AI-Powered Financial Data Assistant Practical Assignment

Submission Deadline

Within 12 hours of receiving this assignment. Submit via Git repository (GitHub/GitLab) with clear commit history and a proper README.md file.

Objective

Build an AI-powered financial data assistant that helps users query and retrieve insights from financial transaction data using semantic similarity and vector embeddings.

The assistant should: - Generate synthetic financial transactions using AI - Store them in a vector database using embeddings - Allow users to ask natural language questions (e.g., “Show my top 5 expenses in September”), and return relevant transactions or summarized insights.

Problem Statement

Create an intelligent financial data assistant that can understand and answer user queries semantically.

Instead of using an existing dataset, generate a dummy dataset of financial transactions using an AI model or rule-based generator.

Each transaction must include the following fields: - id: Unique transaction ID - userId: User identifier - date: Transaction date - description: Transaction details (e.g., “UPI payment to Swiggy”) - amount: Transaction amount - type: “Credit” or “Debit” - category: Expense type (Food, Shopping, Bills, etc.) - balance: Balance after transaction

Assignment Tasks

1. AI-Based Dummy Data Generation

- Generate synthetic transaction data for at least 100–200 transactions per user (minimum 2–3 users).
- Use OpenAI API / Gemini / local LLM or Faker.js / Python Faker.
- Categories: Food, Shopping, Rent, Salary, Utilities, Entertainment, Travel, Others.
- Store the generated data in `/data/transactions.json`.

Example output:

```
{
  "id": "txn_101",
  "userId": "user_1",
  "date": "2024-08-10",
  "description": "UPI payment to Swiggy",
  "amount": 520,
  "type": "Debit",
  "category": "Food",
  "balance": 12480
}
```

2. Embedding Generation

- Use OpenAI (`text-embedding-3-small`) or HuggingFace (`sentence-transformers/all-MiniLM-L6-v2`) for embeddings.
- For each transaction, create a text representation: `"Debit of ₹520 on 2024-08-10 for Swiggy under Food category."`
- Generate embeddings and store them in a vector database (FAISS / Chroma / Pinecone).

3. Semantic Search API

- Build an API that accepts a natural language query.
- Convert query to embedding and perform cosine similarity search in the vector DB.
- Return top 5–10 relevant transactions.

Example Queries: - “Show all UPI transactions above ₹1000.” - “What’s my biggest expense in August?” - “How much did I spend on food last month?”

4. Summarization (Optional)

- After retrieving relevant transactions, summarize using an LLM (GPT-4, Llama-3).
- Example: “You spent ₹12,300 on food in August, mostly through Swiggy and Zomato.”

5. Project Structure

Suggested folder layout:

```
financial-data-assistant/
│
├── data/
│   └── transactions.json          # AI-generated dummy financial
data
├── embeddings/
│   └── vector_store.faiss         # Stored embeddings (if FAISS
used)
├── api/
│   ├── app.js                     # Main API entry point
│   └── routes/
│       └── search.js              # Query endpoint
├── services/
│   ├── dataGenerator.js           # AI-based or rule-based dummy
data generator
│   ├── embeddingService.js        # Generate embeddings
│   ├── vectorSearchService.js     # Vector DB operations
│   └── summarizerService.js       # (Optional) LLM summarizer
├── config/
│   └── dbConfig.js                # Vector DB / MongoDB config
├── README.md
└── package.json
```

Technical Requirements

- Language: Node.js (preferred) or Python
- Dummy Data Generator: Faker.js / OpenAI / Hugging Face
- Embedding Model: OpenAI `text-embedding-3-small` or `sentence-transformers`
- Vector DB: FAISS / Chroma / Pinecone
- Database: JSON file (no real DB required)
- LLM (optional): GPT-4 / Llama-3
- Version Control: Git repository
- Documentation: Proper `README.md`

Expected Deliverables

1. Working codebase implementing:
    - AI-based dummy data generation
    - Embedding generation & vector storage
    - Semantic search retrieval
    - (Optional) Summarization
2. Generated data file (`transactions.json`)
3. Postman collection or cURL commands for API testing
4. README.md containing:
    - Setup and installation
    - How to generate dummy data
    - API endpoints and examples
    - Model & DB configuration
    - Example queries and outputs
5. Submission: Git repository link (public or shared)

Example Interaction

User Query: > "What are my top 3 expenses last month?"

Assistant Response: > - Amazon Purchase – ₹2500 – 2024-08-15 > - Swiggy – ₹1300 – 2024-08-10 > - Big Basket – ₹1150 – 2024-08-18 > > 💬 *Summary:* You spent the most on shopping and food, totaling ₹4,950 in August.

Bonus Enhancements (Optional)

- Add date range filters using natural language parsing.
- Add visual charts of expenses by category.
- Enable multi-user support with isolated vector stores.
- Use a RAG-based pipeline to combine vector retrieval + LLM summarization.

Contact for Queries

- Rutal Jadav: +91 84695 56520
- Jay Karasariya: +91 81558 55663