**Software Requirements Specification (SRS) for FinExpert**

**1. Introduction**

**1.1 Purpose**

FinExpert is a web application designed to help users track their expenses, analyze financial patterns, and receive AI-driven financial advice. The app provides detailed analytics using pie and bar charts, offers budget planning features, and alerts users when they exceed spending limits. Additionally, users receive financial reports via email.

**1.2 Scope**

FinExpert will allow users to:

* Register and log in securely.
* Add expenses under predefined or custom categories.
* View financial analytics through pie and bar charts.
* Receive AI-driven financial advice based on spending patterns.
* Set budgets and receive alerts when limits are exceeded.
* Receive financial reports via email.
* Store and manage data securely in PostgreSQL.
* Automatically allocate budgets across categories based on previous spending patterns.

**1.3 Definitions, Acronyms, and Abbreviations**

* **AI** – Artificial Intelligence
* **LLM** – Large Language Model (for AI-generated financial advice)
* **JWT** – JSON Web Token (for authentication)
* **FCM** – Firebase Cloud Messaging (for notifications)
* **ML** – Machine Learning

**1.4 References**

* PostgreSQL Documentation: https://www.postgresql.org/docs/
* Firebase Authentication: https://firebase.google.com/docs/auth
* NodeMailer for Emails: https://nodemailer.com/

**2. Overall Description**

**2.1 Product Perspective**

FinExpert will be developed as a **web-based** application using React for the frontend and Node.js with Express for the backend. The system will use PostgreSQL as the primary database and integrate AI for financial advice generation.

**2.2 Product Features**

1. **User Authentication**: Register, login, password reset.
2. **Expense Entry**: Add, modify, and delete expenses.
3. **Categories Management**: Predefined & custom expense categories.
4. **Analytics Dashboard**: Pie charts for category spending, bar charts for weekly expenses.
5. **AI-Driven Advice**: ML-based spending insights, LLM-generated recommendations.
6. **Budget Planner**: Users can set budgets and receive alerts if exceeded.
7. **Smart Budget Allocation**: Users can enter a total budget, and the system will allocate amounts to categories based on previous spending patterns.
8. **Email Notifications**: Expense alerts and monthly financial reports with charts.
9. **Data Security**: Encrypted storage and JWT-based authentication.
10. **Offline Mode**: Local data storage for temporary offline access.

**2.3 User Characteristics**

* **Basic Users**: Regular users tracking daily expenses.
* **Financial Enthusiasts**: Users looking for in-depth financial insights.
* **Budget Planners**: Users setting financial goals.

**2.4 Constraints**

* Web-only application.
* AI features depend on cloud-based LLM APIs.
* Requires an internet connection for AI insights and email reports.

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.1.1 User Authentication**

* Users must be able to register with email and password.
* Users can log in using JWT authentication.
* Password reset functionality via email.

**3.1.2 Expense Management**

* Users can add, edit, or delete expenses.
* Each expense must include amount, category, and date.
* Expenses are stored in PostgreSQL.

**3.1.3 Financial Analytics**

* Pie chart for category-wise spending.
* Bar chart for weekly expense trends.

**3.1.4 AI Financial Advice**

* ML model analyzes spending patterns.
* LLM generates personalized saving tips.

**3.1.5 Budget Planning & Alerts**

* Users set budgets per category or overall.
* Alerts via email & push notifications if the budget is exceeded.

**3.1.6 Smart Budget Allocation**

* When a user enters a total budget amount, the system analyzes previous spending habits.
* The budget is automatically distributed across categories based on past spending percentages.
* Users can adjust allocations manually if needed.
* API will return the allocated amounts per category based on historical spending data.

**3.1.7 Financial Reports**

* Monthly reports with charts sent via email.
* Exportable PDF format.

**3.2 Non-Functional Requirements**

* **Scalability**: Should handle multiple concurrent users.
* **Performance**: API response time should be <500ms.
* **Security**: JWT authentication, encrypted user data.
* **Reliability**: 99.9% uptime.
* **Usability**: Intuitive UI/UX design.

**4. System Architecture**

* **Frontend**: React.js
* **Backend**: Node.js (Express)
* **Database**: PostgreSQL
* **AI Services**: Python ML model + LLM API
* **Authentication**: Firebase Auth / JWT
* **Email Service**: NodeMailer

**5. Appendices**

* Further improvements can include multi-currency support, investment tracking, and smart saving goals.