**національний університет БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ україни**

Кафедра комп’ютерних наук

Факультет інформаційних технологій

**Звіт з виробничої практики**

Виконував роботу в групі

Вовк А.О.

Осадчий Д.С

**Виконав**

студент групи ІПЗ-22010бск

**ВОВК А.О.**

Київ 2024

**Вступ**

Боти для навчання є інноваційними інструментами, які активно змінюють підхід до освіти, роблячи його більш доступним, інтерактивним та ефективним. Сучасні технології штучного інтелекту та автоматизації дозволяють створювати інтелектуальних ботів, здатних підтримувати студентів і викладачів у багатьох аспектах навчального процесу.

Боти, які допомагають з розкладом, нагадуваннями про пари та збором відгуків, є надзвичайно корисними інструментами для студентів і викладачів. Ці боти значно полегшують організацію навчального процесу, дозволяючи ефективно управляти розкладом занять, своєчасно нагадувати про важливі події та збирати зворотний зв'язок.

Однією з основних функцій таких ботів є управління розкладом. Вони дозволяють студентам зберігати розклад занять у зручному форматі, отримувати інформацію про майбутні пари та зміни в розкладі. Це допомагає уникнути плутанини і забезпечує, що всі учасники навчального процесу завжди знають, коли і де їм потрібно бути.

Нагадування є ще однією важливою функцією. Боти можуть автоматично нагадувати студентам про початок занять, дедлайни або інші важливі події. Це особливо корисно для тих, хто веде насичений спосіб життя і може випадково пропустити важливу інформацію. Завдяки регулярним нагадуванням студенти завжди в курсі своїх обов'язків і можуть ефективніше планувати свій час.

Збір відгуків також є важливим аспектом роботи таких ботів. Вони дозволяють студентам залишати зворотний зв'язок про заняття, викладачів і загальний навчальний досвід. Це не тільки допомагає викладачам краще розуміти потреби своїх студентів, але й сприяє покращенню якості освіти загалом. Боти можуть автоматично аналізувати зібрані відгуки, що дає можливість швидко реагувати на проблеми і впроваджувати необхідні зміни.

Крім того, такі боти часто інтегруються з іншими освітніми платформами та інструментами, що дозволяє створювати єдине інформаційне середовище. Це робить процес управління навчанням більш ефективним і зручним для всіх учасників.

Отже, боти для управління розкладом, нагадуваннями та збором відгуків є незамінними помічниками у сучасній освіті. Вони допомагають зробити навчальний процес більш організованим, ефективним і зручним, забезпечуючи студентам та викладачам всі необхідні інструменти для успішного навчання та викладання.

Створення власного Telegram-бота може значно покращити взаємодію з користувачами, автоматизувати рутинні завдання та надавати корисну інформацію. Аналіз можливостей та особливостей Telegram-ботів допоможе прийняти обґрунтовані рішення щодо їх розробки та використання. Ось ключові аспекти, які варто врахувати:

**1. Призначення та Функціональність**

**Визначення мети:** Перше, що потрібно зробити — чітко визначити мету бота. Боти можуть виконувати різноманітні функції, такі як інформаційні (новини, погода), обслуговування клієнтів, автоматизація бізнес-процесів, розваги (ігри, чат-боти) та багато інших.

**Розширення функціональності:** Після визначення основної мети, варто розглянути додаткові можливості. Наприклад, інтеграція з базами даних, іншими API, підтримка мультимедіа (зображення, відео, аудіо), планування завдань та надсилання повідомлень на основі часу або подій.

**2. Платформа та Інструменти Розробки**

**Вибір платформи:** Telegram надає потужний API для розробки ботів, що підтримує текстові, голосові та мультимедійні повідомлення, інтерактивні кнопки, опитування та багато іншого.

**Мови програмування:** Найпопулярніші мови для розробки Telegram-ботів — це Python, JavaScript (Node.js), PHP, Java та Go. Вибір залежить від зручності для розробника, наявності бібліотек і фреймворків, а також вимог до продуктивності.

**Фреймворки та бібліотеки:** Існує багато готових рішень для створення Telegram-ботів, таких як python-telegram-bot, node-telegram-bot-api, Telegraf (для JavaScript), JTelegramBot (для Java). Використання фреймворків спрощує процес розробки, обробки запитів та інтеграції з іншими сервісами.

**3. Безпека та Приватність**

**Захист даних:** Оскільки боти можуть обробляти чутливі дані користувачів, важливо забезпечити високий рівень безпеки. Це включає шифрування з'єднань, безпечне зберігання даних та регулярні оновлення для виправлення вразливостей.

**Політика конфіденційності:** Користувачі повинні знати, які дані збираються ботом, як вони використовуються та захищаються. Важливо мати чітку та доступну політику конфіденційності.

**4. Інтерфейс та Зручність Використання**

**Користувацький досвід:** Інтерфейс бота має бути інтуїтивно зрозумілим і простим у використанні. Використання кнопок, меню та команд полегшує навігацію.

**Інтерактивність:** Взаємодія з ботом має бути динамічною та відповідати очікуванням користувачів. Використання інтерактивних елементів, таких як клавіатури та відповіді, покращує користувацький досвід.

**5. Моніторинг та Підтримка**

**Моніторинг роботи:** Важливо мати систему моніторингу для відстеження роботи бота, виявлення помилок і аналізу продуктивності. Це допоможе оперативно реагувати на проблеми та покращувати функціональність.

**Зворотний зв'язок:** Надання можливості користувачам залишати відгуки та пропозиції допоможе вдосконалювати бот та розширювати його функціонал відповідно до потреб користувачів.

**6. Масштабованість**

**Підготовка до зростання:** Варто врахувати можливість масштабування сервера або додатків для обробки великої кількості запитів у майбутньому. Це може включати використання хмарних сервісів, контейнеризацію (наприклад, Docker) та оптимізацію коду.

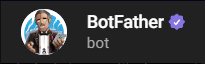
**Висновок**

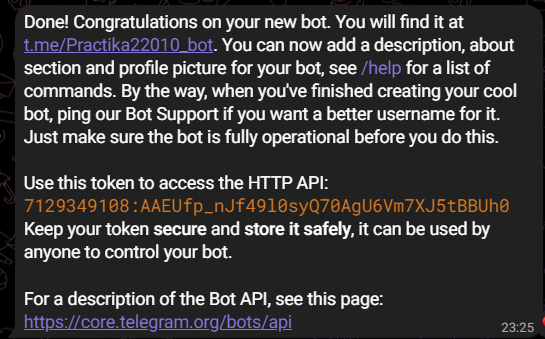
Вибір підходу до розробки Telegram-бота залежить від цілей проекту, наявних ресурсів та технічних вимог. Використання готових бібліотек та фреймворків спрощує процес розробки, а продуманий дизайн і безпека забезпечують надійність і зручність використання бота. Підтримка та оновлення — ключові аспекти для підтримання високої якості сервісу.

**ПОЧАТОК РОБОТИ**

Створення мого Telegram-бота почалося з ідеї створити інструмент, який би допомагав користувачам організовувати свій розклад занять та отримувати нагадування про пари. Першим кроком було ознайомлення з можливостями Telegram API та вибір відповідної мови програмування і фреймворків. Зупинившись на Python через його простоту та численні готові бібліотеки, я розпочав пошук інформації та вивчення документації.

Після цього я створив обліковий запис для бота в Telegram і отримав токен доступу, використовуючи BotFather — офіційного бота Telegram для управління іншими ботами. Це дозволило мені налаштувати основні параметри бота, такі як ім'я та опис.





Далі я встановив необхідні бібліотеки, зокрема python-telegram-bot, яка значно спростила інтеграцію з Telegram API. Вибравши редактор коду, я почав розробку основних функцій бота, таких як обробка команд, введення розкладу та створення нагадувань. Особливу увагу приділяв простоті та інтуїтивності інтерфейсу, щоб користувачі могли легко взаємодіяти з ботом.







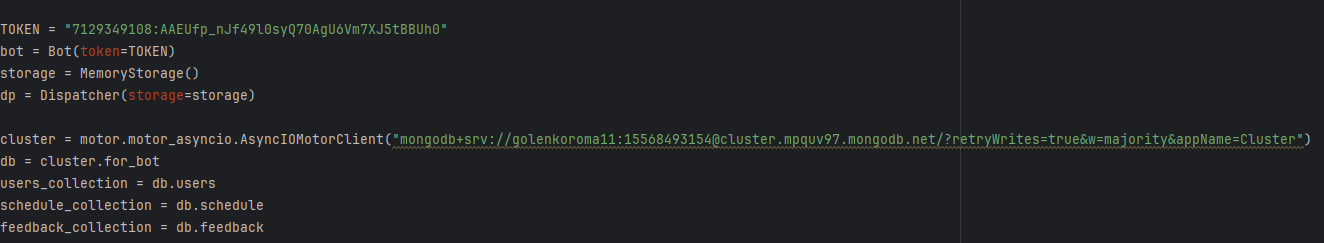
Паралельно з розробкою коду я налаштував базу даних а саме я вибрав MongoDB

**Чому я вибрав MongoDB для мого Telegram-бота:**

1. **Гнучкість даних:** MongoDB дозволяє зберігати дані у вигляді документів, що є більш природним і гнучким способом представлення даних порівняно з таблицями в реляційних базах даних. Це особливо корисно для динамічних або неоднорідних даних, таких як розклади занять або відгуки користувачів.
2. **Масштабованість:** MongoDB легко масштабується, що важливо для додатків, які можуть рости з часом. Ця база даних підтримує горизонтальне масштабування шляхом шардування, що забезпечує високу доступність і стійкість до відмов.
3. **Простота інтеграції:** MongoDB добре інтегрується з різними мовами програмування, включаючи Python, що робить її зручною для розробників. Бібліотеки, такі як pymongo і motor, дозволяють легко працювати з MongoDB з асинхронною обробкою даних, що важливо для ефективного обслуговування запитів користувачів у реальному часі.
4. **Висока продуктивність:** Завдяки своїй структурі MongoDB забезпечує високу швидкість операцій читання і запису, що критично для додатків, де швидкість доступу до даних є ключовою.
5. **Широка підтримка спільноти та документації:** MongoDB має велику спільноту розробників і хорошу документацію, що полегшує вирішення проблем і впровадження нових функцій.

Загалом, вибір MongoDB був зумовлений потребою в гнучкому, масштабованому і швидкому рішенні для зберігання та управління даними, яке також добре інтегрується з обраним стеком технологій для розробки бота

Завершивши розробку основних функцій, я провів тестування бота, щоб виявити та виправити можливі помилки. Тестування включало перевірку коректності роботи команд, обробки помилок та перевірку надійності з'єднання з базою даних.



Першими вагомим результатом був готовий телеграм бот з підлюченою базою даних

**Опис основних частин коду**

**1. Імпорт бібліотек і налаштування**

* **aiogram**: Використовується для створення Telegram-бота. Бібліотека aiogram забезпечує основну функціональність, включаючи обробку повідомлень, збереження стану користувача, створення клавіатур і взаємодію з API Telegram.
* **motor**: Асинхронний клієнт для MongoDB, який дозволяє ботам ефективно взаємодіяти з базою даних без блокування основного потоку подій.

**2. Ініціалізація бота і бази даних**

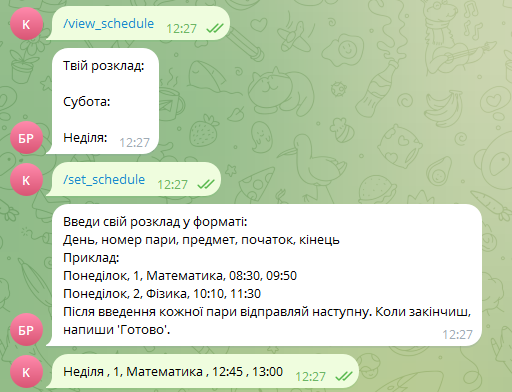
* **Bot і Dispatcher**: Bot представляє об'єкт бота, а Dispatcher відповідає за управління обробкою повідомлень та іншими подіями.
* **MongoDB**: Використовується для збереження даних користувачів, розкладу та відгуків. Колекції users\_collection, schedule\_collection, і feedback\_collection зберігають відповідну інформацію.

**3. Класи станів**

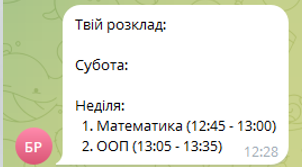
* **ScheduleStates і FeedbackStates**: Ці класи визначають різні стани машини станів (FSM), які допомагають керувати потоком введення/виведення від користувачів.

**4. Функції команд**

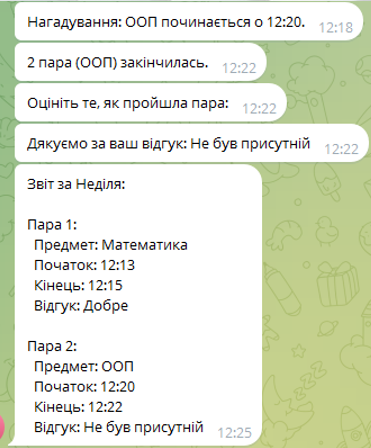
* **/start**: Привітальна команда, яка відправляє повідомлення користувачу з інструкціями щодо доступних команд і додає користувача до бази даних.
* @dp.message(Command(commands=["start"]))  
  async def start(message: types.Message):  
   keyboard = InlineKeyboardMarkup(inline\_keyboard=[  
   [InlineKeyboardButton(text="Звіт за сьогодні", callback\_data="today\_report")]  
   ])  
   await message.answer(  
   "Привіт! Ось команди, які ти можеш використовувати:\n"  
   "/set\_schedule - Встановити свій розклад на тиждень\n"  
   "/view\_schedule - Переглянути свій розклад\n"  
   "/edit\_delete - Редагувати або видалити розклад\n"  
   "/help - Отримати допомогу щодо використання цього бота\n",  
   reply\_markup=keyboard  
   )  
   await add\_user(message.chat.id)
* **/set\_schedule**: Дозволяє користувачам встановлювати свій розклад, вводячи пари у форматі "день, номер, предмет, початок, кінець". Інформація зберігається у базі даних MongoDB.
* @dp.message(Command(commands=["set\_schedule"]))  
  async def set\_schedule\_command(message: types.Message, state: FSMContext):  
   await message.answer(  
   "Введи свій розклад у форматі:\n"  
   "День, номер пари, предмет, початок, кінець\n"  
   "Приклад:\n"  
   "Понеділок, 1, Математика, 08:30, 09:50\n"  
   "Понеділок, 2, Фізика, 10:10, 11:30\n"  
   "Після введення кожної пари відправляй наступну. Коли закінчиш, напиши 'Готово'."  
   )  
   await state.set\_state(ScheduleStates.setting\_schedule)  
   await state.update\_data(schedule=[])



* **/view\_schedule**: Відображає користувачам їх поточний розклад на тиждень.
* @dp.message(Command(commands=["view\_schedule"]))  
  async def view\_schedule\_command(message: types.Message):  
   user\_id = message.chat.id  
   schedules = await schedule\_collection.find({"id": user\_id}).to\_list(None)  
   if schedules:  
   schedule\_text = ""  
   for schedule in schedules:  
   day = schedule.get("day", "Невідомий день")  
   schedule\_text += f"\n{day}:\n"  
   for cls in schedule.get("schedule", []):  
   schedule\_text += f" {cls['number']}. {cls['subject']} ({cls['start']} - {cls['end']})\n"  
   await message.answer(f"Твій розклад:\n{schedule\_text}")  
   else:  
   await message.answer("Ти ще не встановив свій розклад.")



* **/edit\_delete**: Дозволяє редагувати або видаляти наявний розклад. Користувач вибирає день тижня, а потім вводить нову інформацію або вибирає пару для видалення.
* **/today\_report**: Генерує звіт про пари на поточний день, включаючи відгуки, залишені після занять.
* @dp.message(Command(commands=["today\_report"]))  
  async def today\_report\_command(message: types.Message):  
   await generate\_today\_report(message.chat.id)
* async def generate\_today\_report(user\_id):  
   current\_day = datetime.now().strftime("%A")  
    
   # Mapping days in English to Ukrainian  
   day\_map = {  
   "Monday": "Понеділок",  
   "Tuesday": "Вівторок",  
   "Wednesday": "Середа",  
   "Thursday": "Четвер",  
   "Friday": "П'ятниця",  
   "Saturday": "Субота",  
   "Sunday": "Неділя"  
   }  
   day = day\_map.get(current\_day, "Невідомий день")  
    
   # Retrieve schedule for the current day  
   schedule = await schedule\_collection.find\_one({"id": user\_id, "day": day})  
   if schedule and "schedule" in schedule:  
   report\_text = f"Звіт за {day}:\n"  
   for cls in schedule["schedule"]:  
   class\_number = cls['number']  
   subject = cls['subject']  
   start = cls['start']  
   end = cls['end']  
    
   # Retrieve feedback for the class  
   feedback\_doc = await feedback\_collection.find\_one({"user\_id": user\_id, "class\_number": class\_number})  
   feedback\_text = feedback\_doc["feedback"] if feedback\_doc else "Без відгуку"  
    
   report\_text += (  
   f"\nПара {class\_number}:\n"  
   f" Предмет: {subject}\n"  
   f" Початок: {start}\n"  
   f" Кінець: {end}\n"  
   f" Відгук: {feedback\_text}\n"  
   )  
    
   await bot.send\_message(user\_id, report\_text)  
   else:  
   await bot.send\_message(user\_id, "Сьогодні у вас немає пар.")

****

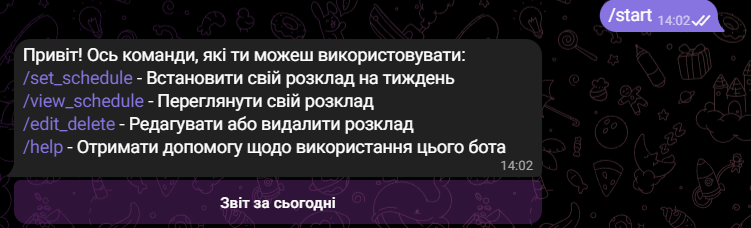
**5. Обробка зворотного зв'язку**

* **request\_class\_feedback**: Після завершення пари користувач отримує запит на оцінку заняття. Відгуки зберігаються у колекції feedback\_collection.

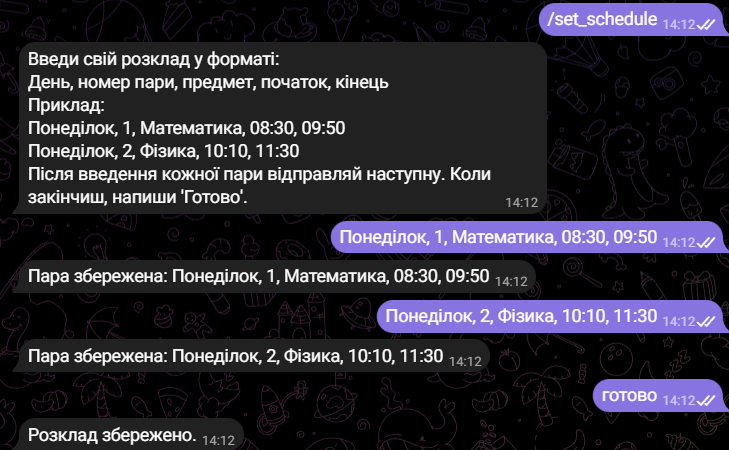
**6. Функції сповіщень**

* **send\_notifications**: Ця функція перевіряє час початку і закінчення пар і надсилає відповідні сповіщення користувачам, зокрема нагадування перед парою та запит на відгук після неї.

**Скріншоти роботи бота**

****

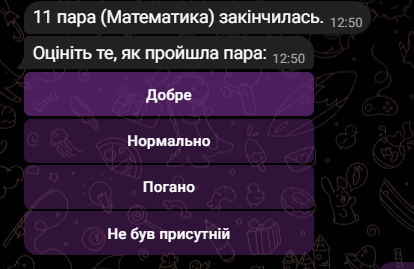
**Додвання пар**

****

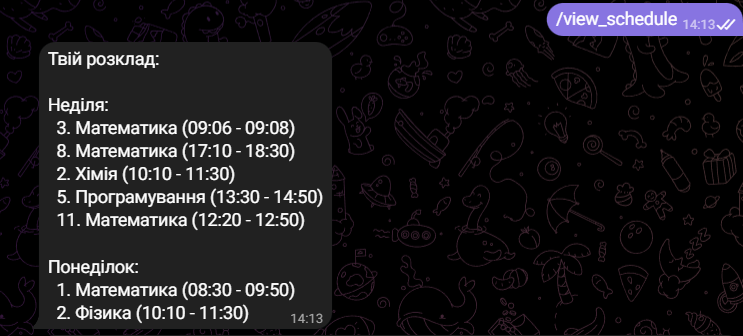
**Нагадування про початок пари**

****

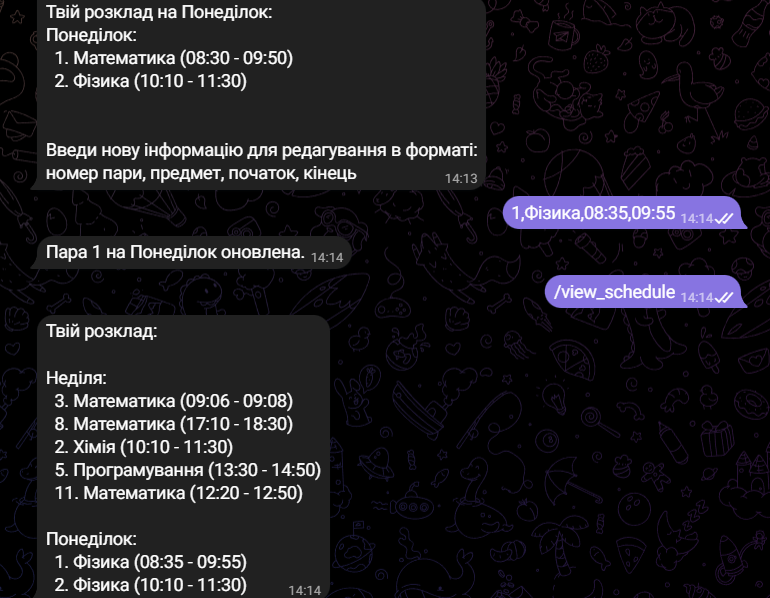
**Опитування пілся пари**

****

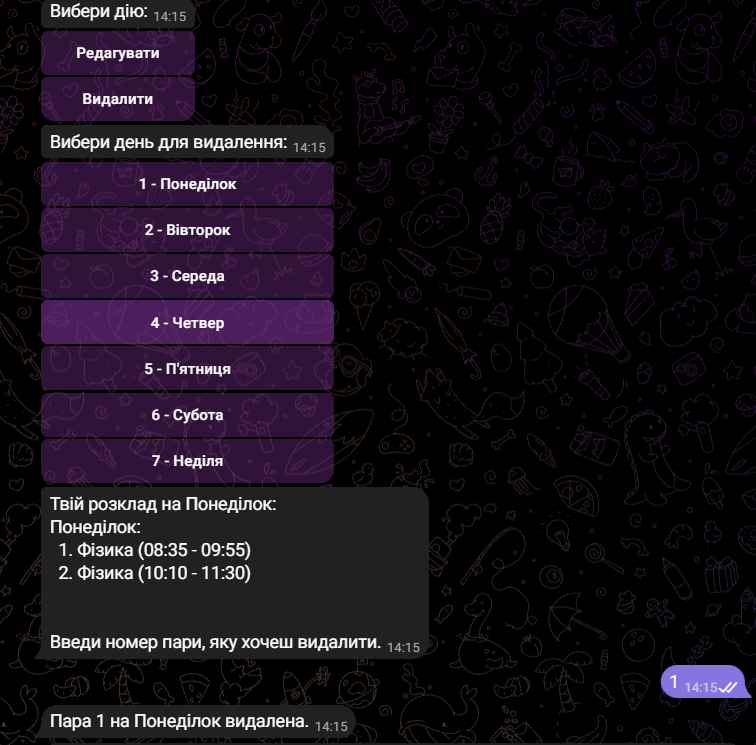
**Перегляд розладу**

****

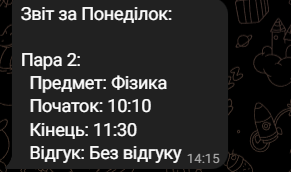
**Редагування**

****

**Видалення**

****

**Звіт за сьгодніші пари та відгуки**

****

**Висновок**

У процесі створення Telegram-бота для управління розкладом занять і збору відгуків, було обрано мовою програмування Python, що забезпечує високу гнучкість і ефективність. Python був обраний через його широкий набір бібліотек, зокрема aiogram для взаємодії з API Telegram та motor для роботи з базою даних MongoDB.

Вибір MongoDB як бази даних обумовлений її здатністю ефективно обробляти неструктуровані дані, що є ключовою перевагою при збереженні різноманітної інформації про користувачів, розклад та відгуки. MongoDB також забезпечує високу масштабованість, що є важливим при збільшенні кількості користувачів та даних.

Основні функції бота включають встановлення, перегляд, редагування та видалення розкладу занять, а також надсилання сповіщень перед початком пари та запити на відгуки після їх завершення. Використання асинхронного програмування дозволяє ефективно обробляти одночасні запити та сповіщення, що значно покращує користувацький досвід.

Цей проект демонструє можливості використання сучасних технологій у створенні корисних інструментів для навчального процесу. Бот не лише спрощує управління розкладом для користувачів, але й забезпечує можливість зворотного зв'язку, що може бути використано для аналізу та покращення навчального процесу.