Design Documentation: Smart Financial Coach

Smart Financial Coach is an AI-powered personal finance application that transforms raw transaction data into actionable financial insights. The application combines behavioral analysis, machine learning, and intuitive visualizations to help users understand their spending patterns, set and track financial goals, manage budgets, and identify savings opportunities.

Core Design Principles

- User-Centric: Focus on actionable insights rather than raw data
- Behavioral Change: Encourage positive financial habits through personalized recommendations
- Transparency: Clear visualization of spending patterns and trends
- Accessibility: Simple CSV upload interface, no complex setup required
- Intelligence: AI-powered insights that learn from user data

Architecture & System Design: High-Level Architecture

CSV Upload → Data Processing → Behavioral Analysis → AI Model (FLAN-T5) → Visualization & UI

Component Architecture

1. Data Layer
    - CSV parsing and validation
    - Transaction data structure (date, amount, merchant, description, category)
    - Monthly aggregation and time-series processing
2. Analysis Layer
    - Behavioral pattern detection

    - Spending habit identification
    - Anomaly detection
    - Savings opportunity calculation
    - Budget tracking and comparison
3. AI/ML Layer
    - Fine-tuned FLAN-T5 model for insight generation
    - Rule-based fallback system
    - Natural language insight formatting
4. Presentation Layer
    - Gradio web interface
    - Interactive visualizations (Plotly)
    - Real-time data updates
    - Responsive UI components

Tech Stack: Core Technologies

Backend & Data Processing

- Python 3.x
- Pandas
- NumPy

Machine Learning & AI

- Transformers (Hugging Face)

- Model: google/flan-t5-small
- PyTorch
- Datasets (Hugging Face)
- Accelerate

Web Interface & Visualization

- Gradio 4.44.1
- Plotly
- Matplotlib

Data Format

- CSV input (date, amount, merchant, description, category)

Development Environment

- Virtual Environment for dependency management
- Requirements management via requirements.txt

Key Features & Implementation

1. Interactive Goal Setting
   Design: Users can set custom financial goals with name, target amount, and timeline.

2. Dynamic Category Budgeting
   Design: Budgets adapt to user's actual categories.
   Implementation: Category detection, real-time updates, and over/under budget indicators.

3. AI-Generated Insights
   Design: Personalized recommendations using fine-tuned FLAN-T5.

Implementation: Model inference, structured prompts, validation, fallback rules.

4. Behavioral Analysis Engine
   Design: Analyze habits, anomalies, subscriptions, and savings opportunities.
   Implementation: Z-scores for anomalies, frequency and pattern recognition, projection calculations.

5. Visual Data Presentation
   Design: Interactive charts and tables for comprehension.
   Implementation: Plotly for interactivity; responsive, color-coded charts.

User Interface Design

Design Elements

- Color Scheme: Purple gradient header, green/yellow/red for status
- Typography: Bold headers, readable sans-serif
- Component Design: KPI and insight cards, color-coded tables
- UX: Tabbed navigation, feedback through colors/icons, mobile-friendly

Data Flow

Input → Processing → Output Pipeline

1. Input: CSV file → DataFrame
2. Processing: Aggregation and analysis
3. Analysis: Pattern and habit detection
4. AI Processing: Summary JSON → AI Model → Text Insights
5. Visualization: Data → Charts and Tables
6. UI Update: Real-time Gradio updates

Future Enhancements

Phase 1 (Short-term): UX Improvements

- Bank Account Integration (Plaid/Yodlee)
- Mobile-Responsive Design
- Export Functionality (PDF/Excel)
- Transaction Search & Filter

Phase 2 (Medium-term): Advanced Analytics

- Spending Forecasting (Prophet, ARIMA, LSTM)
- Natural Language Query
- Personalized Recommendation Engine
- Multi-Currency Support

Phase 3 (Long-term): Advanced Features

- Receipt Upload & OCR
- Social Features and Peer Comparison
- Achievement and Gamification
- Third-Party API

Phase 4 (Enterprise & Scale)

- Multi-User Support
- Advanced Security (2FA, Encryption)

- Cloud Deployment (AWS, GCP, Azure)

Technical Decisions & Rationale

FLAN-T5-small: Efficient, local, adaptable, no API cost
Gradio: Fast prototyping, interactive, easy deployment
CSV Input: Simple, private, flexible
Dynamic Categories: Personalized, relevant, easy to interpret

Performance Considerations

Current: Lazy loading, caching, efficient aggregations, model caching
Future: Database integration, async processing, CDN assets, model quantization

Security & Privacy

Current: Local processing, no persistent data, CSV validation
Future: Data encryption, authentication, audit logging, GDPR compliance