data.v_predict_marks = predict_marks;
delete data.pred_stud_marks;
```