МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАТИКИ ТА КОМП'ЮТЕРНОЇ ТЕХНІКИ

Рівень вищої освіти	Перший (бакалаврський)
Спеціальність	Інформаційні системи та технології
Освітня програма	Інформаційні системи та технології
Група	6.04.126.010.18.1

ДИПЛОМНИЙ ПРОЕКТ

на тему: «Розроблення Telegram-бота для проходження тестів і моніторингу результатів»

Виконав: студент Дмитро РУДНИЦЬКИЙ

Керівник:к.т.н., доцент Ігор Кобзев

Рецензент: доцент кафедри кібербезпеки та Data технологій
Харківського національного
Університету Внутрішніх Справ
Юрій ГОРЕЛОВ

Харків – 2022 рік

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Факультет Інформаційних технологій

Кафедра Інформатики та комп'ютерної техніки

Освітній ступінь Бакалавр

Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри
інформатики та комп'ютерної техніки
_____ проф. Сергій УДОВЕНКО
« 01 » лютого 2022 р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Рудницькому Дмитру Ігоровичу

1. Тема проекту: Розроблення Telegram-бота для проходження тестів і моніторингу результатів

керівник проекту: Кобзев Ігор Володимирович, к.т.н., доцент

затверджені наказом ректора від «01» лютого 2022 року № 178-С

2. Строк подання студентом проекту: 08 червня 2022 року

3. Вихідні дані до проекту: ДСТУ щодо обробки інформації, літературні джерела, матеріали практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Аналіз предметної області
Розділ 2. Технології розробки чат боту
Розділ 3. Розробка програмного забезпечення системи

5. Перелік графічного матеріалу:

Схема обміну даними між користувачом і Telegram ботом, графік популярності мов програмування у комерційних проектах, скріншоти програми.

6. Консультанти розділів дипломного проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання: «01» лютого 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1.	Розроблення плану дипломного проекту, ознайомлення з літературними джерелами за темою	05.02.2022	
2.	Аналіз предметної області	29.02.2022	
3.	Дослідження технології розробки чат боту	22.03.2022	
4.	Розробка програмного забезпечення системи	23.05.2022	
5.	Перевірка чернетки дипломного проекту та внесення змін до неї керівником	29.05.2022	
6.	Перевірка якості дипломного проекту в системі «Антиплагіат»	04.06.2022	
7.	Оформлення дипломного проекту	05.08.2022	
8.	Подання Голові Екзаменаційної комісії щодо захисту дипломного проекту	08.06.2022	

Студент _____ Дмитро Рудницький

Керівник проекту _____ Ігор Кобзев

АНОТАЦІЯ

Пояснювальна записка до дипломного проекту: 53 с., 15 рис., 24 джерел, 1 додаток.

Метою розробки чат-бота є дати можливість користувачам месенджера Телеграм, які цікавляться сферою IT отримати тест та дізнатися свій рівень знань у галузі QA.

Одним із найефективніших способів реклами є чат-ботамітація, що дозволяє проводити масштабні рекламні кампанії брендів. Це пояснюється тим, що в чаті-боті вищий рівень конвертації відкриття повідомлення та перегляду його користувачами, ніж у будь-якого іншого інструменту реклами.

А для користувачів це стане набагато легше, тому що їм не доведеться самостійно шукати потрібну інформацію в інтернеті, а також вони зможуть отримувати її за допомогою чат-бота, який автоматично інформуватиме їх про наявність необхідної інформації в мережі.

При написанні дипломного проекту були використані Інтернет джерела.

ЧАТ-БОТИ, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, TELEGRAM, CRM, QA

ANNOTATION

Explanatory note to the diploma project: p. 57, fig. 15, 24 sources.

The purpose of developing a chatbot is to enable users of Telegram Messenger who are interested in IT to take a test and find out their level of knowledge in the field of QA.

One of the most effective ways to advertise is chat botamitation, which allows you to conduct large-scale advertising campaigns for brands. This is because chatbots have a higher conversion rate than opening and viewing a message than any other advertising tool.

And for users it will be much easier, because they do not have to search for the necessary information on the Internet, and they can get it with a chatbot, which will automatically inform them about the availability of the necessary information online.

CHATBOTS, INFORMATION TECHNOLOGY, TELEGRAM, CRM, QA

ЗМІСТ

ВСТУП

Постійно мінливі ринкові умови, висока швидкість прийняття рішень, багатозадачність в управлінні активами та необхідність зниження ризиків потребують сучасних підходів до організації підприємницької діяльності. Рішення все більш складного внутрішнього та зовнішнього середовища підприємства полягає у комплексній автоматизації бізнес-процесів. Це дозволяє вивільнити цінні ресурси для стратегічного планування та концентрації управління у ключових сферах діяльності компанії. Необхідність автоматизації інформаційних процесів обумовлена збільшенням обсягу інформації в інформаційній системі (ІВ) організації, необхідністю прискорення та використання складніших методів їх обробки.

Автоматизація бізнесу – це частковий або повний переклад стереотипних операцій та бізнес-завдань під управління спеціалізованою інформаційною системою або складним апаратним та програмним забезпеченням. Внаслідок чого відбувається вивільнення людських та фінансових ресурсів для підвищення продуктивності та ефективності праці.

Інформаційні технології – це сукупність методів виробництва, технологічних процесів та програмно-технічних засобів для виконання інформаційних процесів з метою підвищення їх надійності, оперативності та зниження трудомісткості їх використання [22].

На сьогоднішній день, інформаційні технології є одним із основних досягнень діяльності людства. Використовуючи інформаційні технології, можна зробити так, щоб економіка розвивалася і підвищувалася продуктивність праці та заробітна плата [9]. Це полегшить роботу з організації комунікацій на всіх рівнях управління, а також знизить матеріало- та енергоємність окремих виробництв та національної економіки загалом.

Розвинена сфера нових технологій внесла свій внесок у створення інформаційних систем [6]. Для України це дуже важливо – запровадження сучасних ІТ-технологій дозволяє підвищити якість підготовки та прийняття важливих управлінських рішень. З 2001 року в Україні розпочалася активна

робота щодо формування інформаційного суспільства.

За даними Державного агентства з питань науки та інновацій, у 2007 році було прийнято закон України «Про засади розвитку інформаційного суспільства в Україні на 2007—2015 роки».

Головним завданням розвитку інформаційного суспільства в Україні є надання кожній людині на основі широкого використання сучасних інформаційних технологій можливість створювати інформацію та знання, користуватися та обмінятися ними, виробляти товари та надавати послуги з максимальною ефективністю, підвищуючи якість свого життя. Пріоритетним напрямком є розвиток інформаційного суспільства в Україні та впровадження новітніх інформаційних технологій у різні галузі життєдіяльності та діяльності органів державної влади, місцевого самоврядування та організацій.

Національний підхід до розвитку інформаційно-комунікаційних технологій в Україні базується на засадах пріоритету наукового та технічного прогресу над економічним розвитком країни, а також формування необхідних для цього законодавчих та сприятливих економічних умов. Збільшення різноманітності наданих послуг, створення доступу до електронних інформаційних ресурсів; посилення мотивації щодо використання ІКТ;

Розширення спектру послуг за рахунок використання ІКТ у науці, освіті або культурі; підвищення рівня мотивації щодо використання ІКТ у сфері науки, освіти або культури; покращення кадрового потенціалу; посилення мотивації щодо використання ІТС.

Складнощі у цьому процесі пов'язані з недостатньою технологічною базою, складністю фінансування цього процесу, відсутністю достатньої кількості науково-дослідних робіт з формування інформаційного середовища сучасного суспільства та його складових [23].

Сучасні інформаційні системи дозволяють забезпечити:

- Безпосередній та своєчасний доступ до інформації;
- Поліпшення координації роботи внутрішньої діяльності.
- При використанні якісної технології системного аналізу та

проектування операційного управління, можна досягти підвищення ефективності роботи підприємства [14].

На сьогоднішній день Україна має великий потенціал у сфері інформаційних технологій. За рахунок цього її перевага у тому, що вона займається їх створенням. В Україні, як і в інших країнах світу, існує проблема з відсутністю адекватної державної політики щодо виробників. Щоб створити інформаційне суспільство в Україні, необхідно враховувати узгодженість дій усіх гілок влади. Необхідно ухвалити рішення про створення єдиного центру з розробки та впровадження нових інформаційних технологій в Україні [21].

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Тестовий контроль як засіб навчального процесу

Постановка проблеми. Складна та багатогранна реформа системи освіти необхідна для входження України до Болонського процесу. Модернізація змісту вищої школи, яка має бути спрямована на інтеграцію до європейської та світової освіти на засадах Болонської декларації, є пріоритетним напрямком розвитку вищої освіти України. На сьогоднішній день Україна перебуває в сучасних соціально-економічних умовах і молоді люди повинні вміти користуватися сучасними знаннями, щоби бути конкурентоспроможними на ринку праці. Це завдання вирішується через реформування змісту вищої освіти, яке має відповідати міжнародним стандартам.

Навчальний процес при трансферах передбачає вивчення низки навчальних дисциплін (їх перелік наведено у стандарті ECTS), що входять до складу навчального плану більшості європейських університетів. Ці дисципліни включені до навчального плану більшості європейських університетів, тому вони мають право бути включеними до навчального плану. Крім цього, дуже важливо самоорганізація навчального процесу та способи контролю над виконанням завдань, зокрема таких, які відповідали б сучасним уявленням про вищу освіту в Європі.

Відмінні риси Болонської системи – це тестова модульно-рейтингова система контролю якості вищої освіти. Згідно з розпорядженням Міністерства освіти і науки України, тестування на знання української мови проводитиметься у складі модульно-рейтингових систем як діагностичного, стандартизованого засобів вимірювання якості вищої освіти, що розширює можливості для здобуття необхідної кваліфікації та знань, умінь та навичок зручним для кожної людини способом.

Наразі тестування за новою тестовою модульною системою оцінки досягнень студентів стало частиною нових технологій навчання, які знаходять все більше прихильників у навчальних закладах України. На думку авторів, ця

система значною мірою відповідає новим соціально-правовим умовам, які не мають переваги перед традиційними формами перевірки знань.

Виклад основного матеріалу. Контролювати навчальний процес у вищих навчальних закладах необхідно обов'язково, оскільки це один із способів оцінки. Сьогодні найпопулярнішими є усні перевірки знань (опитування), письмова перевірка знання (письмові роботи), співбесіди з допомогою інтерв'ю. Процес складання іспиту – складний процес підбиття підсумків пізнавальної роботи студента. Для того, щоб виявити рівень знань, умінь та навичок студентів, у вищих навчальних закладах все частіше використовується тестовий контроль знань, який дозволяє виявити рівень знань, умінь та навичок студентів.

У цьому тестовий контроль як оптимізації навчального процесу став широко використовуватися у вузах, і навіть у школах. Раніше тестування було предметом дискусії та обговорювалося в психологічній літературі та педагогіці, зараз же тести визнані як інструмент для перевірки знань. Тести – це що? Тести є одним із видів навчальних завдань, які використовуються для контролю та діагностики знань.

Навчальний процес у школі має бути організований таким чином, щоб учні могли отримати знання про правильний підбір тестів та правильні способи їх здачі. Адже не кожен набір питань з варіантами відповідей можна назвати тестом, тому що тестовий контроль має бути:

- Індивідуальний, що передбачає проведення перевірки, оцінки та обліку власних результатів навчання кожного студента.

- Систематичним, регулярним, що передбачає постійний контроль просування кожного студента у навчанні (семестр, рік).

- Об'єктивним, що передбачає виняток із розгляду суб'єктивних, помилкових суджень (і висновків) викладача, що ґрунтуються на недостатньому вивченні студентів або пристрасті до них та пом'якшення впливу соціальних та особистісних факторів, що супроводжують процес оцінювання.

- Прозорим – студентам необхідно зрозуміти, чому їхня відповідь була оцінена саме такою кількістю балів.

- Всебічним (охоплювати всі розділи програми), що передбачає контроль знань теоретичної частини, предметної та загальної умінь та навичок, інтелектуального рівня та загально психічного розвитку студентів, виявлення їх схильностей та здібностей;

- надійним - результати тестового контролю мають відповідати результатам повторних вимірювань.

- точним, тобто при цьому не повинно бути жодної похибки у вимірі даним тестом.

- якісним, який має бути спрямований як на оволодіння предметними знаннями і вміннями, так і на перевірку розвитку загальнопредметних вмінь (аналізувати, міркувати, робити логічні висновки.

На відміну від традиційних форм і методів тестування, тестова перевірка не вимагає додаткових зусиль з боку викладача, вона вбудована в сучасні педагогічні концепції і дозволяє раціональніше використовувати зворотний зв'язок з учнями та виявляти їх прогалини. При цьому необхідно мати перевірену надійну базу тестів для поєднання їх використання з іншими методами контролю, серед існуючих автоматизованих систем тестування вибрати таку, яка максимально уникала б вгадування та зубріння, виробити об'єктивні, чіткі критерії оцінки, зрозумілі не лише екзамену, а й студенту [19].

1.2 Основні вимоги до тестових завдань

Основні вимоги до тестових завдань:

- повинні належати до однієї теми чи дисципліни;
- бути взаємопов’язаними між собою (послідовність у термінології);
- бути взаємодоповнюючими та упорядкованими за труднощами або за логікою;
- форма тесту повинна бути уніфікованою, звичною, зручною;
- терміни та поняття у тестах повинні бути загальновідомі і відповідати вимогам навчальної програми;
- послідовність тестових завдань повинна визначатися за принципом «від простого до складного»;

– завдання повинні бути стислими.

За кількістю завдань існують стислі (до 20 завдань), середні (від 20 до 500 завдань), довгі (більше 500 завдань) тести.

За рівнем засвоєння знань, умінь, навичок тести класифікують на 3 рівні.

Тести першого рівня розподіляють на тести пізнання, тести розрізнення, тести співвіднесення, тести-завдання з вибірковими відповідями.

У тесті пізнання студент дає одну з альтернативних відповідей («так – ні», «є – не є» тощо). У завданні обов'язково фігурує об'єкт, про властивості або характеристики якого студент повинен мати уявлення.

Тести розрізнення разом із завданням містять відповіді, з яких учень повинен обрати одну чи кілька.

Тести співвіднесення пропонують знайти подібність чи розбіжність у вивчених об'єктах і при цьому порівнювані властивості або параметри обов'язково наявні у завданні. Оформлені таким чином тести мають назву вибіркові.

Тести-завдання з вибірковими відповідями. У завданні формулюється умова завдання та всі необхідні вихідні дані, а у відповідях представлено кілька варіантів результату рішення у числовому або буквеному вигляді.

Студент повинен розв'язати завдання і показати, яку відповідь із запропонованих він одержав.

Застосування тестів першого рівня доцільне для проміжного контролю знань студентів в межах викладання окремого курсу.

Перевірку засвоєння на другому рівні можна здійснювати за допомогою наступних тестів – відтворення інформації, розв'язання типових задач.

За оформленням тести відтворення інформації поділяють на тести-підстановки та конструктивні тести.

У тестових завданнях можуть бути різні завдання (словесний текст, чорновий малюнок або схема), в яких пропущені певні компоненти (наприклад, істотна частина слова/фрази, умовні позначення, лінії або елементи зображення тощо). Для вирішення завдання студент повинен відтворити у пам'яті та

заповнити пропуски.

Щоб отримати завдання конструктивного тесту, студент повинен самостійно сконструювати відповідь (наприклад, відтворити формулювання або виконати креслення).

Тому тести другого рівня слід застосовувати для перевірки знань студентів з основних розділів курсів і без знання яких загальне засвоєння дисципліни є проблематичним або взагалі неможливим. Наприклад, для вивчення дисципліни "Документування" такий контроль необхідний після вивчення теми "Оформлення документів за державним стандартом", оскільки вона входить до базової для подальших тем. Такі ж базові теми при вивченні дисципліни Українською Мовою професійного спілкування є теми Документ як вид ділової мови та Усне ділове мовлення.

При відповіді на тести третього рівня (підсумкові) необхідно використовувати засвоєні навички та вміння у нових умовах, у незнайомій ситуації, у практичній діяльності. При проведенні тестів, які можуть бути використані як на практичних заняттях або при підсумковому контролі за весь пройдений курс, необхідно вміти грамотно скласти та виконати тест.

1.3 Основні переваги перевірки знань студентів за тестами

1. Швидкість обробки одержаних результатів. Але зрештою, за налагодженої технологічної схеми, можна довести цей процес до повної автоматизації та забезпечити максимальну об'єктивність.

2. Адекватна оцінка отриманого результату, його незалежність від тестування. Не варто сприймати цю оцінку як оцінку саме сукупності знань студента, оскільки існує категорія студентів, які традиційно не здатні успішно демонструвати свої знання через тестову методологію.

3. Надійність методу вимірювання – це ступінь стійкості результатів, що впливає на точність, з якою можна виміряти ту чи іншу конкретну ознаку. Перевірка надійності методу стосується насамперед відновлення результатів при повторних вимірах. Ступінь надійності методу визначається за допомогою

коефіцієнта надійності.

Для тестування в IT-компаніях використовуються QA-інженери. Щоб правильно провести тестування, необхідно чітко знати, коли саме за справу візьметься тестувальник (якщо це буде проводитися на робочому місці), а також підготувати план тестування, тестову документацію та тестове оточення.

1.4 Постановка задачі

Метою розробки чат-бота є дати можливість користувачам месенджеру Телеграм, які цікавляться сферою IT отримати тест та дізнатися свій рівень знань у галузі QA. Сучасні та ефективні засоби, які використовуються для проведення масштабних рекламних кампаній, це чат-боти. Використовуючи дані статистики за кількістю користувачів месенджерів, можна зробити висновок про те, що чат-боти набагато ефективніші за будь-які інші інструменти реклами.

А для користувачів це стане набагато легше, тому що їм не доведеться самостійно шукати потрібну інформацію в інтернеті, а також вони зможуть отримувати її за допомогою чат-бота, який автоматично інформуватиме їх про наявність необхідної інформації в мережі.

На основі аналізу предметної області в рамках кваліфікаційної роботи необхідно розглянути тестовий контроль як засіб навчального процесу, дослідити основні вимоги до тестових завдань та переваги перевірки знань студентів за тестами, проаналізувати поведінку чат-ботів під час збоїв в Telegram та технологію розробки чат-боту, а також обрати програмне забезпечення для розробки додатку, та виконати практичну реалізацію чат боту мовою програмування Python.

Популярність месенджерів зростає з кожним днем, і зростання популярності даних програм прогнозується мінімум на найближчі 5-10 років вперед. Самі месенджери - це безкоштовне програмне забезпечення, яке встановлюється на смартфон, для обміну повідомленнями, медіа файлами та іншою інформації

РОЗДІЛ 2 ТЕХНОЛОГІЯ РОЗРОБКИ ЧАТ БОТУ

2.1 Технологія розробки чат-боту

Чат-боти – це спеціалізовані частини програмного забезпечення, які покликані допомогти користувачам виконувати різні завдання шляхом впровадження машинного навчання і штучного інтелекту. Це означає, що чат-бот зможе взаємодіяти з вашими співробітниками або клієнтами і допомагати їм без допомоги іншої людини.

Чат-боти виявилися надійними інструментами, які можна використовувати для збільшення продажів і поліпшення маркетингових зусиль. Сьогодні сотні компаній вже використовують можливості віртуальних помічників. Більшість з цих компаній використовують ботів, щоб допомогти споживачам, інші ж компанії знайшли їм застосування в своїх особистих цілях.

Основна причина, по якій боти використовуються, і їх найбільш очевидне застосування, полягає в поліпшенні комунікацій у всій вашій компанії. Крім того, вони здатні виконувати величезну кількість завдань, які мають вплив на вашу організацію.

При цьому щоб створити успішного бота, Ви повинні інвестувати значну кількість часу і зусиль в його розробку і оптимізувати його на регулярній основі.

Створення власного чат-бота

Більшість підприємств малого бізнесу зупиняються на рішенні про купівлю чат-ботів. Існують альтернативні способи, через різні редактори, або так звані «коробкові» рішення, але це мало ефективно. По-перше, розробка бота на замовлення дозволяє адаптувати його абсолютно під будь-яке завдання і користувача. По-друге, замовлення бота у досвідчених розробників, дозволить вам не думати про зміст серверів та іншого.

Як чат-боти змінять роботу всередині бізнесу

Telegram вже є ефективним бізнес-додатком, і таке використання чат-бота може перетворити його в найважливіший інструмент, який буде у ваших співробітників. Використання бота повністю змінить спосіб роботи компаній, оскільки вони зможуть:

1. Покращувати зв'язок

Компанії, які використовують Telegram, часто мають гарний зв'язок з клієнтами, і використання ботів може поліпшити це в усіх напрямках включаючи внутрішню взаємодію співробітників. Замість того, щоб в залежності від членів вашої команди відповідати на запити клієнтів або призначати зустрічі, бот може допомогти організувати зустрічі та надати миттєво інформацію[20].

2. Централізувати завдання

Хоча чат-бот не повинен замінювати важливі компоненти програмного забезпечення, такі як CRM, він може допомогти централізувати всі бізнес-програми. Замість того щоб перемикатися з однієї програми на іншу, ваші співробітники зможуть задати питання безпосередньо вашому боту або просто попросити його виконати будь-які завдання, які їм потрібні.

3. Синхронізувати все відділи

Ефективні бізнес-моделі розбивають свої операції на конкретні відділи. У більшості випадків кожен відділ звертається незалежно, що може створювати проблеми синхронізації і приводити до негативних результатів. Бот може синхронізувати всі відділи, відправляючи масові повідомлення, що підтверджують їх доставку і відповідають на будь-які питання про будь-які зроблені зміни.

4. Надати особистого віртуального помічника для вашої команди

Традиційно тільки лідери в бізнесі мали доступ до особистих помічників. Однак, якщо ви створите бота, ви дозволите всім своїм співробітникам мати спеціального помічника, щоб допомогти їм у вирішенні своїх завдань, не наймаючи нових членів команди.

5. Проаналізувати більше даних

Дані відіграють життєво важливу роль в успіху більшості компаній. При цьому вам часто доводиться наймати аналітиків і виділяти час для інтерпретації цих результатів. Бот може допомогти Вам, сортуючи і аналізуючи дані протягом декількох годин або хвилин. Найкраща частина полягає в тому, що вони зможуть представити свої висновки логічним чином для осіб, які приймають рішення або для всієї організації, в міру необхідності.

6. Спростити завдання

Існує багато задач, які повторюються, але необхідні для операцій будь-якої компанії. Бот може допомогти спростити ці завдання, тому що він може виконувати більшість, якщо не всі, з них. Вам просто потрібно налаштувати свого бота, щоб автоматично подбати про виконання необхідних завдань або дати своїм роботам прямі інструкції, коли вам потрібно, щоб ці дії були виконані.

Якщо ви шукаєте найбільш ефективний спосіб підвищити загальну продуктивність своєї компанії, то впровадження чат-бота може допомогти вам організувати свій бізнес.

Правильне і своєчасне впровадження нових технологій – це те, без чого не може прожити бізнес. Якщо вчасно не задіяти корисну новинку - це обов'язково зроблять конкуренти. І в особі потенційних і діючих клієнтів саме вони будуть виглядати більш прогресивними.

Одним з останніх трендів є використання телеграм-ботів. Їх впровадження дозволяє отримати серйозні позитивні ефекти:

- Залучення нових клієнтів.
- Розвантаження персоналу.
- Утримання клієнтської бази. Розглянемо їх докладніше.

Залучення нових клієнтів через чат-ботів

Телеграм-бот може взяти на себе завдання, пов'язані з першим контактом клієнта з компанією. У нього можна закласти такі функції:

- збір контактів (телефонів, електронних адрес, акаунтів в соцмережах);
- збір готових заявок (на покупку або послугу);
- відправка актуальних цін;
- відповіді на поширені питання.

У чому плюс чат-бота для Telegram:

- бот працює автономно, вимагаючи уваги співробітника тільки тоді, коли клієнт вже «теплий», склав перше враження про компанію, і зацікавився продуктом / послугою;

- бот працює цілодобово, без перерв і вихідних - компанія не упускає

клієнта, навіть якщо він захоче зробити замовлення або дізнатися якусь інформацію в новорічну ніч [17].

Розвантаження персоналу за допомогою чат-ботів

- На спілкування з клієнтами йде значна кількість часу співробітників.

- Причому питання у нових користувачів в більшості своїй повторюються. Як результат - співробітнику доводиться виконувати одну і ту ж роботу постійно, даючи одні й ті ж відповіді [24].

2.2 Вибір програмного забезпечення розробки чат-бота

У сучасному світі дуже багато зав'язано на IT-технологіях, практично в будь-якій компанії працівники використовують різні програми для ефективної та якісної роботи, світ не стоїть на місці, а розвивається і тематика даної статті обумовлена тим, що в рамках дипломної роботи є ідея написання чат- робота в месенджері Telegram, а для написання чат-бота необхідно вибрати мову програмування від якої залежатиме швидкість написання коду, можливості мови тощо.

Мова програмування - це набір правил, які визначають, як виглядає написана комп'ютерна програма і що комп'ютер може виконувати під її контролем. Програма – це код, написаний відповідно до правил цієї мови програмування. Код, з якого складається програма, називається вихідним кодом.

Мови програмування – це формальні штучні мови. Як і природні мови, вони мають алфавіт, словниковий запас, граматику та синтаксис, а також семантику.

Всі мови програмування поділяються на два види - мови низького та високого рівня:

- Мови низького рівня – це спосіб написання комп'ютерних інструкцій апаратною мовою, тобто в машинних кодах (у вигляді послідовності нулів та одиниць). Мови низького рівня жорстко орієнтовані конкретний тип устаткування (система управління процесором, кожен тип процесора має власний машинний код).

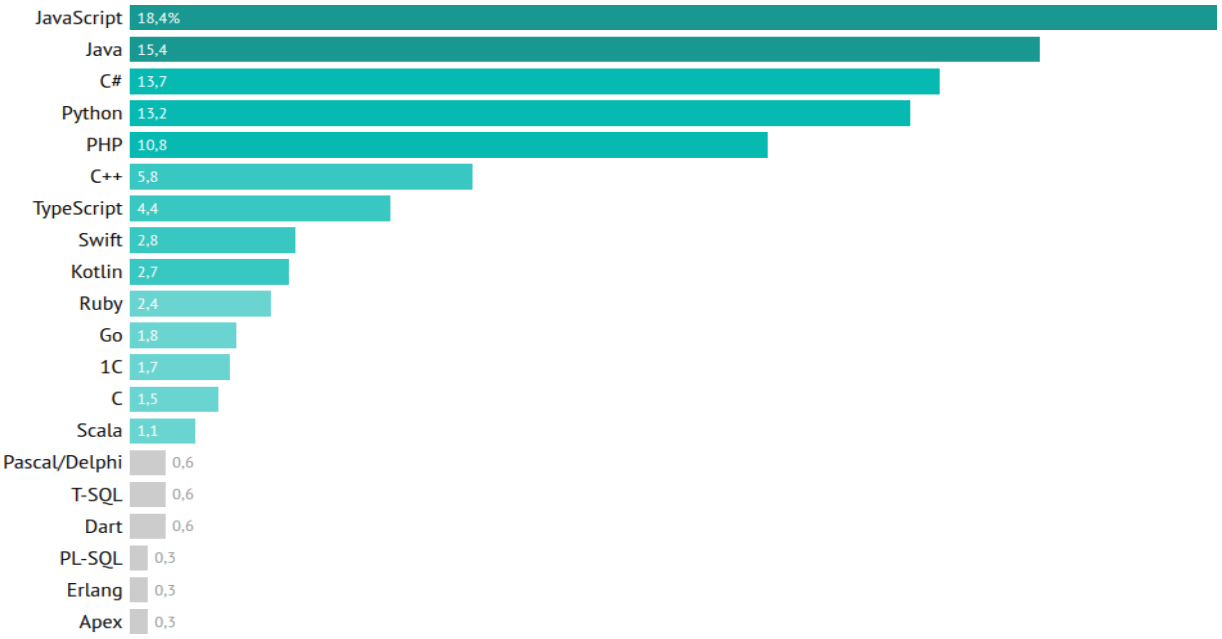
- Мови високого рівня – це мови програмування, які дозволяють записувати програми у зручній для людини формі. Ці мови орієнтовані не так на систему інструкцій тієї чи іншої процесора, але в систему операторів

(інструкцій), характерну написання певного класу алгоритмів.

Мови високого рівня простіше у використанні, оскільки їхнє завдання - обслуговувати потреби програміста, а не визначати можливості комп'ютера. Програми, написані на цих мовах, повинні бути перекодовані - перекладені машинною мовою, щоб перед запуском програм комп'ютер міг їх зрозуміти. Тому системи програмування на Java включають або інтерпретатор мови, або компілятор.

Мови низького рівня, близькі до машинної мови, дозволяють створювати програми, які працюють швидше і дозволяють більш ефективно використовувати ресурси комп'ютера. найкращої мови для написання чат-бота, необхідно вибрати кілька найпопулярніших мов, щоб між ними проводити аналіз, тож у цьому розділі звернемося до статистики за популярністю мов.

На рисунку 2.1 показаний рейтинг мов 2020 у комерційних робочих



проектах.

Рис. 2.1. Рейтинг мов програмування у комерційних проектах, %

За даними можна зробити висновок, що JavaScript значно випереджає Java і є найпопулярнішою мовою програмування. До п'ятірки найкращих мов увійшли також: C#, Python, PHP. На малюнку 7 можна побачити, як змінювалися дані з 2012-2020 років.

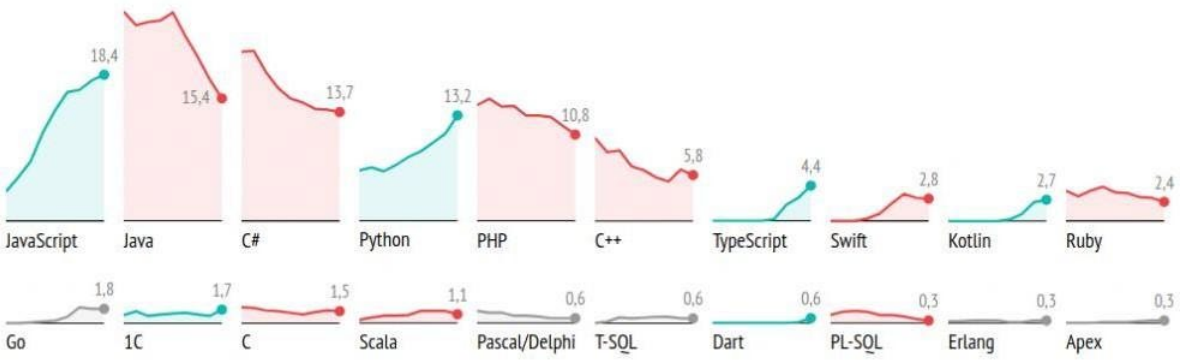


Рис. 2.2. Графік зміни популярності мов програмування, %

На малюнку 2.2 можна дійти невтішного висновку, що популярність мови Java і C# сильно падає, а популярність таких як JavaScript, TypeScript і Python продовжує зростати.

Вывод: итак, по данным, представленным выше самыми популярными языками на данный момент, являются JavaScript, Python и C#, поэтому для анализа и сравнения выберем именно эти языки.

2.3 Працездатність чат-ботів під час збоїв Telegram

Telegram - найнадійніший месенджер, що славиться своєю безпекою і захищеністю. За це він так і подобається мільйонам користувачів по всьому світу.

Проте, часом з'являються повідомлення про збої в Telegram, та й кожен з нас напевно стикався з такою проблемою: повідомлення не доставляються і не приходять, боти не відповідають, канали не оновлюються. Найчастіше проблеми виникають на території декількох країн, але декілька раз спілкування в месенджері було недоступно по всьому світу. Варто віддати належне інженерам і розробникам Telegram, з часом це трапляється все рідше і рідше, але повністю виключити ймовірність виникнення проблем не можна. Від випадковостей ніхто не застрахований, але все ж давайте розглянемо таку тему: « що відбувається, коли Telegram падає, і як поводяться боти під час таких форс-мажорів?»

Географічні збої

Збої бувають різними: іноді, наприклад, якась частина серверів,

розташованих в одному географічному регіоні, з якоїсь причини перестає відповідати. Причин тому може бути декілька: падіння напруги, високе навантаження, фізичні ушкодження. В цьому випадку постраждати можуть користувачі і боти, зареєстровані в країнах, де стався збій, а також канали і групи, власники яких знаходяться в зоні ураження. Наприклад, якщо ваш бот хоститься де-небудь в Польщі, а у Telegram трапився збій в східній Європі, то запити, що приходять на сервера в зоні ураження, не отримують відповіді і «зависають».

Повідомлення користувачам будуть доставлені, як тільки буде відновлена працездатність серверів, а ось боти можуть реагувати по-різному: не отримавши відповіді на запит, програми з низькою стійкістю до відмов можуть зациклитися або вилетіти зовсім, в той час як більш надійні аналоги прийдуть до тями після збою самостійно.

Внутрішні збої

Крім регіональних збоїв, трапляються відмови внутрішніх частин Telegram, не прив'язаної до певного регіону. Так, «впасти» може API Telegram, тобто, безпосередньо механізм взаємодії з ботами. Тоді звичайні користувачі по всьому світу не відчувають труднощів при відправці повідомлень, групи і канали працюють як належить, а ось боти не отримують відповідей від сервера Telegram і завмирають. Звичайна людина, яка відправляє у цей час запит в бота, не побачить причини, і буде просто сердитися на розробників, хоча, на жаль, вони ніяк не можуть вплинути на таку ситуацію. Бот, який не отримав відповіді від API, буде вести себе по-різному в залежності від налаштувань: якщо при розробці не пропрацювати механізм реагування на таку ситуацію, програма може "впасти" і відновлювати працездатність доведеться вручну.

Тайм-аут і реакція на нього

Іноді збої бувають мінорними: виходить з ладу один з серверів великого кластера. Загальна працездатність системи зберігається, але час відповіді на запити збільшується, з точки зору користувача Telegram просто "тупить".

Здавалося б, нічого страшного немає, відповідь адже приходить, просто з затримкою, хіба бот може зламатися від такого? Виявляється, що і таке можливо: якщо під час написання програми не налаштувати поведінку при

тривалому тайм-ауті, то бот може впасти в ступор і не обробляти запити.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1. Функціонал роботи чат-бота

Так як, уже відомо як саме відбувається взаємодія між ботом та сервером, необхідно розробити функціонал. Оскільки цей бот для організації дистанційного навчання, то він продемонструє всі шляхи реалізації обміну даними та файлами між користувачем та викладачем. Тому було вирішено розробити два шляхи реалізації обміну файлами. Перший, який матиме викладач, це обмін за допомогою посилань на вебсервіси. Через те, що лекції відбуваються дистанційно за допомогою відеотелефонного зв'язку, це виявилось гарною можливістю створювати та зберігати запис. Викладач має змогу надіслати посилання на лекцію, або відправити посилання із заздалегідь готовим записом за допомогою чат-бота. Також за бажанням, додати посилання на документи або певні матеріали. Використання саме такого типу обміну даними має низку переваг:

- Інформація в безпеці. При роботі в документі сервіс фіксує будь-яку зміну.
- Доступ з будь-якого місця. Почати працювати в документі можна з одного пристрою, а закінчити на іншому.
- Історія змін. Дає змогу повернутись до попереднього варіанту.
- Спільна робота в документі. Можливість певній категорії людей дозволити редагувати документ, а для інших залишити доступним лише перегляд.
- Можливість залишати в роботі коментарі.
- Економія місця. Всі документи знаходяться у хмарному сховищі.

Але хоч всі можливі сервіси намагаються швидше перейти в режим дистанційного доступу: проведення лекцій, трансляцій, створення документів тощо, але також , ще досі люди користуються створенням файлів на персональних пристроях. Тому в чат-боті продемонстрована можливість, зі

сторони студента, надіслати файли викладачу. За правилами Telegram доступна передача файлів обсягом до п’яти мегабайтів. Також цей бот містить підключену базу даних, для зберігання документів. Як приклад, якщо необхідно розширити можливості цього бота і зробити його для декількох викладачів, і великого потоку студентів.

Оскільки, Telegram бота можна створити на будь-якій серверній мові програмування, так само до нього можна і під’єднати будь-яку базу даних. В залежності від завдання, і необхідної кількості зберігання інформації, на цю мить момент існує широкий вибір баз даних. Для створення чат-бота для організації дистанційного навчання, вибір був зроблений на користь MongoDB. MongoDB - це документно-орієнтована база даних. Завдяки цьому вона працює швидше, має кращу масштабованість, не містить схем, таблиць, зовнішніх ключів та і в цілому досить сильно відрізняється від об’єктно-реляційної бази даних. Спосіб зберігання даних в MongoDB в цьому плані схожий на JSON, хоча формально JSON не використовується. Для зберігання в MongoDB застосовується формат, який називається BSON або скорочення від binary JSON. BSON дозволяє працювати з даними швидше: швидше виконувати пошук і обробку інформації. MongoDB написана на C++, тому її легко перенести на найрізноманітніші платформи. MongoDB може бути розгорнута на платформах Windows, Linux, MacOS, Solaris. Якщо реляційні бази даних зберігають рядки, то MongoDB зберігає документи. На відміну від рядків документи можуть зберігати складну за структурою 32 інформацію. Документ можна уявити як сховище ключів і значень. Ключ - це мітка, з яким асоційована певна частина даних. Також MongoDB та у реляційних баз даних є спільна ознака. У реляційних базах використовується таке поняття як первинний ключ, який являється унікальним ідентифікатором певних полів. Схоже поняття існує і в MongoDB, та називається _id. Різниця полягає в тому, що в реляційних базах є можливість привласнити пустому полю значення NULL, натомість у MongoDB, якщо ключ не отримує значення, то він не використовується у документі. У MongoDB замість таблиць створюються колекції. Відрізняються вони від таблиць тим, що можуть містити об’єкти, які мають різну структуру і різний набір властивостей. Натомість таблиці зберігають лише однотипні жорстко структуровані об’єкти. Для

зберігання даних, MongoDB представляє набір реплік. У цьому наборі є основний вузол, а також може бути набір вторинних вузлів. Всі вторинні вузли зберігають цілісність і автоматично оновлюються разом з оновленням головного вузла. Якщо основний вузол виходить із ладу, то один з вторинних вузлів стає головним. На відміну від реляційних баз даних, MongoDB дозволяє зберігати різні документи з різним набором даних, однак при цьому розмір документа обмежується 16 Мб. Але MongoDB пропонує рішення - спеціальну технологію GridFS , яка дозволяє зберігати дані за розміром більше, ніж 16 Мб. Система GridFS складається з двох колекцій. У першій колекції, яка називається files, зберігаються імена файлів, а також їх метадані, наприклад, розмір. А в іншій колекції, яка називається chunks, у вигляді невеликих сегментів зберігаються дані файлів, зазвичай сегментами по 256 Кб.

Загалом чат-бот, написаний певною мовою програмування, являється серверним додатком, в якому функції чату працюють через власний API. Для того, щоб створити такого бота необхідна певна інфраструктура: хостинг, сервер (фізичний або хмарний) та база даних. Можливості таких чат-ботів обмежуються лише можливостями платформи, на яку вони інтегруються. Натомість чат-бот, який створюється власноруч, за допомогою конструктора, обмежується особливостям сервісу на якому він створюється. Перед тим як створити чат-бота необхідно зрозуміти як саме відбувається взаємодія між користувачем та сервером. Повідомлення, команди і запити, надіслані користувачами, передаються на програмне забезпечення, запущене на серверах розробників. Сервер Telegram, який являється посередницьким та анонімним, обробляє шифрування і здійснює зворотний зв'язок між користувачем і утилітою.

Взаємодія між користувачем і ботом виглядає наступним чином: 1. Користувач надсилає боту команду 2. Бот передає команду на сервер 3. Програма на сервері оброблює отриманий від бота запит 4. Сервер віддає відповідь боту 5. Бот виводить відповідь користувачеві на вікні керування. І цей цикл повторюється раз за разом під час взаємодії з будь-яким телеграмботом. Спілкування відбувається за допомогою простого HTTPS-інтерфейсу, який є спрощеною версією API Telegram. Інакше цей інтерфейс можна назвати програмним каталогом або бот-алгоритмом. Схема обміну даними між

користувачем і Telegram ботом наведена на рис. 3.1.

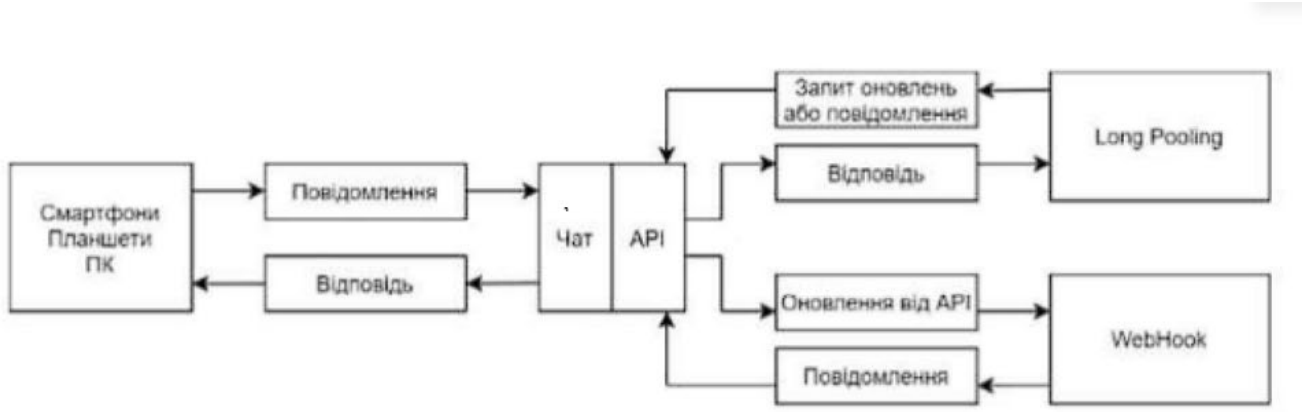


Рис. 3.1. Схема обміну даними між користувачем і Telegram ботом

3.2 Практична реалізація чат-бота мовою програмування Python

Отже, чому я вирішив використовувати aiogram в розробці чат-бота? Тому що на даний момент це найпопулярніша та найефективніша бібліотека для розробки ботів. Головна її перевага - це асинхронність. Що таке асинхроність [8]?

Асинхронність – це буквально реалізація наступного процесу, не чекаючи завершення першого. Наприклад, коли ви вирішили завантажити великий файл, ви не чекаєте завершення завантаження, а водночас можете робити інші дії [3].

Чому не багатопоточність чи багатопроцесорність?

Тому що для мене вони набагато складніші у реалізації, а також у них іноді виникають конфлікти, поява яких складно передбачити [10]. Наприклад, багатопроцесорність на Python має таку особливість як GIL(Global Interpretator Lock) [7]. GIL блокує доступ до пам'яті, якщо вона використовується десь. В aiogram це реалізовано дуже зручно – у бота є диспетчер, який приймає всі можливі тригери – повідомлення від користувача, натискання на кнопку тощо. Ми можемо налаштовувати будь-які фільтри на ці тригери і робити потрібні нам дії.

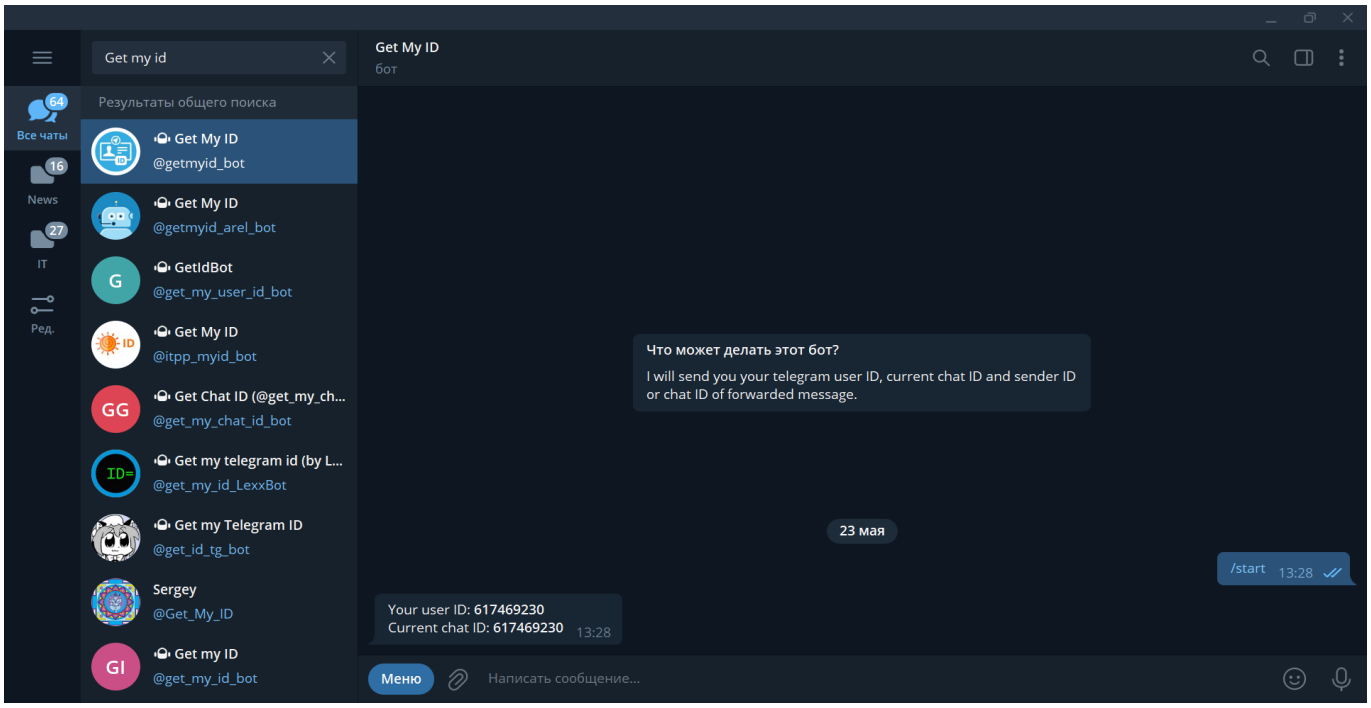


Рис. 3.2. Отримання id за допомогою бота Get_my_id

Далі нам потрібно запам'ятати наш ID та відкрити файл bot.ini. Потім міняємо ID у файлі на наш ID і при бажанні можемо змінити пароль.

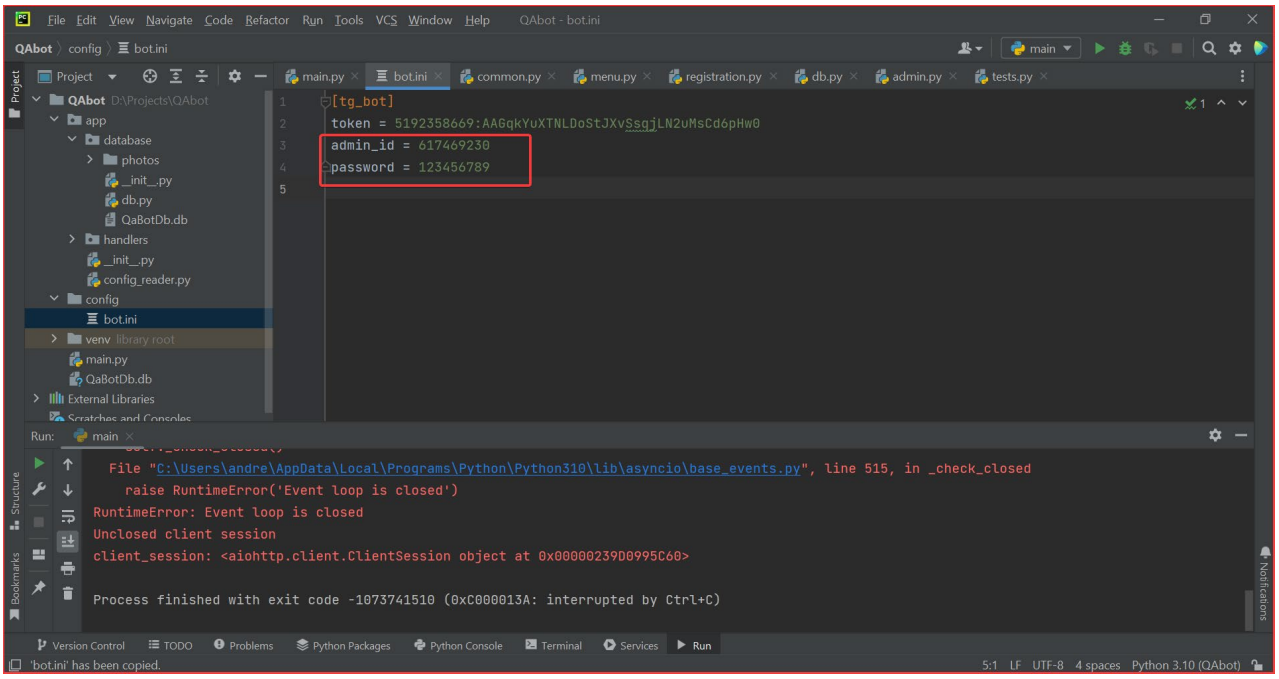


Рис. 3.3. Заміна id у файлі bot.ini

Це називається файл конфігурації, де зберігаються основні змінні. Він написаний окремий файл(config_reader.py), який зчитує ці змінні.

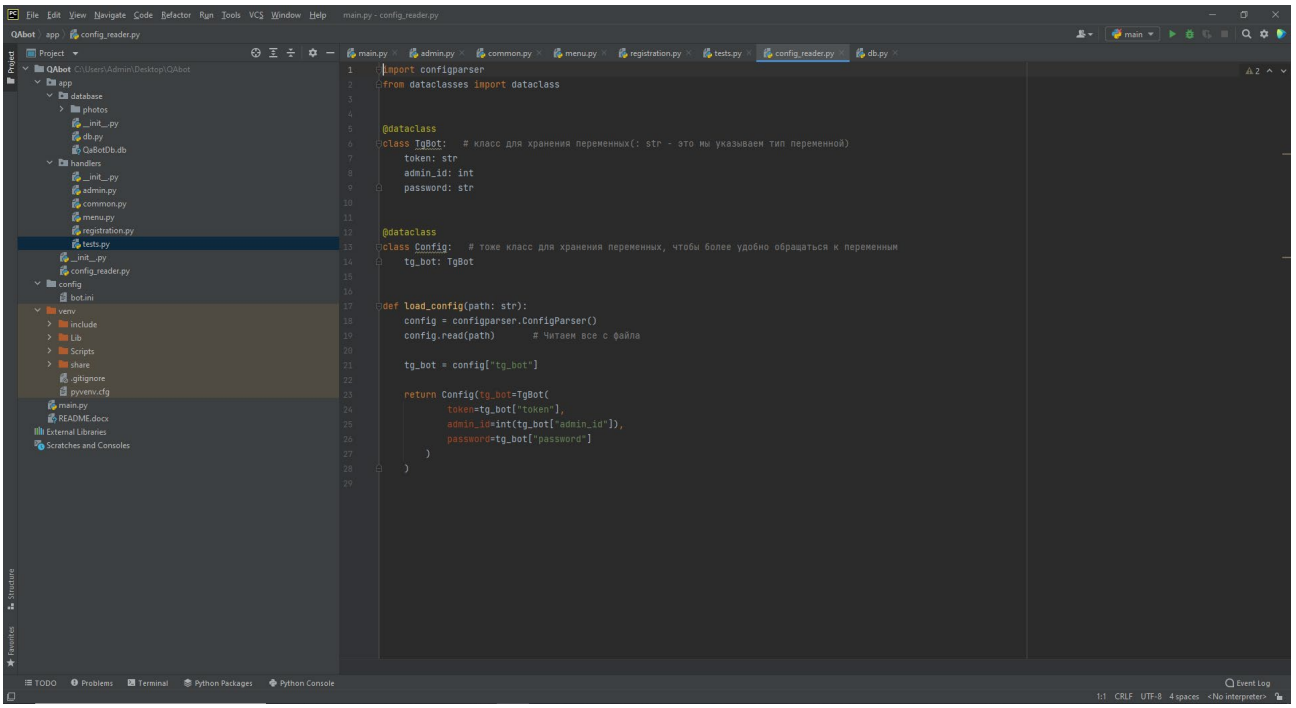


Рис. 3.4. Файл конфігурації, де зберігаються основні змінні.

main.py – головний файл, звідки все починається. У ньому ініціалізується сам бот і запускається. Також у ньому ініціалізуються все фильтры(Хендлеры).

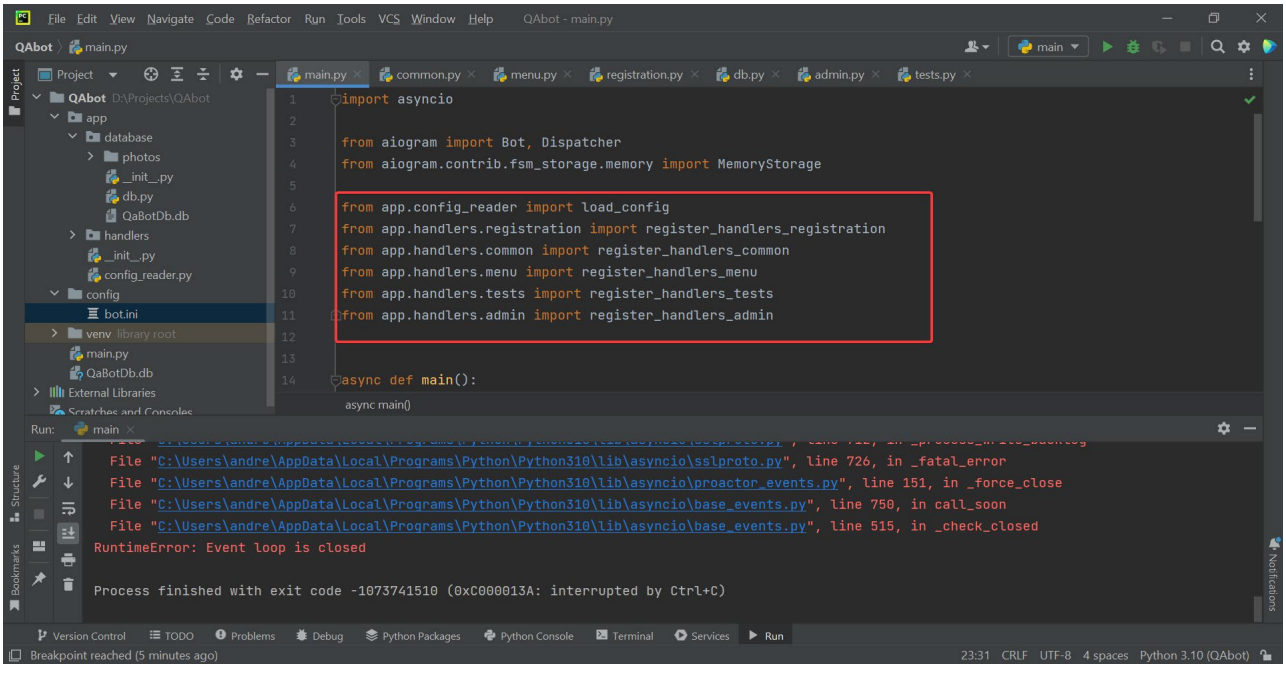


Рис. 3.5. Імпортуємо всі хендлери.

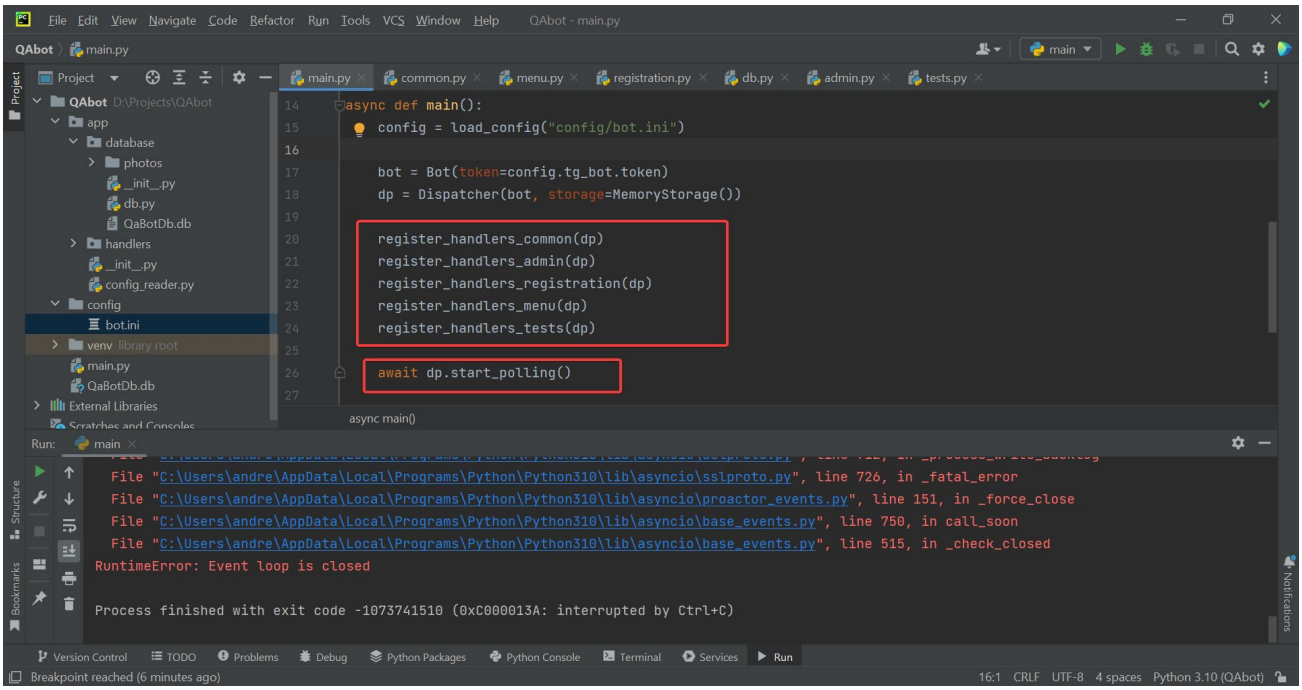


Рис. 3.6. Ініціалізуємо всі хендлери та запускаємо бота.

Інші файли лежать у пакетах (папках). Для того щоб зробити їх пакетами в кожній папці, створюється порожній файл __init__.py . Це потрібно для зручного імпорту потрібних функцій.

common.py - тут лежать функції, що реагують на команди /start/get_admin_status/contacts

```
from aiogram import Dispatcher, types, Bot
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters import Text
from aiogram.dispatcher.filters.state import State, StatesGroup
from app.config_reader import load_config
from app.database.db import user_exists, call_to_admin, add_admin
config = load_config("config/bot.ini")
bot = Bot(token=config.tg_bot.token)
class Order(StatesGroup):
    text = State()
class GetAdminStatus(StatesGroup):
    password = State()
async def set_commands(mode: bool):
    commands = [
        types.BotCommand(command="/registration", description="Зареєструвати користувача"),
        types.BotCommand(command="/contact", description="Зв'язатися с адміном"),
```

```
        types.BotCommand(command="/get_admin_status", description="Отримати статус адміна")
    ]
    admin_commands = [
        types.BotCommand(command="/admin_menu", description="Меню адміна"),
        types.BotCommand(command="/topic_statistics", description="Отримати статистику по темам"),
        types.BotCommand(command="/age_statistic", description="Отримати вікову статистику"),
        types.BotCommand(command="/problem_statistic", description="Отримати статистику помилок")
    ]
    if mode:
        await bot.set_my_commands(commands)
    else:
        await bot.set_my_commands(admin_commands)
async def cmd_start(message: types.Message, state: FSMContext):     # Доделать проверку в базе
    await state.finish()
    await message.answer(
        "Привіт",
        reply_markup=types.ReplyKeyboardRemove()
    )
    await message.answer("Почекай, я перевірю твою наявність в базі")
    if not user_exists(message.from_user.id, "users"):
        await message.answer("Тебе немає в базі \n\n"
                             "Отож давай тебе зареєструємо!\n\n"
                             "Натисни /registration")
        await set_commands(True)
    elif user_exists(message.from_user.id, "admins"):
        await message.answer("Вітаю, ти адмін!\n\n"
                             "Натисни /admin_menu")
        await set_commands(False)
    else:
        await message.answer("Ти вже зареєстрований\n\nНатисни /menu")
        await set_commands(True)
async def contacts(message: types.Message):
    buttons = [
```

```
        types.InlineKeyboardButton(text="Написати повідомлення",
callback_data="call_to_admin")
    ]
    keyboard = types.InlineKeyboardMarkup()
    keyboard.add(*buttons)
    await message.answer("Якщо у тебе виникли питання - напиши повідомлення адмінам",
reply_markup=keyboard)
async def wait_text(call: types.CallbackQuery):
    if not user_exists(call.from_user.id, "blocklist"):
        await Order.text.set()
        await call.message.answer("Напиши своє повідомлення\n\nВоно буде відправлено всім
адмінам")
    else:
        await call.message.answer("Пробач, але ти заблокований")
async def send_message_to_admin(message: types.Message, state: FSMContext):
    call_to_admin("INSERT", message.from_user.id, message.from_user.username, message.text)
    await message.answer("Добре, я відправив твоє повідомлення")
    await state.finish()
async def get_admin_status_state(message: types.Message):
    if user_exists(message.from_user.id, "blocklist"):
        await message.answer("Пробач, але тебе заблокували, як адміна")
    elif not user_exists(message.from_user.id, "admins"):
        await message.answer("Введи пароль")
        await GetAdminStatus.password.set()
    else:
        await message.answer("Ви вже адмін\nНатисни /admin")
async def get_admin_status(message: types.Message):
    if message.text == config.tg_bot.password:
        add_admin(message.from_user.id)
        await message.answer("Вітаю, ви адмін!\n\nПерейти в меню адміна - /admin_menu")
        await bot.send_message(config.tg_bot.admin_id, f"Користувач
@{message.from_user.username}\n"
                                                       f"ID: {message.from_user.id}\n\n"
                                                       f"Увійшов, як адмін")
    else:
        await message.answer("Пароль неправильний")
def register_handlers_common(dp: Dispatcher):
```

```
    dp.register_message_handler(cmd_start, commands="start", state="*")
    dp.register_message_handler(contacts,
                                Text(equals="Контактная інформація ☎", ignore_case=True),
                                commands="contact",
                                state="*")
    dp.register_callback_query_handler(wait_text, Text(startswith="call_to_admin"))
    dp.register_message_handler(send_message_to_admin, state=Order.text)

    dp.register_message_handler(get_admin_status_state, commands="get_admin_status",
state="*")
    dp.register_message_handler(get_admin_status, state=GetAdminStatus.password)
```

Тепер детальніше. У функції cmd_start ми підключаємо всі фільтри. Усередині їх ми можемо записати будь-які фільтри (команда, текст, стан (* - це означає, що викликати при всіх станах))

Також тут є хедлери не тільки на повідомлення, а й на callback. Він реагує, коли ти натискаєш на інлайн кнопку (яка прикріплена до повідомлення).

А тепер про статки. У ботах можна реалізувати таку функцію як кінцеві автомати. Це коли функції послідовно викликаються. Для цього ми імпортуємо FSMContext, State, StatesGroup.

Як реалізуються ці автомати?

```
class GetAdminStatus(StatesGroup):
    password = State()
```

Створюємо дочірній клас, і записуємо всі стани (тут один стан, тобто одна функція викличе цей стан, а друга відреагує на нього)

Викликаємо цей стан. В даному випадку у нас все так працює: людина тисне команду /contacts на цю команду реагує одна функція і відправляє йому повідомлення з кнопкою (надіслати повідомлення). Користувач тисне на кнопку і тут вже реагує ця функція і викликає стан. А наступна функція, яка реагує на викликаний стан спочатку прийме повідомлення від користувача, яке він напише та збереже його в основу.

Далі menu.py - у ньому зберігаються всі основні функції меню для користувача.

```
from aiogram import Dispatcher, types
from aiogram.dispatcher.filters import Text
from aiogram.dispatcher import FSMContext
from app.database.db import get_profile, get_progress, get_themes, get_sources, get_rating
menu_buttons = [
    "Тести 📱",
    "Мій профіль 📄",
    "Рейтинг 🏆",
    "Контактная інформація 📞"
]
async def cmd_menu(message: types.Message, state: FSMContext):
    await state.finish()
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in menu_buttons:
        keyboard.add(name)
    await message.answer("Меню", reply_markup=keyboard)
async def profile(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add("Моя успішність ✏", "Меню")
    profile_list = get_profile(message.from_user.id)
    percent = round(profile_list[4]/profile_list[5] * 100, 3) if profile_list[4] else 0
    await message.answer(
        f"""<u>Інформація про користувача</u>\n
        ID: {profile_list[1]}
        ІП: {profile_list[2]}
        Вік: {profile_list[3]} років
        Дата та час реєстрації: {profile_list[6]}\n
<u>Загальна успішність</u>\n
        Всього розв'язано тестів: {profile_list[5]}
        Кількість правильних відповідей: {profile_list[4]}
        Процент успішності: {percent} %
        """,
        reply_markup=keyboard,
        parse_mode=types.ParseMode.HTML
    )
async def progress(message: types.Message):
```

```
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add("Робота над помилками", "Меню")
    progress_list = get_progress(message.from_user.id)
    themes_list = get_themes()
    # print(progress_list)
    message_text = ""
    for n, item in enumerate(progress_list):
        percent = round(item[2] / item[3] * 100, 3) if item[3] else 0
        message_text += f"<u>Тема №{item[1]}</u> {themes_list[n][0]}\n" \
                        f"Всього розв'язано тестів: {item[3]}\n" \
                        f"Кількість правильних відповідей: {item[2]}\n" \
                        f"Процент успішності: {percent} %\n\n"
    await message.answer(
        message_text,
        reply_markup=keyboard,
        parse_mode=types.ParseMode.HTML
    )
async def sources(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add("Меню")
    progress_list = get_progress(message.from_user.id)
    themes_list = get_themes()
    zero_list = []
    problems_list = []
    for n, item in enumerate(progress_list):
        if item[3] == 0:
            zero_list.append(n + 1)
        elif item[2]/item[3] < 0.6:
            problems_list.append(n + 1)
    sources_list = get_sources(problems_list)
    message_text = "Теми для опрацювання:\n\n"
    for n, item in enumerate(sources_list):
        message_text += f"Тема №{problems_list[n]} {themes_list[n][0]}\n"
        for num, source in enumerate(item):
            message_text += f"{num + 1}. {source[0]}\n"
        message_text += "\n"
    message_text += f"Також рекомендую пройти тести з тем:\n"
```

```
    for item in zero_list:
        message_text += f"{item}. {themes_list[item - 1][0]}\n"
    await message.answer(message_text, reply_markup=keyboard)
async def rating(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add("Меню")
    rating_list = get_rating()
    message_text = "Рейтинг користувачів\n\n"
    for n, item in enumerate(rating_list, 1):
        percent = round(item[1]/item[2] * 100, 2)
        message_text += f"{n}. {item[0]}  ({item[1]}/{item[2]}) {percent} %\n"
    await message.answer(message_text, reply_markup=keyboard)
def register_handlers_menu(dp: Dispatcher):
    dp.register_message_handler(cmd_menu, commands="menu", state="*")
    dp.register_message_handler(cmd_menu, Text(equals="Меню", ignore_case=True))
    dp.register_message_handler(profile, Text(equals="Мій профіль 📃", ignore_case=True))
    dp.register_message_handler(progress, Text(equals="Моя успішність ✏", ignore_case=True))
    dp.register_message_handler(sources, Text(equals="Робота над помилками",
ignore_case=True))
    dp.register_message_handler(rating, Text(equals="Рейтинг 🏆", ignore_case=True))
```

tests.py - функції для проходження тестів.

```
from aiogram import Dispatcher, types
from aiogram.dispatcher.filters import Text
from app.database.db import get_last_test, get_answer_test, save_test_point, get_themes
async def send_other_tests(call: types.CallbackQuery, number: int):
    buttons = [
        types.InlineKeyboardButton(text="А", callback_data=f"{number}_answer_a"),
        types.InlineKeyboardButton(text="Б", callback_data=f"{number}_answer_b"),
        types.InlineKeyboardButton(text="В", callback_data=f"{number}_answer_c"),
        types.InlineKeyboardButton(text="Г", callback_data=f"{number}_answer_d")
    ]
    keyboard = types.InlineKeyboardMarkup(row_width=4)
    keyboard.add(*buttons)

    number_photo = get_last_test(number + 1, call.from_user.id)
    if number_photo:
```

```
        photo = open(r'app\database\photos\{}.png'.format(number_photo), 'rb')
        await call.message.answer_photo(photo=photo, caption="Обери правильну відповідь",
reply_markup=keyboard)
    else:
        await call.message.answer("Ти вже вирішив всі тести з данної теми")
async def send_themes(message: types.Message):
    buttons = []
    for n, item in enumerate(get_themes()):
        buttons.append(types.InlineKeyboardButton(text=item[0], callback_data=f"theme_{n}"))
    keyboard = types.InlineKeyboardMarkup(row_width=1)
    keyboard.add(*buttons)
    await message.answer("Обери тему", reply_markup=keyboard)
async def send_test(call: types.CallbackQuery):
    await send_other_tests(call, int(call.data[-1]))
async def callbacks_num(call: types.CallbackQuery):
    if get_answer_test(call.from_user.id, int(call.data[0])) == call.data[-1]:
        await call.answer("Правильно!")
        save_test_point(call.from_user.id, 1, int(call.data[0]) + 1)
        await call.message.edit_caption("Ви надали правильну відповідь")
    else:
        await call.answer("Не правильно!")
        save_test_point(call.from_user.id, 0, int(call.data[0]) + 1)
        await call.message.edit_caption("Ви надали неправильну відповідь")
    await send_other_tests(call, int(call.data[0]))
def register_handlers_tests(dp: Dispatcher):
    dp.register_message_handler(send_themes, Text(equals="Тести 📝", ignore_case=True))
    dp.register_callback_query_handler(callbacks_num, Text(contains="_answer_"))
    dp.register_callback_query_handler(send_test, Text(startswith="theme_"))
```

Спочатку користувач натискає потрібну кнопку в меню, робота йому надсилає інлайн-клавіатуру з темами. Користувач вибирає одну з них і розпочинається вирішення тестів. Для того щоб відправити перший тест мені довелося зробити окрему функцію, яка запускатиме цикл тест - відповідь - тест.

Тепер, як створюються інлайн клавіатури. Спочатку створюється об'єкт keyboard. Потім до нього додаються інлайн кнопки (я зробив спочатку список з кнопок і потім відразу все додав до клавіатури). Потім потрібно надіслати

повідомлення із прикріпленою клавіатурою. Існують окремі функції для відправки та окремі для отримання відповіді від користувача та перевірки його на збіг з правильною відповіддю. Проблема в тому, що ми отримуємо номер питання, коли відправляємо питання, але не коли обробляємо відповідь. Тому мені довелося передавати відповідь від користувача разом з номером через callback_data (це так би мовити, опис кнопки, її значення). Тому я передаю «номер питання_answer_відповідь». Далі у наступній функції ми аналізуємо відповідь користувача і укладаємо вердикт[18].

Тепер трохи про відповіді.

Тут використовуються кілька способів відповідей:

- Message.answer() - використовується, коли ми прийняли повідомлення від користувача(бот тоді відразу знає, на який id надіслати відповідь).
- Call.message.answer() - використовується, коли ми прийняли callback.
- Bot.send_message() - використовується, коли ми маємо ні прийнятого повідомлення, ні callback. У разі цієї функції ми повинні з дужках вказати id користувача. Є ще різні модифікації такі як message.send_photo(). Тут також нюанс. Бот може надсилати тільки розшифровані фото. Тобто перед відправкою ми спочатку маємо прочитати фото у бінарний код.

При розробці додатку я вирішів використовувати SQLite [1]. Оскільки:

- У ній є той мінімум запитів, який мені потрібен.
- Вона відразу вбудована в Python.
- Вона дуже популярна.

Усі функції із запитами я зберіг в один файл db.py.

```
import sqlite3
db = sqlite3.connect(r"app\database\QaBotDb.db")
cursor = db.cursor()
def user_exists(user_id: int, table: str):
    """Проверяем, есть ли юзер в базе"""
    result = cursor.execute(f"SELECT `id` FROM `{table}` WHERE `user_id` = ?", (user_id,))
    return bool(result.fetchall())
def get_user_id(user_id):
    """Достаем id юзера в базе по его user_id"""
    result = cursor.execute("SELECT `id` FROM `users` WHERE `user_id` = ?", (user_id,))
```

```
    return result.fetchone()[0]
def get_themes():
    """Получаем список тем"""
    result = cursor.execute("SELECT `name` FROM `themes`")
    return result.fetchall()
def get_last_test(id_theme: int, id_user: int):
    """Получаем последний решенный тест по теме"""
    test = cursor.execute("SELECT `total_answers` FROM `users_journal` WHERE `id_user` =
? AND `id_theme` = ?",
                          (get_user_id(id_user), id_theme,)
                          )
    test = test.fetchone()[0]
    number_photo = cursor.execute("SELECT `id` FROM `tests` WHERE `theme` = ?",
(id_theme,))
    try:
        result = number_photo.fetchall()[test][0]
        return result
    except IndexError:
        return None
def add_user(user_id, user_name, user_age):
    """Добавляем юзера в базу"""
    cursor.execute(
        "INSERT INTO `users` (`user_id`, `name`, `age`) VALUES (?, ?, ?)", (user_id, user_name,
user_age)
    )
    for i in range(len(get_themes())):
        cursor.execute(
            "INSERT INTO `users_journal` (`id_user`, `id_theme`) VALUES (?, ?)",
            (get_user_id(user_id), i + 1, )
        )
    return db.commit()
def get_answer_test(user_id, theme):
    """Достаем ответ теста по user_id"""
    result = cursor.execute("SELECT `answer` FROM `tests` WHERE `id` = ?",
(get_last_test(theme + 1, user_id),))
    return result.fetchone()[0]
def save_test_point(user_id, number, theme):      # добавить last_test + 1
```

```
    """Сохраняем полученый балл за тест"""
    cursor.execute(
        "UPDATE `users` "
        "SET `true_answers` = `true_answers` + ?, `total_answers` = `total_answers` + 1 "
        "WHERE `user_id` = ?",
        (number, user_id,)
    )
    test_id = get_last_test(theme + 1, user_id)
    cursor.execute(
        "UPDATE `tests` "
        "SET `true` = `true` + ?, `total` = `total` + 1 "
        "WHERE `id` = ?",
        (number, test_id,)
    )
    cursor.execute(
        "UPDATE `users_journal`"
        "SET `true_answers` = `true_answers` + ?, `total_answers` = `total_answers` + 1 "
        "WHERE `id_user` = ? AND `id_theme` = ?",
        (number, get_user_id(user_id), theme, )
    )
    return db.commit()
def get_profile(user_id: int):
    """Получаем инфу о юзере"""
    result = cursor.execute("SELECT * FROM `users` WHERE `user_id` = ?", (user_id,))
    return result.fetchall()[0] if result else None
def get_progress(user_id: int):
    """Получаем инфу о пользователе по темам"""
    result = cursor.execute("SELECT * FROM `users_journal` WHERE `id_user` = ?",
(get_user_id(user_id),))
    return result.fetchall() if result else None
def get_sources(args: list):
    """Получаем список источников"""
    result = []
    for i in args:
        source = cursor.execute("SELECT `link` FROM `sources` WHERE `id` = ?", (i,))
        result.append(source.fetchall())
    return result
```

```
def get_rating():
    """Получаем рейтинг юзеров ТОП 100"""
    result = cursor.execute("SELECT `name`, `true_answers`, `total_answers` "
                            "FROM `users` ORDER BY `true_answers` DESC LIMIT 100")
    return result.fetchall()
def call_to_admin(request: str, user_id=0, user_name='', message_text=''):
    """Сохраняем сообщение для админов в БД"""
    if request == "DELETE":
        result = cursor.execute(f"SELECT `id`, `user_id` FROM `calls_to_admin` LIMIT 1")
        result = result.fetchall()[0]
        cursor.execute("DELETE FROM `calls_to_admin` WHERE `id` = ?", (result[0],))
        return result[1]
    elif not user_id:
        result = cursor.execute(f"{request} * FROM `calls_to_admin` LIMIT 1")
        db.commit()
        return result.fetchall()
    else:
        result = cursor.execute(f"{request} INTO `calls_to_admin` (`user_id`, `username`,
`text_message`) "
                                f"VALUES (?, ?, ?)",
                                (user_id, user_name, message_text, ))
        db.commit()
        return result
def ban_user(user_id: int):
    """Записуем в блоклист и удаляем сообщение"""
    cursor.execute("INSERT INTO `blocklist` (`user_id`) VALUES (?)", (user_id,))
    cursor.execute("DELETE FROM `calls_to_admin` WHERE `user_id` = ?", (user_id,))
    return db.commit()
def save_test_to_db(content: str, answers: str, true_answer: str):
    """Сохраняем предложеный тест в БД"""
    cursor.execute("INSERT INTO `new_tests` (`content`, `answers`, `true_answer`) "
                   "VALUES(?, ?, ?)",
                   (content, answers, true_answer, ))
    return db.commit()
def get_admins():
    """Получаем список админов"""
    result = cursor.execute("SELECT `user_id` FROM `admins`")
```

```
    return result.fetchall()
def count_new_tests():
    """Получаем количество предложеных тестов"""
    result = cursor.execute("SELECT COUNT(`id`) FROM `new_tests`")
    return result.fetchone()[0]
def get_id_for_send_out(table_1: str):
    """Получаем ID юзера, чтобы отправить ему ответ"""
    result = cursor.execute(f"SELECT `user_id` FROM `{table_1}`")
    return result.fetchall()
def add_admin(user_id: int):
    """Добавляем админа"""
    cursor.execute("INSERT INTO `admins` (`user_id`) VALUES(?)", (user_id,))
    return db.commit()
def ban_admin(admin_id: int, mode: str):
    """Баним админа или наоборот"""
    if mode == "ban":
        cursor.execute("INSERT INTO `blocklist` (`user_id`) VALUES(?)", (admin_id,))
        cursor.execute("DELETE FROM `admins` WHERE `user_id` = ?", (admin_id,))
    else:
        cursor.execute("INSERT INTO `admins` (`user_id`) VALUES(?)", (admin_id,))
        cursor.execute("DELETE FROM `blocklist` WHERE `user_id` = ?", (admin_id,))
    return db.commit()
def get_topic_statistic():
    """Получаем статистику по томам"""
    result = cursor.execute("SELECT `id_theme`, `true_answers`, `total_answers` "
                            "FROM `users_journal` "
                            "GROUP BY `id_theme`")
    return result.fetchall(), get_themes()
def get_age_statistic():
    """Получаем статистику по возрасту"""
    result = cursor.execute("SELECT `age`, COUNT(`age`) FROM `users` GROUP BY `age`")
    return result.fetchall()
def get_problem_statistic():
    """Получаем проблемные тесты"""
    result = cursor.execute("SELECT `id`, `true`, `total` FROM `tests` WHERE `true`/`total` *
100 < 50")
    return result.fetchall()
```

registration.py - Просто функції реєстрації, працюють за принципом кінцевого автомата.

```
from aiogram import Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup
from app.database.db import user_exists, add_user
class Order(StatesGroup):
    waiting_for_name = State()
    waiting_for_age = State()
async def registration_start(message: types.Message):
    if not user_exists(message.from_user.id, "users"):
        await message.answer("Введи свої ім'я та фамілію")
        await Order.waiting_for_name.set()
    else:
        await message.answer("Вибач, але ти вже зареєстрований \n\n"
                             "Натисни /menu щоб перейти в меню")
async def name_chosen(message: types.Message, state: FSMContext):
    if message.text != "Скасувати":
        await state.update_data(name=message.text)
        await message.answer("Напиши свій вік:")
        await Order.waiting_for_age.set()
    else:
        await message.answer("Дію скасовано")
        await state.finish()
async def age_chosen(message: types.Message, state: FSMContext):
    if message.text != "Скасувати":
        try:
            if int(message.text.lower()) not in range(13, 80):
                await message.answer("Будь-ласка , вкажіть свій справжній вік.")
                return
        except ValueError:
            await message.answer("Будь-ласка , вкажіть свій вік цифрами.")
            return
        await state.update_data(chosen_age=message.text.lower())
        user_data = await state.get_data()
        await message.answer(f"Вас звати: {user_data['name']}\nВаш вік:
```

```
{user_data['chosen_age']}.\n\n"
                    "Натисни /menu щоб перейти в меню")
        add_user(message.from_user.id, *user_data.values())
    else:
        await message.answer("Дію скасовано")
    await state.finish()
def register_handlers_registration(dp: Dispatcher):
    dp.register_message_handler(registration_start, commands="registration", state="*")
    dp.register_message_handler(name_chosen, state=Order.waiting_for_name)
    dp.register_message_handler(age_chosen, state=Order.waiting_for_age)
```

admin.py – всі функції для адміну

```
from aiogram import Dispatcher, types, Bot
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters import Text, IDFilter
from aiogram.dispatcher.filters.state import State, StatesGroup
from aiogram.utils.exceptions import ChatNotFound
from app.config_reader import load_config
from app.database.db import call_to_admin, ban_user, save_test_to_db, \
    get_admins, count_new_tests, get_id_for_send_out, ban_admin, \
    get_topic_statistic, get_age_statistic, get_problem_statistic

config = load_config("config/bot.ini")
bot = Bot(config.tg_bot.token)
admins_list = [config.tg_bot.admin_id]
if len(get_admins()):
    for i in get_admins():
        admins_list.append(i[0])
admin_menu_buttons = [
    "Повідомлення",
    "Розіслати повідомлення"
]
class Order(StatesGroup):
    answer = State()
class CreateNewTest(StatesGroup):
    content = State()
    test_answer_1 = State()
    test_answer_2 = State()
```

```
class SendOut(StatesGroup):
    text_message_for_users = State()
    text_message_for_admins = State()
    text_message_for_all = State()
async def admin_menu(message: types.Message, state: FSMContext):
    await state.finish()
    # await bot.set_my_commands(admin_commands)
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in admin_menu_buttons:
        keyboard.add(name)
    if message.from_user.id == admins_list[0]:
        keyboard.add("Перевірити наявність запропонованих тестів")
        keyboard.add("Заблокувати адміна")
    else:
        keyboard.add("Створити тест")
    await message.answer("Меню", reply_markup=keyboard)
async def notification(message: types.Message):
    buttons = [
        types.InlineKeyboardButton(text="Відповісти", callback_data="answer_to_user"),
        types.InlineKeyboardButton(text="Заблокувати користувача",
callback_data="answer_to_user_ban")
    ]
    keyboard = types.InlineKeyboardMarkup()
    keyboard.add(*buttons)
    result = call_to_admin("SELECT")
    if result:
        await message.answer((f"Повідомлення від користувача: @{result[0][2]}\n"
                              f"ID: {result[0][1]}\n\n"
                              f"{result[0][3]}"), reply_markup=keyboard)
    else:
        await message.answer("Повідомлень немає",  reply_markup=keyboard)
async def wait_answer(call: types.CallbackQuery):
    if call.data == "answer_to_user":
        await Order.answer.set()
        await call.message.answer("Очікую на відповідь")
    else:
        await call.message.answer("Користувача заблоковано")
```

```
        user_id = call_to_admin("DELETE")
        ban_user(user_id)
        await bot.send_message(config.tg_bot.admin_id, f"Адмін: @{call.from_user.username}\n"
                                                       f"ID: {call.from_user.id}\n\n"
                                                       f"Заблокував користувача: {user_id}")
        await bot.delete_message(call.from_user.id, call.message.message_id)
async def answer_to_user(message: types.Message, state: FSMContext):
    user_id = call_to_admin("DELETE")
    await bot.send_message(user_id, f"Відповідь від адміна:\n\n{message.text}")
    await bot.send_message(admins_list[0], f"Адмін: @{message.from_user.username}\n"
                                           f"ID: {message.from_user.id}\n\n"
                                           f"Відповів користувачу\n"
                                           f"ID: {user_id}\n\n"
                                           f"{message.text}")
    await state.finish()
    await notification(message)
async def create_test_1(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add("Скасувати")
    await message.answer("Напиши зміст тесту", reply_markup=keyboard)
    await CreateNewTest.content.set()
async def create_test_2(message: types.Message, state: FSMContext):
    if message.text != "Скасувати":
        await message.answer("Напиши варіанти відповідей")
        await state.update_data(content=message.text)
        await CreateNewTest.next()
    else:
        await message.answer("Дію скасовано")
        await state.finish()
async def create_test_3(message: types.Message, state: FSMContext):
    if message.text != "Скасувати":
        await message.answer("Що буде правильною відповідю?")
        await state.update_data(answers=message.text.lower())
        await CreateNewTest.next()
    else:
        await message.answer("Дію скасовано")
        await state.finish()
```

```
async def save_new_test(message: types.Message, state: FSMContext):
    if message.text != "Скасувати":
        try:
            data = await state.get_data()
            save_test_to_db(data["content"], data["answers"], message.text)
            await message.answer("Добре, ваше питання буде розглянуто головним адміном")
            await bot.send_message(config.tg_bot.admin_id, f"Адмін
@{message.from_user.username}\n"
                                                f"ID: {message.from_user.id}\n\n"
                                                f"Запропонував новий тест")
        except KeyError:
            await message.answer("Трапилась помилка, пробуй ще раз")
        finally:
            await state.finish()
    else:
        await message.answer("Дію скасовано")
        await state.finish()


async def amount_new_test(message: types.Message):
    await message.answer(f"Кількість нових запропонованих тестів: {count_new_tests()}")
async def send_out_message_state(message: types.Message):
    buttons = [
        types.InlineKeyboardButton(text="Користувачам", callback_data="send_out_users"),
        types.InlineKeyboardButton(text="Адмінам", callback_data="send_out_admins"),
        types.InlineKeyboardButton(text="Всім", callback_data="send_out_all")
    ]
    keyword = types.InlineKeyboardMarkup()
    keyword.add(*buttons)
    await message.answer("Кому ви хочете розіслати повідомлення:", reply_markup=keyword)
async def wait_for_send_out(call: types.CallbackQuery):
    if call.data.split('_')[2] == "users":
        await SendOut.text_message_for_users.set()
    elif call.data.split('_')[2] == "admins":
        await SendOut.text_message_for_admins.set()
    else:
        await SendOut.text_message_for_all.set()
```

```
    await call.message.answer("Очікую на повідомлення")
async def send_out_to_users(message: types.Message, state: FSMContext):
    send_out_result = "Не знайдені чати:\n\n"
    for n, user_id in enumerate(get_id_for_send_out("users"), 1):
        try:
            await bot.send_message(user_id[0], f"Повідомлення від адміна\n\n{message.text}")
        except ChatNotFound:
            send_out_result += f"{n}. {user_id[0]}\n"
    await message.answer("Я розіслав ваше повідомлення")
    await message.answer(send_out_result)
    await state.finish()


async def send_out_to_admins(message: types.Message, state: FSMContext):
    send_out_result = "Не знайдені чати:\n\n"
    for n, user_id in enumerate(get_id_for_send_out("admins"), 1):
        try:
            await bot.send_message(user_id[0], f"Повідомлення від адміна
@{message.from_user.username}\n"
                                               f"ID: {message.from_user.id}\n\n"
                                               f"{message.text}")
        except ChatNotFound:
            send_out_result += f"{n}. {user_id[0]}\n"
    await message.answer("Я розіслав ваше повідомлення")
    await message.answer(send_out_result)
    await state.finish()
async def send_out_to_all(message: types.Message, state: FSMContext):
    send_out_result = "Не знайдені чати:\n\n"
    n = 0
    for user_id in get_id_for_send_out("users"):
        try:
            await bot.send_message(user_id, f"Повідомлення від адміна\n\n{message.text}")
        except ChatNotFound:
            n += 1
            send_out_result += f"{n}. {user_id[0]}\n"
    for user_id in get_id_for_send_out("admins"):
        try:
```

```
            await bot.send_message(user_id[0], f"Повідомлення від адміна
@{message.from_user.username}\n"
                                               f"ID: {message.from_user.id}\n\n"
                                               f"{message.text}")
        except ChatNotFound:
            n += 1
            send_out_result += f"{n}. {user_id[0]}\n"
    await message.answer("Я розіслав ваше повідомлення")
    await message.answer(send_out_result)
    await state.finish()
async def get_list_admins(message: types.Message):
    buttons = []
    for admin_id in admins_list[1:]:
        buttons.append(types.InlineKeyboardButton(text=f"{admin_id}",
callback_data=f"ban_admin_{admin_id}"))
    keyboard = types.InlineKeyboardMarkup()
    keyboard.add(*buttons)
    await message.answer("Список адміністраторів", reply_markup=keyboard)
async def block_admin(call: types.CallbackQuery):
    admin_id = call.data.split('_')[2]
    if admin_id in admins_list:
        admins_list.remove(admin_id)
    ban_admin(int(call.data.split('_')[2]), mode="ban")
    keyboard = types.InlineKeyboardMarkup()
    keyboard.add(types.InlineKeyboardButton(text="Скасувати дію",
callback_data=f"unban_{admin_id}"))
    await bot.send_message(admin_id, "Вас заблоковано")
    await call.message.answer("Адміна заблоковано", reply_markup=keyboard)
async def unban_admin(call: types.CallbackQuery):
    admin_id = call.data.split('_')[1]
    if admin_id not in admins_list:
        admins_list.append(admin_id)
    ban_admin(int(call.data.split('_')[1]), mode="unban")
    await bot.send_message(admin_id, "Вас розблоковано")
    await call.message.answer("Адміна розблоковано")
async def topic_statistic(message: types.Message):
    result, topic_list = get_topic_statistic()
```

```
    text = "Статистика по темам: \n\n"
    for item in result:
        try:
            text += f"{item[0]}. {topic_list[item[0] - 1][0]} {item[1]}/{item[2]} | " \
                    f"{round(item[1]/item[2] * 100, 2)} %\n"
        except ZeroDivisionError:
            text += f"{item[0]}. {topic_list[item[0] - 1][0]} {item[1]}/{item[2]} | 0.0 %\n"
    await message.answer(text)
async def age_statistic(message: types.Message):
    result = get_age_statistic()
    text = "Статистика по віку: \n\n"
    all_sum = sum([j[1] for j in result])
    for item in result:
        text += f"{item[0]} years - {item[1]} | {round(item[1]/all_sum * 100, 2)} %\n"
    await message.answer(text)


async def problem_statistic(message: types.Message):
    result = get_problem_statistic()
    await message.answer("Проблемні питання")
    for item in result:
        photo = open(r'app\database\photos\{}.png'.format(item[0]), 'rb')
        await message.answer_photo(photo, f"Кількість правильних відповідей: {item[1]}\n"
                                          f"Загальна кількість відповідей: {item[2]}\n"
                                          f"Процент успішності: {round(item[1]/item[2]*100, 2)} %")
def register_handlers_admin(dp: Dispatcher):
    dp.register_message_handler(admin_menu, IDFilter(admins_list), commands=["admin",
"admin_menu"], state="*")
    dp.register_message_handler(notification, IDFilter(admins_list), Text(equals="Повідомлення",
ignore_case=True))
    dp.register_message_handler(create_test_1, IDFilter(admins_list), Text(equals="Створити
тест", ignore_case=True))
    dp.register_message_handler(create_test_2, state=CreateNewTest.content)
    dp.register_message_handler(create_test_3, state=CreateNewTest.test_answer_1)
    dp.register_message_handler(save_new_test, state=CreateNewTest.test_answer_2)
    dp.register_message_handler(amount_new_test, IDFilter(config.tg_bot.admin_id),
                                Text(equals="Перевірити наявність запропонованих тестів",
                                ignore_case=True))
```

```
    dp.register_message_handler(send_out_message_state, IDFilter(admins_list),
                                Text(equals="Розіслати повідомлення",
                                ignore_case=True))
    dp.register_callback_query_handler(wait_for_send_out, Text(startswith="send_out_"))
    dp.register_message_handler(send_out_to_users, state=SendOut.text_message_for_users)
    dp.register_message_handler(send_out_to_admins, state=SendOut.text_message_for_admins)
    dp.register_message_handler(send_out_to_all, state=SendOut.text_message_for_all
    dp.register_callback_query_handler(wait_answer, Text(startswith="answer_to_user"))
    dp.register_message_handler(answer_to_user, state=Order.answer)
    dp.register_message_handler(get_list_admins, IDFilter(config.tg_bot.admin_id),
                                Text(equals="Заблокувати адміна", ignore_case=True))
    dp.register_callback_query_handler(block_admin, Text(startswith="ban_admin_"))
    dp.register_callback_query_handler(unban_admin, Text(startswith="unban_"))
    dp.register_message_handler(topic_statistic, IDFilter(admins_list), commands="topic_statistics",
state="*")
    dp.register_message_handler(age_statistic, IDFilter(admins_list), commands="age_statistic",
state="*")
    dp.register_message_handler(problem_statistic, IDFilter(admins_list),
commands="problem_statistic", state="*")
```

Тут з'являється новий фільтр – IdFilter

Вони додані у тому, щоб простий користувач було викликати ці команди.

ВИСНОВКИ

Сучасні та ефективні засоби, які використовуються для проведення масштабних рекламних кампаній, це чат-боти. Однак, враховуючи той факт, що в чаті-боті вище рівень конвертації відкриття повідомлення та перегляду його користувачами, ніж у будь-якого іншого інструменту реклами, чат-боти значно ефективніший за будь-який інший інструмент реклами.

При цьому для користувачів набагато простіше отримати необхідну інформацію з автоматизованого режиму за допомогою чат-бота, ніж шукати її в інтернеті, що значно підвищує лояльності.

І це не дивно – адже популярність месенджерів зростає з кожним роком, і зростання популярності цих програм очікується максимум протягом п'яти - десяти років.

У роботі будь-якої компанії є безліч завдань, які необхідно виконати для того, щоб отримати прибуток. У цьому випадку бот зможе полегшити завдання, оскільки він може виконувати все, що йому заманеться. А якщо ви хочете самостійно подбати про виконання своїх робіт або дати їм вказівки у разі потреби, то вам достатньо налаштувати свій бот, щоб автоматично виконати всі необхідні дії або дати їм прямі інструкції, якщо це необхідно.

При цьому чат-бот може бути використаний для підвищення загальної продуктивності компанії та організації свого бізнесу [13].

В ході виконання кваліфікаційної роботи було розглянуто тестовий контроль як засіб навчального процесу, досліджено основні вимоги до тестових завдань та переваги перевірки знань студентів за тестами, проаналізована поведінка чат-ботів під час збоїв в Telegram та технологія розробки чат-боту, а також було обране програмне забезпечення для розробки додатку, та виконана практична реалізація чат боту мовою програмування Python.

Вміння правильно використовувати сучасні технології є одним із ключових факторів успішного ведення бізнесу в сучасних умовах. Однак якщо використовувати цю новинку у потрібний час, то вона може бути використана конкурентами. Саме вони будуть більш прогресивними, ніж потенційні клієнти

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SQLite - Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/sqlite/sqlite_python.htm.

2. SQLite, MySQL и PostgreSQL: сравниваем популярные реляционные СУБД [Електронний ресурс] – Режим доступу до ресурсу: https://tproger.ru/translations/sqlite- mysql-postgresql-comparison/.

3. Асинхронное программирование в Python [Електронний ресурс] – Режим доступу до ресурсу: https://tproger.ru/translations/asynchronous-programming-in-python/.

4. Бібліотека освітніх програм від експертів в особистісному розвитку, здоров'я і харчування, лідерстві, фінансах і побудові відносин [Електронний ресурс] – Режим доступу до ресурсу: https://academy.5sfer.com/

5. Види мов програмування [Електронний ресурс]. – Режим доступу до ресурсу: http://csaa.ru/vidy-jazykov-programmirovanija/

6. Єдинак В. С. Розвиток інформаційних технологій в Україні / В. С. Єдинак. // Континент. – 2008. – №5. – С. 289–290.

7. Зачем нужен Python Global Interpreter Lock и как он работает [Електронний ресурс] – Режим доступу до ресурсу: https://tproger.ru/translations/global-interpreter-lock-guide/.

8. Знакомство с aiogram [Електронний ресурс] – Режим доступу до ресурсу: https://mastergroosha.github.io/telegram-tutorial-2/quickstart/.

9. Какорін М. О. Інформаційні технології як фактор інновацій у глобальній фінансовій системі / М. О. Какорін. – 2008. – №5. – С. 106–109.

10. Многопоточность в Python [Електронний ресурс] – Режим доступу до ресурсу: https://coderlessons.com/tutorials/kompiuternoe-programmirovanie/uchebnik-python/28-mnogopotochnost-v-python.

11. Мова програмування Python [Електронний ресурс]. – Режим доступу до ресурсу: https://web-creator.ru/articles/python

12. Модуль sqlite — Работаем с базой данных [Електронний ресурс] – Режим доступу до ресурсу: https://python-scripts.com/sqlite.

13. Не соцмережі, а чат-бот: навіщо бізнесу месенджери

[Електронний ресурс]. – Режим доступу до ресурсу: https://igate.com.ua/news/24336-ne-sotsseti-a-chatbot-zachem-biznesu-messendzhery

14. Пацай Б. Д. Роль інформаційних технологій в управлінні фінансовими ресурсами підприємств / Б. Д. Пацай. – 2008. – №8. – С. 82–84.

15. Поняття мови програмування [Електронний ресурс]. – Режим доступу до ресурсу: https://ibrain.kz/informatika/ponyatie-yazyk-programmirovania

16. Потоковые и многопроцессорные модули на Python [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/FHVP9zN.

17. Приклади використання чат-ботів у бізнесі [Електронний ресурс]. – Режим доступу до ресурсу: https://vc.ru/flood/25197-business-bot

18. Самоучитель PYTHON [Електронний ресурс] – Режим доступу до ресурсу: http://pythoshka.ru/p1138.html.

19. Тестування як засіб оцінки знань та вмінь студентів [Електронний ресурс] – Режим доступу до ресурсу: https://xreferat.com/71/5094-1-testuvannya-yak-zas-b-oc- nki-znan-ta-vm-n-student-v.html.

20. Чат-боти – хто вони та що вміють? [Електронний ресурс]. – Режим доступу до ресурсу: https://efsol.ru/articles/messendzhery-i-chat-boty-dlya-biznesa-dostavki.html

21. Чигасова Н. М. Місце інформаційних технологій у розвитку інформаційного суспільства в Україні / Н. М. Чигасова. – 2007. – №9. – С. 110–113.

22. Шандра В. М. Застосування інформаційних технологій в забезпеченні технологічного оновлення економіки на інноваційній основі / В. М. Шандра. – 2007. – №10. – С. 220–223.

23. Шкраб'юк Н. В. Перспективи розвитку інформаційних технологій в Україні [Електронний ресурс] / Н. В. Шкраб'юк – Режим доступу до ресурсу: http://www.rusnauka.com/17_AND_2010/Informatica/68784.doc.htm.

24. Як використовувати чат-боти у бізнесі: 5 ідей та 5 кейсів [Електронний ресурс]. – Режим доступу до ресурсу: https://vc.ru/services/93850-kak-ispolzovat-chat-boty-v- biznese-5-idey-i-5-keysov

ДОДАТОК А. РОБОТА З ПРОГРАМОЮ

1.Відкрити діалог з чат-ботом «QA bot»

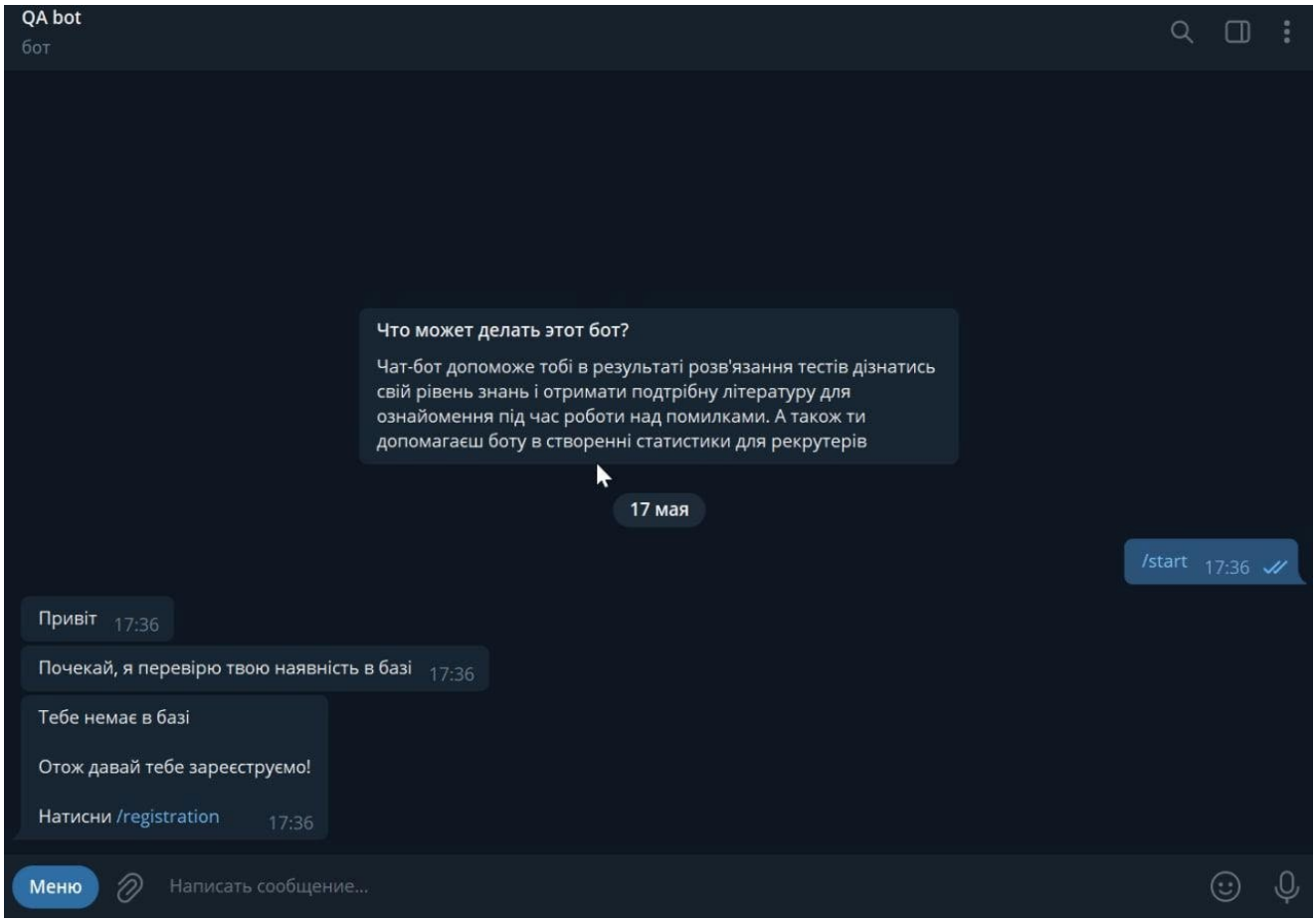


Рис. А.1 – Вікно діалогу з чат-ботом «QA_Project

2.Натиснути на кнопку /registration та ввести им’я та вік

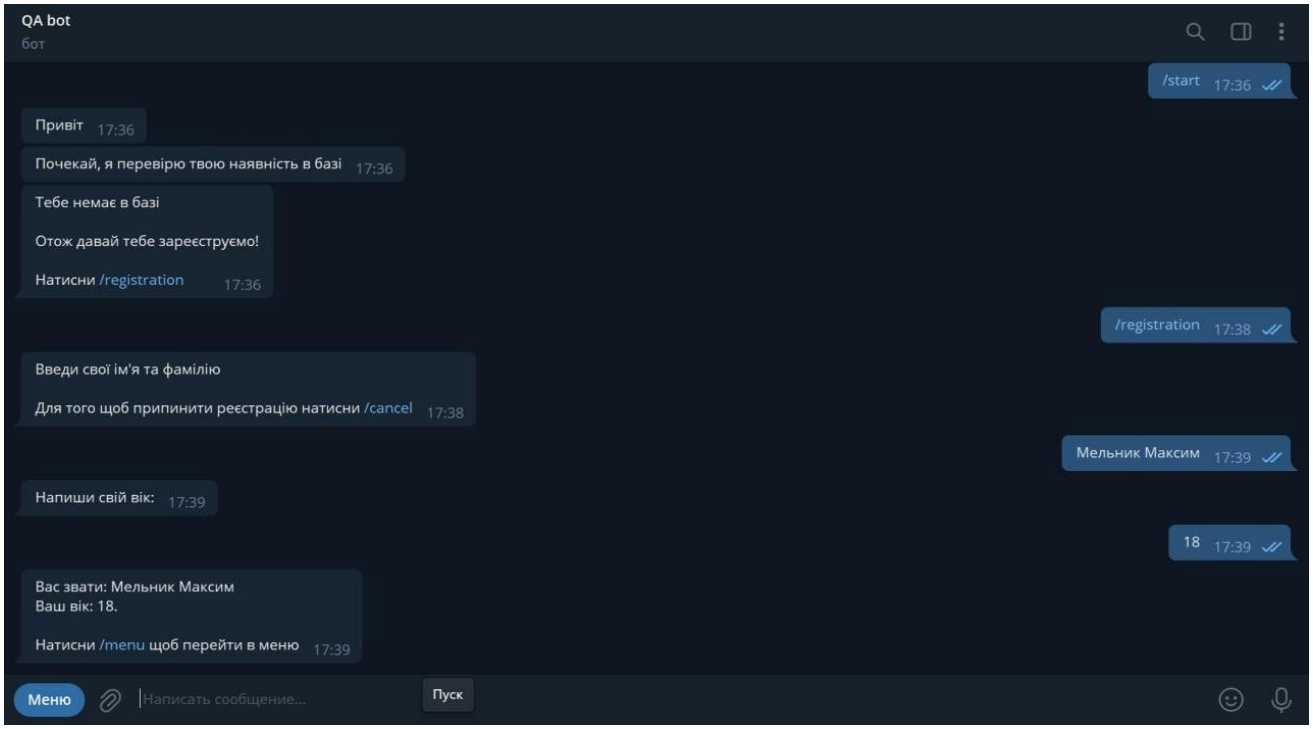


Рис. А.2 – Команда /registration

3.Після вводу им’я, віку, введіть команду /menu

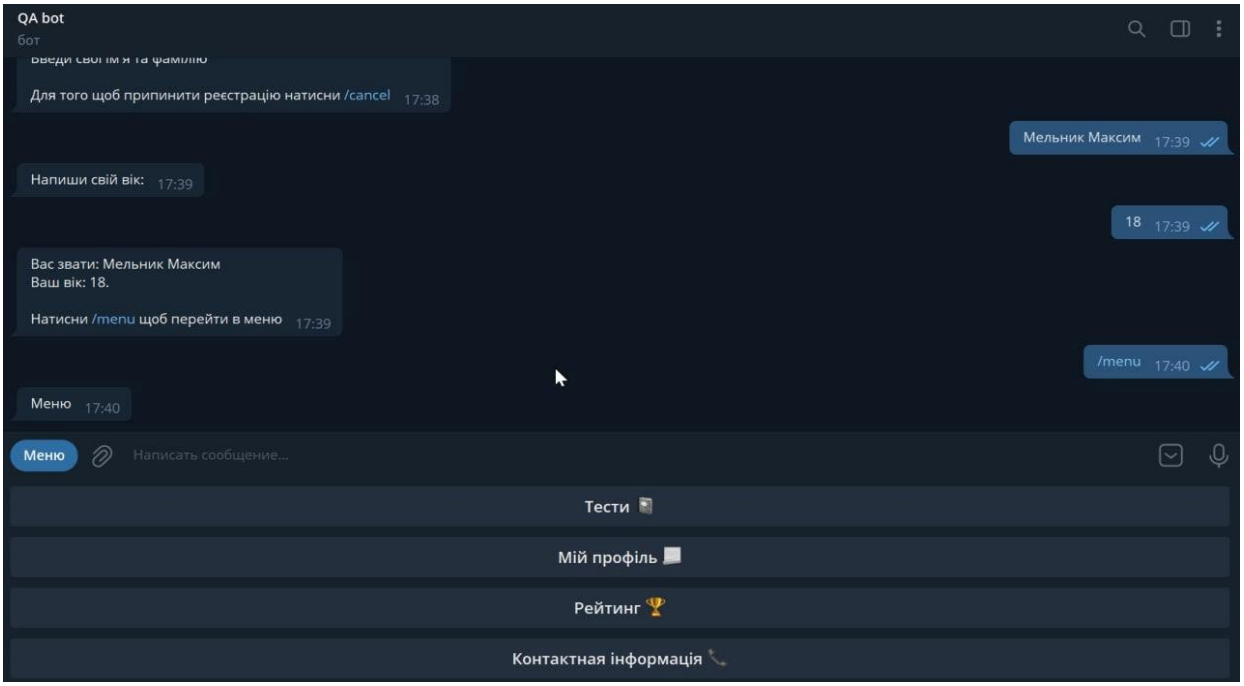


Рис. А.3 – Команда /menu

4.Натисніть кнопку тести щоб обрати тему

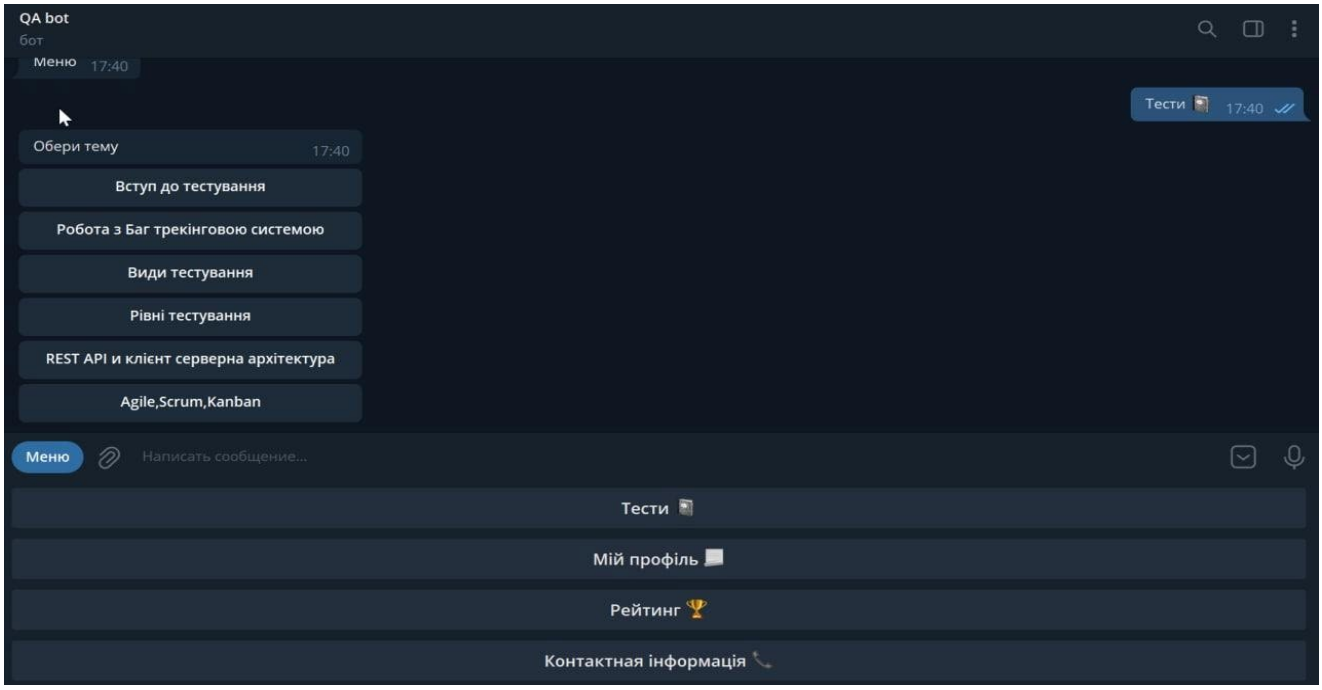


Рис. А.4 – Вибір теми

5.Після вибору теми почніть тестування

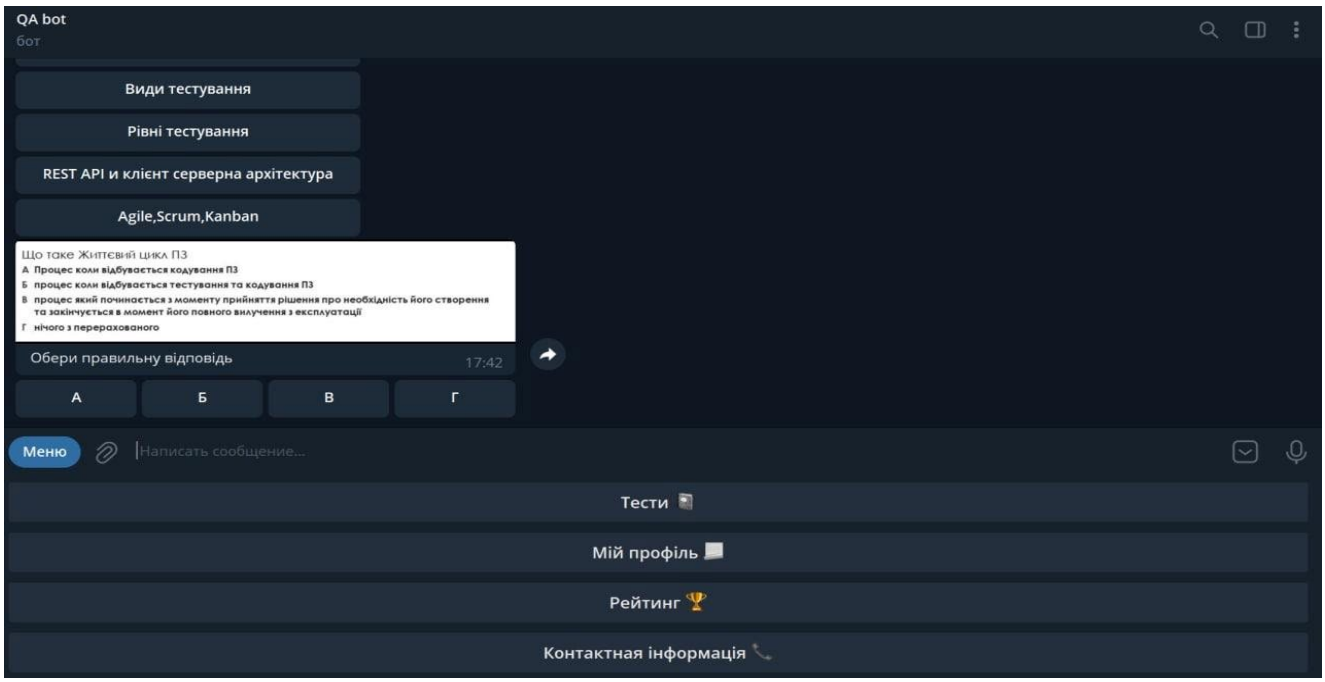


Рис. А.5 – Вибір правильної відповіді

6.Натисніть кнопку Мій профіль в меню щоб побачити інформацію про користувача

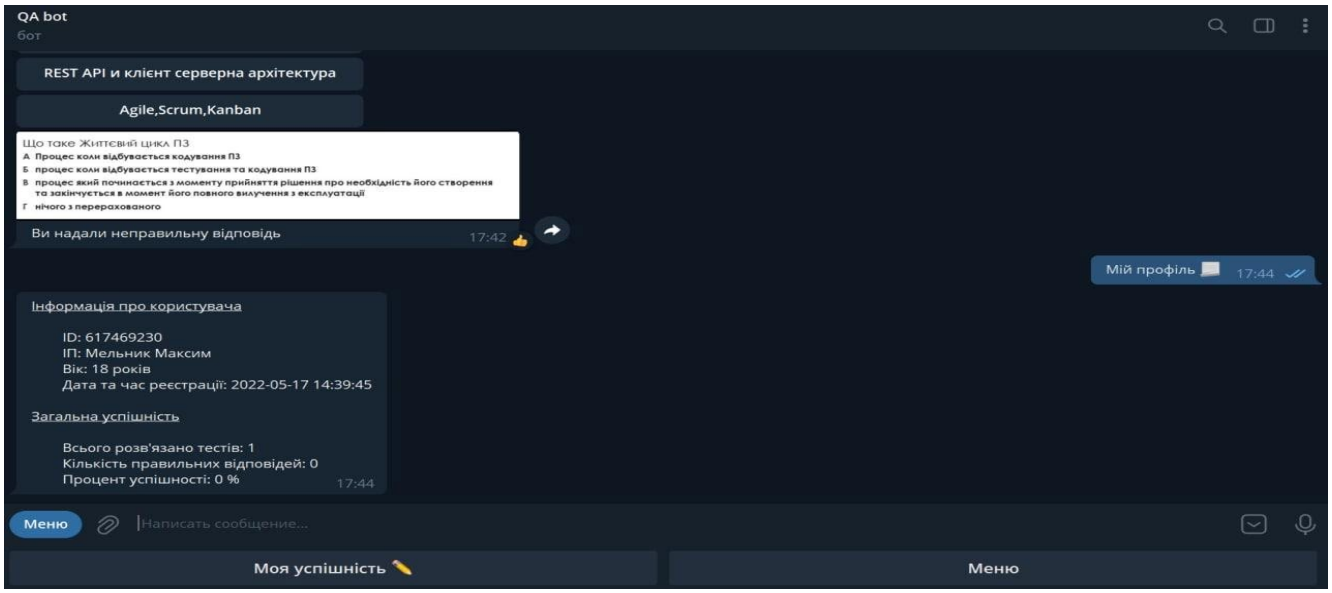


Рис. А.6 - Мій профіль

7.Натисніть кнопку Моя успішність щоб побачити статистику

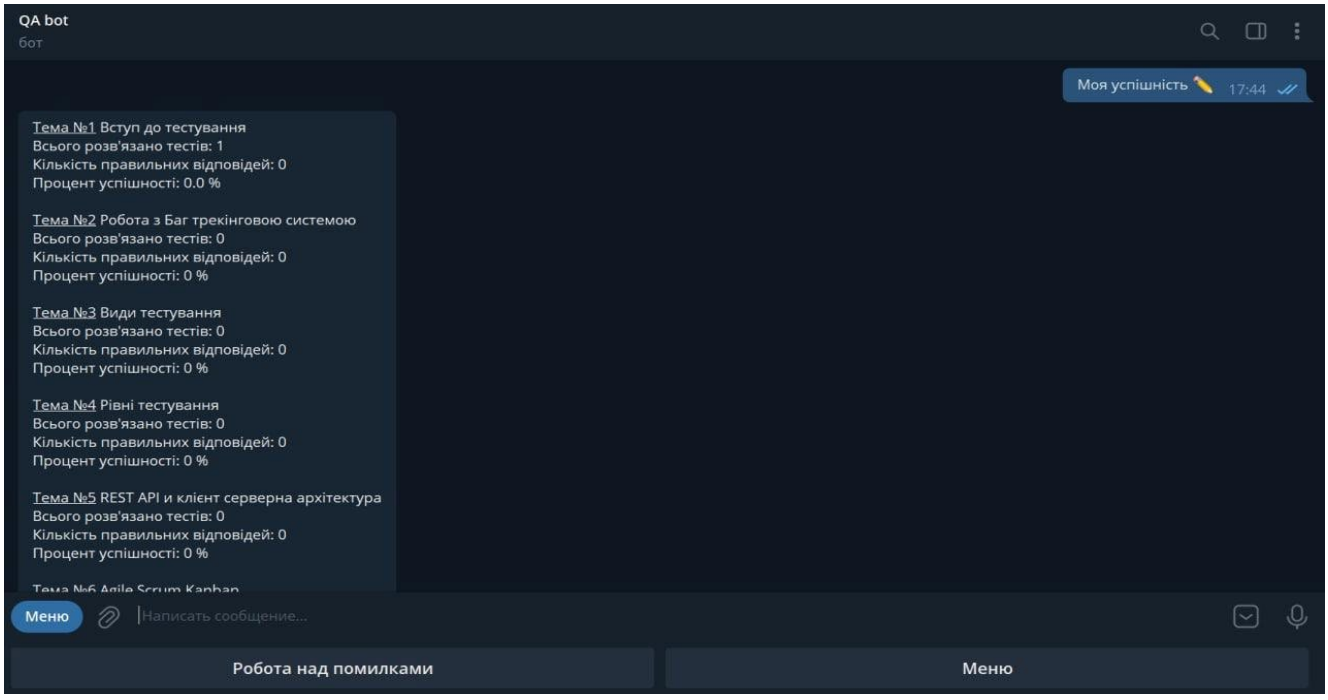


Рис. А.7 - Моя успішність