# Rawa: AI-Powered Financial Assistant with RAG and Flutter Chatbot

A smart system that combines AI, data retrieval, and task automation to help users manage and query their financial data.

## Features

- Natural language understanding for financial queries.
- Real-time answers using a RAG system.
- User-specific knowledge base.
- Secure API with endpoints for data input, reporting, and notifications.
- Mobile chatbot interface using Flutter.
- Hosted API on Render.
- Multi-user architecture with isolated data per user.

## Project Components

### 1. Requirements Analysis

- Identified types of financial data (e.g., invoices, transactions).
- Defined common queries such as: 'What did I spend in April?'.
- Key features: notifications, expense categorization, downloadable reports.

### 2. Data Preparation

- Collected and cleaned structured financial data (CSV, Excel).
- Vectorized data using OpenAI Embeddings.

### 3. Knowledge Base Construction

- Built a vector database using FAISS.
- Implemented multi-tenant setup.
- Developed retrieval system to fetch relevant context.

### 4. RAG Model Integration

- Connected the language model to the vector store.
- Implemented Retrieval-Augmented Generation system.
- Improved answer quality and retrieval accuracy.

### 5. Backend API Development

- Built backend using FastAPI.
- Endpoints: upload data, query RAG, reports, notifications, user settings.
- Hosted API on Render and managed source code on GitHub.

### 6. Flutter Chatbot UI

- Developed mobile app with Flutter.
- Supported login, real-time notifications, report downloads, financial queries.

## Deliverables

- Full-stack AI-powered assistant for financial management.
- Production-ready hosted API.
- User-friendly mobile chatbot interface.
- Easily extendable for enterprise-level features.

## Tech Stack

- Python (FastAPI, OpenAI API)
- Flutter (Frontend)
- Render (API hosting)
- FAISS / Chroma (Vector database)
- GitHub (Code management)

## Sample Code - Vectorization and FAISS Indexing

```python
from langchain.vectorstores import FAISS
from langchain.embeddings import OpenAIEmbeddings
import pandas as pd

data = pd.read_csv("transactions.csv")
texts = [str(row) for row in data.values]
embedding = OpenAIEmbeddings()
vectorstore = FAISS.from_texts(texts, embedding)
```

## FastAPI Backend - Sample Endpoint

```python
from fastapi import FastAPI, UploadFile
from rag_pipeline import query_rag

app = FastAPI()

@app.post("/query")
async def query_endpoint(user_query: str):
    response = query_rag(user_query)
    return {"response": response}
```

## Flutter Chatbot - HTTP Request Example

```dart
import 'package:http/http.dart' as http;
import 'dart:convert';

Future<String> sendQuery(String query) async {
 final response = await http.post(
  Uri.parse('https://rawa-api.onrender.com/query'),
  headers: {'Content-Type': 'application/json'},
  body: jsonEncode({'user_query': query}),
 );

 if (response.statusCode == 200) {
  return jsonDecode(response.body)['response'];
 } else {
  throw Exception('Failed to get response');
 }
}
```

## Report Generation Snippet

```python
from fpdf import FPDF

def generate_report(data, filename="report.pdf"):
  pdf = FPDF()
  pdf.add_page()
  pdf.set_font("Arial", size=12)
  for line in data:
    pdf.cell(200, 10, txt=line, ln=True)
  pdf.output(filename)
```