**Project Title: User Authentication and Interest Management System**

---

**1. Introduction**

This document outlines the requirements for the development of a User Authentication and Interest Management System. The system allows users to register, log in, send interest requests to other users, and respond to received interest requests. Additionally, it includes basic chat functionality between users.

**2. Objectives**

- Implement a user authentication system with registration, login, and logout features.

- Enable users to send and receive interest requests.

- Provide a dashboard where users can view and manage their interest requests.

- Facilitate basic chat functionality between users.

- Ensure proper handling of user interactions with error messages and notifications.

**3. Requirements**

*#3.1. Functional Requirements*

*##3.1.1. User Authentication*

- User Registration:

- Users must be able to register by providing a username and password.

- The system should validate that the two passwords match.

- The system should check if the username already exists and notify the user if it does.

- Upon successful registration, the user should be redirected to the login page with a success message.

- User Login:

- Users must be able to log in using their registered username and password.

- The system should handle cases where the username does not exist or the password is incorrect, displaying appropriate error messages.

- Upon successful login, users should be redirected to their dashboard.

- User Logout:

- Users must be able to log out from the system, which should invalidate their session and redirect them to the login page.

*##3.1.2. Interest Management*

- Send Interest Request:

- Authenticated users should be able to send interest requests to other users.

- The system should prevent sending duplicate interest requests between the same users.

- Upon successful interest request submission, the user should receive a confirmation message.

- Respond to Interest Request:

- Users who receive interest requests should be able to respond with a status of either "Accepted" or "Rejected".

- The system should update the interest request status accordingly and notify the user of the change.

*##3.1.3. Dashboard*

- View Users:

- The dashboard should display a list of all users except the logged-in user.

- View Interest Requests:

- The dashboard should display a list of interest requests the user has received.

- The dashboard should display a list of interest requests the user has sent.

*##3.1.4. Chat Functionality*

- Send and Receive Messages:

- Authenticated users should be able to send messages to other users with whom they have established an interest connection.

- The chat interface should display the username of the message sender.

- View Chat History:

- Users should be able to view past messages in a chat conversation when reloading the chat interface.

*#3.2. Non-Functional Requirements*

*##3.2.1. Performance*

- The system should handle up to 100 concurrent users without significant degradation in performance.

*##3.2.2. Security*

- Passwords should be securely hashed before being stored in the database.

- Sensitive data should be transmitted over HTTPS.

*##3.2.3. Usability*

- The system should provide clear and descriptive error messages to guide users in case of mistakes.

- The user interface should be intuitive and easy to navigate.

*##3.2.4. Scalability*

- The system should be designed to scale horizontally, accommodating more users as needed.

*##3.2.5. Maintainability*

- The code should be modular and adhere to standard coding conventions to facilitate future maintenance and updates.

- Test cases should be included to cover key functionalities.

*#3.3. Technical Requirements*

*##3.3.1. Technology Stack*

- Backend: Django, Django REST Framework

- Frontend: HTML, CSS, JavaScript, Materialise Framework

- Database: PostgreSQL or SQLite (for development purposes)

- Authentication: Django's built-in authentication system

- Web Server: Daphne for production; Django's built-in server for development

*##3.3.2. Environment Setup*

- Python 3.8 or later

- Django 3.2 or later

- PostgreSQL 12 or later (optional)

- Virtual environment for Python package management

*##3.3.3. Dependencies*

- Django

- Django REST Framework

- Django Channels (for chat functionality)

- Other libraries as required by the implementation

**4. Implementation**

*#4.1. Models*

- `User`: Provided by Django's built-in User model.

- `InterestRecords`: Stores interest requests, including sender, receiver, status, and timestamps.

- `ChatMessages`: Stores chat messages between users, including sender, receiver, content, and timestamps.

*#4.2. Views*

- User Views: Registration, login, dashboard, logout, and user creation.

- Interest Views: Sending and responding to interest requests.

- Chat Views: Sending and receiving messages.

*#4.3. Serializers*

- `InterestRecordsSerializer`: Serializes interest request data.

- `ChatMessageSerializer`: Serializes chat message data.

**5. Testing**

- Unit Tests: Test individual functions and methods.

- Integration Tests: Test the interaction between different parts of the system.

- Functional Tests: Test the system's behavior against the defined requirements.

**6. Deployment**

- Deployment will be handled using a suitable platform like AWS,, or a dedicated VPS.

- Ensure environment variables are correctly configured for sensitive data like database credentials and secret keys.

- Set up continuous integration/continuous deployment (CI/CD) pipelines to automate testing and deployment.

**7. Maintenance**

- Regularly update dependencies to their latest versions.

- Monitor logs and performance metrics to identify and resolve issues promptly.

- Provide documentation for developers to understand and contribute to the codebase.

---

This document provides a high-level overview of the project requirements and serves as a guideline for development, testing, and deployment.

***User Interface Notes:-***

1. User should create a accout in the Sign Up page from the login
2. User shold give unique usernames when creating a account
3. Password validators have been disabled for development purpose, so user can select any password
4. Once account is created user will be redirected to the login page and use should login
5. Once logged in, user will be able to see all users to whom they can send interest requests to.
6. User can also see the request sent by them and received.
7. User should **accept the request to chat** with a person
8. Once accepted, user will see **chat icon near the request record**
9. User will be able to send message by clicking send or **by clicking 'Enter'**
10. User will also be able to send chats in **real time**

**Run the Server:-**

**```**

daphne -p 8000 InterestsProject.asgi:application

```