DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DATABASE MANAGEMENT SYSTEM – CO202

PROJECT FILE

DISEASE MANAGEMENT DATABASE

Submitted to:

Prof. Rohit Beniwal

Submitted By:

Uday Singhal(2K22/CO/480)

Unnati (2K22/CO/484)

DISEASE MANAGEMENT DATABASE

ABSTRACT

This project aims to develop a web application that allows users to find the number of patients with a specific disease in a particular city. The application leverages a MySQL database to store patient, disease, and city data. Users can search for diseases topologically, view detailed patient information, and access consolidated statistics such as the total number of patients, gender distribution, and age range. The application is built using Flask, a lightweight web framework for Python, and incorporates essential database management principles.

INTRODUCTION

The rapid growth of healthcare data necessitates efficient management and retrieval systems. This project focuses on developing a Disease Tracker web application to facilitate the monitoring of disease prevalence in various cities. The system enables users to query the database for specific diseases and locations, providing both detailed and consolidated views of the patient data. This application demonstrates the practical implementation of database management concepts, including entity-relationship modeling, normalization, and SQL query execution.

INFORMATION OF ENTITIES

1. **Patient**:
    - **Attributes**: PatientID, Name, Age, Gender, CityID
    - **Description**: Represents individuals who are recorded in the healthcare system. Each patient has a unique identifier, name, age, gender, and an associated city.

2. **Disease**:
    - **Attributes**: DiseaseID, DiseaseName
    - **Description**: Represents diseases that patients might be diagnosed with. Each disease has a unique identifier and a name.

3. **City**:
    - **Attributes**: CityID, CityName
    - **Description**: Represents cities where patients reside. Each city has a unique identifier and a name.

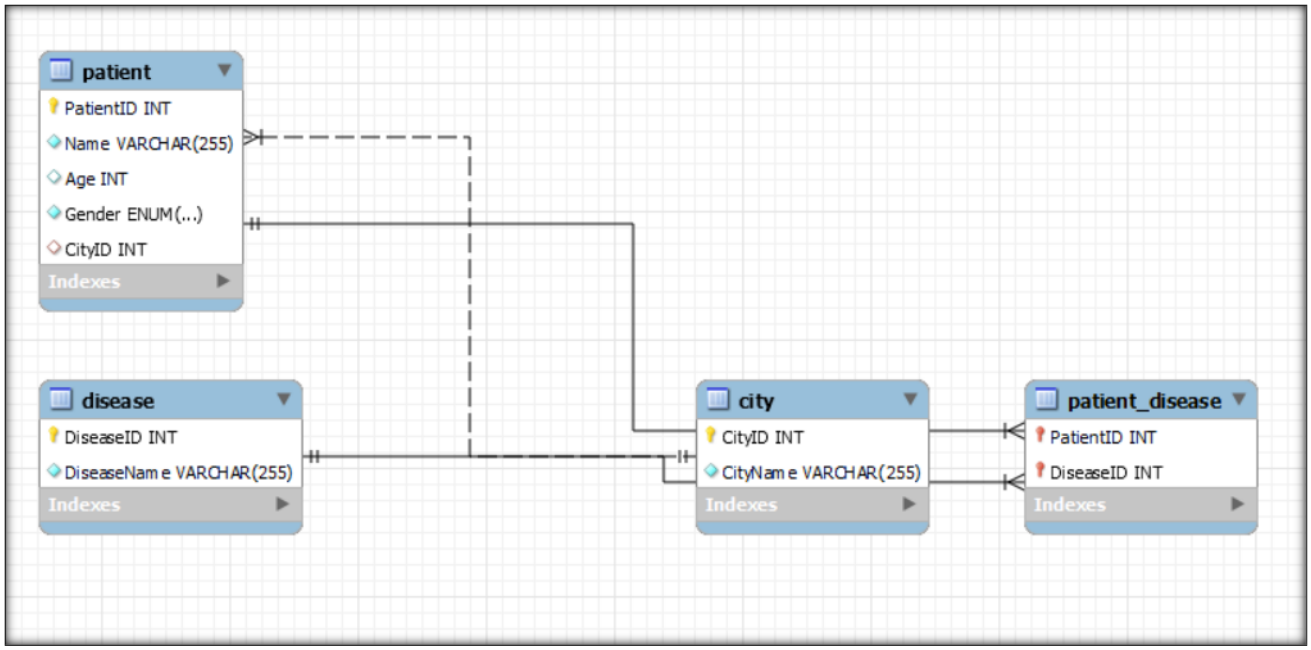
4. **Patient_Disease**:
    - **Attributes**: PatientID, DiseaseID
    - **Description**: Represents the many-to-many relationship between patients and diseases. Each record indicates that a specific patient has been diagnosed with a specific disease.

RELATIONSHIP BETWEEN ENTITIES

- **Patient - City**: Each patient resides in one city (many-to-one relationship). This relationship is represented by the CityID attribute in the Patient entity, which is a foreign key referencing the CityID in the City entity.
- **Patient - Disease**: Patients can have multiple diseases, and each disease can affect multiple patients (many-to-many relationship). This relationship is managed by the Patient_Disease entity, which contains foreign keys referencing both PatientID and DiseaseID.

RELATION SCHEMAS

1. **Patient (PatientID, Name, Age, Gender, CityID)**
    - Primary Key: PatientID
    - Foreign Key: CityID references City.CityID
2. **Disease (DiseaseID, DiseaseName)**
    - Primary Key: DiseaseID
3. **City (CityID, CityName)**
    - Primary Key: CityID
4. **Patient_Disease (PatientID, DiseaseID)**
    - Primary Key: Composite of PatientID and DiseaseID
    - Foreign Key: PatientID references Patient.PatientID
    - Foreign Key: DiseaseID references Disease.DiseaseID



NORMALIZATION

Normalization is a database design technique used to reduce data redundancy and improve data integrity. The tables in this project are normalized to the third normal form (3NF).

1. **First Normal Form (1NF)**: Ensures that each table has a primary key and that each column contains atomic values.
    - Each entity's table has a primary key.
    - All attributes contain atomic (indivisible) values.
2. **Second Normal Form (2NF)**: Ensures that each non-primary key attribute is fully functionally dependent on the primary key.
    - The Patient table's attributes are fully dependent on PatientID.
    - The Patient_Disease table ensures no partial dependencies due to the composite primary key.
3. **Third Normal Form (3NF)**: Ensures that there are no transitive dependencies between non-primary key attributes.
    - All non-primary attributes are dependent only on the primary key.

CREATION OF TABLES

```
-- Create City table
CREATE TABLE City (
    CityID INT PRIMARY KEY,
    CityName VARCHAR(255) NOT NULL
);
-- Create Disease table
CREATE TABLE Disease (
    DiseaseID INT PRIMARY KEY,
    DiseaseName VARCHAR(255) NOT NULL
);
-- Create Patient table
CREATE TABLE Patient (
    PatientID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Age INT NOT NULL,
    Gender VARCHAR(50) NOT NULL,
    CityID INT,
    FOREIGN KEY (CityID) REFERENCES City(CityID)
);
-- Create Patient_Disease table
CREATE TABLE Patient_Disease (
    PatientID INT,
    DiseaseID INT,
    PRIMARY KEY (PatientID, DiseaseID),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DiseaseID) REFERENCES Disease(DiseaseID)
);
```

```
-- Insert data into City table
INSERT INTO City (CityID, CityName) VALUES
(1, 'Delhi'), (2, 'Mumbai'), (3, 'Kolkata'), (4, 'Chennai'), (5, 'Bangalore'),
(6, 'Hyderabad'), (7, 'Ahmedabad'), (8, 'Pune'), (9, 'Surat'), (10, 'Jaipur');

-- Insert data into Disease table
INSERT INTO Disease (DiseaseID, DiseaseName) VALUES
(1, 'Dengue'), (2, 'Malaria'), (3, 'Chikungunya'), (4, 'COVID-19'), (5, 'Influenza'),
(6, 'Tuberculosis'), (7, 'Hepatitis'), (8, 'HIV/AIDS'), (9, 'Cholera'), (10, 'Typhoid');

-- Insert data into Patient table
INSERT INTO Patient (PatientID, Name, Age, Gender, CityID) VALUES
(1, 'Aman Sharma', 34, 'Male', 1), (2, 'Bhavna Singh', 28, 'Female', 2),
(3, 'Chetan Bhagat', 45, 'Male', 3), (4, 'Disha Patani', 32, 'Female', 4),
(5, 'Esha Gupta', 29, 'Female', 5), (6, 'Farhan Akhtar', 50, 'Male', 6),
(7, 'Gaurav Kumar', 36, 'Male', 7), (8, 'Hina Khan', 33, 'Female', 8),
(9, 'Isha Negi', 27, 'Female', 9), (10, 'Jatin Rawat', 40, 'Male', 10),
(11, 'Karan Johar', 48, 'Male', 1), (12, 'Lata Mangeshkar', 75, 'Female', 2),
(13, 'Manish Paul', 39, 'Male', 3), (14, 'Nidhi Agarwal', 30, 'Female', 4),
(15, 'Om Puri', 65, 'Male', 5), (16, 'Priya Sharma', 24, 'Female', 6),
(17, 'Quentin Das', 55, 'Male', 7), (18, 'Rani Mukherjee', 42, 'Female', 8),
(19, 'Sanjay Dutt', 60, 'Male', 9), (20, 'Tina Gupta', 23, 'Female', 10);

INSERT INTO Patient_Disease (PatientID, DiseaseID) VALUES
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10), (11, 1), (12, 2), (13, 3), (14, 4), (15, 5), (16, 6), (17, 7), (18, 8), (19, 9), (20, 10), (21, 1), (22, 2), (23, 3), (24, 4), (25, 5), (26, 6), (27, 7), (28, 8), (29, 9), (30, 10), (31, 1), (32, 2), (33, 3), (34, 4), (35, 5), (36, 6), (37, 7), (38, 8), (39, 9), (40, 10), (41, 1), (42, 2), (43, 3), (44, 4), (45, 5), (46, 6), (47, 7), (48, 8), (49, 9), (50, 10), (51, 1), (52, 2), (53, 3), (54, 4), (55, 5), (56, 6), (57, 7), (58, 8), (59, 9), (60, 10), (61, 1), (62, 2), (63, 3), (64, 4), (65, 5), (66, 6), (67, 7), (68, 8), (69, 9), (70, 10), (71, 1), (72, 2), (73, 3), (74, 4), (75, 5), (76, 6), (77, 7), (78, 8), (79, 9), (80, 10), (81, 1), (82, 2), (83, 3), (84, 4), (85, 5), (86, 6), (87, 7), (88, 8), (89, 9), (90, 10), (91, 1), (92, 2), (93, 3), (94, 4), (95, 5), (96, 6), (97, 7), (98, 8), (99, 9), (100, 10);
```

App.py File

```
from flask import Flask, request, render_template, redirect, url_for, flash
import mysql.connector

app = Flask(__name__)
app.secret_key = 'your_secret_key'  # Ensure you replace this with a strong
secret key

def get_db_connection():
    return mysql.connector.connect(
        host='localhost',
        user='root',  # replace with your MySQL username
        password='Uday@09122004',  # replace with your MySQL password
        database='mydatabase'  # replace with your MySQL database name
    )

@app.route('/')
def welcome():
    return render_template('welcome.html')

@app.route('/search_disease')
def search_disease():
    return render_template('search_disease.html')

@app.route('/search', methods=['POST'])
def search():
    disease = request.form['disease']
    city = request.form['city']

    conn = get_db_connection()
    cursor = conn.cursor()
    query = """
        SELECT Patient.PatientID, Patient.Name, Disease.DiseaseName,
City.CityName, Patient.Age, Patient.Gender
        FROM Patient
        JOIN City ON Patient.CityID = City.CityID
        JOIN Patient_Disease ON Patient.PatientID = Patient_Disease.PatientID
        JOIN Disease ON Patient_Disease.DiseaseID = Disease.DiseaseID
        WHERE Disease.DiseaseName = %s AND City.CityName = %s
    """
    cursor.execute(query, (disease, city))
    results = cursor.fetchall()
    cursor.close()
    conn.close()
```

```
    return render_template('results.html', results=results, disease=disease,
city=city)

@app.route('/consolidated_view')
def consolidated_view():
    disease = request.args.get('disease')
    city = request.args.get('city')

    conn = get_db_connection()
    cursor = conn.cursor()

    query_total = """
        SELECT COUNT(*)
        FROM Patient
        JOIN City ON Patient.CityID = City.CityID
        JOIN Patient_Disease ON Patient.PatientID = Patient_Disease.PatientID
        JOIN Disease ON Patient_Disease.DiseaseID = Disease.DiseaseID
        WHERE Disease.DiseaseName = %s AND City.CityName = %s
    """
    cursor.execute(query_total, (disease, city))
    total_patients = cursor.fetchone()[0]

    query_male = """
        SELECT COUNT(*)
        FROM Patient
        JOIN City ON Patient.CityID = City.CityID
        JOIN Patient_Disease ON Patient.PatientID = Patient_Disease.PatientID
        JOIN Disease ON Patient_Disease.DiseaseID = Disease.DiseaseID
        WHERE Disease.DiseaseName = %s AND City.CityName = %s AND
Patient.Gender = 'Male'
    """
    cursor.execute(query_male, (disease, city))
    male_patients = cursor.fetchone()[0]

    query_female = """
        SELECT COUNT(*)
        FROM Patient
        JOIN City ON Patient.CityID = City.CityID
        JOIN Patient_Disease ON Patient.PatientID = Patient_Disease.PatientID
        JOIN Disease ON Patient_Disease.DiseaseID = Disease.DiseaseID
        WHERE Disease.DiseaseName = %s AND City.CityName = %s AND
Patient.Gender = 'Female'
    """
    cursor.execute(query_female, (disease, city))
    female_patients = cursor.fetchone()[0]
```

```
    query_age = """
        SELECT MIN(Patient.Age), MAX(Patient.Age)
        FROM Patient
        JOIN City ON Patient.CityID = City.CityID
        JOIN Patient_Disease ON Patient.PatientID = Patient_Disease.PatientID
        JOIN Disease ON Patient_Disease.DiseaseID = Disease.DiseaseID
        WHERE Disease.DiseaseName = %s AND City.CityName = %s
    """
    cursor.execute(query_age, (disease, city))
    age_range = cursor.fetchone()

    cursor.close()
    conn.close()

    return render_template('consolidated_view.html',
total_patients=total_patients, male_patients=male_patients,
female_patients=female_patients, age_range=age_range)

if __name__ == '__main__':
    app.run(debug=True)
```

Welcome.html File

```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to Patient Search</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.cs
s">
</head>
<body>
    <div class="container">
        <h1 class="mt-5">Welcome to Patient Search</h1>
        <p class="lead">Please choose a service:</p>
        <ul class="list-group">
            <li class="list-group-item">
                <a href="/search_disease">Search for diseases by cityname</a>
            </li>
        </ul>
    </div>
</body>
</html>
```

Search_diseases.html File

```
<!DOCTYPE html>
<html>
<head>
    <title>Search Disease by Location</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.cs
s">
    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
    <script src="https://code.jquery.com/ui/1.12.1/jquery-
ui.min.js"></script>
    <link rel="stylesheet"
href="https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
    <script>
        $(document).ready(function() {
            var diseases = ["Cancer", "Diabetes", "COVID-19", "Flu",
"Hypertension"]; // Example diseases
            var cities = ["New York", "Los Angeles", "Chicago", "Houston",
"Phoenix"]; // Example cities

            $("#disease").autocomplete({
                source: diseases
            });

            $("#city").autocomplete({
                source: cities
            });
        });
    </script>
</head>
<body>
    <div class="container">
        <h1 class="mt-5">Search Disease by Location</h1>
        <form action="/search" method="post">
            <div class="form-group">
                <label for="disease">Disease:</label>
                <input type="text" id="disease" name="disease" class="form-
control" required>
            </div>
            <div class="form-group">
                <label for="city">City:</label>
                <input type="text" id="city" name="city" class="form-control"
required>
            </div>
            <button type="submit" class="btn btn-primary">Search</button>
```

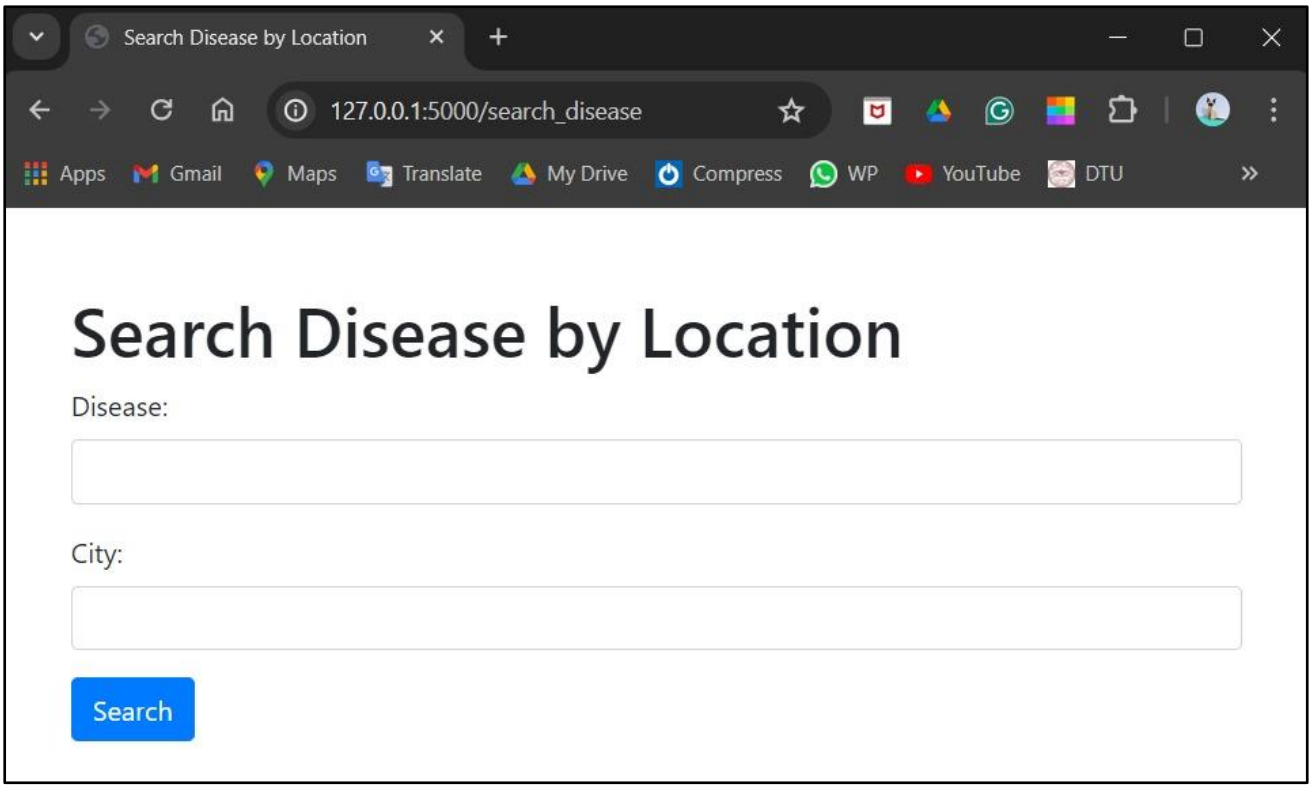
```
        </form>
    </div>
</body>
</html>
```

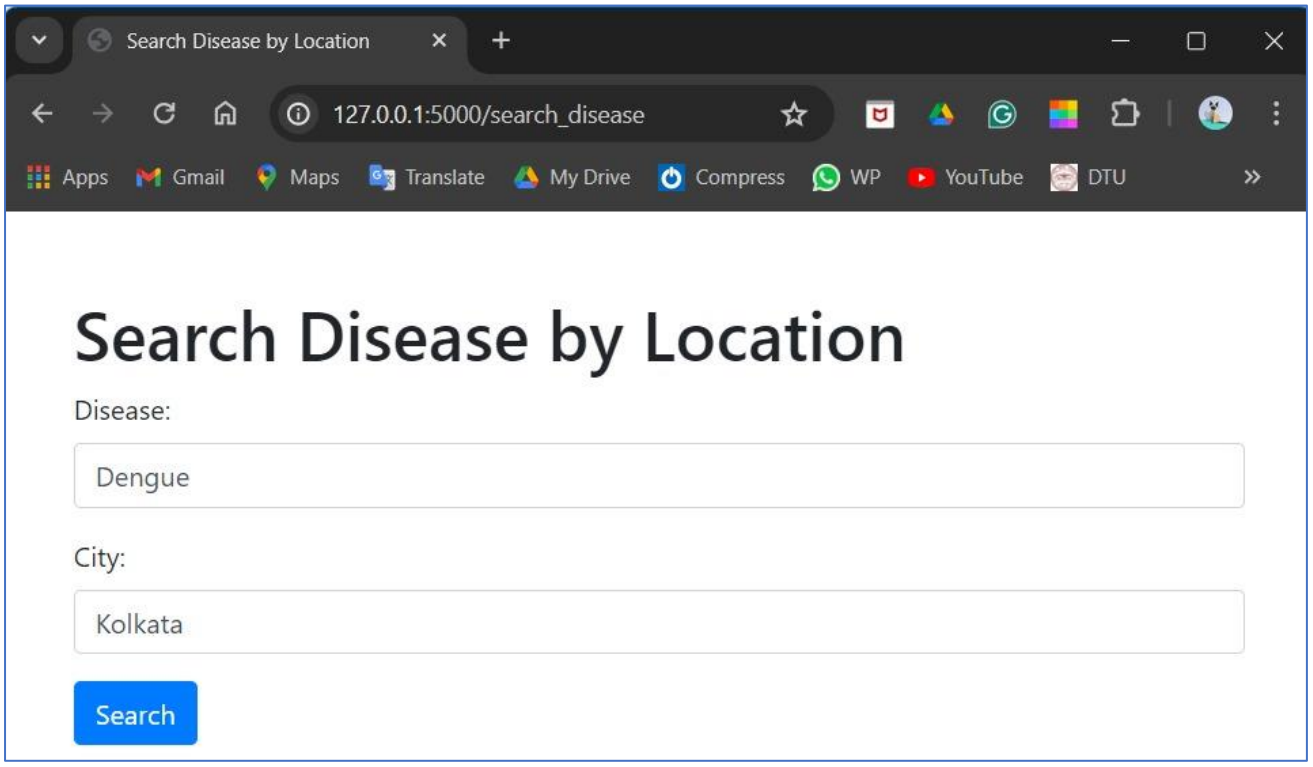
Consolidated_view.html File

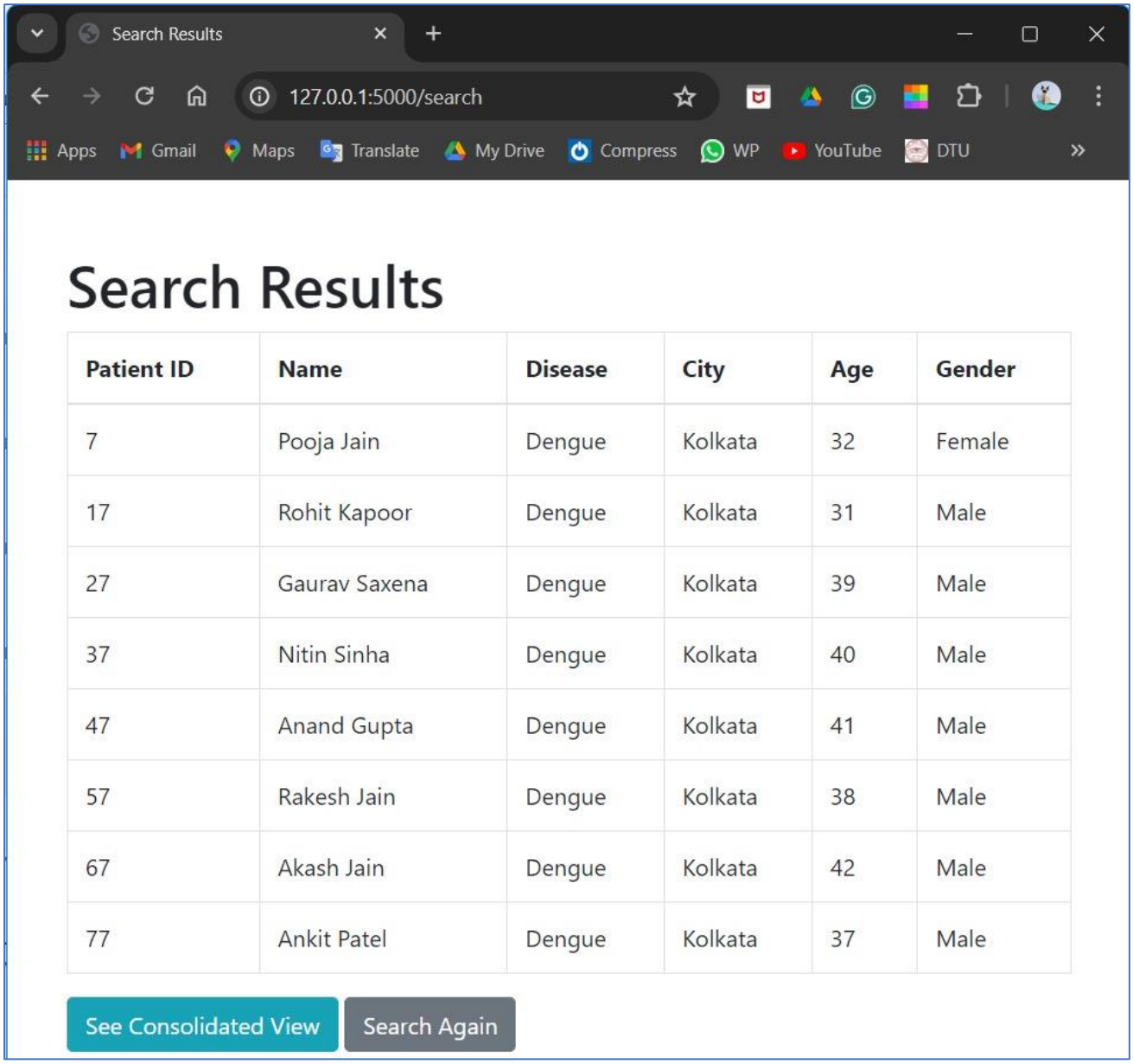
```
<!DOCTYPE html>
<html>
<head>
    <title>Consolidated View</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.cs
s">
</head>
<body>
    <div class="container">
        <h1 class="mt-5">Consolidated View</h1>
        <p>Total Patients: {{ total_patients }}</p>
        <p>Number of Male Patients: {{ male_patients }}</p>
        <p>Number of Female Patients: {{ female_patients }}</p>
        <p>Age Range: {{ age_range[0] }} - {{ age_range[1] }}</p>
        <a href="/" class="btn btn-secondary">Back to Search</a>
    </div>
</body>
</html>
```

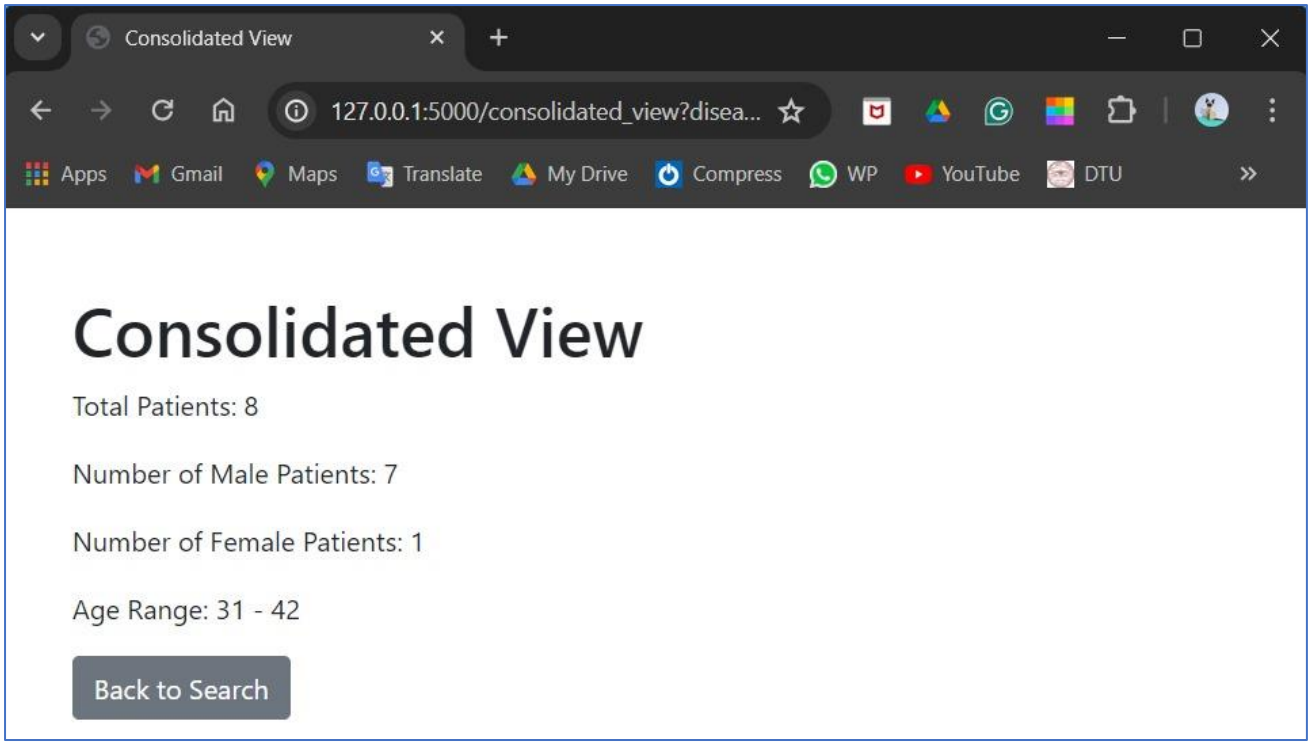
Results.html File

```
<!DOCTYPE html>
<html>
<head>
    <title>Search Results</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.cs
s">
</head>
<body>
    <div class="container">
        <h1 class="mt-5">Search Results</h1>
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th>Patient ID</th>
                    <th>Name</th>
                    <th>Disease</th>
                    <th>City</th>
                    <th>Age</th>
                    <th>Gender</th>
                </tr>
            </thead>
            <tbody>
                {% for row in results %}
                <tr>
                    <td>{{ row[0] }}</td>
                    <td>{{ row[1] }}</td>
                    <td>{{ row[2] }}</td>
                    <td>{{ row[3] }}</td>
                    <td>{{ row[4] }}</td>
                    <td>{{ row[5] }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        <a href="/consolidated_view?disease={{ disease }}&city={{ city }}"
class="btn btn-info">See Consolidated View</a>
        <a href="/" class="btn btn-secondary">Search Again</a>
    </div>
</body>
</html>
```


Search Disease by Location
127.0.0.1:5000/search_disease
Apps
Gmail
Maps
Translate
My Drive
Compress
WP
YouTube
DTU
Search Disease by Location
Disease:
City:
Search


Search Disease by Location
127.0.0.1:5000/search_disease
Apps
Gmail
Maps
Translate
My Drive
Compress
WP
YouTube
DTU
Search Disease by Location
Disease:
Dengue
City:
Kolkata
Search


Search Results
127.0.0.1:5000/search
Apps
Gmail
Maps
Translate
My Drive
Compress
WP
YouTube
DTU
Search Results
Patient ID
Name
Disease
City
Age
Gender
7
Pooja Jain
Dengue
Kolkata
32
Female
17
Rohit Kapoor
Dengue
Kolkata
31
Male
27
Gaurav Saxena
Dengue
Kolkata
39
Male
37
Nitin Sinha
Dengue
Kolkata
40
Male
47
Anand Gupta
Dengue
Kolkata
41
Male
57
Rakesh Jain
Dengue
Kolkata
38
Male
67
Akash Jain
Dengue
Kolkata
42
Male
77
Ankit Patel
Dengue
Kolkata
37
Male
See Consolidated View
Search Again




```
(venv) C:\Users\udaya\my_flask_app>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production d
eployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 598-967-308
127.0.0.1 - - [27/May/2024 15:19:05] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2024 15:19:11] "GET /search_disease HTTP/1.1" 20
0 -
127.0.0.1 - - [27/May/2024 15:19:21] "POST /search HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2024 15:19:26] "GET /consolidated_view?disease=D
engue&city=Kolkata HTTP/1.1" 200 -
```

CONCLUSION

This DBMS project successfully demonstrates the practical application of database management principles through the development of a web-based disease tracking system. By leveraging Flask for the web framework and MySQL for the database, the project provides a robust platform for querying and managing patient data based on disease and location. The structured approach of entity-relationship modeling ensured a clear understanding of the data relationships, while normalization techniques helped in reducing data redundancy and maintaining data integrity.

The system enables users to search for disease prevalence in specific cities, view detailed patient information, and access consolidated statistics that offer insights into patient demographics and disease distribution. The integration of SQL for database interactions and the use of HTML for dynamic web pages highlight the seamless interplay between backend and frontend technologies.

Overall, this project underscores the importance of efficient database design and implementation in addressing real-world healthcare data management challenges. The successful creation and population of the database, along with the development of an interactive web application, demonstrate the efficacy of combining theoretical knowledge with practical skills in DBMS. This project serves as a solid foundation for further enhancements, such as incorporating more complex queries, adding new features, and expanding the database to handle larger datasets.