МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Державний університет Житомирська політехніка

Кафедра комп`ютерних наук

ЗВІТ
З лабораторної роботи

з дисципліни: «Тестування, верифікація та валідація ПЗ»
на тему:
«Створення telegram-бота для тестування рівня знань»

Студентка І курсу групи ІСТ-21-1
Городецька Вікторія Валеріївна

Викладач: Граф М. С.

Житомир – 2022

Лабораторна робота

Мета: розробити мобільний чат-бот для тестування знань.

Практичне значення: розроблений чат-бот буде використовуватись під час проведення співбесіди, для тестування претендента на посаду, на результати якого в подальшому буде опиратись менеджер по набору персоналу. Дане тестування дозволить оцінити рівень знань претендента, щоб отримати бачення про його рівень кваліфікації.

РОЗДІЛ 1 АНАЛІЗ ДЖЕРЕЛА ДАНИХ ТА НАПИСАННЯ ПРОГРАМНОГО КОДУ

1.1 Характеристика джерела даних для проведення аналізу

В якості джерела даних було використано систему керування базами даних MongoDB.

В ході проектування БД в системі керування базами даних MongoDB створено наступні колекції:

- Bot
- Users

Для збереження даних про питання та відповіді тесту була призначена колекція Bot (таблиця 1.1 та рис. 1.1).

Таблиця 1.1

Колекція Bot

Назва	Тип даних	Опис
id	Int32	Ідентифікатор запитання
text	String	Текст запитання
answers	Array	Масив даних з відповідями
correct	Int32	Правильна відповідь

Дані у колекцію User генеруються автоматично при старті, та змінюються впродовж усього тестування (таблиця 1.2 та рис. 1.2).

Таблиця 1.2

Колекція Users

Назва	Тип даних	Опис
chat_id	Int32	Ідентифікатор чату (користувача)
is_passing	Boolean	Статус тестування (проходить)
is_passed	Boolean	Статус тестування (пройдено)
question_index	Int32	Номер запитання на даний момент проходження тестування
answers	Array	Масив з обраними відповідями

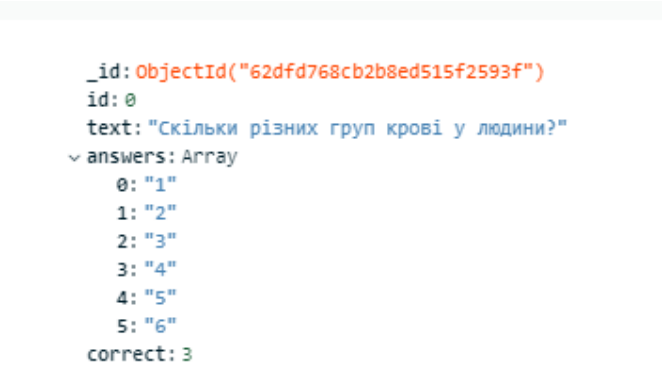

Рис. 1.1. Візуалізація даних колекції Bot в MongoDB

Рис. 1.2. Візуалізація даних колекції Users в MongoDB

1.2 Створення Telegram бота та реалізація програмного коду

Перед написання програмного коду за допомогою офіційного бота Telegram @BotFather потрібно створити нового бота, та отримати його API Токен (рис. 1.3).

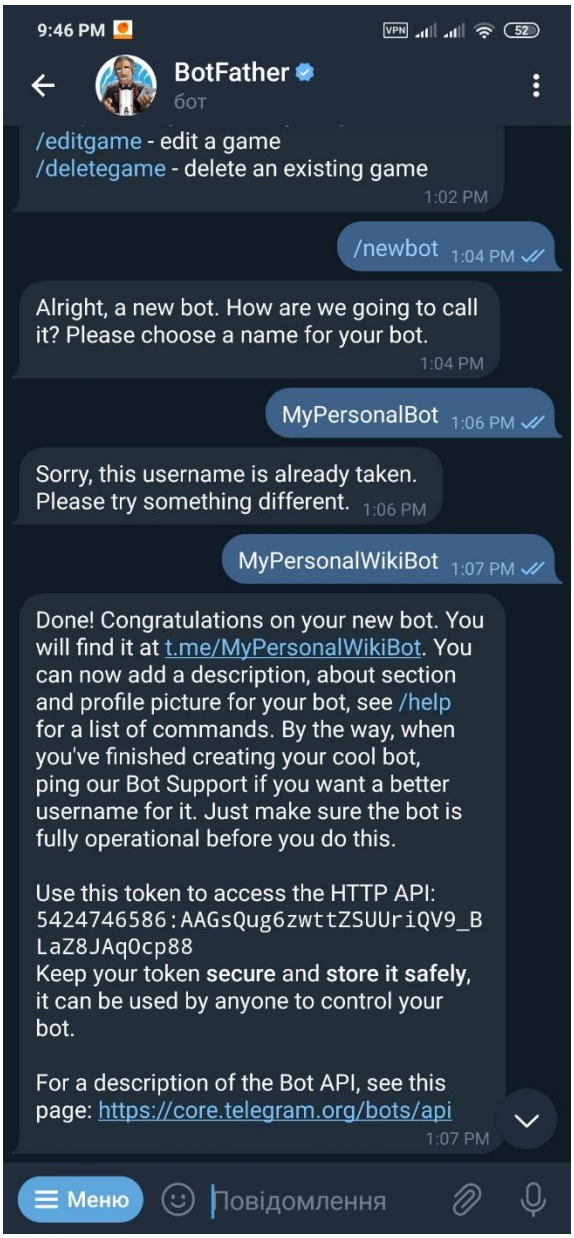

Рис. 1.3. Створення бота

Початком написання коду є підключення бібліотек pyTelegramBotAPI та pymongo. Фрагмент коду підключення бібліотек:

```
import telebot
from pymongo import MongoClient
```

Далі було виконано підключення API токена бота. Фрагмент коду підключення токена:

```
bot = telebot.TeleBot("5424746586:AAGX3cqvuxfO8lI_cdXe_NsWJPfswSNRxI")
```

Наступним кроком підключаємо БД MongoDB, отримуємо колекції та підраховуємо кількість питань, створивши клас DataBase в якому будуть функції для керування БД. Фрагмент коду наведено нижче:

```
class DataBase:
  def __init__(self):
   cluster =MongoClient("mongodb+srv://user:12345@cluster0.0dsh8.mongodb.net/?retry
   Writes=true&w=majority")
    self.db = cluster["WikiBot"]
    self.users = self.db["Users"]
    self.questions = self.db["Bot"]
    self.questions_count = len(list(self.questions.find({})))
```

Ідентифікація користувача буде відбуватися по chat_id. Створюємо функцію яка повертає користувача по chat_id, якщо користувача немає в БД, створиться новий. Фрагмент коду наведено нижче:

```
user = self.users.find_one({"chat_id": chat_id})
                if user is not None:
                        return user
                user = {
                        "chat_id": chat_id,
                        "is_passing": False,
                        "is_passed": False,
                        "question_index": None,
                        "answers": []
                }
                self.users.insert_one(user)
                return user
```

Створюємо функцію яка буде змінювати параметри користувача по chat_id. Фрагмент коду наведено нижче:

```
def set_user(self, chat_id, update):
    self.users.update_one({"chat_id": chat_id}, {"$set": update})
```

Створюємо функцію, яка буде повертати питання по його id. Фрагмент коду наведено нижче:

```
def get_question(self, index):
    return self.questions.find_one({"id": index})
db = DataBase()
```

Створюємо функцію яка буде викликатися командою /start, отримаємо користувача, якщо користувач проходив тест, тобто якщо параметр is_passed=true, тоді буде відправлено повідомлення, що другий раз тест пройти неможливо. Якщо користувач проходить тест в даний момент, тобто параметри is_passed=false та is_passing=true, нічого відправлятися не буде. Фрагмент коду наведено нижче:

```
@bot.message_handler(commands=["start"])
def start(message):
    user = db.get_user(message.chat.id)
    if user["is_passed"]:
        bot.send_message(message.from_user.wid, "Ви вже пройшли це
тестування. Вдруге пройти неможливо")
        return
    if user["is_passing"]:
        return
    db.set_user(message.chat.id, {"question_index": 0, "is_passing": True})
  user = db.get_user(message.chat.id)
    post = get_question_message(user)
    if post is not None:
        bot.send_message(message.from_user.id, post["text"],
reply_markup=post["keyboard"])
```

Створюємо функцію яка буде повертати текст і клавіатуру питання. Якщо користувач пройшов тест, тобто коли question_index буде дорівнювати кількості запитань, буде рахуватися відсоток правильних відповідей, в інших випадках

створюється пуста клавіатура. Проходимось по усіх відповідях, та додаємо необхідні кнопки в клавіатуру. Оновлюємо параметри користувача Створюємо змінну з текстом повідомлення, далі повернемо словник з текстом і клавіатурою. Фрагмент коду наведено нижче:

```
def get_question_message(user):
    if user["question_index"] == db.questions_count:
        count = 0
        for question_index, question in enumerate(db.questions.find({})):
            if question["correct"] == user["answers"][question_index]:
                count += 1
        percents = round(100 * count / db.questions_count)

        if percents < 40:
            smile = "😕"
        elif percents < 60:
            smile = "😐"
        elif percents < 90:
            smile = "😀"
        else:
            smile = "😎"
        text = f"Ви відповіли правильно на {percents}% питань {smile}
        db.set_user(user["chat_id"], {"is_passed": True, "is_passing": False})
        return {
            "text": text,
            "keyboard": None
        }
    question = db.get_question(user["question_index"])
    if question is None:
        return
    keyboard = telebot.types.InlineKeyboardMarkup()
    for answer_index, answer in enumerate(question["answers"]):
        keyboard.row(telebot.types.InlineKeyboardButton(f"{chr(answer_index +
97)}) {answer}",
callback_data=f"?ans&{answer_index}"))
```

```
    text = f"Питання №{user['question_index'] + 1}\n\n{question['text']}"
    return {
        "text": text,
        "keyboard": keyboard
    }
```

Створюємо функцію, яка буде викликатися коли користувач натиснув на відповідь, в ній перевіряться параметри is_passed та is_passing, далі додаємо відповідь користувача в його відповіді. Фрагмент коду наведено нижче:

```
@bot.callback_query_handler(func=lambda query: query.data.startswith("?ans"))
def answered(query):
    user = db.get_user(query.message.chat.id)

    if user["is_passed"] or not user["is_passing"]:
        return
    user["answers"].append(int(query.data.split("&")[1]))
    db.set_user(query.message.chat.id, {"answers": user["answers"]})
    post = get_answered_message(user)
    if post is not None:
        bot.edit_message_text(post["text"], query.message.chat.id, query.message.id,
                              reply_markup=post["keyboard"])
```

Далі створюємо функцію, яка буде повертати що потрібно відповісти користувачу, в ній створюємо змінну з текстом, в змінну додаємо номер і текст питання. Проходимось по усіх відповідях, галочкою відмітимо правильну відповідь, хрестиком неправильну Створимо кнопку «Далі». Фрагмент коду наведено нижче:

```
def get_answered_message(user):
    question = db.get_question(user["question_index"])
    text = f"Питання №{user['question_index'] + 1}\n\n{question['text']}\n"
    for answer_index, answer in enumerate(question["answers"]):
        text += f"{chr(answer_index + 97)}) {answer}"
        if answer_index == question["correct"]:
            text += " ✓"
        elif answer_index == user["answers"][-1]:
            text += " ✗"
```

```
        text += "\n"
    keyboard = telebot.types.InlineKeyboardMarkup()
    keyboard.row(telebot.types.InlineKeyboardButton("Далі", callback_data="?next"))
    return {
        "text": text,
        "keyboard": keyboard
    }
```

Створюємо функцію, яка буде викликатися при натисненні кнопки «Далі», в ній збільшуємо індекс питання на 1, та відправляємо текст питання. Фрагмент коду наведено нижче:

```
@bot.callback_query_handler(func=lambda query: query.data == "?next")
def next(query):
    user = db.get_user(query.message.chat.id)

    if user["is_passed"] or not user["is_passing"]:
        return
    user["question_index"] += 1
    db.set_user(query.message.chat.id, {"question_index": user["question_index"]})
    post = get_question_message(user)
    if post is not None:
        bot.edit_message_text(post["text"], query.message.chat.id, query.message.id,
                              reply_markup=post["keyboard"])
```

РОЗДІЛ 2 ТЕСТУВАННЯ ТА АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

2.1 Тестування Telegram бота

Тестування бота здійснювалося вручну в додатку Telegram. Для початку тестування потрібно відкрити чат з ботом, знайшовши його в пошуку за нікнеймом @MyPersonalWikiBot або перейшовши за посиланням яке було видане ботом @BotFather. Процес пошуку та відкритий чат з ботом зображено на рис. 2.1 та рис 2.2.

Рис. 2.1 Пошук telegram бота

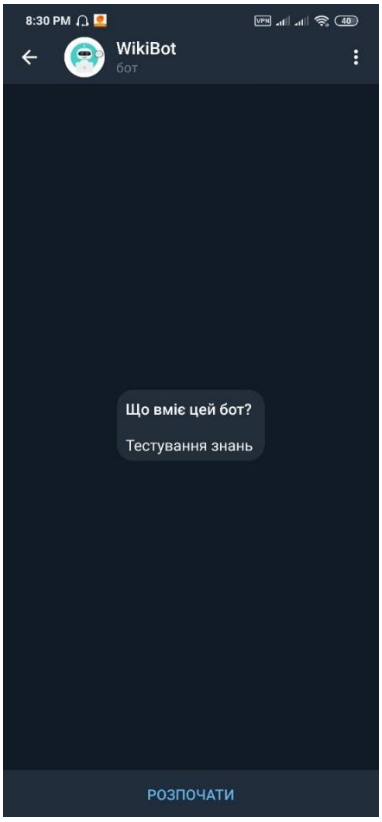

Рис. 2.2 Вікно чату з ботом

Для того щоб запустити бота, потрібно натиснути клавішу «Розпочати», або відправити команду /start. Після запуску бота, одразу з’явиться перше запитання (рис. 2.3).

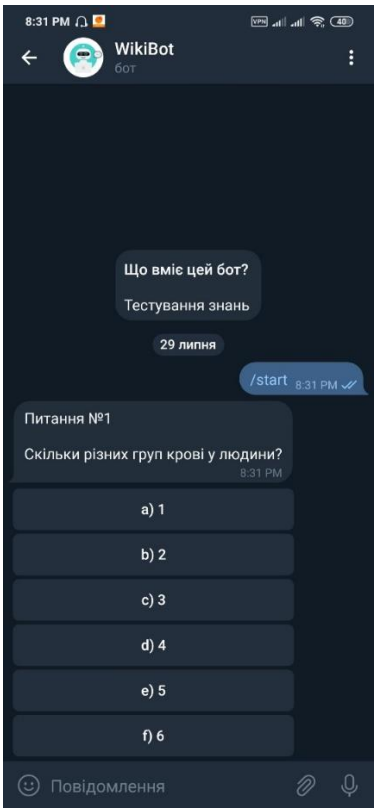

Рис. 2.3 Запуск бота

Після вибору відповіді, бот замінює повідомлення на нове, в якому відображаються дані відповіді, і кнопка «Далі», щоб перейти до наступного питання. Галочкою позначається правильна відповідь, хрестиком неправильна (рис.2.4 та рис.2.5).

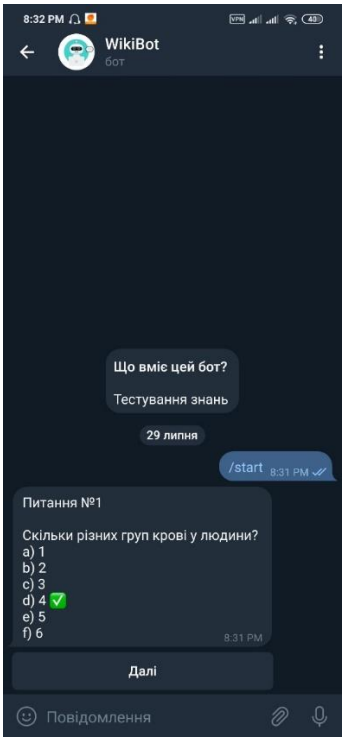

Рис. 2.4 Повідомлення про правильну відповідь

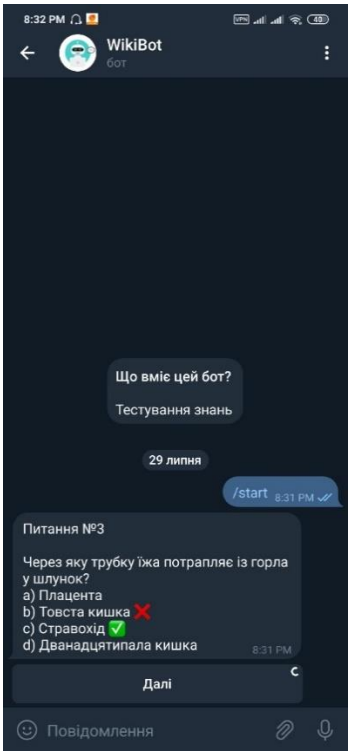

Рис. 2.4 Повідомлення про неправильну відповідь

Після проходження усього тесту, з’являється повідомлення з кількістю правильних відповідей у відсотках (рис. 2.5)

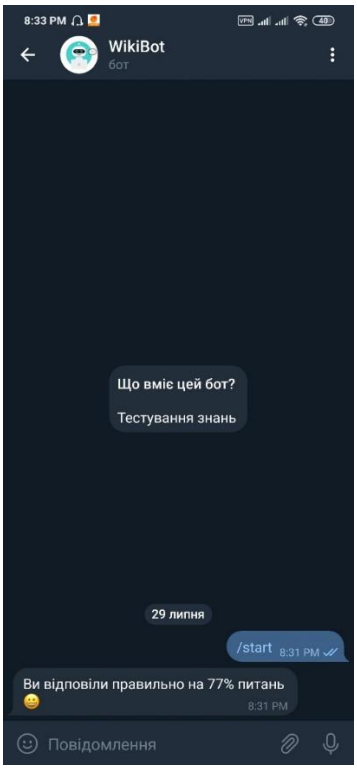

Рис. 2.5 Результат проходження тесту

При повторній спробі пройти тест з’явиться повідомлення «Ви вже пройшли це тестування. Вдруге пройти неможливо» (рис. 2.6).

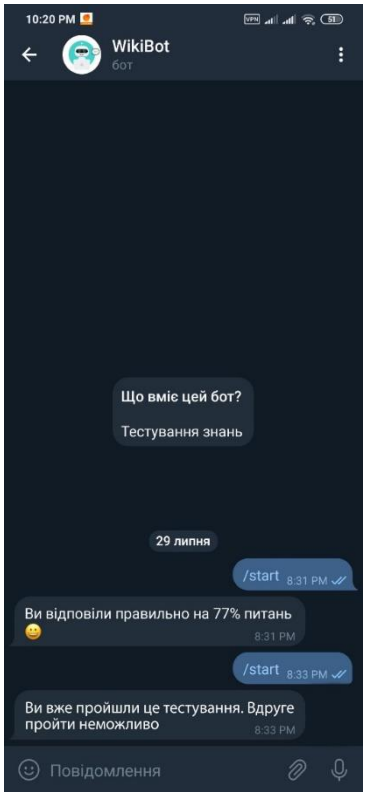

Рис. 2.6 Повторна спроба пройти тестування

ВИСНОВОК

В даній лабораторній роботі було створено telegram-бота для тестування рівня знань за допомогою офіційного бота додатку Telegram @BotFather, реалізовано та детально описано програмний код, який було написано на багатоцільовій мові програмування Python, з використанням таких бібліотек як pyTelegramBotAPI та pymongo.

Було проведено тестування розробленого telegram-бота. Проведено аналіз різних сценаріїв, наприклад проведення повторного тестування для перевірки знань.

Програмний код створеного додатку:

```
import telebot
from pymongo import MongoClient
bot = telebot.TeleBot("5424746586:AAGX-3cqvuxfO8lI_cdXe_NsWJPfswSNRxI")
class DataBase:
    def __init__(self):
        cluster = MongoClient("mongodb+srv://user:12345@cluster0.0dsh8.mongodb.net/?retryWrites=true&w=majority")
        self.db = cluster["WikiBot"]
        self.users = self.db["Users"]
        self.questions = self.db["Bot"]
        self.questions_count = len(list(self.questions.find({})))
    def get_user(self, chat_id):
        user = self.users.find_one({"chat_id": chat_id})
        if user is not None:
            return user
        user = {
            "chat_id": chat_id,
            "is_passing": False,
            "is_passed": False,
            "question_index": None,
            "answers": []
        }
        self.users.insert_one(user)
        return user
    def set_user(self, chat_id, update):
        self.users.update_one({"chat_id": chat_id}, {"$set": update})
    def get_question(self, index):
        return self.questions.find_one({"id": index})
db = DataBase()
@bot.message_handler(commands=["start"])
def start(message):
    user = db.get_user(message.chat.id)
    if user["is_passed"]:
```

```
                bot.send_message(message.from_user.wid, "Ви вже пройшли це тестування. Вдруге пройти неможливо")
                return
        if user["is_passing"]:
                return
        db.set_user(message.chat.id, {"question_index": 0, "is_passing": True})
        user = db.get_user(message.chat.id)
        post = get_question_message(user)
        if post is not None:
                bot.send_message(message.from_user.id, post["text"], reply_markup=post["keyboard"])

@bot.callback_query_handler(func=lambda query: query.data.startswith("?ans"))
def answered(query):
        user = db.get_user(query.message.chat.id)
        if user["is_passed"] or not user["is_passing"]:
                return
        user["answers"].append(int(query.data.split("&")[1]))
        db.set_user(query.message.chat.id, {"answers": user["answers"]})
        post = get_answered_message(user)
        if post is not None:
                bot.edit_message_text(post["text"], query.message.chat.id, query.message.id,
                                      reply_markup=post["keyboard"])
@bot.callback_query_handler(func=lambda query: query.data == "?next")
def next(query):
        user = db.get_user(query.message.chat.id)
        if user["is_passed"] or not user["is_passing"]:
                return
        user["question_index"] += 1
        db.set_user(query.message.chat.id, {"question_index": user["question_index"]})
        post = get_question_message(user)
        if post is not None:
                bot.edit_message_text(post["text"], query.message.chat.id, query.message.id,
                                      reply_markup=post["keyboard"])
def get_question_message(user):
        if user["question_index"] == db.questions_count:
```

```
                count = 0
                for question_index, question in enumerate(db.questions.find({})):
                        if question["correct"] == user["answers"][question_index]:
                                count += 1
                percents = round(100 * count / db.questions_count)
                if percents < 40:
                        smile = "😕"
                elif percents < 60:
                        smile = "😐"
                elif percents < 90:
                        smile = "😀"
                else:
                        smile = "😎"
                text = f"Ви відповіли правильно на {percents}% питань {smile}"
                db.set_user(user["chat_id"], {"is_passed": True, "is_passing": False})
                return {
                        "text": text,
                        "keyboard": None
                }

        question = db.get_question(user["question_index"])
        if question is None:
                return
        keyboard = telebot.types.InlineKeyboardMarkup()
        for answer_index, answer in enumerate(question["answers"]):
                keyboard.row(telebot.types.InlineKeyboardButton(f"{chr(answer_index + 
97)}) {answer}",callback_data=f"?ans&{answer_index}"))
        text = f"Питання №{user['question_index'] + 1}\n\n{question['text']}"
        return {
                "text": text,
                "keyboard": keyboard
        }
    def get_answered_message(user):
        question = db.get_question(user["question_index"])
        text = f"Питання №{user['question_index'] + 1}\n\n{question['text']}\n"
```

```
        for answer_index, answer in enumerate(question["answers"]):
            text += f"{chr(answer_index + 97)}) {answer}"
            if answer_index == question["correct"]:
                text += " ✓"
            elif answer_index == user["answers"][-1]:
                text += " ✗"
            text += "\n"

        keyboard = telebot.types.InlineKeyboardMarkup()
        keyboard.row(telebot.types.InlineKeyboardButton("Далі", callback_data="?next"))
        return {
            "text": text,
            "keyboard": keyboard
        }
bot.polling()
```