BUSINESS REQUIREMENTS DOCUMENT

Project Name:

SMART BUDGETING TOOL

1. INTRODUCTION

1.1 Purpose

The purpose of this Business Requirements Document (BRD) is to define and describe the requirements for the Smart Budgeting Tool component of a personal finance management application. This tool will help users manage their budgets effectively by providing functionalities for budget creation, expense tracking, and financial visualization.

1.2 Scope

The Smart Budgeting Tool will include features for:

- Creating and managing category multiple budgets
- Tracking and categorizing expenses
- Receiving alerts related to budget limits
- Visualizing budget data through interactive charts and reports
- The frontend will be developed using Angular, while the backend will be built with Java Spring Boot microservices.

1.3 Definitions

- Budget: A plan that allocates financial resources over a specified period to achieve financial goals.
- Expense: A monetary outflow from the user’s finances used to track spending.
- Analytics: A visualization of the budget and expense to track finances interactively.

1.4 Stakeholders

- End Users: Individuals who want to manage their budgets more effectively.
- Development Team: Engineers and developers responsible for implementing and maintaining the application.

2. BUSINESS OBJECTIVES

2.1 Goals

- Enhance Budget Management: Enable users to create, adjust, and monitor multiple budgets.
- Improve Expense Tracking: Facilitate the logging and categorization of expenses.
- Provide Real-Time Insights: Offer users dynamic visualizations and notifications to help them stay within their budget limits.
- Increase Financial Control: Equip users with tools to make informed financial decisions based on their budget data.

2.2 Benefits

- Dynamic Budget Creation: Users can set up and manage multiple budgets, adjusting them as needed.
- Automated Expense Tracking: Reduces manual effort by automatically categorizing and summarizing expenses.
- Proactive Alerts: Keeps users informed when approaching or exceeding budget limits.
- Interactive Visualization: Provides graphical representations of budget data to help users understand their spending patterns and adjust as needed.

3. FUNCTIONAL REQUIREMENTS

3.1 Interactive Budget Planner

3.1.1 Budget Creation and Management

- Requirement: Users must be able to create and manage multiple budgets.
- Details:
    - Create Budget: Users can set up new budgets with customizable categories (e.g., groceries, rent) and budget limits.
    - Edit Budget: Users can modify existing budgets, including updating category limits and adding or removing categories.
    - Delete Budget: Users can remove budgets that are no longer needed.
    - View Budgets: Users can view a list of all their budgets and their current status.

3.1.2 Budget Category Management

- Requirement: Users must be able to add, edit, and delete budget categories.
- Details:

  - Add Category: Users can define new expense categories within a budget.
  - Edit Category: Users can update category names and limits.
  - Delete Category: Users can remove categories that are no longer relevant.
  - Set Limits: Users can specify limits for each category, including setting monthly or yearly limits.

3.1.3 Budget Visualization

- Requirement: The application will provide dynamic visualizations of budget data.
- Details:
  - Charts and Graphs: Display budget data using charts
  - Budget Status: Show current spending against budget limits, including overage and remaining budget.
  - Category Breakdown: Visualize spending by category to highlight areas of high or low expenditure.

3.2 Expense Tracker

3.2.1 Expense Logging

- Requirement: Users must be able to log and categorize expenses.
- Details:
  - Enter Expense: Users can input expense details, including amount, category, date, and description.
  - Categorize Expense: Users can assign expenses to specific categories within their budgets.
  - Edit Expense: Users can update or correct previously entered expense details.
  - Delete Expense: Users can remove expenses that were entered in error.

3.2.2 Expense Summary

- Requirement: Users must be able to view a summary of their expenses.
- Details:
  - Expense Report: Generate reports summarizing expenses by category and time period.
  - Budget Impact: Show how expenses impact the current budget status and remaining budget.

3.2.3 Expense Trends

- Requirement: Users should be able to view trends in their spending.
- Details:
  - Historical Data: Display trends and patterns in expenses over time.
  - Category Trends: Visualize how spending in different categories changes over time.

4. NON-FUNCTIONAL REQUIREMENTS

4.1 Performance

- The application must handle multiple concurrent users with minimal latency.
- Data processing and visualizations should be responsive and provide real-time updates.

4.2 Security

- The application must include strong authentication and authorization mechanisms to protect user data.
- User data must be encrypted during transmission and at rest.

4.3 Usability

- The user interface should be intuitive and user-friendly, with clear navigation and helpful tooltips.
- The application must be accessible on various devices, including desktops, tablets, and smartphones.

5. COMPONENTS AND SERVICES

Frontend (Angular):

- Components:
    - BudgetPlannerComponent: For creating and managing budgets.
    - AnalyticsComponent: For displaying visualizations of budget and expense data.
    - ExpenseTrackerComponent: For logging and viewing expenses.
- Services:
    - BudgetService: Handles HTTP requests for creating, retrieving, and updating budgets.
    - ExpenseService: Manages HTTP requests for logging and retrieving expenses.
    - User Service: The User Service is used to register and login user and store the user details in it.

Backend (Java Spring Boot):

- Services:
    - BudgetService: Contains business logic for managing budgets and interacts with the database.
    - ExpenseService: Handles business logic for managing expenses and integrates with the database.
    - User Service: The User Service functions as a gateway for the microservices, handling user authentication and registration. It also incorporates Spring Security to ensure secure login and access management.

6. INTERACTION

- Frontend and Backend Communication:
    - HTTP Requests: Angular frontend communicates with Spring Boot backend via HTTP requests for budget and expense operations.
    - Data Binding: Data models are exchanged between frontend and backend to keep the application synchronized.

- Real-Time Updates:
    - Angular Components: Reactively update the UI based on data changes from the backend.
    - Spring Boot Services: Process requests and manage data, ensuring consistency and reliability.