AccuKnox – AI/ML Trainee Assignment

Name: Harshitha D G
Portfolio: https://harshithadevanga.lovable.app
Email: harshithadg97@gmail.com
GitHub: https://github.com/HarshithaDevanga
LinkedIn: https://linkedin.com/in/harshitha-d-g

Assignment 1

API Data Retrieval and Storage: You are tasked with fetching data from an external REST API, storing it in a local SQLite database, and displaying the retrieved data. The API provides a list of books in JSON format with attributes like title, author, and publication year.

(Assuming All APIs are REST APIs returning JSON. SQLite is used as a local lightweight database. CSV files are clean and available locally.)

I would approach as the goal is to collect book data from an external REST API and store it in a local database so that it can be queried and reused later. I first call the API using Python's requests library. Since the API returns data in JSON format, it can easily be converted into Python dictionaries.

Then, I create a SQLite database and a table named books with columns title, author and publication_year. Each book record from the API is inserted into this table and Finally, I read the stored data back and display it. SQLite is chosen because it is lightweight, does not need a server and works well for local applications.

Sample Code:

```
import requests
import sqlite3

#Fetch data from API
url =”https://example.com/api/books”
response = requests.get(url)
books = responses.json()

#Create SQLite database
Conn = sqlite3.connec(“books.db”)
Cursor = conn.cursor()


Cursor.execute(“ “ “
CREATE TABLE IF NOT EXISTS books (
  Id INTEGER PRIMARY KEY,
  Title TEXT,
  Author TEXT.
  Year INTEGER
)
“ “ “)
```

```
#Insert records
For book in books:
cursor.execute(
       "INSERT INTO books (title, author, year) VALUES (? ? ?)",
        (book["title"], book["author"], book["publication_year"])
)

conn.commit()

#Read and display
cursor.execute("SELECT title , author, year FROM books")
for row in cursor.fetchall():
       print(row)
conn.close()
```

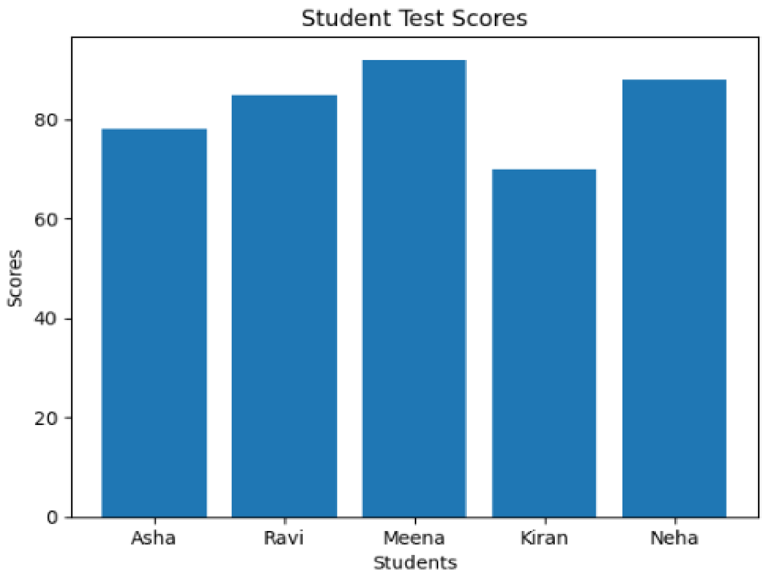
This method allows the data to be used even when the API is not available.

Data Processing and Visualization: Given a dataset containing information about students' test scores, fetch the data from an API, calculate the average score, and create a bar chart to visualize the data

First, I fetch all the student data from a REST API which returns a list of students with their names and test scores in JSON format. This data is converted into Python objects so that it can be processed. Next, I extract all the scores into a list and calculate the average score using a simple math formula:

Average=Number of students/Sum of all scores

After that I use the matplotlib library to create a bar chart where:
The x-axis represents student names
The y – axis represents their test scores



This makes it esay to visually compare student perform and identify who scored higher or lower.

Sample Python Code:

```
Import requests
Import matplotlib.pyplot as plt
```

```
# fetch students dat from API
url =”https://example.com/api/students”
students = requests.get(url).json()

#extract names and scores
names =[s[“name”] for s in students]
scores = [s[“score”]for s in students]

# Calculate average
average_score = sum(scores) / len(scores)
print("Average Score:", average_score)

# Create bar chart
plt.bar(names, scores)
plt.xlabel("Students")
plt.ylabel("Scores")
plt.title("Student Test Scores")
plt.xticks(rotation=45)
plt.show()
```

<u>CSV Data Import to a Database: </u>Write a Python script that reads data from a CSV file containing user information (e.g., name, email) and inserts it into a SQLite database.

First, I read the CSV file using Python’s csv module and then create a SQLite database and a table named users with columns name and email. Each row from the CSV file inserted into the database using SQL queries.After inserting the data, it is stored permanently and can be searched or reused later.

Code:

```
import csv
import sqlite3

#create or connect to database
conn = sqlite3.connect(“users.db”)
cursor = conn.cursor()

#create users table
cursor.execute(“ “ “
CREATE TABLE IF NOT EXISTS users (
        Id INTEGER PRIMARY KEY,
        name TEXT,
        email TEXT
)
“ “ “)

#Read CSV and insert data
```

```
With open(“users.csv”, “r”) as file:
        Reader = csv.DictReader(file)
        For row in reader:
           cursor.execute(
                “INSERT INTO users(name, email) VALUES (?, ?)”,
                (row[“name”], row[“email”])
            )
conn.commit()
conn.close()
```

By moving data from CSV file into database Data becomes structured, searching and filtering become easy,Duplicate or invalide entries cn be controlled and it can be used by applications, API or dashboards.

Send a link to the most complex Python code you have written

https://github.com/HarshithaDevanga/deepfake-audio-detection

This project is one of my most technically involved python implementation. The goal of the system is to distinguish between real and AI generated deepfake audio samples. Deepfake audio is a growing threat in media integrity and security, so a reliable detection system has high tral-world impact.This is Complex because Signal Processing + Deep Learning: This is the combination of extracting audio signal features and learning using deep learning techniques, which are both highly advanced Model Training Management: It encompasses the management of batch processing, the use of the GPU, training callbacks, and controlling overfitting Real-world Security Relevance: Assists in detecting the generated content that could be used for malicious phishing or impersonation campaigns.This reflects my capacity for creating fully functional intelligent systems, right from the data preprocessing stage, training, evaluating, and final deployment.

Send a link to the most complex database code you have written

[(23) Projects | Harshitha D G | LinkedIn]()

This is the most complex project I have done so far involving databases as it simulates the actual world railway booking system.Unlike a CRUD system, a railway reservation system requires atomic consistency; in other words, a single seat cannot be reserved more than once, and every event has rules.

Database Design

The system uses multiple related tables:

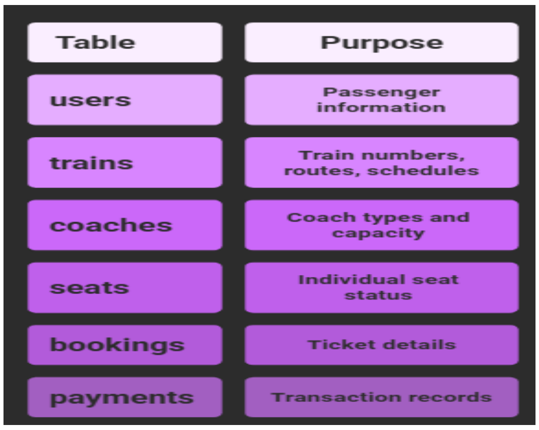
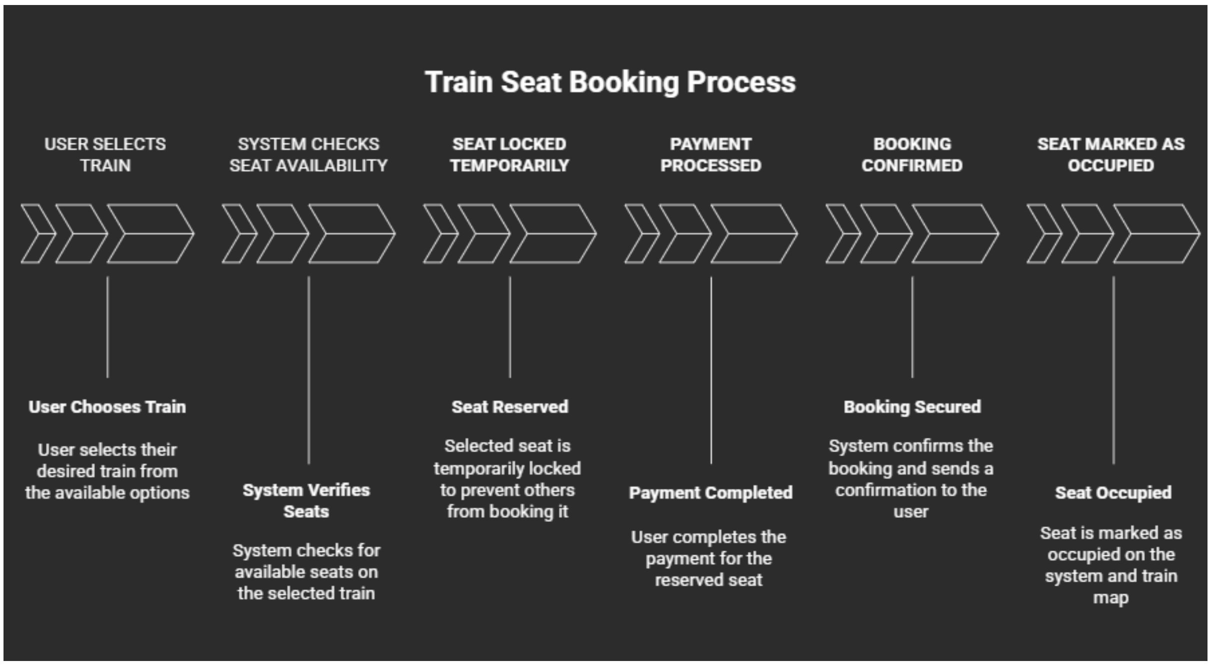


Table	Purpose
users	Passenger information
trains	Train numbers, routes, schedules
coaches	Coach types and capacity
seats	Individual seat status
bookings	Ticket details
payments	Transaction records

How bookings work:


Train Seat Booking Process
USER SELECTS TRAIN
SYSTEM CHECKS SEAT AVAILABILITY
SEAT LOCKED TEMPORARILY
PAYMENT PROCESSED
BOOKING CONFIRMED
SEAT MARKED AS OCCUPIED
User Chooses Train
User selects their desired train from the available options
System Verifies Seats
System checks for available seats on the selected train
Seat Reserved
Selected seat is temporarily locked to prevent others from booking it
Payment Completed
User completes the payment for the reserved seat
Booking Secured
System confirms the booking and sends a confirmation to the user
Seat Occupied
Seat is marked as occupied on the system and train map

Example Databse Logic :

```
SELECT seat_id
FROM seats
WHERE train_id = 101 AND is_booked = 0
LIMIT 1;

UPDATE seats
SET is_booked = 1
WHERE seat_id = 45;

INSERT INTO bookings(user_id, train_id, seat_id, status)
VALUES (12, 101, 45, 'CONFIRMED');
```

This project is Databse-Intensive as it uses relational integrity and foreign keys, it requires transaction safety, handels real time seat availability, prevents data inconsistency and simulates real world booking system.

ASSIGNMENT 2

1. Where would you rate yourself on (LLM, Deep Learning, AI, ML). A, B, C [A = can code independently; B = can code under supervision; C = have little or no understanding]

LLM – A: I think my skills in large language models are pretty solid right now. I can put together systems on my own, like chatbots or those RAG pipelines, and even workflows that chain prompts together. Ive used stuff like LangChain, Hugging Face, and Ollama to integrate models without much help. Still, Im keeping up with learning more about fine-tuning and optimizing big models at scale, it seems like theres always something new.

AI – A : When it comes to overall AI, I feel confident designing full systems from start to finish. That means mixing data pipelines with machine learning models, APIs, and even user interfaces to make actual apps that work. I get the basics of system design and how to deploy them, but Im always trying to dig deeper into it.

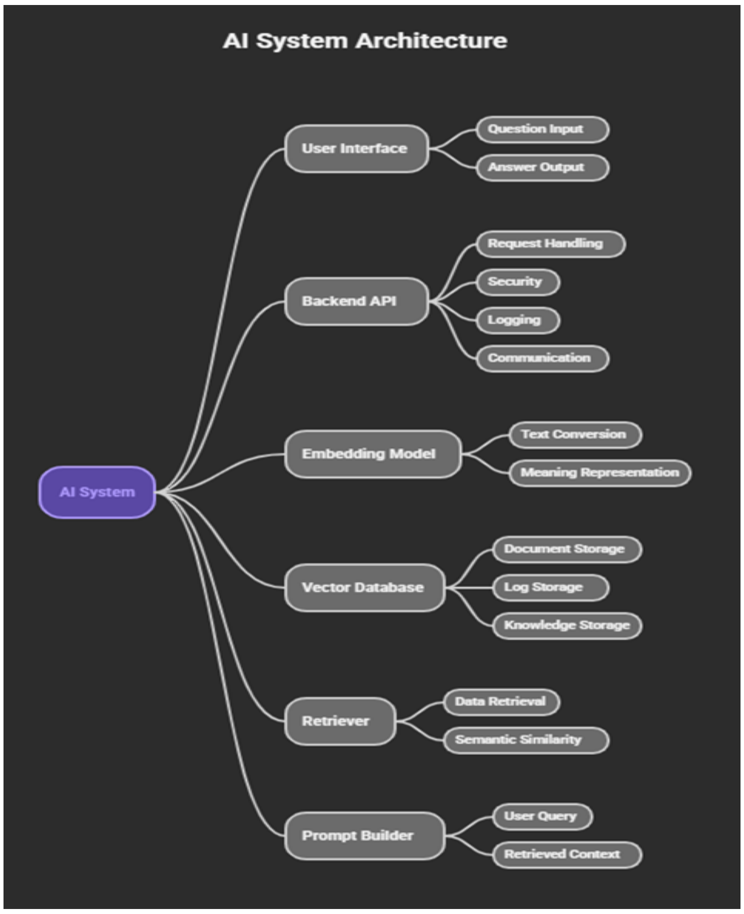
Deep Learning – B: Deep learning is a bit different for me. Ive built some neural networks, like CNNs for things such as detecting deepfake audio in real scenarios. I know about feature extraction, how training goes, loss functions, and evaluating them. Now Im working on getting better at more advanced designs, it might take a while.

Machine Learning – B: For machine learning in general, I can handle preprocessing data, training models, checking how they perform, and getting pipelines into apps. But optimization at an expert level and handling huge ML systems, thats where Im still building up, kind of growing into it.

A means I can code and build systems independently, B means I can build under supervision and continue learning.

2. What are the key architectural components to create a chatbot based on LLM? Please explain the approach on a high-level

A modern chatbot that uses LLMs, its not really just some language model spitting out answers to questions or anything. I mean, it feels like theres more going on there, kind of a bunch of layers working together to pull in the right info, think about it a bit, and then come up with responses that actually make sense.The way its built at a basic level, you have these different parts that make up the whole thing.



A person's inquiry is entered into the chatbot. After entering the inquiry, it gets passed to the API of the backend before getting sent to the embedding model. Once the embedding has been calculated, it will be used in submitting a query to a vector database in order to retrieve relevant articles (knowledge). When the relevant articles (knowledge) are retrieved from the vector database, they will be added to a structured prompt that will then be sent to the large language model (LLM). Once the LLM has received the structured prompt, it will generate a response to the inquiry by using both the inquiry and retrieved articles (knowledge). Once the generated response has been created by the LLM, it will be sent back to the user. The method in which this response generation occurs is called "Retrieval-Augmented Generation" (RAG) because the generated response is based upon actual data instead of simply a memory recall from the LLM.

3. Please explain vector databases. If you were to select a vector database for a hypothetical problem (you may define the problem) which one will you choose, and why?

A Vector Database stores and retrieves embeddings of Data (text/image/audio) with Similarity Search, rather than the traditional "exact words" searches of the traditional database.

As a vector database allows you to search by Similarity Search, rather than by Exact Word; therefore, vector databases have been created mainly for use with LLMs (large language models) and their capabilities to support RAG (Retrieval Augmented Generation, or Retrieval augmented Generations) by providing Relevant Search Context when answering user Questions.

Solution Example: Suppose a system/project for creating Cloud Security Tool or Assistant to respond to a user question (based on all the relevant Data via):

- Kubernetes Security log files
- Runtime Policies
- Incident Reports

To create the Tool, all three types of documentation above must first be generated to form embeddings to feed into the vector database and then create a query to retrieve the data most relevant to answering the user's question (from the Vector Database) and providing it to the LLM for Generating the most accurate response possible.

Best of all: I would choose Qdrant because it is a free/open-source application (which is a must for anyone dealing with Security). It has fast Similarity Searching capability and allows for filtering based on Metadata (i.e. TIMESTAMP, SEVERITY, etc.). It's a great fit to work with either LangChain, along with other LLM Pipelines and supports Deployment On-Premise; which is critical for Sensitive Data.