Personal Finance Tracker – Project Documentation

Overview

Project Name: Personal Finance Tracker
Company: ABCD Ltd
Platform: React Native (with Expo CLI) – Supports Android, iOS, and Web (via Expo for Web)
Backend Integration: Axios HTTP client with MockAPI
Purpose: Empower users to better manage personal finances by tracking expenses, visualizing spending habits, and eventually offering budgeting and notifications.

Key Features (Phase 1)

1. **User Login**
    - GET /users?username={email} to retrieve user
    - Client-side password validation
2. **Create Expense**
    - POST /expenses with fields: date, category, amount, description, userId
3. **View Expense Details**
    - GET /expenses/{expenseId}
4. **List All Expenses**
    - GET /expenses (filtered client-side or via userId param if supported)
5. **Delete Expense**
    - DELETE /expenses/{expenseId}

New Additions

A. Long Landing Page (Initial Screen Before Login)

Sections:

- Hero (Headline, CTA)
- App Features (Icons + Descriptions)
- User Benefits
- How It Works (Steps)
- Testimonials (Placeholder)
- Security Reassurance
- Final CTA to Login

B. Enhanced Dashboard (Post-Login Home)

Components:

- Welcome Message
- Summary Cards (Total Expenses, # of Transactions)
- Charts:
    - Pie/Donut for category distribution
    - Line/Bar for spending trends
- Recent Transactions (3–5)
- Quick Actions: Add Expense, View All Transactions

Application Architecture

Frontend: React Native with Expo CLI (supports Web and Android)
State Management: React Context API for global user session + local component state
HTTP Library: Axios
Backend API: MockAPI v1

Screen Flow Diagram

1. Landing Page →
2. Login Screen →
3. Dashboard →
4. Add Expense →
5. View Expense Details →
6. All Transactions

Component Structure Ideas

Landing Page:

- HeroSection
- FeaturesSection
- BenefitsSection
- HowItWorksSection
- TestimonialsSection
- SecuritySection

- FooterCTA

Login Screen:

- LoginForm
- Validation & Error UI

Dashboard:

- WelcomeBanner
- SummaryCards
- ExpenseCharts (CategoryChart + TimelineChart)
- RecentTransactions
- QuickActionsPanel

Expense Screens:

- AddExpenseForm
- ExpenseDetailCard
- ExpenseList

State Management Strategy

- **User Context:** Stores current user session (email, name, userId)
- **Local State:** For expense form inputs, UI toggles, and error messages
- **Charts:** Transform fetched expenses into datasets using helper functions

---

Data Handling for Charts

- **Pie/Donut Chart:** Group expenses by category, sum values
- **Line/Bar Chart:** Group expenses by day or week, aggregate totals
- All transformations done after fetching with GET /expenses

Implementation Plan

1. Set up Expo React Native environment (with Web support)
2. Create routing/navigation for screens
3. Build Landing Page (scrollable, multiple sections)
4. Implement Login functionality
5. Store user session context after login
6. Build Dashboard with:
    - Welcome Message
    - Summary Cards
    - Chart integrations
    - Recent Transactions
7. Build Add Expense screen
8. Build View Expense details screen
9. Build All Expenses list with filters
10. Implement Delete functionality
11. Add input validation and error handling across the app

UI/UX Considerations

- Clean and minimal design with clear CTA buttons
- Scrollable and informative Landing Page
- Visually engaging Dashboard with charts and summaries
- Responsive design for Android and Web
- Use standard UI feedback: toasts, loaders, and error messages

---

Error Handling & Validation

- Email format and required fields on login and forms
- Amount must be a positive number
- Handle API errors with fallback UI (e.g., "Unable to load expenses")
- Show loaders while waiting on API

API Endpoints Overview

Base URL: `https://67ac71475853dfff53dab929.mockapi.io/api/v1`

Users

- **GET /users?username={email}**
    - **Usage:** Login validation
    - **Client-side:** Match password in response

Expenses

- **POST /expenses**
    - Create new expense (fields: date, category, amount, description, userId)
- **GET /expenses/{id}**
    - Get single expense details
- **GET /expenses**
    - Fetch all expenses
    - Filter client-side if no userId filtering
- **DELETE /expenses/{id}**
    - Delete specific expense

Pages to Develop

1. **Landing Page** (scrollable intro page)
2. **Login Screen**
3. **Dashboard Screen**
4. **Add New Expense Screen**
5. **View Expense Details**
6. **All Expenses (Transaction List)**

README Summary (for your project)

Project: Personal Finance Tracker (React Native w/ Expo)

Overview:
A cross-platform finance tracking app supporting Android and Web using Expo. Allows users to log in, track expenses, view charts and summaries, and manage transaction history.

Tech Stack:

- React Native + Expo
- Axios (API requests)
- Context API (global state)
- MockAPI for backend simulation

Screens Included:

- Landing Page
- Login
- Dashboard (charts, stats, actions)
- Add Expense
- View Expense
- Expense List

API Endpoints: See [API Endpoints Overview] section.

How to Run:

1. Install Expo CLI
2. Clone repository
3. Run `npm install`
4. Launch with `npx expo start`
5. Test on Android/Web

Let me know if you’d like this exported to PDF, Markdown, or Google Docs.