LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in ECSc

Submitted By

Name:- Bhairav Ganguly

Roll No: 2230246

Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Table of Content

Experiment Number	7
Experiment Title	Natural Language Database Interaction with LLMs
Date of Experiment	18/03/2025
Date of Submission	24/03/2025

Github link: https://github.com/Bhairavg7/AD_LAB_7.git

1. Objective:-

To interact with databases using natural language queries powered by LLMs.

2. Procedure:-

Detailed Procedure:

1. Set up a MySQL database and populate it with sample data.
2. Integrate an LLM to convert natural language queries into SQL commands.

3. Code:-

app.py

```
 from flask import Flask, render_template, request
import pymysql
import groq
import os
from dotenv import load_dotenv

# Load environment variables
try:
    load_dotenv(encoding='utf-8')
    if not os.getenv('GROQ_API_KEY'):
        print("Warning: GROQ_API_KEY not found in .env file")
except Exception as e:
    print(f"Error loading .env file: {e}")
    print("Please ensure .env file exists and is properly formatted")

app = Flask(__name__)

# Database configuration
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Binbud123$',
    'database': 'adlab',
    'cursorclass': pymysql.cursors.DictCursor  # Ensures results are dictionaries
}

# Initialize Groq client
groq_client = groq.Client(api_key=os.getenv('GROQ_API_KEY'))

def get_db_connection():
    try:
        return pymysql.connect(**db_config)
    except pymysql.MySQLError as e:
        print(f"Database Connection Error: {e}")
        return None

def test_db_connection():
    conn = None
    cursor = None
    try:
        conn = get_db_connection()
        if conn:
            cursor = conn.cursor()
            cursor.execute("SELECT 1")
            cursor.fetchall()
            print("Database connection successful!")
            return True
        else:
            print("Database connection failed!")
            return False
    except pymysql.MySQLError as e:
        print(f"Database connection failed: {e}")
        return False
    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()

def natural_to_sql(natural_query):
    prompt = f"""
    Convert the following natural language query to a MySQL query.
    The query should be for a restaurants table with columns:
    id, name, special_dish, rating, location, cuisine, contact_number, opening_hours, created_at

    Natural language query: {natural_query}

    Return only the SQL query without any explanation.
    """

    try:
        response = groq_client.chat.completions.create(
            messages=[{"role": "user", "content": prompt}],
            model="mixtral-8x7b-32768",
            temperature=0.2,
        )
        sql_query = response.choices[0].message.content.strip()
        sql_query = sql_query.replace('\_', '_').replace('\*', '*').replace('\\', '')  # Clean query
        return sql_query
    except Exception as e:
        print(f"Error in natural_to_sql: {e}")
        raise Exception("Failed to convert natural language to SQL query")
```

```
def execute_query(sql_query):
    conn = None
    cursor = None
    try:
        conn = get_db_connection()
        if not conn:
            raise Exception("Failed to connect to database")

        cursor = conn.cursor(pymysql.cursors.DictCursor)  # ✅ Fixed line
        print(f"Executing query: {sql_query}")  # Debug print
        cursor.execute(sql_query)
        results = cursor.fetchall()
        return results if results else []
    except pymysql.MySQLError as e:
        print(f"MySQL Error: {e}")
        raise Exception(f"Database error: {str(e)}")
    except Exception as e:
        print(f"General error: {e}")
        raise Exception(f"Error: {str(e)}")
    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()

@app.route('/', methods=['GET', 'POST'])
def index():
    results = []
    natural_query = ''
    sql_query = ''
    error = None

    if request.method == 'POST':
        natural_query = request.form['natural_query'].strip()
        if not natural_query:
            error = "Please enter a query"
        else:
            try:
                sql_query = natural_to_sql(natural_query)
                if sql_query:
                    results = execute_query(sql_query)
                    if not results:
                        error = "No results found for your query"
                else:
                    error = "Failed to generate SQL query"
            except Exception as e:
                error = str(e)
                results = []

    return render_template('index.html',
                         results=results,
                         natural_query=natural_query,
                         sql_query=sql_query,
                         error=error)

if __name__ == '__main__':
    if test_db_connection():
        app.run(debug=True)
    else:
        print("Please check your database configuration.")
```

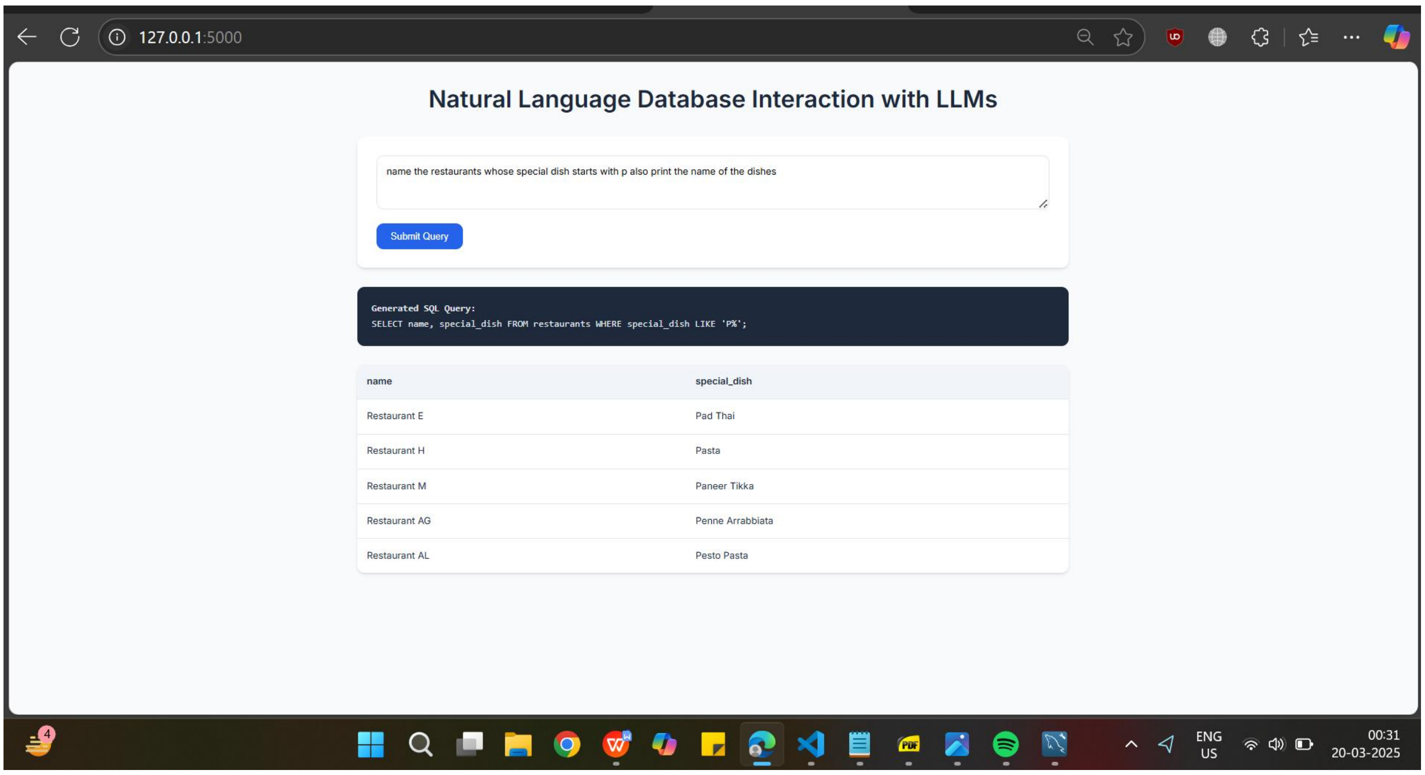
MySQL code for table:

```
CREATE TABLE restaurants (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    special_dish VARCHAR(255) NOT NULL,
    rating DECIMAL(2, 1) CHECK (rating >= 0 AND rating <= 5),
    location VARCHAR(255) NOT NULL,
    cuisine VARCHAR(100),
    contact_number VARCHAR(15),
    opening_hours VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);


INSERT INTO restaurants (name, special_dish, rating, location, cuisine, contact_number,
opening_hours) VALUES
('Restaurant A', 'Butter Chicken', 4.5, 'Delhi', 'Indian', '9876543210', '10 AM - 10 PM'),
('Restaurant B', 'Kung Pao Chicken', 4.0, 'Mumbai', 'Chinese', '9876543211', '11 AM - 11 PM'),
('Restaurant C', 'Margherita Pizza', 3.5, 'Bangalore', 'Italian', '9876543212', '9 AM - 9 PM'),
('Restaurant D', 'Tacos', 5.0, 'Chennai', 'Mexican', '9876543213', '12 PM - 12 AM'),
('Restaurant E', 'Pad Thai', 4.2, 'Kolkata', 'Thai', '9876543214', '10 AM - 10 PM'),
('Restaurant F', 'Sushi', 4.8, 'Hyderabad', 'Japanese', '9876543215', '11 AM - 10 PM'),
('Restaurant G', 'Biryani', 4.6, 'Ahmedabad', 'Indian', '9876543216', '10 AM - 11 PM'),
('Restaurant H', 'Pasta', 4.1, 'Pune', 'Italian', '9876543217', '11 AM - 10 PM'),
('Restaurant I', 'Chow Mein', 3.9, 'Jaipur', 'Chinese', '9876543218', '10 AM - 10 PM'),
('Restaurant J', 'Nachos', 4.3, 'Surat', 'Mexican', '9876543219', '12 PM - 11 PM'),
('Restaurant K', 'Tom Yum Soup', 4.7, 'Lucknow', 'Thai', '9876543220', '10 AM - 10 PM'),
('Restaurant L', 'Ramen', 4.4, 'Indore', 'Japanese', '9876543221', '11 AM - 10 PM'),
('Restaurant M', 'Paneer Tikka', 4.5, 'Nagpur', 'Indian', '9876543222', '10 AM - 10 PM'),
('Restaurant N', 'Lasagna', 3.8, 'Coimbatore', 'Italian', '9876543223', '9 AM - 9 PM'),
('Restaurant O', 'Burrito', 4.2, 'Visakhapatnam', 'Mexican', '9876543224', '12 PM - 12 AM'),
('Restaurant P', 'Green Curry', 4.6, 'Vadodara', 'Thai', '9876543225', '10 AM - 10 PM'),
('Restaurant Q', 'Sashimi', 4.1, 'Nashik', 'Japanese', '9876543226', '11 AM - 10 PM'),
('Restaurant R', 'Dumplings', 4.3, 'Mysore', 'Chinese', '9876543227', '10 AM - 10 PM'),
('Restaurant S', 'Quesadilla', 4.0, 'Rajkot', 'Mexican', '9876543228', '12 PM - 11 PM'),
('Restaurant T', 'Massaman Curry', 4.5, 'Bhubaneswar', 'Thai', '9876543229', '10 AM - 10 PM'),
('Restaurant U', 'Fried Rice', 4.2, 'Guwahati', 'Chinese', '9876543230', '11 AM - 10 PM'),
('Restaurant V', 'Cheese Naan', 4.6, 'Agra', 'Indian', '9876543231', '10 AM - 10 PM'),
('Restaurant W', 'Spaghetti', 4.1, 'Dehradun', 'Italian', '9876543232', '9 AM - 9 PM'),
('Restaurant X', 'Chili Con Carne', 4.3, 'Ranchi', 'Mexican', '9876543233', '12 PM - 12 AM'),
('Restaurant Y', 'Spring Rolls', 4.7, 'Patna', 'Thai', '9876543234', '10 AM - 10 PM'),
('Restaurant Z', 'Tempura', 4.4, 'Jodhpur', 'Japanese', '9876543235', '11 AM - 10 PM'),
('Restaurant AA', 'Fish Curry', 4.5, 'Kochi', 'Indian', '9876543236', '10 AM - 10 PM'),
('Restaurant AB', 'Fettuccine Alfredo', 3.8, 'Tirupati', 'Italian', '9876543237', '9 AM - 9 PM'),
('Restaurant AC', 'Enchiladas', 4.2, 'Kolkata', 'Mexican', '9876543238', '12 PM - 12 AM'),
('Restaurant AD', 'Green Papaya Salad', 4.6, 'Chennai', 'Thai', '9876543239', '10 AM - 10 PM'),
('Restaurant AE', 'Miso Soup', 4.1, 'Hyderabad', 'Japanese', '9876543240', '11 AM - 10 PM'),
('Restaurant AF', 'Chole Bhature', 4.5, 'Delhi', 'Indian', '9876543241', '10 AM - 10 PM'),
('Restaurant AG', 'Penne Arrabbiata', 3.9, 'Mumbai', 'Italian', '9876543242', '9 AM - 9 PM'),
('Restaurant AH', 'Fajitas', 4.3, 'Bangalore', 'Mexican', '9876543243', '12 PM - 12 AM'),
('Restaurant AI', 'Tom Kha Gai', 4.7, 'Ahmedabad', 'Thai', '9876543244', '10 AM - 10 PM'),
('Restaurant AJ', 'Soba Noodles', 4.4, 'Pune', 'Japanese', '9876543245', '11 AM - 10 PM'),
('Restaurant AK', 'Aloo Gobi', 4.5, 'Jaipur', 'Indian', '9876543246', '10 AM - 10 PM'),
('Restaurant AL', 'Pesto Pasta', 3.8, 'Surat', 'Italian', '9876543247', '9 AM - 9 PM'),
('Restaurant AM', 'Chimichangas', 4.2, 'Lucknow', 'Mexican', '9876543248', '12 PM - 12 AM'),
('Restaurant AN', 'Red Curry', 4.6, 'Kolkata', 'Thai', '9876543249', '10 AM - 10 PM'),
('Restaurant AO', 'Udon Noodles', 4.1, 'Hyderabad', 'Japanese', '9876543250', '11 AM - 10 PM');
```

```
select * from restaurants;
```

4. Results/Output:-

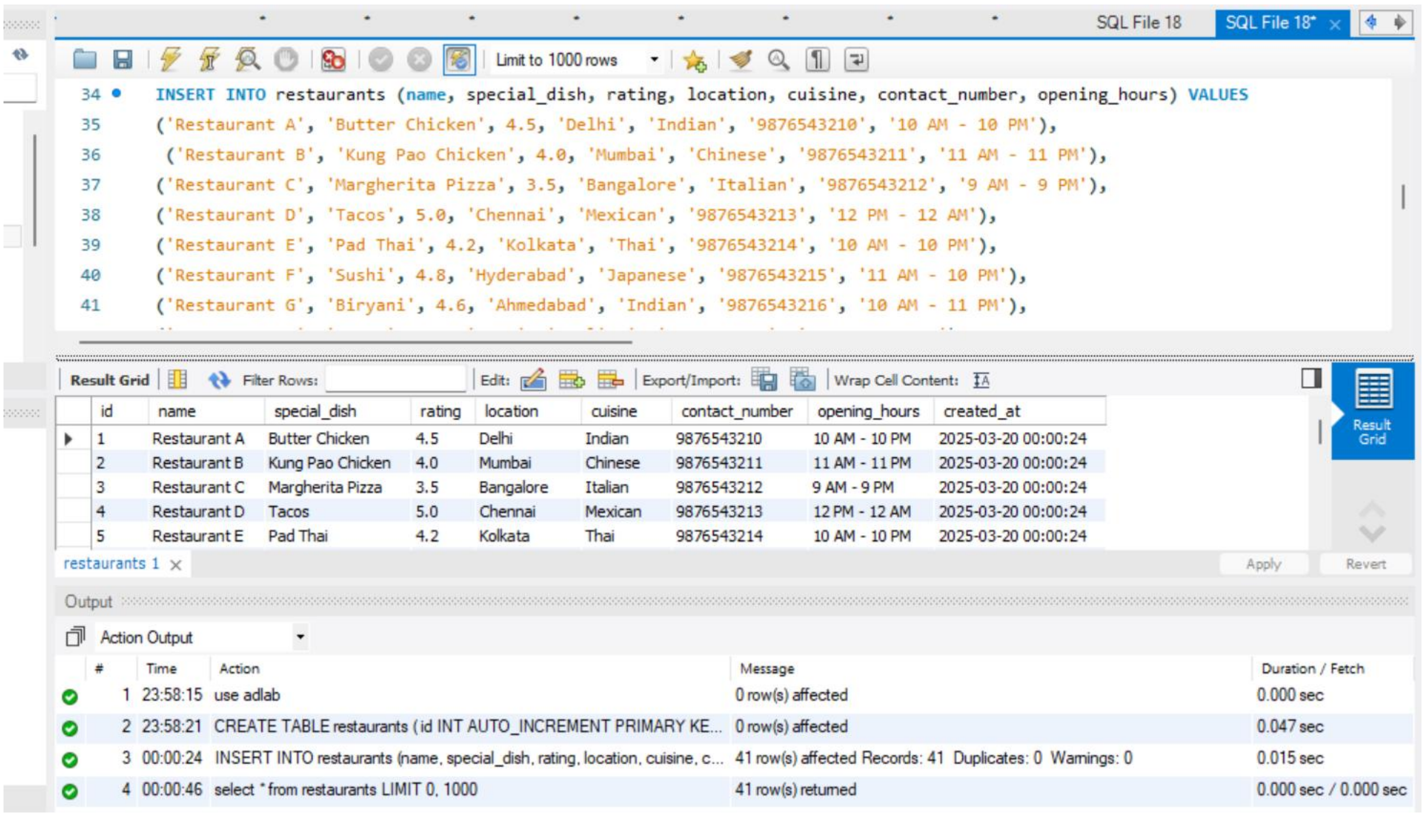


Natural Language Database Interaction with LLMs

name the restaurants whose special dish starts with p also print the name of the dishes

```
Generated SQL Query:
SELECT name, special_dish FROM restaurants WHERE special_dish LIKE 'P%';
```

name	special_dish
Restaurant E	Pad Thai
Restaurant H	Pasta
Restaurant M	Paneer Tikka
Restaurant AG	Penne Arrabbiata
Restaurant AL	Pesto Pasta



```sql
INSERT INTO restaurants (name, special_dish, rating, location, cuisine, contact_number, opening_hours) VALUES
('Restaurant A', 'Butter Chicken', 4.5, 'Delhi', 'Indian', '9876543210', '10 AM - 10 PM'),
 ('Restaurant B', 'Kung Pao Chicken', 4.0, 'Mumbai', 'Chinese', '9876543211', '11 AM - 11 PM'),
('Restaurant C', 'Margherita Pizza', 3.5, 'Bangalore', 'Italian', '9876543212', '9 AM - 9 PM'),
('Restaurant D', 'Tacos', 5.0, 'Chennai', 'Mexican', '9876543213', '12 PM - 12 AM'),
('Restaurant E', 'Pad Thai', 4.2, 'Kolkata', 'Thai', '9876543214', '10 AM - 10 PM'),
('Restaurant F', 'Sushi', 4.8, 'Hyderabad', 'Japanese', '9876543215', '11 AM - 10 PM'),
('Restaurant G', 'Biryani', 4.6, 'Ahmedabad', 'Indian', '9876543216', '10 AM - 11 PM'),
```

id	name	special_dish	rating	location	cuisine	contact_number	opening_hours	created_at
1	Restaurant A	Butter Chicken	4.5	Delhi	Indian	9876543210	10 AM - 10 PM	2025-03-20 00:00:24
2	Restaurant B	Kung Pao Chicken	4.0	Mumbai	Chinese	9876543211	11 AM - 11 PM	2025-03-20 00:00:24
3	Restaurant C	Margherita Pizza	3.5	Bangalore	Italian	9876543212	9 AM - 9 PM	2025-03-20 00:00:24
4	Restaurant D	Tacos	5.0	Chennai	Mexican	9876543213	12 PM - 12 AM	2025-03-20 00:00:24
5	Restaurant E	Pad Thai	4.2	Kolkata	Thai	9876543214	10 AM - 10 PM	2025-03-20 00:00:24

	#	Time	Action	Message	Duration / Fetch
✓	1	23:58:15	use adlab	0 row(s) affected	0.000 sec
✓	2	23:58:21	CREATE TABLE restaurants (id INT AUTO_INCREMENT PRIMARY KE...	0 row(s) affected	0.047 sec
✓	3	00:00:24	INSERT INTO restaurants (name, special_dish, rating, location, cuisine, c...	41 row(s) affected Records: 41 Duplicates: 0 Warnings: 0	0.015 sec
✓	4	00:00:46	select * from restaurants LIMIT 0, 1000	41 row(s) returned	0.000 sec / 0.000 sec

5. Remarks:-

Signature of the Student
Bhairav Ganguly (Name of the Student)

Signature of the Lab Coordinator
____
(Name of the Coordinator)