# Smart Budget Manager - Complete Documentation

## Smart Budget Manager - Complete Documentation

Table of Contents

1. Overview

2. Technology Stack

3. Project Structure

4. Features Documentation

5. UI/UX Design

6. Code Architecture

7. Page-by-Page Breakdown

8. Component Documentation

9. Styling and Theming

10. Installation and Setup

Overview

Smart Budget Manager is a modern web application designed specifically for middle-class families and students to manage their finances effectively. The app provides intelligent budget tracking, expense monitoring, family collaboration features, and AI-powered financial insights.

What This App Does

- Track Expenses: Automatically categorize and monitor daily spending

- Budget Planning: Create and manage monthly/weekly budgets

- Family Collaboration: Share budgets and expenses with family members

- Smart Analytics: AI-powered insights and spending pattern analysis

- Goal Setting: Set and track financial goals

- Bill Reminders: Never miss important bill payments

- Multi-language Support: Available in multiple languages

- Dark/Light Theme: Switch between themes for comfortable viewing

## Technology Stack

Frontend Technologies

- React 18

- TypeScript

- Vite

- Tailwind CSS

- Shadcn/ui

- React Router

- Recharts

- Lucide React

State Management & Data

- React Query (@tanstack/react-query)

- React Context

- Local Storage

Styling & UI

- Tailwind CSS

- CSS Variables

- Animations

- Responsive Design

Project Structure

src/

├── components/

│ ├── ui/

│ ├── AppSidebar.tsx

│ ├── DashboardLayout.tsx

│ ├── AuthContext.tsx

│ ├── LanguageContext.tsx

│ ├── ThemeProvider.tsx

│ └── VoiceAssistant.tsx

├── pages/

│ ├── Index.tsx

│ ├── Login.tsx

│ ├── Dashboard.tsx

│ ├── BudgetPlanner.tsx

│ ├── Analytics.tsx

│ ├── ExpenseTracker.tsx

│ └── [other pages]

├── hooks/

└── lib/

## Features Documentation

1. Dashboard (Main Hub)

What it does: Central control center showing financial overview

Key Features:

- Quick Stats Cards

- Daily Budget Limiter

- Visual Charts

- Recent Transactions

- Add Transaction Form

How it works:

- React state for transaction data

- Recharts for visualizations

- Progress bars for budget

- Real-time financial metrics

2. Budget Planner

What it does: Budget creation and management

Key Features:

- 50-30-20 Rule

- Category Management

- Progress Tracking

- Family Budget Template

- Monthly/Weekly Views

How it works:

- Tabbed interface

- Real-time calculations

- Local storage persistence

3. Analytics & Insights

What it does: Advanced analysis with AI insights

Key Features:

- Trend Analysis

- Category Breakdown

- AI Insights

- Future Predictions

- Savings Optimization

How it works:

- Chart types: bar, line, pie

- Time-based filtering

- Pattern recognition

- Predictive modeling

4. Expense Tracker

What it does: Expense recording and categorization

Key Features:

- Quick Entry

- Auto Categorization

- Receipt Scanning

- Recurring Expenses

- Search & Filter

5. Multi-language Support

What it does: Multiple language accessibility

How it works:

- Language context provider

- Translation keys

- Dynamic switching

- Local storage preference

6. Theme Management

What it does: Light/Dark theme switching

How it works:

- Theme context provider

- CSS variables

- System preference detection

- Theme persistence

## UI/UX Design

Design Philosophy

- Clean & Minimal

- Accessibility First

- Mobile Responsive

- Intuitive Navigation

Color Scheme

Light Theme: whites, grays, blue/green/orange accents

Dark Theme: dark grays/blacks, softer colors

Typography

- Headers: Bold, large

- Body Text: System fonts

- Data Display: Monospace

- Interactive: Clear buttons

Layout Structure

- Sidebar Navigation

- Main Content Area

- Card-based Design

- Progressive Disclosure

## Code Architecture

Component Structure

Component-based with reusable UI blocks

Example Structure

Dashboard component:

- State for data

- Effects for loading

- Layout with StatsCards, Charts, Activity

State Management

1. Local State:

`const [transactions, setTransactions] = useState([])`

2. Context State:

`const { user, login, logout } = useAuth()`

`const { theme, toggleTheme } = useTheme()`

3. React Query:

`const { data, isLoading } = useQuery({ queryKey: ['transactions'], queryFn: fetchTransactions })`

Data Flow

1. User Interaction

2. State Update

3. Re-render

4. Persistence

5. Visual Feedback

## Page-by-Page Breakdown

1. Index.tsx

Purpose: Landing page

Structure:

- Header

- Hero Section

- Features Grid

- Benefits Section

- Statistics

- Footer

Code Highlights:

Responsive feature grid layout with cards and hover animation

2. Dashboard.tsx

Purpose: Financial overview

Structure:

- Quick Stats

- Budget Progress

- Charts

- Transaction Forms

- Recent Activity

Code Highlights:

Dynamic calculations and chart rendering with Recharts

3. BudgetPlanner.tsx

Purpose: Budget creation

Structure:

- Tabbed Views

- Budget Rule Visualizer

- Category Management

- Progress Trackers

Code Highlights:

Budget rule math:

needs = monthlyIncome \* 0.5

wants = monthlyIncome \* 0.3

savings = monthlyIncome \* 0.2

4. Analytics.tsx

Purpose: Financial insights

Structure:

- Metrics Dashboard

- Tabs for Trends, Categories, Insights, Predictions

- AI Insights

- Forecasts

Code Highlights:

Multiple chart types and AI-based recommendations

## Component Documentation

## Core Components

1. DashboardLayout.tsx

Purpose: Wrapper for dashboard pages

Features:

- Header

- Sidebar

- Main Content

- Voice Assistant

2. AppSidebar.tsx

Purpose: Navigation menu

Features:

- Collapsible

- Grouped Items

- Active State Highlight

- User Info

3. ThemeProvider.tsx

Purpose: Theme management

Features:

- User preference saving

- System theme detection

- CSS variable control

4. AuthContext.tsx

Purpose: Authentication

Features:

- Login/Logout

- User info

- Protected Routes

- Session Persistence

UI Components (Shadcn/ui used)

- Card

- Button

- Input

- Select

- Progress

- Tabs

- Alert

## Styling and Theming

Tailwind CSS Approach

Utility-first CSS with consistent reusable styles

Example:

.gradient-card

@apply bg-gradient-to-br from-white to-gray-50 shadow-lg rounded-xl