

Business Requirements Document (BRD)

Project Title: Budget Tracking Application with AI

Submitted by: JOTHSHANA S M (7376221CS181)

Contact: jothshana123@gmail.com, jothshana.cs22@bitsathy.ac.in

Date: 22th June 2025

GitHub Profile: [GitHub profile URL](#)

Hosted Links:

Component	Hosting Platform	URL / Notes
Frontend	Netlify	Budget Tracker Frontend
Backend	Render	Budget Tracker Backend
Database	MongoDB Atlas	mongodb+srv://<username>:<password>@cluster.mongo db.net

Deliverables:

GitHub source code	Project repository URL
Frontend README	README (Frontend)
Backend README	README (Backend)
Frontend .env	Sample .env (Frontend)
Backend .env	Sample .env (Backend)
Postman	APIs
Demo Working Video	Demo
UI/UX Wireframe	Figma UI
Flow Diagram	Flow chart
AI Model Used	NLP, gemini

1. Executive Summary

This document outlines the requirements and expectations for a **Budget Tracking Application enhanced with AI capabilities**, designed to help users efficiently manage their personal finances. The system aims to simplify the process of tracking income, expenses, budgeting, and decision-making by providing real-time analytics, intelligent suggestions, and visual reporting.

2. Project Objectives

- Enable users to manage and monitor their **monthly finances**.
 - Track **income and expenses** with proper categorization.
 - Allow users to **set budgets** and get notified of overspending.
 - Visualize financial data through **dashboards and charts**.
 - Utilize **AI** to deliver personalized recommendations based on user behavior.
 - Provide the ability to **export reports** in PDF and CSV formats.
 - Ensure the application is **mobile responsive** and user-friendly.
-

3. Target Users

- Individuals who want to track their personal spending
 - College students and working professionals managing monthly budgets
 - Non-finance background users seeking a simple UI/UX experience
 - Freelancers and gig workers who have variable income and need better financial control
 - Early-stage entrepreneurs and small business owners managing basic operational expenses
-

4. Core Functional Requirements

Feature	Description
User Authentication	Secure Sign Up / Login using JWT

Transaction Management	Add/Edit/Delete income and expense entries
Category Tagging	Classify transactions (e.g., Food, Rent, Travel)
Budget Setup	Set and track monthly spending limits
Dashboard	Visual overview using pie/bar charts
Reporting	Generate and export monthly financial summaries
Responsive UI	Mobile-first responsive design
Data Persistence	Store user data securely using MongoDB

5. AI Feature Requirements

- **Note Analysis (NLP):** Extracts keywords from transaction notes to understand spending behavior.
- **Spending Profile:** Identifies top spending categories and labels users (e.g., "Foodie", "Health Spender").
- **Income vs. Expense Ratio:** Highlights financial balance.
- **Gemini AI Suggestions:** Provides personalized tips based on user behavior.
- **Smart Recommendations:** Suggests similar users with better habits for comparison and learning.

6. Non-Functional Requirements

- **Performance:** Fast load times and smooth UI transitions
 - **Security:** Encrypted JWT-based authentication and secure data handling
 - **Scalability:** Designed to support multi-user environments
 - **Portability:** Easily deployable to any cloud (AWS, GCP, Azure)
 - **Maintainability:** Modular and well-documented code structure
-

7. Tech Stack Overview

Layer	Technologies
Frontend	React.js (with Vite), Tailwind CSS, Redux
Backend	Node.js, Express.js
Database	MongoDB
Auth	JWT (JSON Web Token)
Charts	Recharts / Chart.js
AI	NLP (via natural, compromise and OpenAI API)
Deployment	Netlify/ Render/ Atlas MongoDB
Testing	Jest (Frontend), Postman (API testing)

8. High-Level Architecture

Frontend (React)

↔ **Backend (Express API)**

↔ **MongoDB Database**

- Each user has their own transactions and budgets stored securely.
 - The AI module runs on the backend, categorizing entries or suggesting budget changes.
 - Charts are generated on the frontend from API responses.
-

9. Modules Breakdown














Module	Components
Auth Module	Login, Register, Forgot Password
Dashboard	Summary Cards, Income vs Expenses Chart, Alerts
Transactions	Add/Edit/Delete, List by month/category
Budget Manager	Set monthly budget, Remaining budget tracker
Reports	Export buttons, Date filters, Category-wise split
AI Recommendation	Personalized tips, Budget suggestions
Admin/Settings (Optional)	Dark mode toggle, Profile update, Logout

10. Development Timeline (3-Day Plan)

Day	Tasks
Day 1	Wireframes, BRD, UI setup, React + Express boilerplate, DB schema, Auth module
Day 2	Core features (CRUD), Budget module, Charts, AI features, Export functions
Day 3	Testing, Final fixes, Docs, Architecture diagrams, Deployment, Demo recording









11. Folder Structure

Frontend(including files)

- └─  client
 - └─  public
 - └─ _redirects
 - └─ image.png
 - └─ vite.svg
 - └─  src
 - └─ App.css
 - └─ App.jsx
 - └─  assets
 - └─ react.svg
 - └─  components
 - └─  dashboard
 - └─ CategoryCharts.jsx
 - └─ MonthlySummary.jsx
 - └─ SummaryCards.jsx
 - └─ TransactionFormModal.jsx
 - └─ TransactionTable.jsx
 - └─  layout
 - └─ AppLayout.jsx
 - └─ SideBar.jsx
 - └─ TopBar.jsx
 - └─ Loader.jsx
 - └─ ProtectedRoute.jsx
 - └─  reports
 - └─ ExportButtons.jsx
 - └─  Transactions
 - └─ TransactionForm.jsx
 - └─ TransactionsTable.jsx
 - └─  utils
 - └─ api.jsx
 - └─  context
 - └─ AuthContext.jsx
 - └─ BudgetAlertContext.jsx
 - └─ index.css
 - └─ main.jsx
 - └─  pages
 - └─ AIRecommandations.jsx
 - └─ BudgetFix.jsx
 - └─  Dashboard
 - └─ Dashboard.jsx
 - └─ Login.jsx
 - └─ MoneyAssistant.jsx
 - └─ Register.jsx

- └─ Reports.jsx
- └─ Transactions.jsx
- └─ .env
- └─ .env.example
- └─ .gitignore
- └─ eslint.config.js
- └─ index.html
- └─ package-lock.json
- └─ package.json
- └─ postcss.config.js
- └─ README.md
- └─ tailwind.config.js
- └─ vite.config.js

Backend(including files)

- └─  server
 - └─  ai
 - └─ analyzeNote.js
 - └─ recommend.js
 - └─  config
 - └─ db.js
 - └─  controllers
 - └─ ai_Controller.js
 - └─ aiController.js
 - └─ alertController.js
 - └─ authController.js
 - └─ budgetCategoryController.js
 - └─ transactionController.js
 - └─  middleware
 - └─ authMiddleware.js
 - └─  models
 - └─ BudgetCategory.js
 - └─ Transaction.js
 - └─ User.js
 - └─  routes
 - └─ aiRoutes.js
 - └─ alertRoutes.js
 - └─ authRoutes.js
 - └─ budgetCategoryRoutes.js
 - └─ dashboard.js
 - └─ transactionRoutes.js
 - └─  utils
 - └─ geminiProfile.js
 - └─ geminiSuggest.js
- └─ .env
- └─ .env.example

- └─ .gitignore
- └─ package-lock.json
- └─ package.json
- └─ README.md
- └─ server.js

12. APIs overview

Endpoint	Method	Description
/api/auth/register	POST	Register a new user
/api/auth/login	POST	Login user
/api/transactions	GET	Get all transactions
/api/transactions	POST	Create a new transaction
/api/transactions/by-month	GET	Get transactions of a specific month
/api/transactions/yearly-summary	GET	Get income/expense for the year
/api/transactions/income	GET	Get income-only transactions
/api/transactions/expense	GET	Get expense-only transactions
/api/budget-category	POST	Create category (income/expense)
/api/budget-category/:type	GET	Get categories by type
/api/ai/chat	POST	Chat with assistant

13. API Payloads

Register

POST <http://localhost:5000/api/auth/register>

Body:

```
{  
  "name": "username",
```



```
"email": "username@example.com",  
"password": "password"  
}
```

Response:

```
{  
  "_id": "6856928180bdfd107cabe671",  
  "name": "username",  
  "email": "username@example.com",  
  "token": "xxxxxtoken"  
}
```

Login

POST <http://localhost:5000/api/auth/login>

Body:

```
{  
  "email": "username@example.com",  
  "password": "password"  
}
```

Response:

```
{  
  "_id": "6856928180bdfd107cabe671",  
  "name": "username",  
  "email": "username@example.com",  
  "token": "xxxxxtoken"  
}
```

Transactions

Get All Transactions

GET <http://localhost:5000/api/transactions>

Response:

```
[  
  {  
    "_id": "68553b0356ba405c5e9b8e2b",  
    "user": "68551c33dd148e785cf9fce3",  
    "type": "income",  
    "category": "dddssaa",  
    "amount": 23,  
    "note": "dummy",  
    "createdAt": "2025-06-19T00:00:00.000Z",  
    "updatedAt": "2025-06-19T00:00:00.000Z",  
    "__v": 0  
  }  
]
```

```
},  
...  
]
```

Add a Transaction

POST <http://localhost:5000/api/transactions>

Body:

```
{  
  "type": "expense",  
  "category": "Groceries",  
  "amount": 1200,  
  "note": "Weekly supermarket shopping",  
  "date": "2025-06-20T00:00:00.000Z",  
  "createdAt": "2025-06-20T00:00:00.000Z"  
}
```

Response:

```
{  
  "user": "68551c33dd148e785cf9fce3",  
  "type": "expense",  
  "category": "Groceries",  
  "amount": 1200,  
  "note": "Weekly supermarket shopping",  
  "_id": "6856956280bfd107cabe692",  
  "createdAt": "2025-06-20T00:00:00.000Z",  
  "updatedAt": "2025-06-20T00:00:00.000Z",  
  "__v": 0  
}
```

Get Transactions by Month

GET <http://localhost:5000/api/transactions/by-month?month=2025-06>

Response:

```
[  
  {  
    "_id": "6856330ddb5e369c57f89f8e",  
    "user": "68551c33dd148e785cf9fce3",  
    "type": "expense",  
    "category": "cloths2",  
    "amount": 100,  
    "note": "velavan shopping",  
    "createdAt": "2025-06-21T00:00:00.000Z",  
    "updatedAt": "2025-06-21T00:00:00.000Z",
```

```
"__v": 0
},
...
]
```

Get Expense Transactions

GET <http://localhost:5000/api/transactions/expense>

Response:

```
[
  {
    "_id": "6856956280bfdf107cabe692",
    "user": "68551c33dd148e785cf9fce3",
    "type": "expense",
    "category": "Groceries",
    "amount": 1200,
    "note": "Weekly supermarket shopping",
    "createdAt": "2025-06-20T00:00:00.000Z",
    "updatedAt": "2025-06-20T00:00:00.000Z",
    "__v": 0
  },
  ...
]
```

Get Income Transactions

GET <http://localhost:5000/api/transactions/income>

Response:

```
[
  {
    "_id": "68552a1cfd05c237600995da",
    "user": "68551c33dd148e785cf9fce3",
    "type": "income",
    "category": "cloths",
    "amount": 30,
    "note": "dresses for winter",
    "createdAt": "2025-05-20T09:30:04.623Z",
    "updatedAt": "2025-05-20T09:30:04.623Z",
    "__v": 0
  },
  ...
]
```

Budget Category

Add Budget Category (Expense)

POST http://localhost:5000/api/budget-category

Body:

```
{  
  "category": "Cloths",  
  "percentage": 25,  
  "type": "expense"  
}
```

Response:

```
{  
  "user": "68551c33dd148e785cf9fce3",  
  "category": "Cloths",  
  "percentage": 25,  
  "type": "expense",  
  "_id": "6856966080bfdf107cabe699",  
  "createdAt": "2025-06-21T11:24:16.248Z",  
  "updatedAt": "2025-06-21T11:24:16.248Z",  
  "__v": 0  
}
```

Add Budget Category (Income)

POST http://localhost:5000/api/budget-category

Body:

```
{  
  "category": "incomeCategorycheck",  
  "type": "income"  
}
```

Response:

```
{  
  "user": "68551c33dd148e785cf9fce3",  
  "category": "incomeCategorycheck",  
  "percentage": null,  
  "type": "income",  
  "_id": "685696e480bfdf107cabe69c",  
  "createdAt": "2025-06-21T11:26:28.223Z",  
  "updatedAt": "2025-06-21T11:26:28.223Z",  
  "__v": 0  
}
```

14. Conclusion

This budget tracker is a robust, scalable, and user-friendly application aimed at helping individuals track and optimize their spending. The combination of visual insights, smart budgeting, and AI-powered suggestions ensures a complete personal finance management solution.
