Software Engineering Take Home Assignment

The provided code defines a Python class named ultraChatBot, which serves as a framework for a WhatsApp-based chatbot designed to collect patient data. The bot interacts with users, prompts them for various pieces of information, securely stores the data, and allows exporting it to CSV or Excel formats. Here's an overview of its architecture:

- Messaging Platform Interface:

  The chatbot interacts with users via the WhatsApp messaging platform. I have used the api endpoint exposed by ultramessage.com for usng the whatsapp business api.
  It sends messages to users, prompts them for information, and receives responses.

- Chatbot Framework:
  The ultraChatBot class serves as the core framework for the chatbot. It manages conversation states, handles incoming messages, processes user inputs, and stores collected data securely.

- Data Storage
  Patient data is stored securely using SQLite, a lightweight relational database management system. The patient_records table stores encrypted patient data, ensuring confidentiality and data integrity. There are additional checks that were included to make sure that the user inputs the data following proper pattern.

- Encryption
  Sensitive patient data is encrypted before storage using the Fernet symmetric encryption algorithm. Encryption keys are generated securely to protect patient confidentiality.

- Export Functionality
  The chatbot provides options to export collected data to CSV and Excel formats. Exported files contain patient data in a structured format, facilitating further analysis and processing.

Libraries Used:

- requests:
  Used for making HTTP requests to the Ultra Messaging API for sending and receiving messages on WhatsApp.

- cryptography:
  Provides encryption and decryption capabilities using various cryptographic algorithms.
  Utilized for encrypting sensitive patient data before storing it in the SQLite database.

- pymongo:
  A Python driver for MongoDB, used for interacting with MongoDB databases (although not explicitly used in this code).

- sqlite3:
  A built-in Python module for SQLite database operations. Used for creating and managing the SQLite database for storing patient records securely.

- csv:
  A built-in Python module for reading and writing CSV files. Used for exporting patient data to CSV format.

- openpyxl:
  A Python library for reading and writing Excel files (XLSX). Used for exporting patient data to Excel format.

Setup Guide:

Step 1: install flask

we need to deploy the server using the FLASK framework.
The FLASK allows to conveniently respond to incoming requests and process them.

```
pip install flask
```

Step 2 : install ngrok

for local development purposes, a tunneling service is required. I have used ngrok for the same.

Step3: start WhatsApp Chatbot project

Run FLASK server by running the following command.

```
flask run
```

Run ngrok by running the following command

```
./ngrok http 5000
```

Step 4: Set URL Webhook in Instance settings.

After run ngrok, we need to update the instance id on the website for the messages to be received. The website is https://user.ultramsg.com/app/instances/instance.php
Once the instance is created on the site, the instance id and token numbers are replaced in the code for the connection to be established.

Security Measures Implemented

- Data Encryption: The code includes functionality for encrypting patient data using the cryptography library's Fernet module before storing it in the database. This ensures that sensitive patient information is protected from unauthorized access.
- The code communicates with the UltraMsg API using unencrypted HTTPS request which ensures secure transmission of the data over the network.

Potential Vulnerabilities and Mitigations (Future Scope):

- I am currently using SQLLite database which can be a potential threat. Instead, we can use mongoDb with encrypted database credentials.
- The code contains hardcoded API credentials (self.ultraAPIUrl and self.token), which is a security risk. If the code is compromised, these credentials can be exposed, leading to unauthorized access to the API. Rather we can store API credentials securely, such as in environment variables or a secure key management system. Avoid hardcoding sensitive information in the codebase.
- Currently everyone can download the .csv files which can be a potential threat as there is no access control or protection mechanism. To mitigate this, we can implement access control mechanism for restricting the access.
- Also there is no authentication and authorization mechanisms implemented here. We can have some role specific access provided to clients in order to have more secured trasactions.