Smart Financial Coach – Design Documentation

Smart Financial Coach – Design Documentation

1. App Design Overview

The **Smart Financial Coach** is a local-first financial assistant that combines **machine learning regression** with a **generative AI orchestrator**.

• **User Interaction:** Command-line interface (CLI).
• **Data Input:** User provides a `.csv` file of bank transactions (`Date, Description, Amount, Running Bal.`).
• **Orchestrator (LLM):** A local LLM (via Ollama) interprets user queries, classifies them into one of three categories:
• **Regression task:** Forecast future balance using ML.
• **General task:** Provide advice or explanations using the LLM.
• **Clarification task:** Ask the user for missing parameters (e.g., time horizon).
• **ML Forecasting Engine:**
• Uses **Polynomial Regression (degrees 1–3)** with **cross-validation** to forecast account balances.
• Handles both relative queries (“in 30 days”) and absolute queries (“on August 10th”).
• **Session Memory:** Keeps past conversation context to support follow-up questions.
• **Responsible AI:**
• Data stays local; no external services.
• Deterministic ML for numeric forecasts, AI only for interpretation and explanations.

---

2. Tech Stack

• **Programming Language:** Python 3.9+
• **Libraries:**
• `requests` → Communicate with Ollama API
• `numpy` → Numeric operations
• `scikit-learn` → Polynomial regression (LinearRegression + PolynomialFeatures)
• `matplotlib` (optional) → Visualizations (used in presentation/demo)
• **LLM Runtime:** [Ollama](https://ollama.ai/) for running local models (e.g., `llama3.1`)
• **Interface:** Command-line interface (CLI)
• **Data Format:** CSV (comma-separated values, with balances and transactions)

---

3. Design Components

a. CLI Layer

• Handles user input/output
• Manages session loop (exit/quit commands supported)

b. Orchestrator (LLM via Ollama)

• Classifies queries into regression/general/clarify
• Parses absolute dates (`August 10th`) and converts them into ISO format
• Extracts numeric parameters (e.g., horizon_days, target thresholds)

c. Regression Forecasting Engine

• Builds a **time series** of balances from CSV
• Fits **Polynomial Regression (degree auto-selected via CV)**
• Outputs forecast date, balance, model degree, and cross-validation RMSE

d. Session Memory

• Stores prior user and assistant messages
• Provides history to the orchestrator for contextual understanding

---

4. Example User Flows

1. **Forecasting Query (Regression)**
• User: *“How much money will I have in my account on August 10th?”*
• Orchestrator → classify as regression, extract target date.
• Forecasting engine → predict balance on 2025-08-10.
• Assistant: *“My ML regression forecast for 2025-08-10 is $X (degree=2, CV_RMSE=Y).”*

2. **Advice Query (General)**
• User: *“How can I cut down on my spending?”*
• Orchestrator → classify as general.
• Assistant: *“You could reduce dining out expenses and set a monthly budget limit.”*

3. **Clarification Query**
• User: *“Will I have enough money soon?”*
• Orchestrator → classify as clarify.
• Assistant: *“Could you specify how many days or weeks ahead you want me to forecast?”*

---

5. Future Enhancements

1. **UI Layer:** Add a simple web or mobile interface.
2. **Visual Forecasts:** Display balance projections as interactive charts.
3. **Advanced ML Models:** Incorporate ARIMA or LSTM for time-series forecasting.
4. **Subscription Detection:** Implement pattern recognition for recurring charges.
5. **Budgeting Features:** Allow goal-setting and alerts for overspending.
6. **External API Integration:** Connect to real-time bank feeds (with secure tokens).
7. **Personalization:** Provide custom saving strategies based on spending habits.

8. **Privacy Enhancements:** Explore federated learning to improve models across users without sharing raw data.