Software Requirements Specification (SRS) Document

Project Name: BudgetIQ
Version: 1.0
Date: October 2025
Author: Krishna Bhatt

1. Introduction

1.1 Purpose

The purpose of this SRS document is to define the functional and non-functional requirements of the **BudgetIQ** application — a smart personal finance management system that allows users to track expenses, set budgets, manage recurring transactions, and achieve financial goals through intelligent insights and automation.

1.2 Scope

BudgetIQ is a web-based application built using **React (Frontend)**, **Spring Boot (Backend)**, and **MySQL (Database)**.
It enables users to:

- Manage daily expenses and income.
- Create and monitor budgets.
- Set and track financial goals.
- Automate recurring transactions.
- Gain insights from analytics and smart alerts.
- Access secure, cloud-synced data across devices.

Future versions (v2.0) will introduce **AI-based recommendations**, **cloud synchronization**, and a **chat-based financial assistant**.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
AI	Artificial Intelligence
JWT	JSON Web Token
API	Application Programming Interface
SRS	Software Requirements Specification
2FA	Two-Factor Authentication
UI	User Interface

1.4 References

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
- Project Overview Document – *BudgetIQ (v1.0)*
- Spring Boot and React official documentation

2. Overall Description

2.1 Product Perspective

BudgetIQ is a standalone web application with client-server architecture. The frontend (React.js) interacts with the backend (Spring Boot REST APIs), which connects to a relational MySQL database.
Planned cloud integration will allow users to access synchronized data from any device securely.

2.2 Product Functions

Core functions of BudgetIQ include:

- Expense and income management
- Budget creation and tracking
- Goal setup and progress tracking
- Automatic recurring transaction management

- Smart alerts and notifications
- Analytics dashboard for spending visualization
- Secure authentication and authorization
- Two-Factor Authentication (2FA) for enhanced security
- *(Upcoming)* AI-powered insights and cloud sync

2.3 User Characteristics

Users are individuals who manage personal or small business finances. They are expected to have basic computer literacy and an understanding of budgeting and expenses.

2.4 Constraints

- Application should support **web browsers** (Chrome, Edge, Firefox).
- Internet connection is required for cloud sync and AI features.
- Data privacy and security must comply with **industry standards (e.g., HTTPS, encryption)**.

2.5 Assumptions and Dependencies

- Users will register using a valid email ID.
- Cloud and AI features depend on external APIs (e.g., OpenAI API, AWS).
- The system will be deployed in a secure server environment.

3. System Features

3.1 Expense Tracking

- Users can add, edit, and delete expense records.
- Each expense includes date, category, payment method, and amount.
- Expenses can be filtered and summarized monthly.

3.2 Budget Planning

- Users can define monthly or custom budgets.
- System displays total expenses versus budgeted amount.

- Alerts when spending exceeds thresholds.

3.3 Goal Tracking

- Users can create financial goals with a target amount and date.
- Progress is tracked as users allocate savings toward the goal.
- Dashboard shows percentage completion for each goal.

3.4 Automatic Recurring Transactions

- Users can schedule recurring income or expense transactions.
- The system auto-generates these entries based on the defined frequency.
- Users can edit or disable recurring schedules anytime.

3.5 Smart Alerts & Notifications

- Alerts for approaching budget limits, due recurring transactions, or goal completion milestones.
- Configurable alert frequency and delivery mode (email or dashboard).

3.6 Analytics Dashboard

- Visual representation of spending trends using charts and graphs.
- Category-wise and time-based analysis to improve budgeting decisions.

3.7 Authentication & Security

- JWT-based authentication for user sessions.
- Passwords stored using encryption.
- **Two-Factor Authentication (2FA)** for additional login security.

3.8 Future Features (v2.0)

- AI-based financial assistant for predictive analytics.
- Cloud sync for multi-device access.
- Mobile application (Android/iOS).

4. External Interface Requirements

4.1 User Interface

- Responsive web UI designed using React.js.
- Clean dashboard layout showing key metrics and insights.
- Intuitive forms for transaction and goal input.

4.2 Hardware Interfaces

- Compatible with desktop, laptop, and tablet devices.

4.3 Software Interfaces

- Frontend communicates with backend via RESTful APIs (JSON format).
- Database interface through Spring Data JPA.
- Future integration with third-party APIs (banking, OpenAI).

4.4 Communication Interfaces

- HTTPS protocol for secure communication between client and server.
- Email or push notifications for alerts.

5. Non-Functional Requirements

5.1 Performance

- The system should handle at least **500 concurrent users** efficiently.
- Response time for API calls should be under **2 seconds** under normal load.

5.2 Security

- All data transmissions use **HTTPS encryption**.
- Passwords stored with hashing (BCrypt).
- Role-based access control for user privileges.

5.3 Reliability

- The system must ensure **99% uptime** under stable network conditions.
- Automatic data backup (planned in v2.0).

5.4 Maintainability

- Modular code structure allows easy feature addition and debugging.

5.5 Usability

- Minimal learning curve for new users.
- Consistent, user-friendly interface design.

6. System Evolution

BudgetIQ is designed for continuous improvement.
Upcoming versions will include:

- Integration of AI for predictive financial insights.
- Cloud synchronization and mobile app development.
- Multi-language support for global reach.

7. Appendix

- Version 1.0 focuses on web-based personal finance management.
- Future releases will expand functionality and scalability based on user feedback.