AI Financial Advisor — AI Rule Engine + LLM Challenge 2025

1. Summary

A financial health analysis system that evaluates personal financial data, applies rule-based scoring, and generates AI-powered insights to help users understand and improve their financial well-being.

2. Problem & Domain

Domain: Personal Finance Management

- **Why it matters**: Financial literacy and awareness are crucial for personal well-being, yet many individuals struggle with understanding their financial health.
- **Use Cases**:
  - Personal budget analysis
  - Financial habit tracking
  - Spending pattern recognition
  - Savings optimization

3. Architecture

```
graph LR
    A[User Input] --> B[Rule Engine]
    B --> C[Score + Tags]
    C --> D[LLM Insight Generation]
    D --> E[Final API Response]

    subgraph AI Financial Advisor
        B -->|Applies Rules| C
        C -->|Sends Analysis| D
    end

    E -->|Returns| F[User]
```

Components:

1. **Data Ingestion**: Accepts JSON input with financial data
2. **Rule Engine**: Processes data through financial rules
3. **Scoring System**: Generates scores and tags
4. **LLM Layer**: Produces natural language insights
5. **API Layer**: REST endpoint for interaction

Dependencies:

- Python 3.8+
- FastAPI
- OpenAI Python Client
- loguru (for logging)
- uvicorn (ASGI server)

4. Data & Schema

Input Format (JSON):

```
{
  "Name": "string",
  "currency": "string",
  "target_net_worth": number,
  "current_net_worth": number,
  "set_salary": number,
  "history": [
    {
      "month": "YYYY-MM",
      "salary": number,
      "spending": number,
      "saving": number,
      "metadata": {
        "primary_expense_category": "string",
        "investment_allocation": number,
        "note": "string"
      }
    }
  ]
}
```

5. Rule Engine

Financial Rules

Rule	Condition	Effect
Savings Rate	< 10% of income	-4 points
Savings Rate	10-20% of income	+1 point
Savings Rate	> 20% of income	+3 points
Spending Trend	Increasing > 30%	-3 points
Spending Trend	Decreasing > 10%	+2 points
Income Stability	Fluctuating income	-2 points
Burn Rate	> 80% of income	-3 points
Burn Rate	< 30% of income	+3 points
Investment Allocation	> 50% of savings	+2 points

Scoring System

- Final score is normalized between -10 and 10
- Negative scores indicate areas needing attention
- Positive scores indicate good financial health
- Tags are generated based on rule triggers

6. LLM Layer

Provider: Hugging Face (moonshotai/Kimi-K2-Thinking)

Prompt Template:

```
You are a financial advisor analyzing the following financial data:

{financial_data}

Analysis Results:
{analysis_results}

Provide clear, actionable insights and recommendations in 2-3 paragraphs. Focus on:
1. Key financial health indicators
2. Areas of concern
3. Practical recommendations
4. Long-term financial planning advice
```

Safety Measures:

- Input validation
- Output sanitization
- Error handling for API failures
- Rate limiting
- Token limits

7. API Usage

Endpoint: `POST /analyze`

Request Example:


```
{
  "Name": "Kwabena Boateng",
  "currency": "GHS",
  "target_net_worth": 1000000,
  "current_net_worth": 5000,
  "set_salary": 8500,
  "history": [
    {
      "month": "2024-01",
      "salary": 8500,
      "spending": 4200,
      "saving": 4300,
      "metadata": {
        "primary_expense_category": "Rent/Utilities",
        "investment_allocation": 0
      }
    }
  ]
}
```


Response Example:


```
{
  "status": "success",
  "analysis": {
    "tags": ["High Savings Rate", "Stable Spending", "Healthy Burn Rate"],
    "score": 8,
    "note": "Score is between -10 to 10"
  },
  "insight": "Your financial health is strong with a high savings rate..."
}
```

Error Response:

```
{
  "status": "error",
  "message": "Invalid input format",
  "details": "Field 'history' is required"
}
```

8. Quickstart

1. Clone the repository
2. Install dependencies:

   ```
   pip install -r requirements.txt
   ```

3. Set up environment variables:

   ```
   cp .env.example .env
   # Edit .env with your API keys
   ```

4. Run the application:

   ```
   uvicorn main:app --reload
   ```

5. Access the API at `http://localhost:8000`
6. Use the `/analyze` endpoint with your financial data

Documentation last updated: November 2025 For BUILD SciTech AI Rule Engine + LLM Challenge 2025