Smart Financial Journal

Problem

People often view their spending as just numbers, without pausing to consider why they made each purchase. Even when they want to improve, they may not know what to fix. Whether it's emotional spending, impulse buys, or low-benefit purchases that don't align with their goals. Without reflection, it's difficult to recognize patterns, learn from mistakes, or develop healthier financial habits.

Solution

The Smart Financial Journal encourages users to reflect on their daily purchases, similar to how a personal journal helps people reflect on their life. Users log transactions with notes and categories, and Gemini AI impersonates the user, generating a first-person reflection that critiques spending choices while offering motivation. This self-reflective perspective highlights patterns, celebrates good decisions, and points out wasteful habits, transforming raw spending into actionable insights and a daily financial score that guides better choices over time.

Target Users

- Young adults and students looking to build good financial habits.
- Freelancers and gig workers with variable incomes who need help with budgeting and saving.
- Anyone looking to gain a clear understanding of their spending and find actionable ways to save more effectively

Tech Stacks

- **Runtime:** Node.js (development + Firebase Functions runtime).
- **Security:** Firebase Authentication (scopes user data by UID).
- **Frontend:** React, React DOM, React Router DOM (for navigation).
- **Build Tool:** Vite.
- **Backend:** Firebase Cloud Functions (server-side logic, Gemini API integration).
- **Styling:** Tailwind CSS.
- **API:** Google Gemini API (called securely via Cloud Functions).
- **Database:** Firebase Firestore (stores transactions, user settings, progress).

Core Features

1. **User Sign In**

Securely authenticates users via Firebase (email/password) to access the app. Supports sign-in and sign-up. User ID (UID) scopes data in Firebase Auth. Styled in Tailwind CSS.

1. Displays a form with email/password fields and a “Sign In”/“Sign Up” toggle.
2. Redirects to Setting (first use) or Journal (returning user).
3. Error handling: Shows alerts (e.g., “Invalid email”).

2. Setting

Collects user settings to personalize the app: Name, Salary (budget), Savings Goal (amount + date), Journal Time (for reminders). Styled in Tailwind CSS.

- Form inputs: Name (First Name), Salary (>0), Goal Amount (>0), Goal Date (future), Journal Time (HH:MM AM/PM).
- Validation: Alerts for invalid inputs.
- Settings are saved in Firestore under the signed-in user’s UID.
- Redirects to Journal Page on submit.

3. Daily Spending Journal

Allows users to log and reflect on daily purchases, helping them confront their spending habits. Displays a list of transactions (from Firestore) with editable fields for Category and Notes. Styled in Tailwind CSS.

AI Integration (via Secure Cloud Functions)

- Transactions and user settings are sent to a Firebase Cloud Function instead of directly calling Gemini from the frontend.
- The Cloud Function securely stores secret API keys and calls Gemini server-side to generate a tailored response.

Feedback highlights:

- Legitimate/essential spends.
- Wasteful or emotional purchases (e.g., stress shopping, boredom buying, celebration splurging).
- Empty or vague notes flagged as small but risky impulse buys.
- Links spending to broader impacts on health, well-being, and financial goals.

Output:

- A shortened categorized list showing the essential purchases and the wasteful purchases side by side for quick review.

- A first-person reflective analysis, written in the user's own voice, that is self-critical but motivational.
- A Daily Financial Score summarizing spending performance, returned by the Cloud Function.

Future Implementation

User Transactions: Integrate real bank transactions using a secure third-party service, like Plaid, to verify user identity and safely share financial data with the application.

Settings & Security: Ensure all user data, including future transactions and settings, is fetched and processed server-side via Firebase Cloud Functions to maximize security and control.

Daily Transactions & Reminders: Implement automatic daily transaction logging, rotating between days, with notifications to remind users to update their journal.

Customizable AI Analysis: Allow users to prioritize what matters most in transaction analysis , such as health, convenience, mood, or financial impact, and adjust the tone of AI feedback, from non-judgmental to more critical, based on their preferences.

Structured Insights: Utilize models, like sentiment models, to provide structured insights. This will help better identify mood or emotional spending patterns, giving the AI a clearer and more consistent understanding of the user's behavior.

Financial Reality Checks: Compare daily spending against budgets and overall financial goals, providing users with an end-of-day reality check on how their decisions impact their finances.

Progress Summaries: Provides weekly or monthly statistics of the user's overall progression based on the value scores from each log. Highlights patterns and behaviors critiqued in past spending, giving the user a clear view of their progress, shortcomings, and actionable steps to improve moving forward.