AI Chat Application with Flask & OpenAI Integration

Presenter: [T.Pavan], Software Trainee

Project Overview

Objective: Develop a secure web application enabling users to interact with OpenAI's GPT-3.5 model.

Key Features:

- User authentication (signup/login)
- Secure password management
- Personalized AI chat history
- Contextual AI responses
- Chat history management

Technology Stack

- **Backend:** Flask (Python)
- **Database:** SQLite with SQLAlchemy ORM
- **Authentication:** Flask-Login, Flask-WTF, Flask-Bcrypt
- **AI Integration:** OpenAI GPT-3.5 via API
- **Frontend:** HTML5, Jinja2 Templates
- **Environment Management:** python-dotenv

User Interface Overview

Login Page: User authentication with form validation.

Signup Page: New user registration with validation checks.

Chat Interface:

- Input field for user queries.
- Display area for AI responses.
- Chat history displayed with timestamps.
- Option to clear chat history.

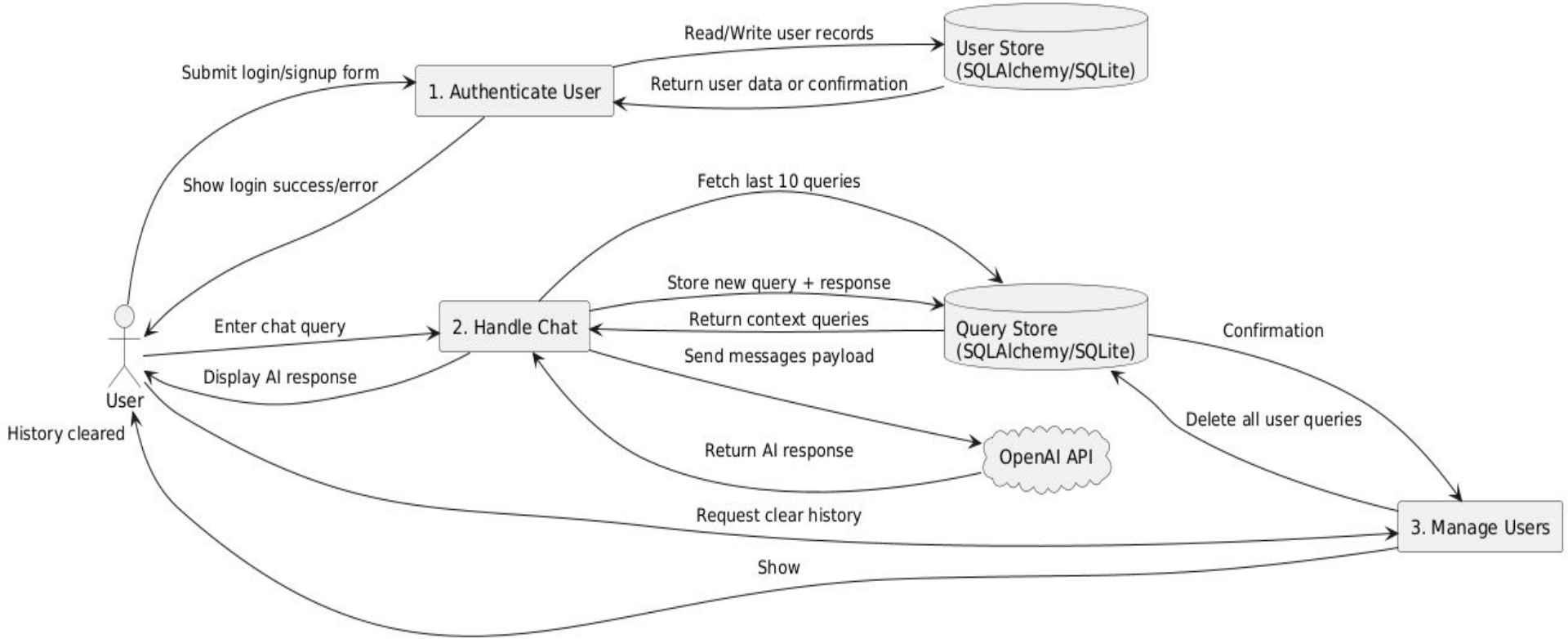
Security Measures

- **Password Handling:** Passwords hashed using Bcrypt before storage.
- **Session Management:** Flask-Login ensures secure user sessions.
- **CSRF Protection:** Implemented via Flask-WTF forms.
- **API Key Management:** OpenAI API key stored securely using environment variables (.env file).

AI Integration Details

- **Model Used:** OpenAI's GPT-3.5-turbo.
- **Contextual Responses:** System retrieves the last 10 user interactions to provide context-aware responses.
- **API Interaction:** Utilizes OpenAI's ChatCompletion endpoint with parameters like temperature and max_tokens for response control.

Data Flow Diagram(DFD)



Challenges & Solutions

Challenge: Ensuring secure password storage.

Solution: Implemented Bcrypt hashing for all passwords.

Challenge: Managing user sessions securely.
Solution: Utilized Flask-Login for session management.

Challenge: Handling API key security.
Solution: Stored API keys in a .env file and accessed them using python-dotenv

Thank you!