# Voice-Enabled Data Science Q&A Bot using TDD.

## Project Overview

This project is a voice-enabled Q&A chatbot built using Streamlit, designed to answer only data science-related questions. Users can ask questions through voice, and the app processes them using speech recognition and a large language model hosted locally via Ollama.

### Key Objectives:

* Enable voice input and spoken responses.
* Ensure the model answers only data science-related questions.
* Use Test-Driven Development (TDD) to validate components.

## Tech Stack

|  |  |
| --- | --- |
| Component | Technology Used |
| Frontend | Streamlit |
| Voice Recognition | speech\_recognition |
| Text-to-Speech | pyttsx3 |
| Language Detection | langdetect |
| LLM Integration | Ollama (mistral model) |
| Testing | Pytest + Mocking |

## Key Features

* Voice-based question input (Speech-to-Text)
* Responses generated via Ollama LLM (Mistral)
* Text-to-speech replies using pyttsx3
* Language filtering (only accepts English)
* TDD-based unit testing for reliability

## Testing Strategy (TDD)

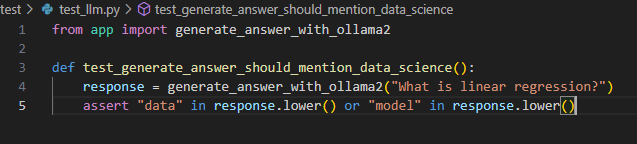
We followed a Test-Driven Development (TDD) approach:

### Steps:

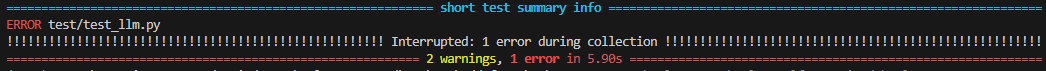
1. Write failing tests first based on expected behavior.

Example:

test\_llm.py



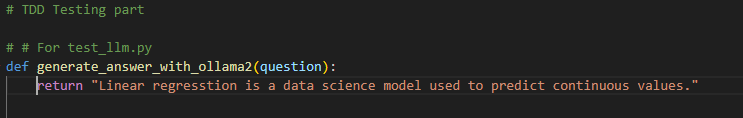
Result,



1. Implement only enough code to pass the test.

Example:

app.py



Result,

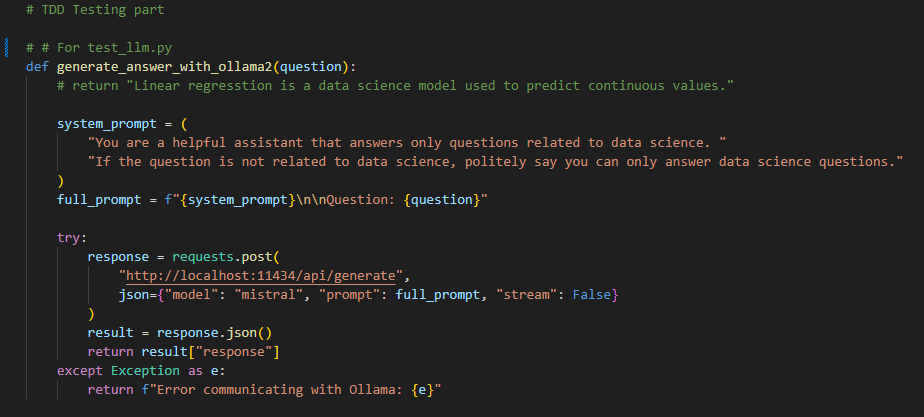
A screenshot of a computer program

AI-generated content may be incorrect.

1. Refactor and optimize the implementation.

Example:

app.py



Result,

A screenshot of a computer

AI-generated content may be incorrect.

1. Repeat.

Example:

* test\_voice
* test\_language

## Unit Tests Overview

|  |  |  |
| --- | --- | --- |
| Module | Function | Test Case |
| app.py | generate\_answer\_with\_ollama2 | Ensures LLM returns a data science-related response |
| app.py | get\_voice\_input2 | Simulates microphone input and checks the returned question text |
| app.py | detect\_language | Confirms correct detection of English input text |