INTERNATIONAL BURCH UNIVERSITY

FACULTY OF ENGINEERING, NATURAL AND MEDICAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

PROJECT : BASIC CHATBOT

IBU 008 Programming I

Seida Subašić

Sarajevo, 2023

1. Introduction

In today's tech-driven world, chatbots are developed to make talking to machines feel easy and friendly. This project aims to simplify how users interact with technology by creating a basic user-friendly chatbot that handles logins, registrations, and provides answers to common questions.

The main goals of the project are to create a versatile chatbot that manages user logins, responds to questions, and showcases the practical use of natural language to make user interactions smoother. This report explains how the chatbot works, from its design to how it handles user logins and answers questions. It also discusses why the chatbot is essential for improving user engagement, sharing information, and connecting with the community. The chatbot goes beyond just answering questions. It's designed to streamline user interactions, making communication more straightforward and efficient. It contributes to the evolution of technology by enhancing accessibility and user interaction.

In the following sections, the report will dive into the details of the chatbot project. It will cover how it's designed, how it works with logins and registrations, and its role in answering questions. Finally, the report will wrap up with insights into the project's achievements and thoughts on future developments.

2. Program Functionalities

1. User Registration and Login:
   - Description: Allows users to register by choosing a unique username, a 4-character password, and providing an email. Registered users can log in using their credentials.
   - Importance: Essential for personalized interactions and maintaining user-specific information.

2. Login Authentication and Account Locking:
   - Description: Implements a secure login process, limiting failed attempts to 5. After exceeding this limit, the account is locked for enhanced security.
   - Importance: Enhances security by preventing unauthorized access and protects user accounts from potential breaches.

3. Question-Response Interaction with Chatbot:
   - Description: Users can ask the chatbot predefined questions, receiving predefined responses. If the user is logged in, the questions are logged.
   - Importance: Provides a conversational interface for users to seek information and engages them in an interactive experience.

4. Question History Logging:
   - Description: Logs questions asked by users when interacting with the chatbot. This information is stored in a file for future reference.
   - Importance: Enables users to review past interactions, facilitating a personalized and informed experience.

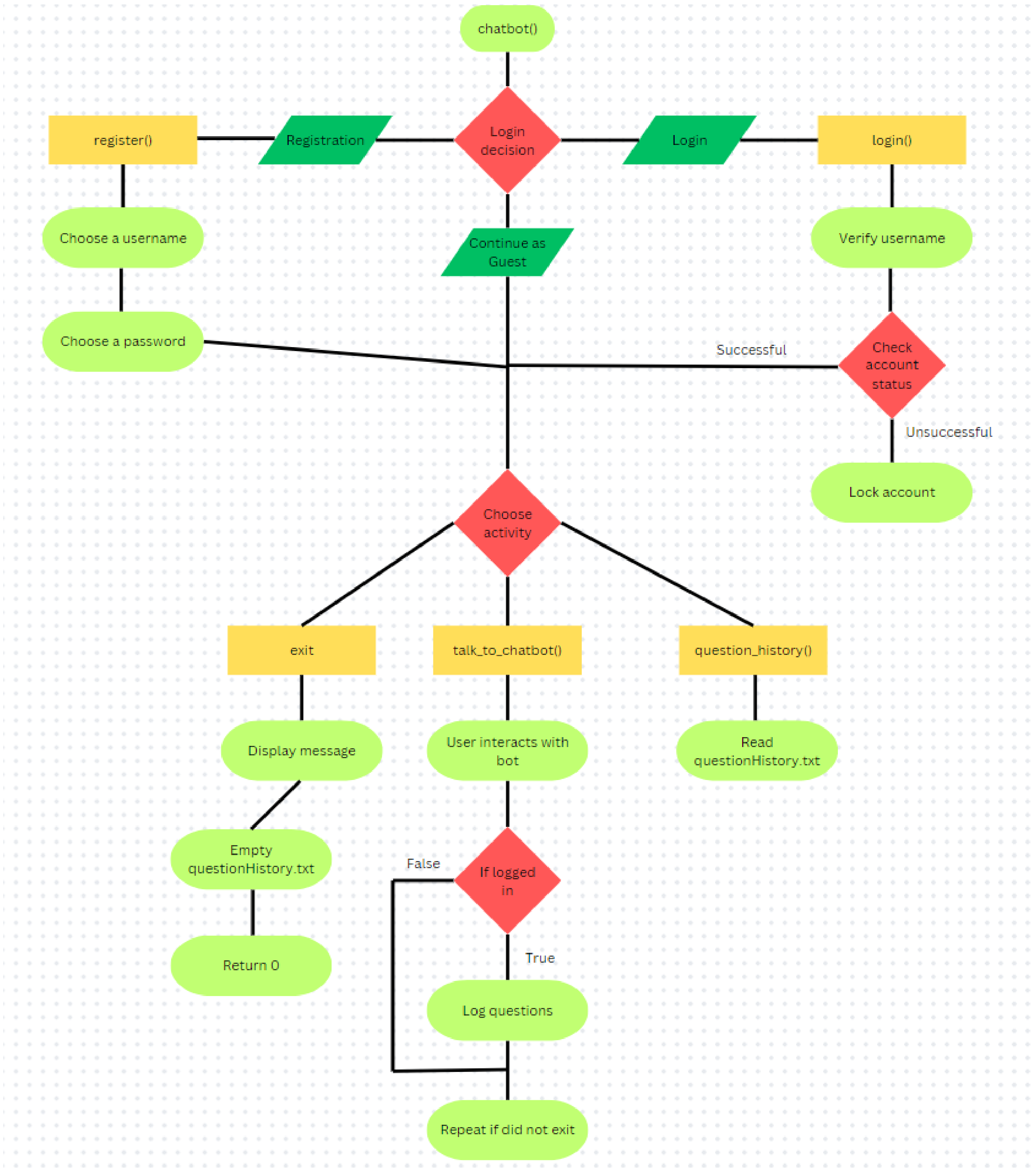
5. User Account Status Management:
   - Description: Tracks the status of user accounts, allowing for account locking and unlocking based on login attempts.
   - Importance: Enhances account security and prevents unauthorized access by implementing status-based controls.

6. User Registration Error Handling:
   - Description: Ensures that users choose a unique username and a 4-character password during the registration process.
   - Importance: Maintains data integrity and prevents registration errors, ensuring a smooth onboarding experience.

7. Question History Display:
   - Description: Reads and displays the logged questions from the question history file when requested by the user.
   - Importance: Offers users a retrospective view of their interactions, promoting transparency and accountability.

8. File Content Deletion:
   - Description: Allows for the deletion of content in the question history file, providing users with the option to clear their interaction history.
   - Importance: Offers users control over their data and privacy by enabling them to manage stored information.

3. Flowcharts



Flowchart Description for ChatBot Project:

1. Start Program:
   - Execution begins at the start of the program.

2. User Interaction:
   - Users decide whether to continue as a guest, log in, or register.

3. Guest or Log In:
   - If continuing as a guest, the program proceeds to the chatbot options.
   - If logging in or registering, users are directed to the respective functionalities.

4. Log In/Register:
   - Users provide necessary information for login or registration.

5. Check Login:
   - For login, the program verifies the credentials, locks the account after 5 failed attempts, or proceeds to chatbot options if successful.

6. Chatbot Options:
   - Users can talk to the chatbot, see question history, or exit the program.

7. Talk to Chatbot (User Interaction):
   - Users interact with the chatbot, asking questions and receiving predefined responses.

8. Ask Question:
   - User inputs a question, and the chatbot provides a response.

9. Log Question (if logged in):
   - If the user is logged in, the question is logged for future reference.

10. End Interaction:
    - The chatbot interaction concludes.

11. See Question History:
    - Users choose to view their question history.

12. Display Questions:
    - The program reads and displays logged questions from the file.

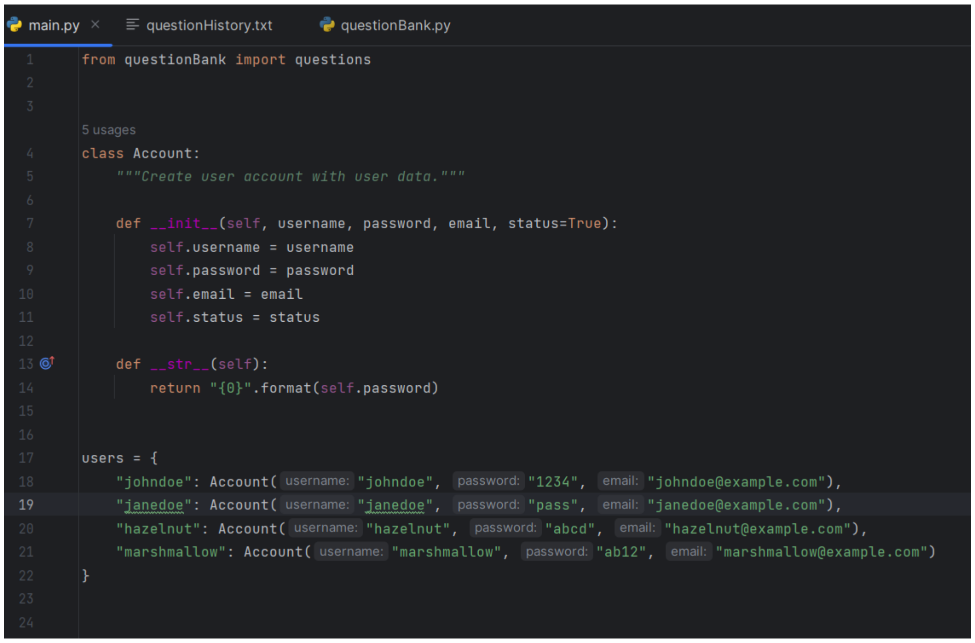
13. Delete File Content (if any):
    - Users can choose to delete the content of the question history file.

14. End Program:

   - Execution concludes.

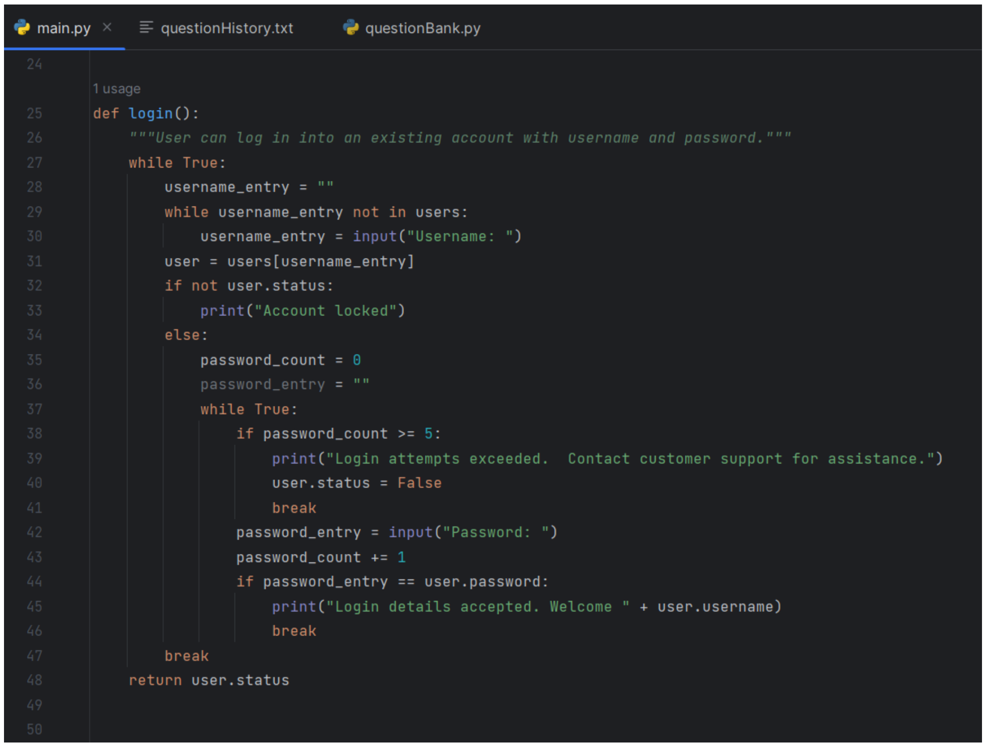
This flowchart description outlines the step-by-step process of user interaction, login or registration, chatbot interactions, question logging, and question history management in the ChatBot project. Each decision point and action contributes to the overall functionality of the program.

4. Code



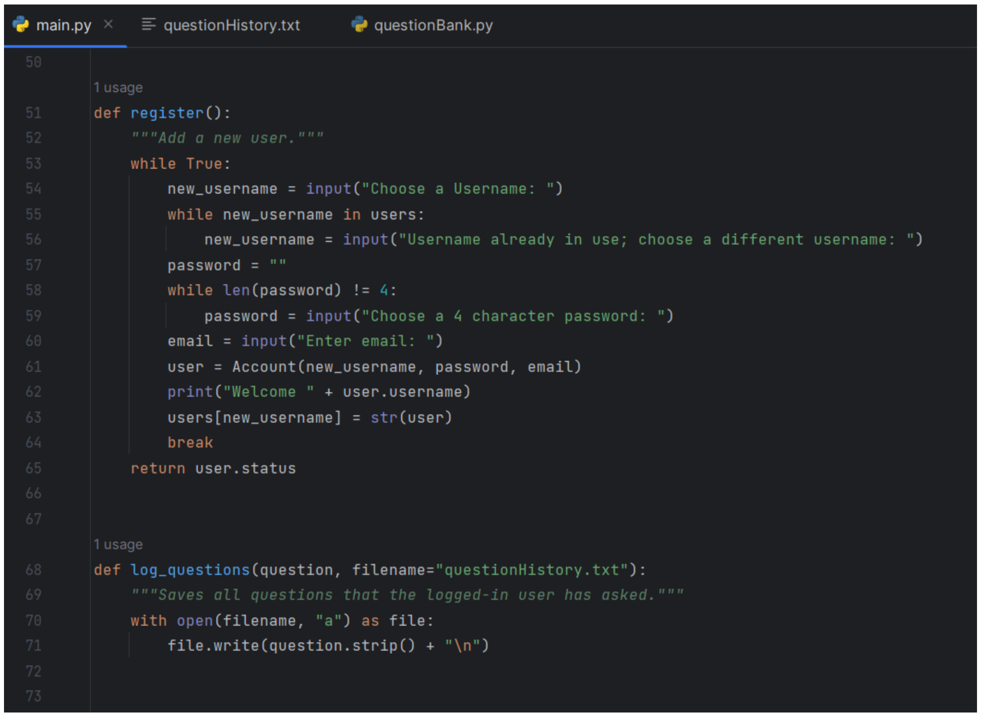
main.py × questionHistory.txt questionBank.py

```
from questionBank import questions


5 usages
class Account:
    """Create user account with user data."""

    def __init__(self, username, password, email, status=True):
        self.username = username
        self.password = password
        self.email = email
        self.status = status

    def __str__(self):
        return "{0}".format(self.password)


users = {
    "johndoe": Account( username: "johndoe",  password: "1234",  email: "johndoe@example.com"),
    "janedoe": Account( username: "janedoe",  password: "pass",  email: "janedoe@example.com"),
    "hazelnut": Account( username: "hazelnut",  password: "abcd",  email: "hazelnut@example.com"),
    "marshmallow": Account( username: "marshmallow",  password: "ab12",  email: "marshmallow@example.com")
}
```

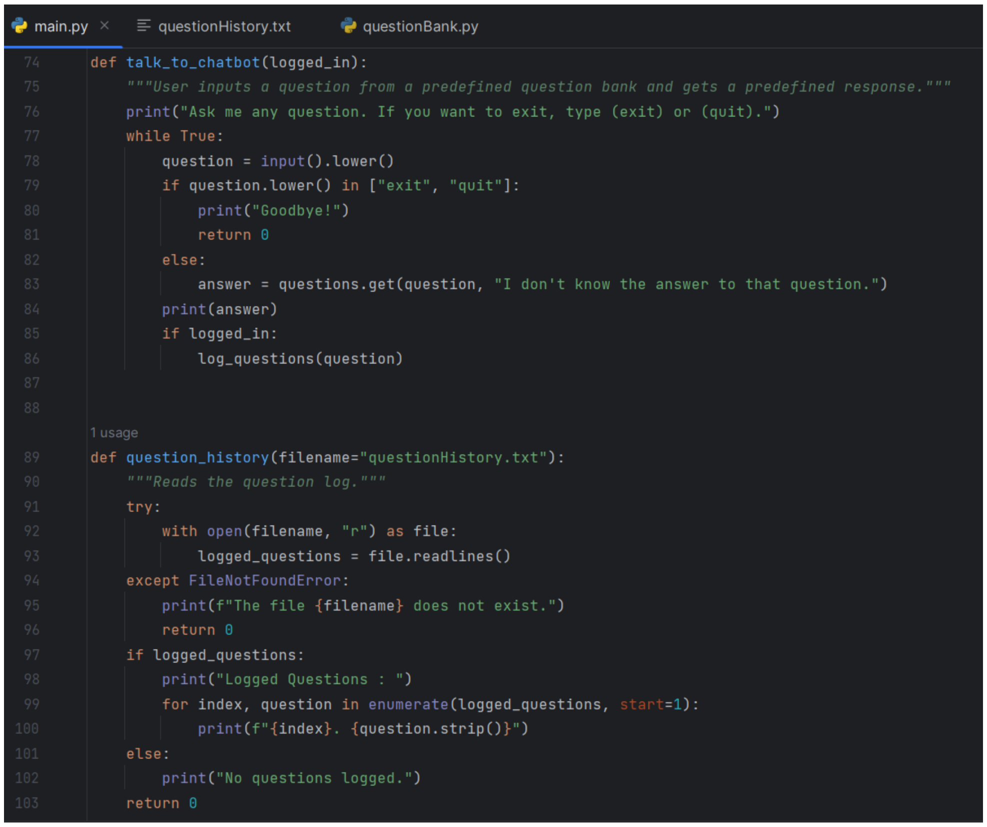


main.py × questionHistory.txt questionBank.py

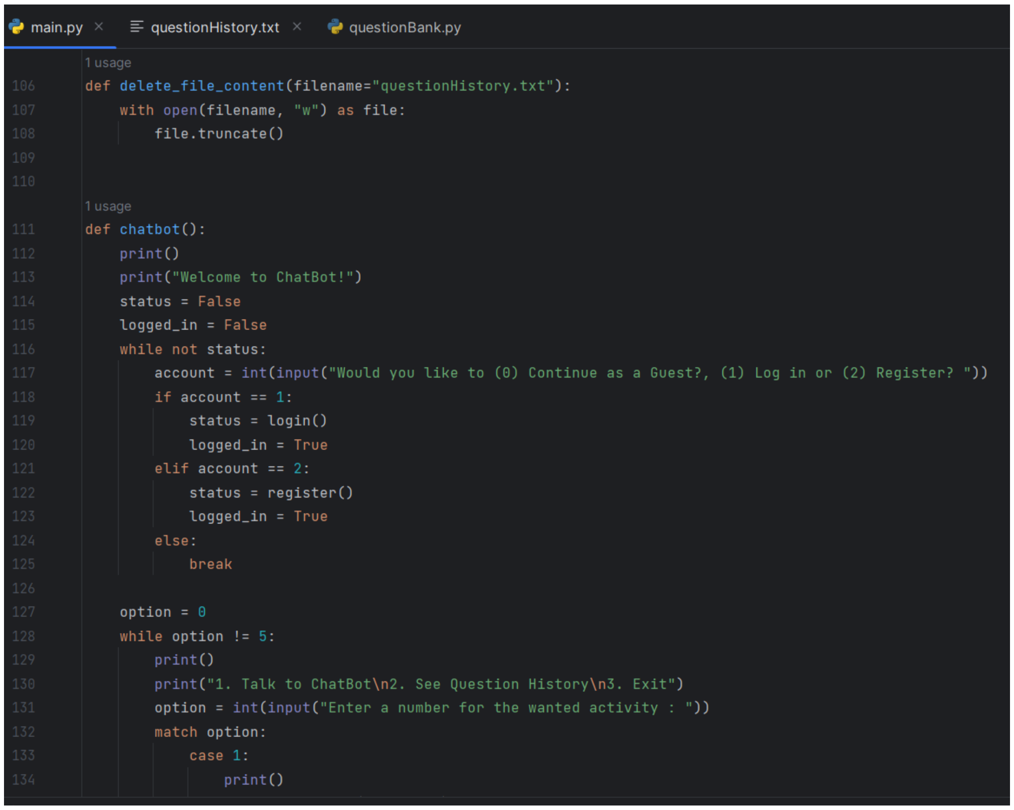
```
1 usage
def login():
    """User can log in into an existing account with username and password."""
    while True:
        username_entry = ""
        while username_entry not in users:
            username_entry = input("Username: ")
        user = users[username_entry]
        if not user.status:
            print("Account locked")
        else:
            password_count = 0
            password_entry = ""
            while True:
                if password_count >= 5:
                    print("Login attempts exceeded.  Contact customer support for assistance.")
                    user.status = False
                    break
                password_entry = input("Password: ")
                password_count += 1
                if password_entry == user.password:
                    print("Login details accepted. Welcome " + user.username)
                    break
        break
    return user.status
```


main.py × questionHistory.txt questionBank.py

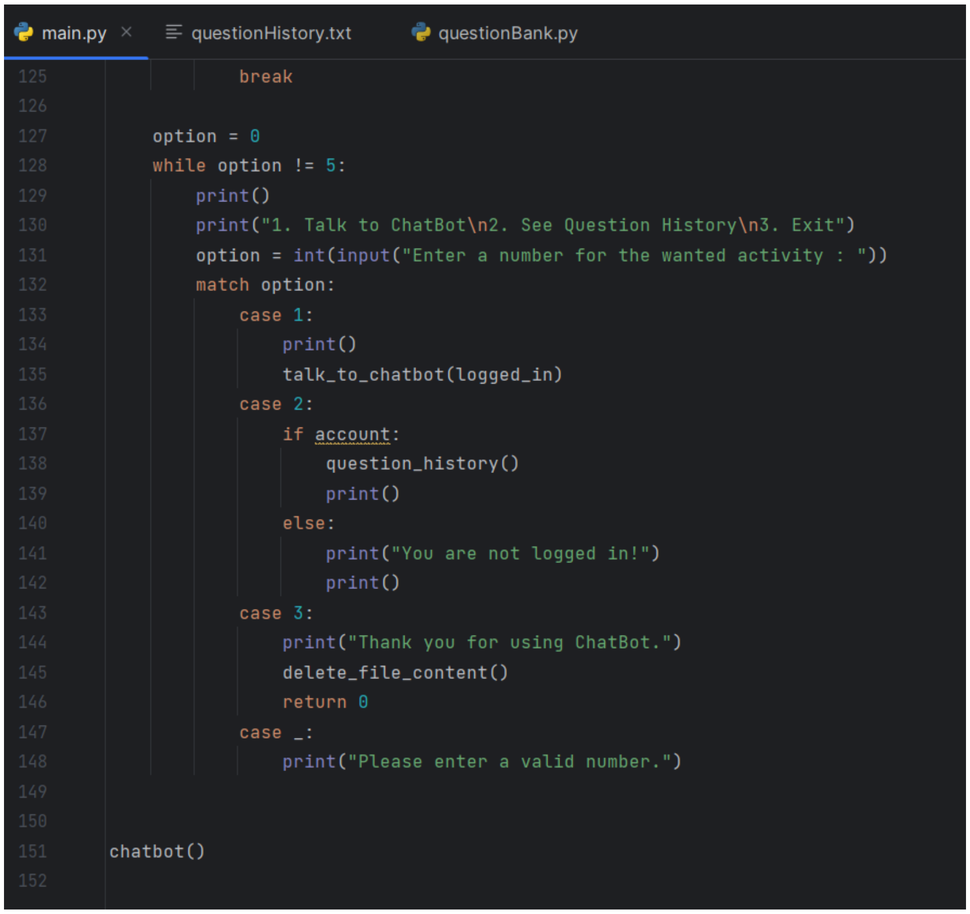
```
1 usage
def register():
    """Add a new user."""
    while True:
        new_username = input("Choose a Username: ")
        while new_username in users:
            new_username = input("Username already in use; choose a different username: ")
        password = ""
        while len(password) != 4:
            password = input("Choose a 4 character password: ")
        email = input("Enter email: ")
        user = Account(new_username, password, email)
        print("Welcome " + user.username)
        users[new_username] = str(user)
        break
    return user.status


1 usage
def log_questions(question, filename="questionHistory.txt"):
    """Saves all questions that the logged-in user has asked."""
    with open(filename, "a") as file:
        file.write(question.strip() + "\n")


```


main.py × questionHistory.txt questionBank.py

```
def talk_to_chatbot(logged_in):
    """User inputs a question from a predefined question bank and gets a predefined response."""
    print("Ask me any question. If you want to exit, type (exit) or (quit).")
    while True:
        question = input().lower()
        if question.lower() in ["exit", "quit"]:
            print("Goodbye!")
            return 0
        else:
            answer = questions.get(question, "I don't know the answer to that question.")
        print(answer)
        if logged_in:
            log_questions(question)


1 usage
def question_history(filename="questionHistory.txt"):
    """Reads the question log."""
    try:
        with open(filename, "r") as file:
            logged_questions = file.readlines()
    except FileNotFoundError:
        print(f"The file {filename} does not exist.")
        return 0
    if logged_questions:
        print("Logged Questions : ")
        for index, question in enumerate(logged_questions, start=1):
            print(f"{index}. {question.strip()}")
    else:
        print("No questions logged.")
    return 0
```



```python
def delete_file_content(filename="questionHistory.txt"):
    with open(filename, "w") as file:
        file.truncate()


def chatbot():
    print()
    print("Welcome to ChatBot!")
    status = False
    logged_in = False
    while not status:
        account = int(input("Would you like to (0) Continue as a Guest?, (1) Log in or (2) Register? "))
        if account == 1:
            status = login()
            logged_in = True
        elif account == 2:
            status = register()
            logged_in = True
        else:
            break

    option = 0
    while option != 5:
        print()
        print("1. Talk to ChatBot\n2. See Question History\n3. Exit")
        option = int(input("Enter a number for the wanted activity : "))
        match option:
            case 1:
                print()
```



```python
            break

    option = 0
    while option != 5:
        print()
        print("1. Talk to ChatBot\n2. See Question History\n3. Exit")
        option = int(input("Enter a number for the wanted activity : "))
        match option:
            case 1:
                print()
```

```python
                talk_to_chatbot(logged_in)
            case 2:
                if account:
                    question_history()
                    print()
                else:
                    print("You are not logged in!")
                    print()
            case 3:
                print("Thank you for using ChatBot.")
                delete_file_content()
                return 0
            case _:
                print("Please enter a valid number.")


chatbot()
```

5. Results

This section will enhance the clarity by incorporating visual representations of the ChatBot program. The inclusion of screenshots offers readers a direct view of the program's user interface and specific outcomes, providing a comprehensive understanding of its functionalities.

User Registration and Login Screenshots:

Description:
- This screenshot illustrates the user registration process.
- Users are prompted to choose a unique username, input a 4-character password, and provide an email.
- The purpose is to showcase the user-friendly registration interface and collect essential user information.
- The login interface is presented to users who choose to log in.
- Users input their username and password for authentication.
- This screenshot emphasizes the simplicity of the login process and user credential verification.

Chatbot Interaction Screenshots:

Chatbot Interface:

Description:
- Users can interact with the chatbot through this interface.
- The chatbot prompts users to input questions or commands.
- This screenshot demonstrates the conversational aspect of the program and the user's interaction with the chatbot.

Chatbot Response:

Description:
- After a user inputs a question, the chatbot responds with a predefined answer.
- This screenshot highlights the chatbot's role in providing informative responses to user queries.
- It showcases the seamless interaction between the user and the chatbot.

Question History Management Screenshots:

Question History Display:

Description:
- Users can view their question history through this interface.
- The program displays a list of logged questions.

- This screenshot allows users to review past interactions, promoting transparency and user engagement.

File Content Deletion Option:

Description:
- Users have the option to delete the content of the question history file.
- This screenshot emphasizes user control over data and privacy.
- It showcases the functionality to manage stored information within the program.

By including these screenshots, the report aims to provide a visual walkthrough of key aspects of the ChatBot program, enriching the reader's understanding of its user interface and functionalities. Each screenshot is accompanied by a detailed description to enhance context and facilitate a comprehensive review of the program's visual components.