Medical Chatbot

Session: 2022 – 2026

Submitted by:

Ameer Hamza	2022-CS-17
Muhammad Zubair	2022-CS-20

Submitted to:

Sir Waqas Ali

Department of Computer Science
University of Engineering and Technology Lahore, Pakistan

Contents

1 Introduction

The Medical Chatbot is a secure web application for people who would like to access medical data by uploading PDF files and communicate with a chatbot. It is developed and tested using Django framework. The focus of this project is on protecting patient personal health information (PHI) in a Zero Trust security model and using some advanced cryptographic algorithms and robust authentication features to assure confidentiality of the data and user verification as well as implementation of medical data protection standards. This documentation will give you an overview of the project objectives, the Security implementation and of the operational workflow.

2 Existing Problem

The introduction of a medical chatbot which could assist with processing of confidential medical information and assist users in using medical services also posed significant security risks.

- **User unauthentication:** Without adequate user authentication mechanisms, unauthorized users may create accounts or connect to the system, which can lead to account spoofing and misuse of medical data.
- **Vulnerable User Profiles:** Maintaining or transmitting user credentials (for example, passwords) in plaintext exposed them to interception or database hacking and could expose user accounts.
- **Unencrypted medical data:** Uploaded PDFs and user chats with the chatbot were open to unauthorized access/interception and were subject to potential PHI (Protected Health Information) leaks.
- **Trust & Verification:** Lack of a "never trust, always verify" approach caused vulnerabilities in user sessions, data storage, and communication paths, increasing the chance of security problems.

These vulnerabilities require an extremely robust security posture to safeguard user data, maintain system integrity, and adhere to regulatory requirements.

3 Project Overview

The Medical Chatbot solves the need to create a secure platform for medical documents processing and user-specific medical information. The system allows users to upload PDF files with their medical records, which are processed to allow interactive conversation with a chatbot. Security of the medical data is of great importance due to the sensitive nature of this, so the project uses a Zero Trust security model to reduce the risks.

3.1 Objectives

- Ensure strong user authentication through OTP-based verification during signup.
- Protect user credentials using RSA encryption to prevent unauthorized access.
- Secure uploaded PDFs and chatbot conversations using AES encryption for confidentiality.
- Implement Zero Trust practices to eliminate implicit trust and reduce attack surface.
- Provide a simple interface so users can upload documents and interact with chatbots easily.

3.2 Key Features

- **OTP Verification on Signup:** Uses email-based one-time passwords for user verification.
- **RSA-Encrypted Credentials:** Secures user credentials using asymmetric encryption.
- **PDF Upload and Processing:** Allows users to upload medical PDFs and interact with them.
- **AES-Encrypted Data:** Protects PDFs and chatbot conversations for data confidentiality.
- **Zero Trust Security:** Requires continuous verification and encryption throughout the system.

4 Solution Overview

To meet the identified security challenges, the project provides an enterprise-wide security framework specifically addressed through the following solutions:

- **OTP-Based Authentication:** Verifies user identity during signup using email-delivered OTPs to prevent fraudulent account creation.
- **RSA Encryption for Credentials:** Efficiently encrypts user credentials using asymmetric encryption during storage and transmission.
- **AES Encryption for Data:** Applies AES encryption to uploaded PDFs and chatbot conversations to maintain confidentiality.
- **Zero Trust Architecture:** Adopts a "never trust, always verify" model for continuous authentication and data protection.
- **Secure Application Configurations:** Utilizes Django's built-in security features (e.g. Secure email delivery) to minimize system vulnerabilities.

These solutions collectively address the previously identified vulnerabilities, resulting in a secure and compliant medical chatbot system.

5 System Architecture

5.1 Technology Stack

- **Framework:** Django 4.2.20, a Python-based web framework known for security and scalability.
- **Database:** SQLite3 for development, with the option to upgrade to PostgreSQL for production.
- **Frontend:** Django templates with static CSS and JavaScript for a responsive interface.
- **Email Service:** Gmail SMTP for secure OTP delivery.

5.2 5.2 System Components

The system is composed of multiple integrated components, each responsible for handling critical functions ranging from secure user authentication to encrypted data processing. These components collectively ensure a robust, secure, and user-friendly platform for interacting with medical data through a chatbot interface.

User Management Module:

- This module is responsible for handling all user-related operations, including account creation, login, and OTP-based identity verification. OTPs are delivered securely via email to prevent unauthorized access.
- User credentials are encrypted using RSA (Rivest–Shamir–Adleman) asymmetric encryption before being stored in the system database, reducing the risk of data breaches.
- It ensures that no sensitive information is stored or transmitted in plaintext, adhering to security best practices and data protection standards.

PDF Upload Module:

- This component allows users to upload medical documents in PDF format through an intuitive interface. All uploads are scanned and encrypted immediately to ensure privacy.
- Once uploaded, the system parses these documents and extracts key information, which is later utilized by the chatbot for intelligent conversations.
- The extracted data is temporarily stored in an encrypted form to facilitate real-time interaction without compromising confidentiality.

Chatbot Engine:

- The chatbot serves as the user's primary interface for interacting with their medical data. It interprets user queries and responds based on the information derived from the uploaded PDF.
- All chatbot conversations are encrypted using AES (Advanced Encryption Standard) to preserve the confidentiality of sensitive health information.
- The engine is optimized for contextual understanding and secure, real-time communication with users.

Security Layer:

- This foundational layer integrates multiple security mechanisms to protect user data. RSA encryption secures credentials, while AES encryption secures uploaded documents and chat history.
- OTP-based user verification ensures that only legitimate users can access the system.
- The implementation follows a Zero Trust Architecture, meaning that every access request is authenticated and validated, regardless of origin or user role.
- This model minimizes the possibility of lateral movement by attackers within the system.

Session Management:

- Secure session handling is achieved through encrypted cookies with strict expiration rules, preventing session hijacking and unauthorized reuse.
- Sessions are continuously monitored and revalidated to maintain the principles of Zero Trust and ensure that no implicit trust is granted after initial login.
- The session layer also logs critical security events, which can be audited for compliance and incident response.

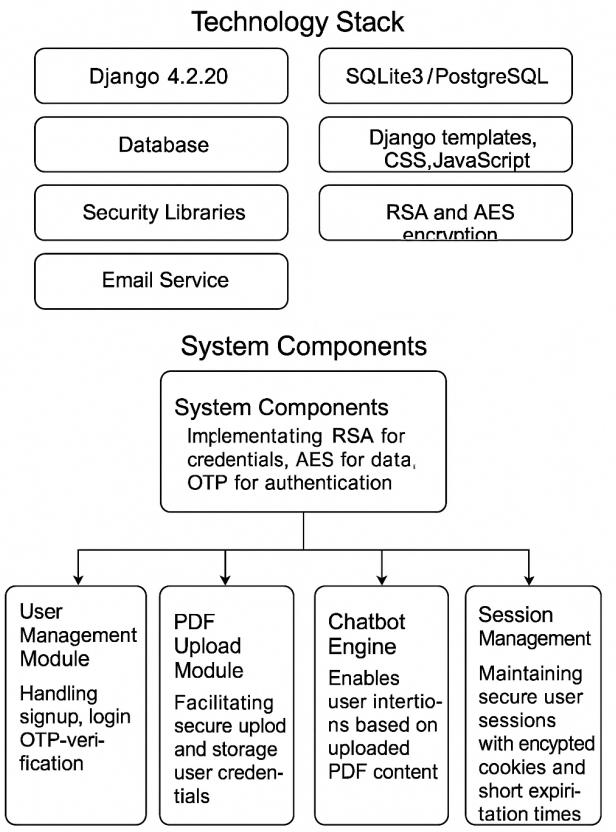

Figure 1: Secure Medical Chatbot System Architecture

6 Zero Trust Security Model

The Zero Trust security model is a cybersecurity framework that assumes no user, device, or network is inherently trustworthy, adhering to the principle of "never trust, always verify." This approach eliminates implicit trust by requiring continuous authentication, authorization, and encryption for every access request, regardless of the user's location or credentials. In the Medical Chatbot project, Zero Trust is foundational to protecting sensitive personal health information (PHI). It is implemented through robust mechanisms such as OTP-based authentication for signup and login, RSA encryption for user credentials, and AES encryption for PDFs and chat data. User sessions are validated on every request with short-lived, encrypted cookies. Access controls restrict users to their own data, and secure communication channels

(e.g., HTTPS, TLS for email) prevent interception. By adopting Zero Trust, the system minimizes attack surfaces, mitigates risks of data breaches, and aligns with medical data protection standards like HIPAA, ensuring a secure and compliant platform for medical interactions.

7 Security Implementation

7.1 OTP on Signup and Login

Purpose: To verify user identity during account creation and after failed login attempts, preventing unauthorized access.

Process:

- **Signup:** During signup, users provide an email address and password. A 6-digit OTP is generated and sent to the user's email via a secure SMTP connection. The OTP remains valid for 5 minutes and is stored in a hashed format. Users must enter the correct OTP to complete the signup process.
- **Login:** If a user enters an incorrect username or password three times, an OTP is sent to their registered email for re-verification. The user must enter the correct OTP to proceed with the login. After three failed attempts, the account is blocked for 30 minutes to prevent brute-force attacks.

Security Benefits:

- Ensures only legitimate users can create accounts.
- Reduces risks of automated bot registrations and phishing attempts.

Configuration:

- Uses Django's email backend with Gmail SMTP.
- Employs a dedicated sender address and app-specific password for authentication.

7.2 RSA Encryption for User Credentials

Purpose: To protect user credentials from exposure during storage and transmission.

Process:

- An RSA key pair is generated, with the private key securely stored on the server.
- User credentials (e.g., passwords) are encrypted using the RSA public key before storage in the database.
- During login, the private key decrypts the credentials for verification.

Security Benefits:

- Asymmetric encryption ensures credentials remain secure even if the database is compromised.
- Prevents plaintext exposure during network transmission.

Implementation:

- Seamlessly integrates with Django's authentication system.

7.3 AES Encryption for PDFs and Chat Data

Purpose: To ensure the confidentiality of uploaded medical PDFs and user interactions with the chatbot.

Process:

PDF Encryption:

- AES is used, with a unique encryption key generated per user session.
- Uploaded PDFs are encrypted before storage in the designated media directory.
- PDFs are decrypted temporarily in memory during processing for chatbot responses.

Chat Encryption:

- User inputs and chatbot responses are encrypted before database storage.

- Data is decrypted only for display to the authenticated user.

Security Benefits:

- Protects sensitive medical data from unauthorized access at rest and in transit.
- Supports compliance with data protection regulations like HIPAA.

Configuration:

- PDFs are stored in a secure media directory with restricted access.
- Session-based encryption keys are managed securely to prevent exposure.

7.4 Zero Trust Security Implementation

Purpose: To enforce a "never trust, always verify" approach, ensuring no user, device, or network is inherently trusted.

Process:

Continuous Authentication:

- OTP verification is required for signup, ensuring only verified users gain access.
- User sessions are validated on every request, with short-lived cookies expiring after 10 minutes of inactivity.

Data Protection:

- All sensitive data (credentials, PDFs, chats) is encrypted using RSA and AES, both at rest and in transit.
- Encryption keys are managed securely, with access restricted to authorized processes.

Access Control:

- Only authenticated users can upload PDFs or interact with the chatbot.
- Role-based access ensures users can only view their own data.

Secure Communication:

- Session cookies are encrypted and transmitted over HTTPS to prevent hijacking.

Application Hardening:

- Secure configurations, such as expiring sessions on browser close, reduce attack surfaces.

Security Benefits:

- Eliminates implicit trust, ensuring all actions are verified.
- Reduces risks of unauthorized access, data breaches, and session attacks.
- Aligns with Zero Trust principles outlined by NIST, enhancing system resilience.

Configuration:

- Sessions are configured to expire quickly and require re-authentication.

8 User Workflow

8.1 User Signup

- The user accesses the signup page and enters their email and password.
- The system generates a 6-digit OTP and sends it to the user's email.
- The user enters the OTP to verify their identity.
- The password is encrypted using RSA and stored securely.
- Upon successful verification, the user is registered and redirected to the login page.

8.2 User Login

- The user enters their email and password on the login page.
- The system decrypts the stored password using RSA and verifies the credentials.
- A secure session is created, granting access to the chatbot interface.

8.3 PDF Upload

- The authenticated user uploads a medical PDF via the chatbot interface.
- The PDF is encrypted using AES and stored in a secure media directory.
- The system processes the PDF to extract relevant medical information.

8.4 Chatbot Interaction

- The user interacts with the chatbot, asking questions based on the uploaded PDF.
- User inputs and chatbot responses are encrypted using AES before storage.
- The chatbot generates responses using the processed PDF data, displayed securely to the user.

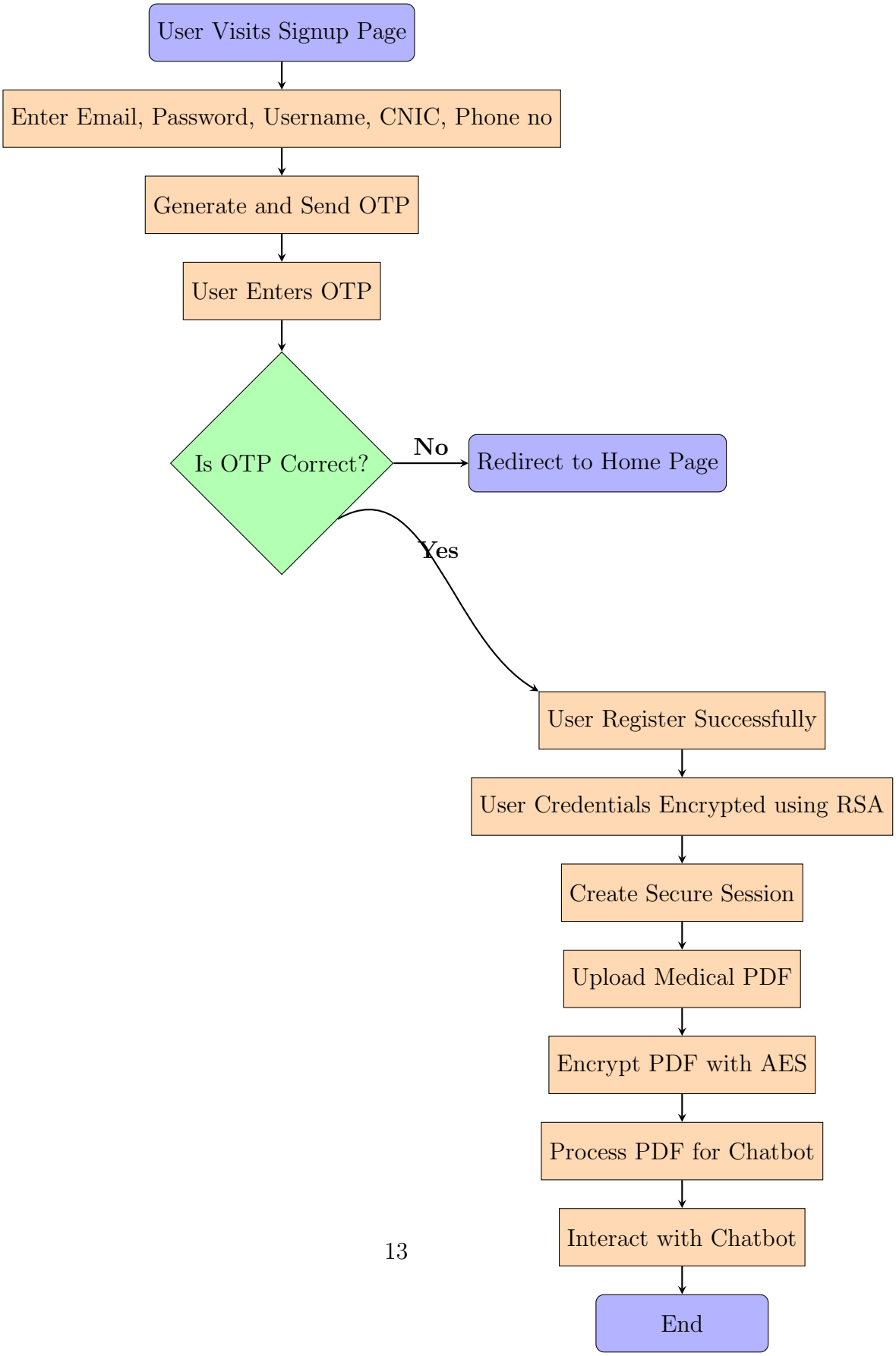

Figure 2: User Workflow Diagram for Secure Medical Chatbot

9 Application Interface Visuals

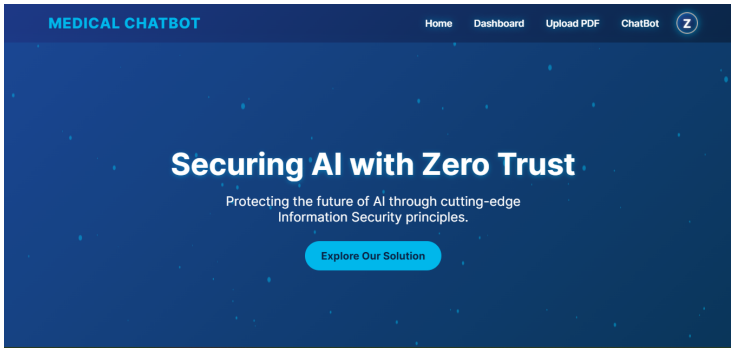

Figure 3: Home page

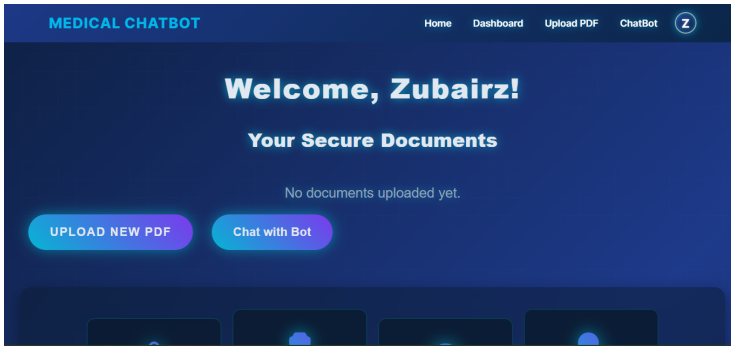

Figure 4: User Dashboard

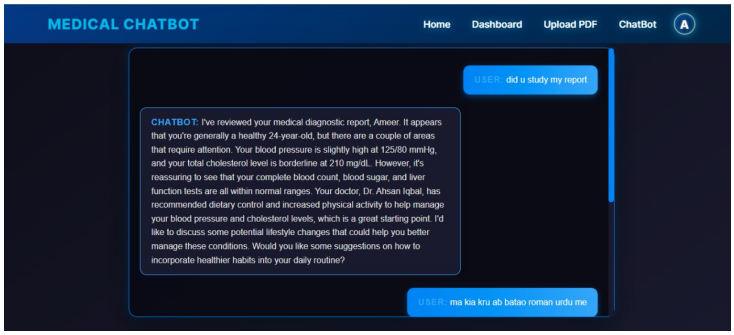

Figure 5: Chatbot

10 Future Works

To further strengthen the Medical Chatbot system, the following improvements are proposed:

1. **Production-Grade Deployment:** Transition the system to a secure production environment using a robust web server such as `Gunicorn` behind `Nginx`, along with a scalable relational database like `PostgreSQL`. Sensitive configurations such as API keys and passwords will be managed securely using environment variables. This deployment strategy ensures high availability, performance, and strong security.

2. **Regulatory Compliance Audit:** Conduct a formal audit to ensure full compliance with healthcare regulations such as **HIPAA** and **GDPR**. This will include implementing features like data anonymization, role-based access controls, and detailed audit logging, which are essential for handling sensitive medical data and building user trust.

3. **Advanced Key Management:** Integrate a **Hardware Security Module (HSM)** for the secure generation, storage, and management of RSA and AES encryption keys. Incorporating key rotation policies and limiting key exposure further supports the Zero Trust Security architecture and enhances overall cryptographic strength.

4. **Enhanced Chatbot Intelligence:** Improve the chatbot's capabilities using advanced **Natural Language Processing (NLP)** techniques and integration with verified medical knowledge bases. This will increase the accuracy of responses while maintaining end-to-end AES encryption for all interactions to ensure data confidentiality.

11 Conclusion

The Medical Chatbot provides a secure platform for interacting with medical documents, leveraging a Zero Trust model. It uses OTP-based authentication, RSA encryption for credentials, and AES encryption for PDFs and chats to protect sensitive data. Django's secure configurations enhance resilience against threats. Future enhancements, including production deployment and compliance audits, will improve reliability and regulatory adherence. This project balances usability with robust security, applying advanced information security principles to a medical use case.