FUNCTIONAL REQUIREMENTS DOCUMENT

Project Name:

SMART BUDGETING TOOL

1. Introduction

This document outlines the functional and non-functional requirements for the Smart Budgeting Tool, which is a component of a personal finance management application designed to help users manage their budgets effectively.

2. Functional Requirements

2.1 Interactive Budget Planner

2.1.1 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.

2.1.2 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.

2.1.3 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.

2.2 Expense Tracker

2.2.1 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.

2.2.2 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.

2.2.3 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.

2.2.4 Budget Alerts

Requirement: Notify users when they approach or exceed their budget limits.

Details:

Threshold Alerts: Users receive alerts when spending reaches predefined thresholds (e.g., 50%,80% of budget).

-Overage Alerts Notify users when spending exceeds budget limits.

NON-FUNCTIONAL REQUIREMENTS DOCUMENT

1. Introduction

This document outlines the non-functional requirements for the Smart Budgeting Tool, ensuring that the application meets performance, security, usability, and compatibility standards.

2. Non-Functional Requirements

2.1 Performance

Requirement: The application must handle multiple concurrent users with minimal latency.

Details:

Concurrent Users: The system should be able to support a high number of simultaneous users without performance degradation.

Response Time: Data processing and visualizations should be responsive and provide real-time updates.

2.2 Security

Requirement: The application must include strong authentication and authorization mechanisms to protect user data.

Details:

User Authentication: Incorporate secure login processes, such as JWT-based authentication.

Data Encryption: User data must be encrypted during transmission (e.g., HTTPS) and at rest (e.g., encrypted databases).

Access Control: Ensure that users only have access to their own data, with no unauthorized access to others' budgets or expenses.

2.3 Usability

Requirement: The user interface should be intuitive and user-friendly, with clear navigation and helpful tooltips.

Details:

UI/UX Design: The application should be easy to navigate, with a focus on simplicity and ease of use.

Device Compatibility: The application must be accessible on various devices, including desktops, tablets, and smartphones.

Accessibility: Ensure the application adheres to accessibility standards to accommodate users with disabilities.

2.4 Compatibility

Requirement: The application must be compatible across various platforms and environments.

Details:

Cross-Browser Support: The frontend must function correctly across popular web browsers (e.g., Chrome, Firefox, Edge).

API Compatibility: Ensure that the backend services can interact seamlessly with the frontend components, regardless of the platform or device.