Here’s your **Super Comprehensive README.md** file for the **Car Clinic Smart Repair Advisor** GitHub repo.

markdown

CopyEdit

# 🧠 Car Clinic Smart Repair Advisor 🚗🛠️

A fully autonomous, LLM-powered data pipeline that extracts real-world automotive problems from Reddit, structures them, augments and tags them, then matches them with the most appropriate Car Clinic repair branch based on semantic similarity and service expertise.

---

## 📌 Table of Contents

- [🚀 Project Overview](#-project-overview)

- [📈 System Architecture](#-system-architecture)

- [🛠️ Features](#️-features)

- [🧪 Pipeline Phases](#-pipeline-phases)

- [🧬 Data Flow Diagram](#-data-flow-diagram)

- [🗂 Directory Structure](#-directory-structure)

- [⚙️ Tech Stack](#️-tech-stack)

- [⚡ Getting Started](#-getting-started)

- [🧠 Prompt Engineering](#-prompt-engineering-principles)

- [🗓 Roadmap](#-roadmap)

- [🧾 License](#-license)

- [👨‍💻 Author](#-author)

---

## 🚀 Project Overview

Car Clinic Smart Repair Advisor is an intelligent system that reads thousands of Reddit threads from car repair subreddits, cleans and structures the data using LLMs, tags the content, and then recommends the best-fit repair branch using embeddings and semantic similarity.

It enables:

- Real-world problem mining from community data

- Problem-solution extraction using local LLMs

- Multilingual data augmentation

- Semantic & tag-based repair center recommendation

---

## 📈 System Architecture

```plaintext

Reddit

|

v

[Data Extractor] ──▶ [Data Cleaner (LLM)] ──▶ [Augmenter + Translator]

|

v

[Tag Generator (LLM/Rule-Based)]

|

v

[Embedding Generator (SBERT/Instructor)]

|

v

[Branch Matcher (Tags + Vectors + Filters)]

|

v

[Best Repair Branch Recommendation Output]

**🛠️ Features**

* ✅ Reddit daily data ingestion
* ✅ Prompt-based problem-solution parsing (LLM offline)
* ✅ Data augmentation (paraphrasing, noise injection)
* ✅ Multilingual translation (back-translation)
* ✅ Automatic tagging of issues & solutions
* ✅ Embedding generation for both user queries and branches
* ✅ Branch matching based on semantic + skill alignment
* ✅ GitHub Actions + Prefect integration
* ✅ CLI-ready, modular architecture

**🧪 Pipeline Phases**

**✅ Phase 1: Reddit Data Extraction**

* Uses PRAW and HTTPX to fetch Reddit data.
* Handles pagination, limits, retries.
* Saves to /data/raw/.

**🔍 Phase 2: LLM-Based Data Cleaning**

* Uses offline LLM (e.g., DeepSeek) for structured parsing.
* Removes bots, ensures quality comments.
* Outputs saved in /data/cleaned/.

**🎭 Phase 2.5: Data Augmentation**

* Paraphrasing using NLPAug or templates.
* Noise injection simulates user error.
* Boosts training and inference variety.

**🌍 Phase 2.6: Translation (Optional)**

* Applies back-translation for multilingual support.
* Uses offline translator or HuggingFace-compatible models.

**🏷️ Phase 3: Tag Generator**

* Applies LLM or rule-based taggers to cleaned text.
* Tags both problems and solutions.
* Saves tagged JSON to /data/tagged/.

**🔢 Phase 4: Embedding Generator**

* SBERT or Instructor-XL encodes cleaned posts and branch descriptions.
* Branch data manually curated.
* Saves embeddings in /data/embeddings/.

**🧭 Phase 5: Branch Recommender**

* Matches tags + embeddings.
* Filters by geography or availability.
* Produces explainable top-N branch matches.

**🧪 Phase 6: Local Testing**

* Test each module independently.
* End-to-end tests using small batches.
* Store test sets in /docs/.

**🔄 Phase 7: Prefect Orchestration**

* Modularize each phase into flows.
* Enable scheduling, retries, logs, CLI interface.

**☁️ Phase 8: GitHub Actions & Deployment**

* Schedule end-to-end daily runs.
* Future-proof with Docker containerization.

**📘 Phase 9: Documentation & Finalization**

* Maintain /docs/ for every module.
* Include architectural decisions and sample data.
* Keep this README up-to-date.

**🧬 Data Flow Diagram**

mermaid

CopyEdit

graph TD

A[Reddit Data Extractor] --> B[LLM Data Cleaner]

B --> C[Augmenter: Paraphrase + Noise]

C --> D[Translator: Back-Translation]

D --> E[Tag Generator]

E --> F[Embedding Generator]

F --> G[Branch Matcher]

G --> H[Recommended Repair Branch]

**🗂 Directory Structure**

css

CopyEdit

📁 Car\_Clinic\_Project/

│

├── data/

│ ├── raw/

│ ├── cleaned/

│ ├── tagged/

│ ├── embeddings/

│ └── samples/

│

├── reddit\_data\_extractor/

│ ├── flow.py

│ ├── extractor.py

│ └── utils.py

│

├── reddit\_data\_cleaner/

│ ├── flow.py

│ ├── preprocessor.py

│ ├── llm\_runner.py

│ └── postprocessor.py

│

├── data\_augmenter/

│ ├── flow.py

│ ├── augmenter.py

│

├── translator/

│ ├── flow.py

│ ├── translator.py

│

├── tag\_generator/

│ ├── flow.py

│ ├── tagger.py

│

├── embedding\_generator/

│ ├── flow.py

│ ├── embedder.py

│

├── branch\_recommender/

│ ├── flow.py

│ ├── matcher.py

│ ├── ranker.py

│

├── prefect\_flows/

│ └── car\_clinic\_pipeline.py

│

├── .github/

│ └── workflows/

│ └── main.yaml

│

├── docs/

│ ├── architecture.png

│ ├── prompt\_templates.md

│ └── data\_samples.json

│

├── README.md

└── requirements.txt

**⚙️ Tech Stack**

| **Category** | **Tool/Library** |
| --- | --- |
| **Data Source** | Reddit (via PRAW & HTTPX) |
| **LLM** | DeepSeek / GPTQ (Offline Inference) |
| **Embedding** | SBERT / Instructor-XL |
| **Tagging** | Prompt-based or rule-based |
| **Workflow** | GitHub Actions + Prefect |
| **Language** | Python 3.10+ |
| **Containers (future)** | Docker |
| **Storage** | Local filesystem (S3-ready) |

**⚡ Getting Started**

**1. Clone the Repository**

bash

CopyEdit

git clone https://github.com/Ibrahim-Hegazi/Car\_Clinic\_Project\_Revamped.git

cd Car\_Clinic\_Project\_Revamped

**2. Create Virtual Environment**

bash

CopyEdit

python -m venv .venv

source .venv/bin/activate # On Windows: .venv\Scripts\activate

pip install -r requirements.txt

**3. Run Locally**

bash

CopyEdit

python prefect\_flows/car\_clinic\_pipeline.py

**4. Run GitHub Actions**

* See .github/workflows/main.yaml for automated job triggers (e.g., daily ETL).

**🧠 Prompt Engineering Principles**

Prompt design matters. Each phase uses different prompt templates that are:

* ✅ Few-shot or instruction-based
* ✅ Language-sensitive (translatable)
* ✅ Format-aware (returning JSON)
* ✅ Hallucination-resistant

See /docs/prompt\_templates.md for examples.

**🗓 Roadmap**

* Reddit data extraction
* LLM-based data cleaning
* Tagging and schema design
* Semantic embeddings
* Branch recommender
* GitHub & Prefect orchestration
* API Interface (Flask/FastAPI)
* Frontend dashboard (optional)
* Language expansion (Arabic, Spanish)
* Docker-based deployment

**🧾 License**

This project is licensed under the **MIT License**. See LICENSE for more info.

**👨‍💻 Author**

**Ibrahim Hegazi**  
*Data Engineer, ML Enthusiast*  
[LinkedIn](https://linkedin.com/in/ibrahim-hegazi) • [GitHub](https://github.com/Ibrahim-Hegazi)

*If you found this helpful, consider giving the project a ⭐ on GitHub.*

yaml

CopyEdit

---

Let me know if you'd like to generate badges (e.g., Python version, build status, license) or have the README translated into Arabic for dual-language visibility.

Ask ChatGPT

**Updated Readme**

# 🧠 Car Clinic Smart Repair Advisor 🚗🛠️

A fully autonomous, LLM-powered data pipeline that extracts real-world automotive problems from Reddit, structures them, augments and tags them, and intelligently recommends the most appropriate Car Clinic repair branch using embeddings and service expertise.

---

## 📌 Table of Contents

- [🚀 Project Overview](#-project-overview)

- [📈 System Architecture](#-system-architecture)

- [🛠️ Features](#️-features)

- [🧪 Pipeline Phases](#-pipeline-phases)

- [🧬 Data Flow Diagram](#-data-flow-diagram)

- [🗂 Directory Structure](#-directory-structure)

- [⚙️ Tech Stack](#️-tech-stack)

- [⚡ Getting Started](#-getting-started)

- [🧠 Prompt Engineering](#-prompt-engineering-principles)

- [🗓 Roadmap](#-roadmap)

- [🧾 License](#-license)

- [👨‍💻 Author](#-author)

---

## 🚀 Project Overview

\*\*Car Clinic Smart Repair Advisor\*\* uses Reddit posts as a foundation to build a localized emergency assistant and mechanic helper. It ingests Reddit car problem threads, cleans them using local LLMs, augments and tags them, then maps the issue to the best repair branch nearby.

This enables:

- ⚙️ Real-time, explainable repair suggestions

- 🤖 Offline, privacy-friendly inference

- 🌍 Multilingual support and understanding

- 🧭 Nearest optimal repair branch recommendations

---

## 📈 System Architecture

```plaintext

Reddit

|

v

[Data Extractor] ──▶ [Data Cleaner (LLM)] ──▶ [Augmenter + Translator]

|

v

[Tag Generator (LLM/Rule-Based)]

|

v

[Embedding Generator (SBERT/Instructor)]

|

v

[Branch Matcher (Tags + Vectors + Filters)]

|

v

[Best Repair Branch Recommendation Output]

🛠️ Features

✅ Reddit daily data ingestion

✅ LLM-powered structured problem–solution extraction

✅ Data augmentation (paraphrasing, typos, slang)

✅ Multilingual translation (back-translation)

✅ Repair problem & solution tagging

✅ Semantic embedding generation

✅ Nearest repair branch recommendation

✅ Modular CLI-compatible pipeline

✅ Full Prefect + GitHub Actions orchestration

🧪 Pipeline Phases

✅ Phase 1: Reddit Data Extraction

Fetches threads from target subreddits using PRAW + HTTPX

Saves raw JSON data to /data/raw/

🔍 Phase 2: LLM-Based Data Cleaning

Uses offline LLMs like DeepSeek or Zephyr to extract problem/solution pairs

Invalid, bot, or low-effort threads are filtered

Cleaned data saved to /data/cleaned/

🎭 Phase 2.5: Data Augmentation

NLPAug/Custom logic for paraphrasing and noise injection

Expands variation and robustness

🌍 Phase 2.6: Translation (Optional)

Back-translation for non-English support

Uses offline or HuggingFace translation models

🏷️ Phase 3: Tag Generator

Applies rule-based or LLM-driven tags to problems and solutions

Tags mapped to mechanical specialties/tools

🔢 Phase 4: Embedding Generator

SBERT or Instructor-XL used for sentence embeddings

Embeds both post and branch capabilities

🧭 Phase 5: Branch Recommender

Matches user problem tags and embeddings with branches

Suggests nearest viable repair location

🧪 Phase 6: Testing & Evaluation

Module-level and end-to-end testing

Sample data in /docs/

🔄 Phase 7: Prefect Orchestration

Chains the full flow in a DAG

Provides observability, retry, and logs

☁️ Phase 8: GitHub Actions & Deployment

Auto-runs extraction and cleaning daily

Ready for Docker containerization

📘 Phase 9: Documentation

Full guides in /docs/

Includes architecture and sample JSON data

🖥️ Phase 10: Local Chatbot Interface (optional)

Load processed dataset locally to run LLM-assisted chat

Mechanics can input issues and receive structured advice

🔌 Phase 11: Backend Integration

RESTful or FastAPI endpoint for chatbot

POST user issue → return solution + repair branch recommendation

🧬 Data Flow Diagram

mermaid

Copy

Edit

graph TD

A[Reddit Data Extractor] --> B[LLM Data Cleaner]

B --> C[Augmenter: Paraphrase + Noise]

C --> D[Translator: Back-Translation]

D --> E[Tag Generator]

E --> F[Embedding Generator]

F --> G[Branch Matcher]

G --> H[Recommended Repair Branch]

🗂 Directory Structure

bash

Copy

Edit

📁 Car\_Clinic\_Project/

│

├── data/

│ ├── raw/

│ ├── cleaned/

│ ├── tagged/

│ ├── embeddings/

│ └── samples/

│

├── reddit\_data\_extractor/

│ ├── flow.py

│ ├── extractor.py

│ └── utils.py

│

├── reddit\_data\_cleaner/

│ ├── flow.py

│ ├── preprocessor.py

│ ├── llm\_runner.py

│ └── postprocessor.py

│

├── data\_augmenter/

│ ├── flow.py

│ ├── augmenter.py

│

├── translator/

│ ├── flow.py

│ ├── translator.py

│

├── tag\_generator/

│ ├── flow.py

│ ├── tagger.py

│

├── embedding\_generator/

│ ├── flow.py

│ ├── embedder.py

│

├── branch\_recommender/

│ ├── flow.py

│ ├── matcher.py

│ ├── ranker.py

│

├── prefect\_flows/

│ └── car\_clinic\_pipeline.py

│

├── .github/

│ └── workflows/

│ └── main.yaml

│

├── docs/

│ ├── architecture.png

│ ├── prompt\_templates.md

│ └── data\_samples.json

│

├── README.md

└── requirements.txt

⚙️ Tech Stack

Category Tool/Library

Data Source Reddit (PRAW, HTTPX)

LLM DeepSeek / GPTQ (Offline)

Embedding Model SBERT / Instructor-XL

Workflow Prefect + GitHub Actions

Prompting JSON-returning, robust

Language Python 3.10+

Deployment (opt) Docker

Storage Local (S3-ready)

⚡ Getting Started

1. Clone the Repository

bash

Copy

Edit

git clone https://github.com/Ibrahim-Hegazi/Car\_Clinic\_Project\_Revamped.git

cd Car\_Clinic\_Project\_Revamped

2. Set Up Environment

bash

Copy

Edit

python -m venv .venv

source .venv/bin/activate # Windows: .venv\Scripts\activate

pip install -r requirements.txt

3. Run Locally

bash

Copy

Edit

python prefect\_flows/car\_clinic\_pipeline.py

4. Run via GitHub Actions

Workflow file: .github/workflows/main.yaml

Triggered on push, PR, or schedule

🧠 Prompt Engineering Principles

Each prompt is:

🧭 Clear on role and format

📎 JSON-structured

🔒 Resistant to hallucination

🧩 Few-shot capable

🔄 Flexible for translation/augmentation

See /docs/prompt\_templates.md for full prompt examples and guidelines.

🗓 Roadmap

✅ Reddit scraping automation

✅ LLM offline extraction

✅ Tag schema + generator

✅ Embedding + similarity matcher

✅ Branch recommender

✅ GitHub CI/CD & Prefect orchestration

🔄 REST API endpoint (Flask/FastAPI)

🔄 Frontend UI (optional)

🔄 Docker deployment

🔄 Arabic / Spanish support

🧾 License

This project is licensed under the MIT License. See LICENSE for full text.

👨‍💻 Author

Ibrahim Hegazi

Data Engineer • NLP Builder • System Designer

GitHub

LinkedIn

⭐ If this repo helps your work or inspires you, feel free to give it a star!

Readme file 3  
  
  
**ChatGPT said:**

Here’s a **fully comprehensive README.md** tailored for your car\_clinic\_chatbot/ project. It explains the purpose, setup, pipeline structure, folder organization, tech stack, and instructions for both local development and CI/CD.

markdown

CopyEdit

# 🛠️ Car Clinic Chatbot 🚗

A fully autonomous system that scrapes car-related issues from Reddit, cleans and tags them using LLMs, generates embeddings, and recommends the best Car Clinic repair branch — all wrapped in an interactive chatbot with FastAPI.

---

## 📌 Project Overview

\*\*Car Clinic Chatbot\*\* is a modular data-driven application designed to:

- Extract car issues from Reddit

- Clean posts into structured problem–solution pairs using LLMs

- Translate, paraphrase, and enrich the data

- Auto-generate semantic tags and embeddings

- Match the issue with the best Car Clinic branch

- Provide results through a chatbot interface

---

## 🧱 Architecture

```mermaid

graph TD

A[Reddit Scraper] --> B[LLM Cleaner]

B --> C[Data Augmenter]

C --> D[Tag Generator]

D --> E[Embedding Generator]

E --> F[Branch Recommender]

F --> G[Chatbot API (FastAPI)]

**🧑‍💻 Tech Stack**

| **Layer** | **Tools / Frameworks** |
| --- | --- |
| **Backend API** | FastAPI, Pydantic |
| **Orchestration** | Prefect |
| **LLMs** | OpenAI, HuggingFace Transformers, MarianMT |
| **Embedding** | Sentence Transformers, FAISS |
| **Scraping** | PRAW, HTTPX |
| **Containerization** | Docker, Docker Compose |
| **CI/CD** | GitHub Actions |

**🗂️ Project Structure**

bash

CopyEdit

car\_clinic\_chatbot/

│

├── .github/ # GitHub Actions workflows

│ └── workflows/

│ └── main.yaml

│

├── README.md # You're here

├── requirements.txt # Main dependencies

├── .env # Environment variables

├── .gitignore # Ignored files & secrets

├── Dockerfile # Container image definition

├── docker-compose.yml # Optional orchestration

├── main.py # FastAPI entry point

├── prefect\_flows.py # Entry for end-to-end orchestration

├── llm\_prompt\_templates.py # Prompt templates for LLM tasks

│

├── config/ # Central app configuration

│ └── settings.py

│

├── data/ # Data lifecycle directory

│ ├── raw/ # Raw Reddit data

│ ├── cleaned/ # Structured problem → solution pairs

│ ├── augmented/ # Translated/paraphrased versions

│ ├── tagged/ # Tagged semantic data

│ ├── embeddings/ # Vector stores

│ │ ├── problems/

│ │ └── branches/

│ └── examples.jsonl # Manually verified examples

│

├── python\_scripts/ # Core data pipeline modules

│ ├── reddit\_data\_extractor/

│ │ ├── extractor.py, flow.py, etc.

│ ├── reddit\_data\_cleaner/

│ │ ├── llm\_cleaner.py, flow.py, etc.

│ ├── data\_augmenter/

│ │ ├── translator.py, paraphraser.py, etc.

│ ├── tag\_generator/

│ │ ├── tagger.py, constants.py, etc.

│ ├── embedding\_generator/

│ │ ├── embedder.py, flow.py, etc.

│ └── branch\_recommender/

│ ├── matcher.py, ranker.py, etc.

│

├── chatbot/ # Chat logic and decision engine

│ ├── query\_classifier.py

│ ├── retriever.py

│ ├── reasoner.py

│ ├── branch\_suggester.py

│ ├── formatter.py

│ └── fallback.py

│

├── api/ # FastAPI backend

│ ├── routes/

│ │ ├── chat.py

│ │ └── recommend.py

│ ├── models.py

│ ├── dependencies.py

│ └── server.py

│

├── tests/ # Unit + integration tests

│ ├── unit/

│ ├── integration/

│ └── test\_data/

│

└── docs/ # Documentation assets

├── architecture\_diagram.png

├── dag\_flow.png

├── embedding\_guide.md

├── branch\_matching.md

├── api\_schema.md

└── sample\_output\_examples.md

**⚙️ Installation**

bash

CopyEdit

# Clone the repo

git clone https://github.com/your-username/car\_clinic\_chatbot.git

cd car\_clinic\_chatbot

# Create virtual environment

python -m venv .venv

source .venv/bin/activate # or .venv\Scripts\activate on Windows

# Install dependencies

pip install -r requirements.txt

# Create .env file with API keys and secrets

cp .env.example .env

**🚀 Running the App Locally**

bash

CopyEdit

# Run FastAPI server

uvicorn main:app --reload

Navigate to: <http://localhost:8000/docs> for the interactive API.

**🧭 Running the Data Pipeline (Prefect)**

bash

CopyEdit

# Execute the end-to-end data pipeline

python prefect\_flows.py

You can also run flows individually from their respective flow.py files inside python\_scripts/.

**🐳 Docker Support**

bash

CopyEdit

# Build and run container

docker build -t car\_clinic\_chatbot .

docker run -p 8000:8000 car\_clinic\_chatbot

Or with Compose:

bash

CopyEdit

docker-compose up --build

**🔄 GitHub Actions (CI/CD)**

* Located at .github/workflows/main.yaml
* Actions include:
  + Running tests on push
  + Formatting/linting checks
  + Optional deployment or Prefect flow triggers

**✅ Testing**

bash

CopyEdit

# Run all tests

pytest tests/

**🧠 Key Features**

* ✅ Fully modular architecture
* ✅ LLM-based data cleaning & reasoning
* ✅ Multilingual & slang-resilient augmentation
* ✅ Semantic search and recommendation engine
* ✅ FastAPI-powered chatbot interface
* ✅ Automated CI/CD with GitHub Actions

**📬 Future Improvements**

* Deploy API on cloud (e.g., GCP, Azure)
* Integrate with Car Clinic's internal CRM
* Add feedback collection & retraining loop
* Monitor logs and alerts via Grafana + Prometheus

**🙋‍♂️ Contributing**

1. Fork the repo
2. Create your feature branch (git checkout -b feature/my-feature)
3. Commit your changes (git commit -am 'Add some feature')
4. Push to the branch (git push origin feature/my-feature)
5. Open a pull request

**📄 License**

This project is licensed under the MIT License — see the LICENSE file for details.

**📞 Contact**

**Author:** Ibrahim Hegazi  
**Email:** your.email@example.com  
**GitHub:** [github.com/Ibrahim-Hegazi](https://github.com/Ibrahim-Hegazi)

Built with 🧠, 🧰, and ☕️ by car nerds for car nerds.