BudgetMasters Project Specification

1. Overview

Budget Masters is a fully functional web based financial utility designed to help users successfully track their money. These features include the creation of budget, organizing and recording expenditure in the various expense categories, the expenditure graph for easy display of expenditure patterns. Unlike most of the systems, users are only required to enter a username, which makes the system easier and simple to use. The innovative banner to enhance the user activity is in the form of a leaderboard that ranks users according to their financial savings efficiency.

2. Core Features

1. User Accounts: Login with a simple username (no passwords required).
2. Budget Creation: Users can create budgets, set limits, and categorize expenses.
3. Expense Tracking: Log expenses with details (amount, date, category, description) and filter expenses as needed.
4. Data Visualization: View spending patterns using D3.js visualizations including pie charts for category distribution and bar charts for spending trends.
5. Leaderboard: Users compete based on savings efficiency, with rankings updated in real-time.
6. In-App Help: Comprehensive in-app support through a searchable Help Center.

3. Frontend Specification

The front end of BudgetMasters is implemented using HTML, CSS, and JavaScript, ensuring responsiveness and an intuitive user experience on both desktop and mobile devices.

Key Pages:

- Login Page (index.html): Users log in or register using a simple username-based system.
- Dashboard (dashboard.html): Provides an overview of users' budgets, expenses, and total savings. Users can create new budgets and manage existing ones.

- Expenses Page (expenses.html): Allows users to log, view, edit, and delete expenses. Users can filter expenses by category, budget, and time range.
- Charts Page (charts.html): Displays visual analytics using D3.js, featuring:
    - Category spending pie chart with percentage labels
    - Interactive filters for budget and time range selection
    - Dynamic updates and responsive sizing
- Leaderboard (leaderboard.html): Displays a savings leaderboard, ranking users based on their budget efficiency. The top three users are highlighted.
- Help Page (help.html): Offers support through help topics, FAQs, and tutorials to assist users in navigating and using the app.

Frontend Technologies:

- HTML/CSS: Provides the structure and layout of the application.
- JavaScript: Implements dynamic behavior and page interactivity.
- D3.js: Used for rendering interactive data visualizations, including pie charts, bar charts, and dynamic updates.

4. Backend Specification

The backend is built using Node.js with Express for server-side logic and MongoDB (via Mongoose) for database operations.

Server (server.js):

- Handles incoming API requests.
- Manages authentication, session tracking, and API routing.

Database (db.js):

- Establishes and maintains a connection with the MongoDB database.

API Endpoints:

Method	Endpoint	Description	Input	Output
POST	/api/users/login	Login or create user	Username	User object
GET	/api/budgets/:id	Retrieve budgets for a user	User ID	Array of budgets
POST	/api/budgets	Create a new budget	Budget details	Budget object
POST	/api/expenses	Log a new expense	Expense details	Expense object
GET	/api/expenses/budget/:id	Retrieve expenses for a budget	Budget ID	Array of expenses
GET	/api/leaderboard	Get leaderboard rankings	Time Range	Leaderboard data

Database Models (Mongoose):

- User.js: Stores user information and links to user budgets.
    - Fields: _id, username, budgets (Array of Budget IDs), totalSavings, savingsPercentage
- Budget.js: Represents a user's budget.
    - Fields: _id, userId, name, categories (array of {name, limit, spent}), totalLimit, totalSpent
- Expense.js: Represents an expense tied to a specific budget.
    - Fields: _id, budgetId, amount, category, description, date

Controller Files:

- users.js: Handles user-related actions like registration and login.
- budgets.js: Manages budget-related actions (create, retrieve, update, delete budgets).
- expenses.js: Handles expense-related actions (add, edit, view, delete expenses).

5. Reflections

What Went Well:

- Smooth Implementation of Leaderboard: The logic for calculating savings efficiency and ranking users dynamically worked as intended.
- Interactive Visualizations: The use of D3.js provided powerful and customizable data visualizations that enhanced the user experience.
- Dynamic Data Updates: The charts successfully update in real-time as users filter their data.
- Responsive Design: The use of CSS media queries and responsive design principles ensured that the app functioned well on both desktop and mobile devices.
- Intuitive Navigation: The sidebar navigation allowed for quick and easy access to key pages.

What Could Be Improved:

- User Feedback for Errors: While users receive basic feedback, more informative error messages would improve user experience.
- Chart Performance: For users with large amounts of data, chart rendering and updates could be optimized.
- Database Efficiency: For large datasets, indexing key fields and optimizing queries will be essential.

6. Technologies Used

- Frontend:
    - HTML/CSS for structure and styling
    - JavaScript for client-side logic
    - D3.js for data visualizations
- Backend:
    - Node.js and Express for server
    - Mongoose and MongoDB for database
- Development Tools:
    - dotenv for environment variable management

7. Directory Structure

Copy

```
project-root/
  ├── public/
  │   ├── index.html
  │   ├── dashboard.html
  │   ├── expenses.html
  │   ├── charts.html
  │   ├── leaderboard.html
  │   ├── help.html
  │   ├── css/style.css
  │   ├── js/main.js
  │   ├── js/dashboard.js
  │   ├── js/expenses.js
  │   ├── js/charts.js
  │   ├── js/leaderboard.js
  │   ├── js/help.js
  ├── server/
  │   ├── server.js
  │   ├── controllers/
  │   │     ├── users.js
  │   │     ├── budgets.js
  │   │     ├── expenses.js
  │   ├── models/
  │   │     ├── User.js
  │   │     ├── Budget.js
  │   │     ├── Expense.js
  │   ├── db.js
```