**BudgetBuddy Project (Frontend) Documentation**

**1. Introduction**

BudgetBuddy is a comprehensive mobile application designed to help users manage their personal finances effectively. The app is built using React Native and utilizes various libraries and tools to provide a seamless user experience.

**2. Project Structure**

The project follows a well-organized folder structure to ensure code maintainability and readability. Here's a detailed overview of the folders and files:

budget-buddy/

├── app/

│   ├── (auth)/

│   │   ├── forgot-password/

│   │   │   └── index.tsx

│   │   ├── login/

│   │   │   └── index.tsx

│   │   ├── otp/

│   │   │   └── index.tsx

│   │   └── register/

│   │       └── index.tsx

│   ├── (root)/

│   │   ├── (drawer)/

│   │   │   ├── (tabs)/

│   │   │   │   ├── budget/

│   │   │   │   │   └── index.tsx

│   │   │   │   ├── expenses/

│   │   │   │   │   └── index.tsx

│   │   │   │   ├── home/

│   │   │   │   │   └── index.tsx

│   │   │   │   └── profile/

│   │   │   │       └── index.tsx

│   │   │   ├── onboarding/

│   │   │   │   └── index.tsx

│   │   │   └── reports/

│   │   │       └── index.tsx

│   │   └── \_layout.tsx

│   ├── assets/

│   │   ├── adaptive-icon.png

│   │   ├── favicon.png

│   │   ├── icon.png

│   │   └── splash.png

│   └── index.tsx

├── assets/

│   └── images/

│       └── logo.png

├── components/

│   ├── global/

│   │   ├── CustomModal.tsx

│   │   └── Header.tsx

│   └── home/

│       └── ExpenseForm.tsx

├── services/

│   ├── authService.ts

│   ├── expenditureService.ts

│   └── goalsService.ts

├── stores/

│   ├── auth.ts

│   └── date.ts

├── utils/

│   └── fetcher.ts

├── .env

├── .gitignore

├── App.tsx

├── app.json

├── babel.config.js

├── package.json

├── README.md

└── tsconfig.json

**2.1 App Directory**

The app/ directory contains the main application code and follows the Expo Router convention. It is organized into sub-directories based on the app's navigation structure.

 (auth)/: Contains authentication-related screens such as login, registration, forgot password, and OTP verification.

 (root)/: Represents the root navigator of the app.

 (drawer)/: Implements a drawer navigation for the main app screens.

 (tabs)/: Implements a bottom tab navigation for the main app screens, including budget, expenses, home, and profile.

 onboarding/: Contains the onboarding screen for new users.

 reports/: Contains the reports screen for displaying financial reports.

 \_layout.tsx: Defines the layout for the root navigator.

 assets/: Contains app-specific assets such as icons and splash screens.

 index.tsx: The entry point of the app.

**2.2 Assets Directory**

The assets/ directory contains static assets used in the app, such as images and logos.

**2.3 Components Directory**

The components/ directory contains reusable components used throughout the app. It is organized into sub-directories based on the component's scope.

 global/: Contains globally used components such as CustomModal and Header.

 home/: Contains components specific to the home screen, such as ExpenseForm.

**2.4 Services Directory**

The services/ directory contains modules responsible for making API requests to the backend server.

 authService.ts: Handles authentication-related API requests.

 expenditureService.ts: Handles expense-related API requests.

 goalsService.ts: Handles savings goals-related API requests.

**2.5 Stores Directory**

The stores/ directory contains state management modules using the zustand library.

 auth.ts: Manages the user's authentication state.

 date.ts: Manages the selected date for expense tracking.

**2.6 Utils Directory**

The utils/ directory contains utility functions and helper modules.

 fetcher.ts: Defines a custom data fetching function used with the swr library.

**2.7 Configuration Files**

The project includes various configuration files at the root level:

 .env: Contains environment variables used in the app.

 .gitignore: Specifies files and directories to be ignored by Git.

 App.tsx: The main entry point of the application.

 app.json: Configuration file for Expo app settings.

 babel.config.js: Configuration file for Babel, the JavaScript compiler.

 package.json: Defines project dependencies and scripts.

 README.md: Provides an overview of the project and installation instructions.

 tsconfig.json: Configuration file for TypeScript compiler options.

**3. Key Libraries and Tools**

The BudgetBuddy app utilizes several key libraries and tools to enhance its functionality and development experience:

 React Native: A framework for building native mobile apps using React.

 Expo: A platform for building and deploying universal React apps.

 Expo Router: A file-based routing system for Expo apps.

 React Navigation: A library for handling navigation in React Native apps.

 React Hook Form: A library for managing form state and validation.

 Zod: A TypeScript-first schema validation library.

 Tamagui: A UI kit for building cross-platform apps with React Native.

 Nativewind: A library for using Tailwind CSS in React Native.

 SWR: A React Hooks library for data fetching and caching.

 Zustand: A small and fast state management library for React.

 React Native Chart Kit: A library for creating charts in React Native.

 React Native Toast Message: A library for displaying toast messages in React Native.

 Axios: A promise-based HTTP client for making API requests.

**4. App Screens and Components**

**4.1 Authentication Screens**

 Login Screen (app/(auth)/login/index.tsx): Allows users to log in to their accounts.

 Registration Screen (app/(auth)/register/index.tsx): Allows new users to create an account.

 Forgot Password Screen (app/(auth)/forgot-password/index.tsx): Allows users to reset their passwords.

 OTP Verification Screen (app/(auth)/otp/index.tsx): Allows users to verify their accounts using a one-time password.

4.2 Main App Screens

 Home Screen (app/(root)/(drawer)/(tabs)/home/index.tsx): Displays an overview of the user's financial information.

 Budget Screen (app/(root)/(drawer)/(tabs)/budget/index.tsx): Allows users to manage their budget and track expenses.

 Expenses Screen (app/(root)/(drawer)/(tabs)/expenses/index.tsx): Displays a list of the user's expenses.

 Profile Screen (app/(root)/(drawer)/(tabs)/profile/index.tsx): Allows users to view and edit their profile information.

 Onboarding Screen (app/(root)/(drawer)/onboarding/index.tsx): Provides an onboarding experience for new users.

 Reports Screen (app/(root)/(drawer)/reports/index.tsx): Displays financial reports and analytics.

**4.3 Reusable Components**

 CustomModal (components/global/CustomModal.tsx): A customizable modal component for displaying alerts and confirmations.

 Header (components/global/Header.tsx): A header component used across different screens.

 ExpenseForm (components/home/ExpenseForm.tsx): A form component for adding and editing expenses.

**5. API Services**

The app communicates with the backend server through API services defined in the services/ directory:

 Auth Service (services/authService.ts): Handles authentication-related API requests, such as login, registration, and password reset.

 Expenditure Service (services/expenditureService.ts): Handles expense-related API requests, such as fetching expenses, adding new expenses, and updating expenses.

 Goals Service (services/goalsService.ts): Handles savings goals-related API requests, such as fetching goals and adding new goals.

**6. State Management**

The app uses the zustand library for state management. The main store modules are defined in the stores/ directory:

 Auth Store (stores/auth.ts): Manages the user's authentication state, including login status and user information.

 Date Store (stores/date.ts): Manages the selected date for expense tracking and reporting.

**7. Styling and Theming**

The app uses a combination of nativewind and tamagui for styling and theming:

 Nativewind: Allows the use of Tailwind CSS classes in React Native components, providing a utility-first approach to styling.

 Tamagui: Provides a set of pre-styled components and theme management, ensuring a consistent look and feel throughout the app.

8. Data Fetching and Caching

The app uses the swr library for data fetching and caching. The utils/fetcher.ts file defines a custom data fetching function that is used with swr to retrieve data from the backend API and cache it for improved performance.

**9. Form Handling and Validation**

The app uses the react-hook-form library for form handling and the zod library for form validation:

 React Hook Form: Provides a simple and efficient way to manage form state and handle form submissions.

 Zod: Allows defining validation schemas for form inputs, ensuring data integrity and providing meaningful error messages.

**10. Charts and Visualization**

The app uses the react-native-chart-kit library to display charts and visualizations, such as pie charts and bar charts, for representing financial data in a visually appealing way.

**11. Notifications and Alerts**

The app uses the react-native-toast-message library to display notifications and alerts to the user, providing feedback on actions and events within the app.

**12. Environment Variables**

The app uses environment variables to store sensitive information and configuration settings. The .env file at the root level contains the necessary environment variables, which are loaded into the app using the react-native-dotenv library.

**13. Conclusion**

The BudgetBuddy app is a comprehensive personal finance management solution built with React Native and Expo. It leverages various libraries and tools to provide a robust and user-friendly experience for tracking expenses, managing budgets, and gaining insights into financial health.

The codebase follows a modular and organized structure, making it easy to navigate and maintain. The use of reusable components, state management with zustand, and data fetching with swr ensures a scalable and efficient application architecture.

With its intuitive user interface, powerful features, and seamless integration with the backend API, BudgetBuddy empowers users to take control of their finances and make informed financial decisions.