Software Requirements Specification (SRS) for PocketPulse Expense Tracker

Project By:-
Siddharth Pillai
Sparsh Jain

Mentored By:-
Dr. Shaurya Gupta

Index

Software Requirements Specification (SRS) for Pocket Pulse Expense Tracker

1. Introduction

1.1 Purpose

This SRS document describes the requirements for an Expense Tracker application that enables users to track, categorize, and analyze their expenses efficiently. The system aims to provide an intuitive interface, data visualization tools, and budget alerts to enhance financial awareness.

1.2 Scope

The application will:

- Provide an interactive dashboard for tracking expenses.
- Allow users to add, edit, and categorize expenses.
- Generate reports and graphical insights on spending habits.
- Enable users to set budget limits and receive alerts.
- Ensure data security through authentication and encryption.

1.3 Definitions, Acronyms, and Abbreviations

- **Expense Tracker**: A financial tool that helps users monitor spending.
- **UI/UX**: User Interface/User Experience.
- **JWT**: JSON Web Token (authentication mechanism).
- **React.js/React Native**: Frontend frameworks for web and mobile applications.
- **SRS**: Software Requirements Specification.

1.4 References

1. Personal Finance Management: Best Practices – Journal of Financial Planning (2021).
2. Smith, J. (2020): "Designing User-Centric Financial Apps" – Tech Journal.
3. Mint: Budget, Bills, & Finance – https://mint.intuit.com.
4. YNAB (You Need A Budget) – https://www.ynab.com.

1.5 Overview

This document outlines the functional and non-functional requirements, system design, and technical specifications of the Expense Tracker application.

2. Overall Description

2.1 Product Perspective

The Expense Tracker is a standalone web and mobile application that helps users manage their finances effectively. It provides a user-friendly interface for tracking and analyzing expenses.

2.2 Product Features

- Real-time expense tracking.
- Categorization of expenses for better insights.
- Budget setting and alert notifications.
- Graphical reports for financial analysis.
- Secure user authentication and data encryption.

2.3 User Classes and Characteristics

- **Individuals**: Users who want to manage personal finances efficiently.
- **Small Businesses**: Entrepreneurs looking for simple expense management.
- **Financial Enthusiasts**: Users interested in detailed financial insights.

2.4 Operating Environment

- **Platforms:** Web (React.js), Mobile (React Native).
- **Backend:** Node.js with Express.js.
- **Database:** PostgreSQL/MongoDB/Firebase Firestore.
- **Hosting:** AWS/Heroku/Firebase/Vercel.

2.5 Design and Implementation Constraints

- The application must support real-time data updates without performance issues.
- Secure authentication must be implemented using JWT or Firebase Authentication.

2.6 Assumptions and Dependencies

- Users will have access to the internet to sync data.
- The system assumes users will enter accurate financial data.

3. Functional Requirements

3.1 User Authentication

- The system shall allow users to register, log in, and reset passwords securely.

3.2 Expense Management

- Users shall be able to add, edit, and delete expense entries.
- Users shall categorize expenses for better analysis.

3.3 Budgeting and Alerts

- The system shall allow users to set budget limits.
- Users shall receive alerts when nearing or exceeding budgets.

3.4 Reporting and Visualization

- The system shall generate graphical reports of spending patterns.
- Users shall be able to view monthly and yearly expense summaries.

3.5 Data Security

- All user data shall be encrypted for security.
- Authentication shall be implemented via JWT or Firebase Authentication.

4. Non-Functional Requirements

4.1 Performance

- The system shall provide real-time response to user interactions.
- The application shall load within 3 seconds under normal conditions.

4.2 Usability

- The UI shall be designed for ease of navigation and minimal learning curve.

4.3 Compatibility

- The application shall be cross-platform (web and mobile).
- The system shall support multiple devices with different screen sizes.

4.4 Maintainability

- The system shall be built using modular architecture for future updates.

5. System Design

5.1 System Architecture

- **UI Layer**: Manages user interface and interactions.
- **Backend Layer**: Handles API requests and business logic.
- **Database Layer**: Stores user and expense data securely.

5.2 Data Flow

1. **User logs in** → Access dashboard.
2. User adds/edits an expense → Stored in database.
3. **User views reports** → Data retrieved and visualized.
4. **User sets budget alerts** → Notifications triggered based on spending.

6. Other Non-Functional Attributes

6.1 Reliability

- The system shall handle simultaneous users without downtime.

6.2 Security

- Data encryption and authentication shall be implemented.

6.3 Portability

- The system shall support migration to different hosting providers if needed.

7. Appendices

- **Setup Instructions:** Guide to setting up the development environment.
- **User Manual:** Instructions on using the Expense Tracker features.
- **API Documentation:** Detailed endpoints and request-response structures.