AutoAid Pro

Final Capstone Presentation

AI Car Troubleshooter

Safety-first diagnosis with Mini-RAG, agent actions, and full-stack demo UI.

Prepared by: Ahmad Obeid

AI Engineering Bootcamp - Final Hackathon

Date: 2026

Problem and Project Goals

Problem

- Drivers report vague symptoms and cannot judge risk quickly.
- Online advice may be unsafe, generic, or not vehicle-specific.
- Need clear next actions with priority: keep user and vehicle safe.

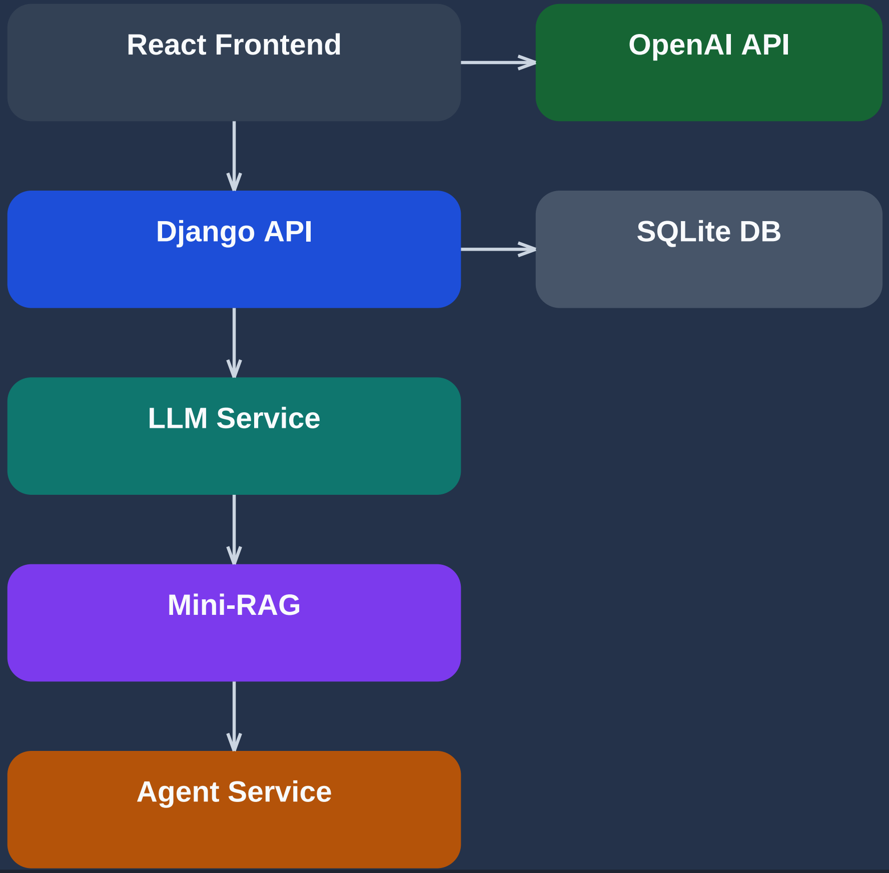
Goals

- Deliver structured triage: green, yellow, red.
- Use uploaded docs via Mini-RAG and show citations.
- Trigger agent actions: checklist, escalate, resolve.
- Provide graceful fallback if LLM is unavailable.

System Overview

Core Components

- **Frontend UI**
  - Manual vehicle and case creation forms
  - Document upload from local storage
  - Chat and follow-up interaction
- **Backend API**
  - Django REST endpoints for all workflows
  - Validation and persistence
- **AI Layer**
  - LLM diagnosis service + safety processing
  - Mini-RAG retrieval and citation injection
- **Agent Layer**
  - Auto or forced actions
  - Auditable action and note logs



UI Overview and User Journey

Single-page workflow for live demo

1) Create Vehicle

User fills manual fields: make, model, year, transmission, fuel, mileage.

2) Create Case

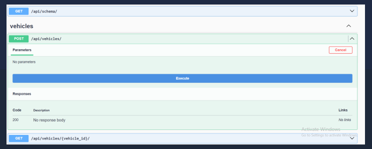
Attach vehicle_id, then provide initial problem title and first message.

3) Upload Document

Select local file (.pdf/.txt/.md/.doc/.docx) and metadata.

4) Chat + Agent

Receive diagnosis with citations; run agent actions and inspect logs.



```
/api/schema/
```

vehicles

```
/api/vehicles/
```

Parameters

No parameters

Responses

Code	Description	Links
200	No response body	No links

```
/api/vehicles/{vehicle_id}/
```

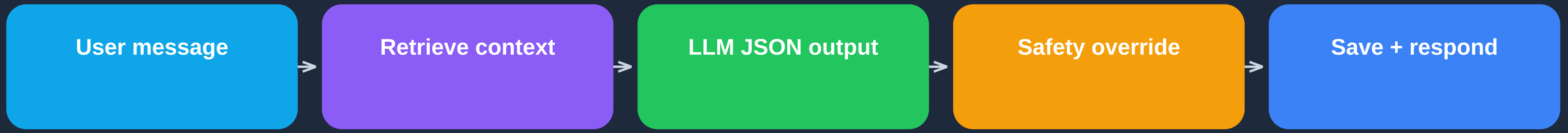
Functionalities and Features

What the system does today

- Vehicle profile management with typed fields and validation.
- Case lifecycle tracking with status and risk level updates.
- Symptom logging for both user and assistant turns.
- Versioned diagnosis records with confidence and latency.

- Mini-RAG ingestion and retrieval with citation objects.
- Safety overrides for red-flag phrases (cannot stop, smoke, fuel leak).
- Agent actions: auto, checklist, escalate, resolve.
- Operational logs: actions and notes endpoints for traceability.

AI Pipeline - Mini-RAG + Diagnosis + Safety



User message → Retrieve context → LLM JSON output → Safety override → Save + respond

- If documents exist, retrieved chunks become rag_context.
- If no docs are found, system answers normally using case and vehicle context.
- Structured output includes triage, causes, actions, and follow-up questions.
- Unsafe DIY instructions are sanitized before response is returned.

Backend API Surface

Primary Endpoints

Core

```
GET   /api/health/
POST  /api/vehicles/
POST  /api/cases/
POST  /api/chat/
POST  /api/cases/{id}/symptoms/
```

RAG + Agent

```
POST  /api/rag/documents/upload/
POST  /api/cases/{id}/agent/run/
GET   /api/cases/{id}/actions/
GET   /api/cases/{id}/notes/
GET   /api/docs/  (Swagger UI)
```

- Typed serializers enforce payload validity.
- OpenAPI schema provides interactive endpoint testing.
- All core flows are testable from Swagger and Postman.

How to Start the System

Local run steps (backend + frontend)

Backend (Django)

```
python -m venv .venv
.venv\Scripts\activate
pip install -r requirements.txt
python manage.py migrate
python manage.py runserver

# .env (local only)
OPENAI_API_KEY=your_key
OPENAI_MODEL=gpt-4o-mini
```

Frontend (Vite + React)

```
npm install
npm run dev

# frontend .env
VITE_API_BASE_URL=http://127.0.0.1:8000/api

Create Vehicle -> Create Case
Upload Doc -> Chat -> Agent Logs
```

Security note: never commit .env or virtual environment folders to GitHub.

Demo Script for Judges

- **Step 1: Create a vehicle**
  - Input make/model/year and confirm a new vehicle_id is returned.
- **Step 2: Create a case**
  - Attach vehicle_id and add initial problem statement.
- **Step 3: Upload a knowledge document**
  - Use a local troubleshooting file to enable grounded retrieval.
- **Step 4: Ask diagnostic question**
  - Show triage, causes, safe actions, and source citations.
- **Step 5: Continue with one follow-up**
  - Answer follow-up question and show improved recommendation.
- **Step 6: Run agent action**
  - Trigger escalate or checklist and display action and notes logs.

Performance, Safety, and Hardening

Performance

- Latency tracked per diagnosis (latency_ms).
- Token usage logged (tokens_input, tokens_output).
- Model metadata stored for auditing and benchmarking.
- Fallback mode prevents downtime during API issues.

Safety

- Red-flag keyword override forces high-risk triage.
- Unsafe instructions replaced with mechanic-safe actions.
- Escalation path and stop-driving reasons persisted.
- Short follow-up loop to reduce ambiguity before action.

Limitations and Roadmap

Next Iterations

- Improve retrieval ranking and chunk relevance scoring.
- Add semantic risk classifier beyond keyword matching.
- Introduce authentication and role-based access control.
- Deploy production stack: PostgreSQL + workers + object storage.

- Support deeper bilingual (Arabic/English) troubleshooting.
- Add analytics dashboard for case outcomes and resolution times.
- Integrate OBD sensor data for stronger signal quality.
- Expand test coverage with automated red-flag regression suite.

Thank You

AutoAid Pro

AI-powered, safety-first car troubleshooting assistant

Q and A

Live demo available: end-to-end flow with Mini-RAG and agent logs