Design Document - RAG Conflict Detection App

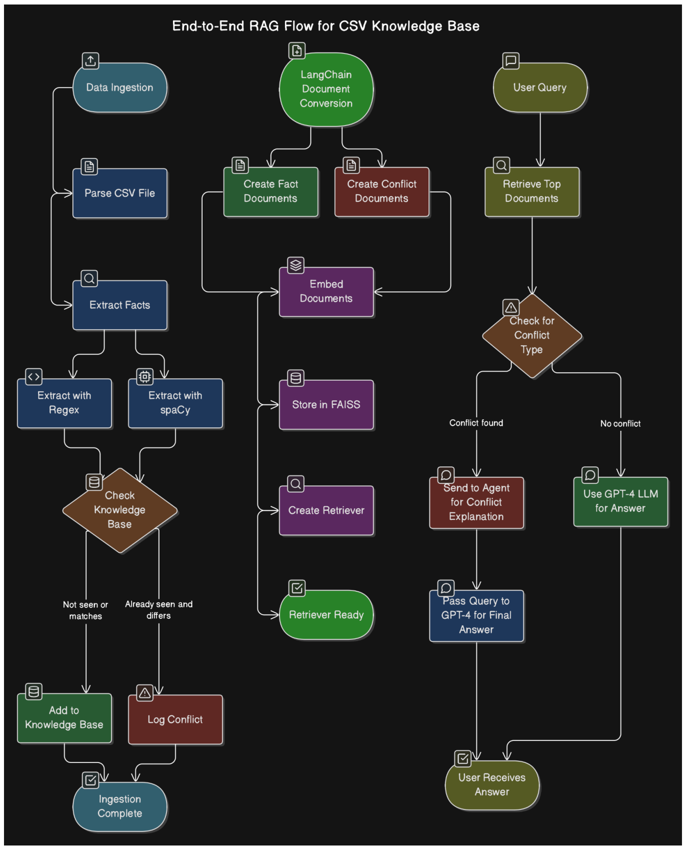
Project Overview

The RAG Conflict Detection App is a system that allows users to upload CSV documents, extract structured facts using regex and NLP, and build a knowledge base to handle user queries. The system automatically detects conflicts when newer documents contradict existing facts and uses a Retrieval-Augmented Generation (RAG) pipeline to provide informed, context-aware answers.

The application integrates LangChain with FAISS for semantic search and uses GPT-4 as the backend LLM, with additional GPT-4 agents to explain detected conflicts in human-readable terms.

This app is designed to be containerized (using Docker), stored in a cloud container registry (AWS ECR), and planned for future deployment on AWS ECS Fargate, with infrastructure management handled via Infrastructure as Code (Terraform).

Detailed System Architecture



Frontend

Built with Streamlit, providing a simple web interface for users to upload CSV files and enter natural language queries.

Fact Extraction Layer

Uses Python regular expressions (re) and the spaCy NLP pipeline to scan uploaded CSV files, extract structured relationships (like “A = B”, “X has 10”, etc.), and normalize these facts into a dictionary-based knowledge base.

Any contradictions (i.e., same key, different value) are logged into a dedicated conflict log.

Vector Database + Retrieval Layer

Converts ingested documents into LangChain Document objects.

Uses OpenAI embeddings and stores vector representations in FAISS for fast similarity search.

On user queries, retrieves semantically relevant chunks of knowledge or conflict descriptions.

LLM & Conflict Agent Layer

GPT-4 handles all generation tasks, including answering user questions and explaining why a conflict exists.

A LangChain agent (ConflictExplainer) wraps GPT-4 to generate human-readable explanations for conflicting facts.

Containerization

The entire app (including Python dependencies and environment setup) is packaged into a Docker image for consistent builds across development, staging, and future production.

Cloud Infrastructure

Docker images are pushed to AWS ECR (Elastic Container Registry).

ECS Fargate will be set to deploy containers, configured behind an Application Load Balancer (ALB) for public web access.

Infrastructure will be set to be fully managed with Terraform, defining ECS clusters, task definitions, security groups, networking, and load balancing.

Tools & Technologies

Layer	Tools

App Framework	Streamlit
NLP + Extraction	Python re, spaCy
LLM + Agent	LangChain, GPT-4 (OpenAI API)
Embeddings + Vector DB	OpenAI Embeddings, FAISS
Containerization	Docker
Cloud Platform	AWS (ECR, ECS Fargate, ALB, IAM, VPC)
IaC	Terraform
CI/CD (Planned)	GitHub Actions

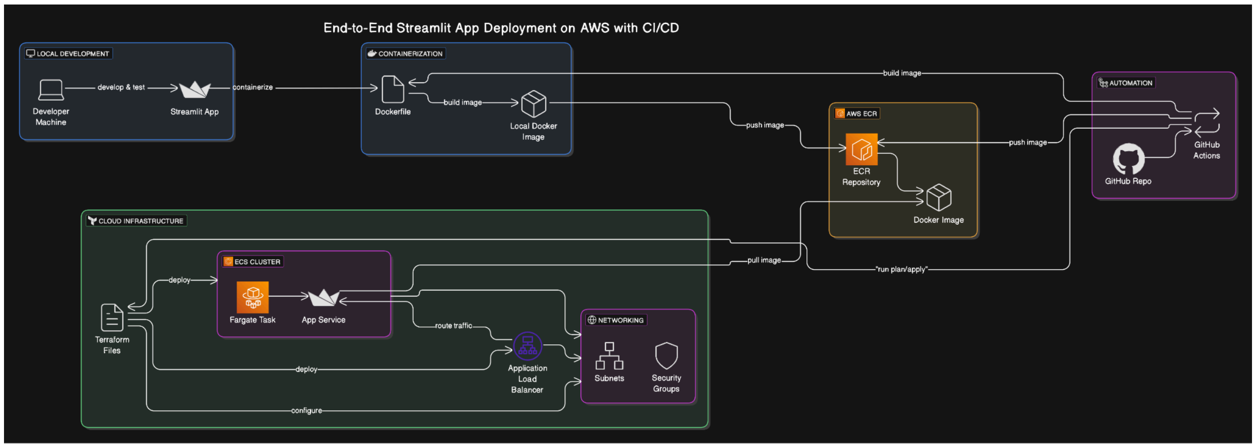
Assumptions

Input CSV files follow a consistent format with extractable fact statements.

OpenAI API keys are securely provided (via .env or AWS Secrets Manager).

The AWS account used has permissions to provision ECS, ECR, IAM roles, ALBs, and associated networking resources.

Current Deployment Plan



Build the Docker image locally using:

- docker build -t rag-conflict-app .

Push the image to AWS ECR:

- docker tag + docker push commands.

Use Terraform to deploy:

- Defines ECS cluster, Fargate services, task definitions, security groups, and ALB setup.
- Commands: terraform init → terraform plan → terraform apply.

Expose the app through the ALB endpoint for end-user access.

Roadmap and Future Improvements

Phase 1

Streamlit app with conflict detection and RAG answering.

Manual deploy via Docker + Terraform.

Phase 2

Add CloudWatch logging and monitoring for app health and performance.

Integrate GitHub Actions for automated CI/CD, including Docker build + ECR push + Terraform apply on new commits.

Use AWS Secrets Manager or Parameter Store to manage sensitive keys securely.

Implement autoscaling policies on ECS Fargate to handle variable loads.

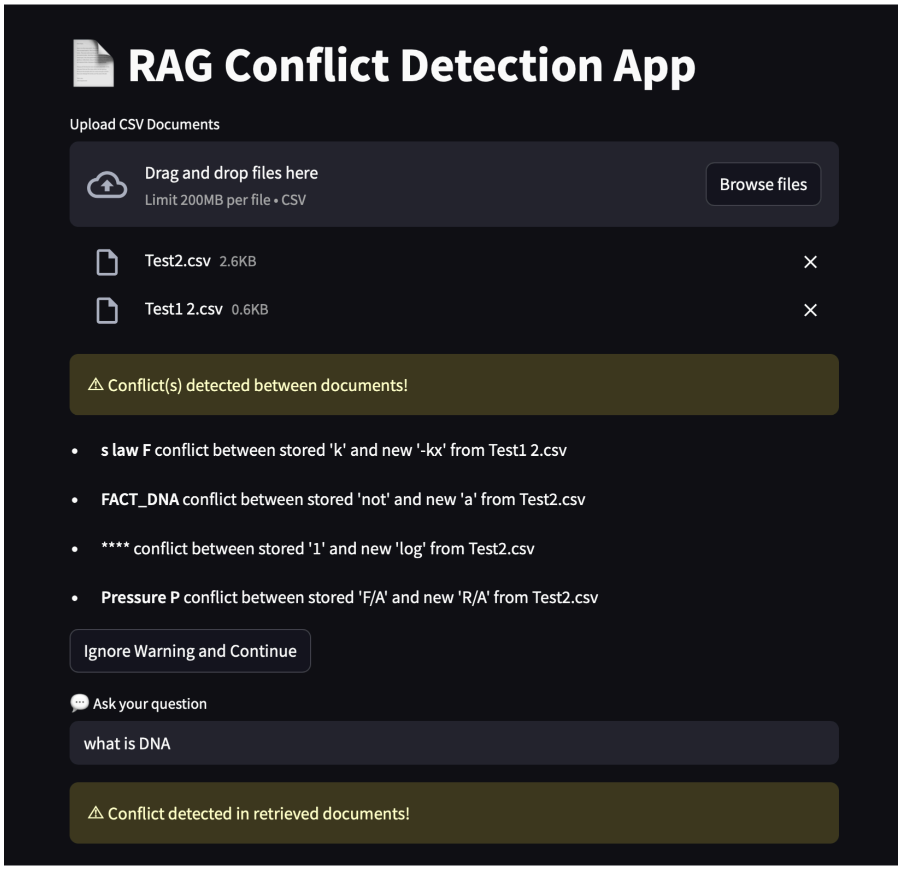
Add unit and integration tests for core extraction and RAG pipelines.

Expand the retrieval logic to combine text + metadata for more precise results.

Improve the Streamlit frontend design for better user experience, including multi-language support and mobile responsiveness.

Introduce user authentication for restricted access if needed.

Working App Demo



📄 RAG Conflict Detection App

Upload CSV Documents

Drag and drop files here
Limit 200MB per file • CSV

Browse files

Test2.csv 2.6KB

Test1 2.csv 0.6KB

⚠ Conflict(s) detected between documents!

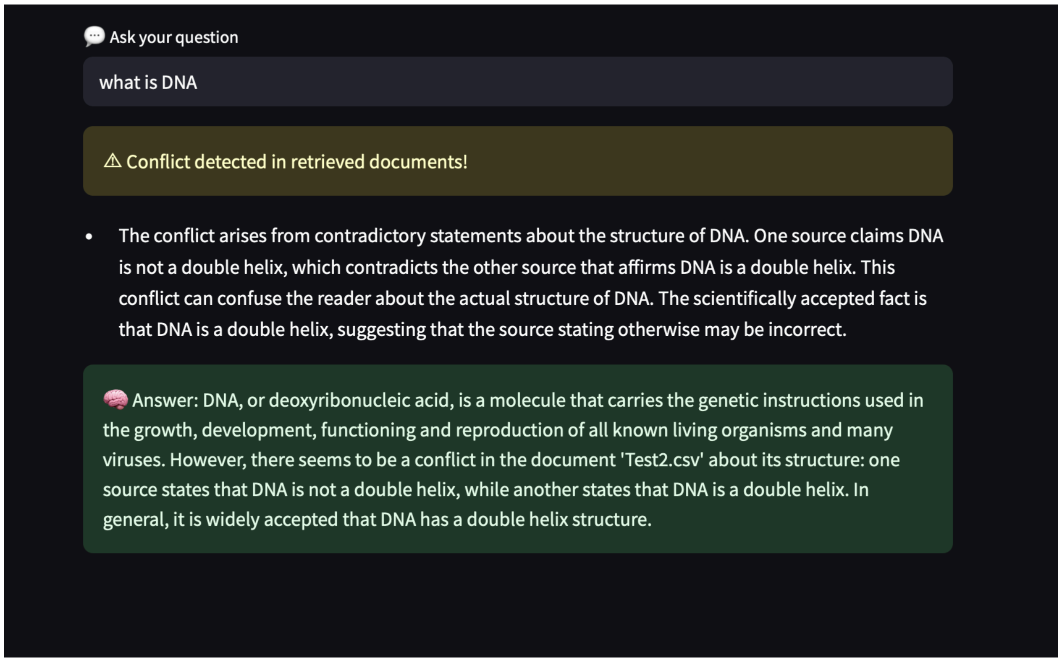
- **s law F** conflict between stored 'k' and new '-kx' from Test1 2.csv
- **FACT_DNA** conflict between stored 'not' and new 'a' from Test2.csv
- **** conflict between stored '1' and new 'log' from Test2.csv
- **Pressure P** conflict between stored 'F/A' and new 'R/A' from Test2.csv

Ignore Warning and Continue

💬 Ask your question

what is DNA

⚠ Conflict detected in retrieved documents!



💬 Ask your question

what is DNA

⚠ Conflict detected in retrieved documents!

- The conflict arises from contradictory statements about the structure of DNA. One source claims DNA is not a double helix, which contradicts the other source that affirms DNA is a double helix. This conflict can confuse the reader about the actual structure of DNA. The scientifically accepted fact is that DNA is a double helix, suggesting that the source stating otherwise may be incorrect.

🧠 Answer: DNA, or deoxyribonucleic acid, is a molecule that carries the genetic instructions used in the growth, development, functioning and reproduction of all known living organisms and many viruses. However, there seems to be a conflict in the document 'Test2.csv' about its structure: one source states that DNA is not a double helix, while another states that DNA is a double helix. In general, it is widely accepted that DNA has a double helix structure.