system_require.md

System Requirements Document

1. Introduction

The purpose of this document is to outline the functional and non-functional requirements for the Realtime Pool Voting application. This application consists of a ReactJS frontend, a Python FastAPI backend, and utilizes Firebase as the database. The document will cover the features, performance expectations, scalability, security, and any other specific requirements of the application.

2. Functional Requirements

2.1 User Management

- The application should allow users to register and create an account.
- Users should be able to log in and authenticate themselves.
- User roles should be implemented to differentiate between administrators and regular users.

2.2 Pool Creation and Management

- Users should be able to create new pools for voting.

- Pool creators should have the ability to define the options for voting.
- The application should support real-time updates to the pool status and results.

2.3 Voting Process

- Users should be able to view and participate in active pools.
- Each user should have the ability to cast a single vote for their preferred option.
- The application should prevent users from voting multiple times in the same pool.

2.4 Results and Analytics

- The application should display real-time results as votes are cast.
- Pool creators and administrators should have access to detailed analytics and statistics about the voting process.

3. Non-Functional Requirements

3.1 Performance

- The application should be responsive and provide a smooth user experience.
- Real-time updates should be fast and efficient, ensuring minimal delay between vote submission and result display.

3.2 Scalability

- The system should be able to handle a large number of concurrent users and pools without significant performance degradation.
- The database and backend should be scalable to accommodate increasing data and user load.

3.3 Security

- User authentication and authorization should be implemented securely.
- The application should protect against common security vulnerabilities, such as cross-site scripting (XSS) and SQL injection attacks.
- Data transmission between the frontend, backend, and database should be encrypted using secure protocols.

3.4 Integration with Firebase

- The backend should integrate seamlessly with Firebase as the database.
- Firebase security rules should be implemented to restrict access to sensitive data and operations.

4. Additional Requirements

- The application should have a user-friendly and intuitive interface.
- Error handling and validation should be implemented to provide meaningful feedback to users.
- The codebase should follow best practices and be well-documented for future maintenance and enhancements.

5. Constraints

- The frontend should be developed using ReactJS.
- The backend should be developed using Python FastAPI.
- The database should be implemented using Firebase.

6. Assumptions

- Users will have access to modern web browsers that support the required technologies.
- Firebase will be properly configured and accessible for database operations.

7. Dependencies

- ReactJS
- Python FastAPI
- Firebase

8. Risks and Mitigation

- Risk: High user load may impact performance.
  - Mitigation: Perform load testing and optimize the application for scalability.
- Risk: Security vulnerabilities may expose user data.
  - Mitigation: Implement secure authentication and authorization mechanisms, and regularly update and patch dependencies.

9. Conclusion

This System Requirements Document outlines the functional and non-functional requirements for the Realtime Pool Voting application. It provides a comprehensive overview of the features, performance expectations, scalability, security, and other specific requirements of the application. Following these requirements will ensure the successful development and deployment of the application.