**ASSIGNMENT**

COM - 402

**BY**

Pratham Suri

2022A6R034

Semester: 4th

Department of Computer Science and Engineering

****

**Model Institute of Engineering & Technology (Autonomous)**

(Permanently Affiliated to the University of Jammu, Accredited by NAAC with “A” Grade)

Jammu, India

2024



**Subject Name:** Database Management System

**Subject Code:** COM-402

**Due Date:** 20-05-2024

|  |  |  |  |
| --- | --- | --- | --- |
| **Question Number** | **Course Outcomes** | **Bloom’s Level** | **Maximum Marks** |
| Q1 | CO3, CO4, CO5 | 6 | 20 |
| Total Marks | | | 20 |
| Faculty Signature:  Email: navin.cse@mietjammu.in | | | |

* **Objective:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Q.NO.** | **Questions** | **Marks** | **Total Marks** |
| **1** | **SQL as a Data Manipulation Language**  1.Create the tables for the Company database in your text, and populate with data.  2. Create a simple desktop app to load, add and delete the data from database. [Use any language Python tk, c#, .net etc.]  3. Create a Mini Project report for the application you have created. | **5** | **5** |
| **2** | **Practice writing Queries** | **7.5** | **7.5** |
| **3** | **Database Design Assignment**  **Part I -- the ER Diagram**  **Part II -- Normalization of the Universal Relation** | **7.5** | **7.5** |

**ABSTRACT**

Jammu Gems is a comprehensive desktop application tailored to enhance the exploration and management of notable places in Jammu city. The application provides a user-friendly interface for both residents and visitors, facilitating the discovery, categorization, and favoriting of various locations including hotels, restaurants, bakeries, and temples. Powered by MySQL for reliable data management, Jammu Gems ensures efficient and robust handling of data, enabling users to search for places, view detailed information, and store favorite spots for quick access.

Key features include an intuitive browsing experience, detailed place descriptions, and a streamlined method for managing favorite locations. Users can easily add places to their favorites list, view their favorites, and remove places as needed. The application supports image display for a better visual representation of the places. By providing these features, Jammu Gems aims to make the exploration of Jammu city both enjoyable and straightforward, offering a reliable tool for discovering the hidden gems and popular attractions of the city.

* **TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| S. No. | Title | Page Number |
| 1. | Introduction | 5-7 |
| 2. | Literature Overview | 7-8 |
| 3. | System Design | 8-11 |
| 4. | Implementation | 12-27 |
| 5. | Conclusion | 28 |
| 6. | References | 29 |

**1. INTRODUCTION**

Exploring a new city or finding hidden gems in your hometown can be a daunting task, often leaving residents and visitors overwhelmed with options and lacking reliable information. **Jammu Gems** addresses this challenge by providing a comprehensive desktop application designed to simplify the exploration and management of notable places in Jammu city. With its user-friendly interface, Jammu Gems allows users to easily discover, categorize, and favorite various locations such as hotels, restaurants, bakeries, and temples. Powered by MySQL, the application ensures efficient data management, helping users search for places, view detailed information, and maintain a personalized list of favorite spots. By offering detailed descriptions, images, and streamlined favorite management, Jammu Gems transforms the way people explore and enjoy the city, making it both enjoyable and straightforward.

**1.1 BACKGROUND**

Jammu, a city rich in culture and history, offers numerous attractions including temples, markets, bakeries, and hotels. However, finding reliable information about these places can be challenging, especially for newcomers and tourists. This often leads to time-consuming and inefficient searches.

Jammu Gems was developed to address this issue, providing a centralized, user-friendly desktop application that simplifies the exploration and management of notable places in Jammu city. Powered by MySQL, the application offers comprehensive and easily accessible data, allowing users to discover, categorize, and favorite various locations effortlessly. By offering detailed descriptions and images, Jammu Gems enhances the user experience, transforming the way people explore and enjoy the city.

**1.2 PROBLEM STATEMENT**

Exploring a new city or discovering hidden gems in one's hometown can be daunting due to scattered and unreliable information. Residents and visitors of Jammu face challenges in finding comprehensive and trustworthy details about local attractions such as hotels, restaurants, bakeries, and temples. Traditional methods of research are time-consuming and inefficient, often leading to frustration. Additionally, there is a lack of tools for personalizing and managing favorite spots, which diminishes the overall user experience. Jammu Gems addresses these issues by offering a user-friendly desktop application that centralizes information, provides detailed descriptions and images, and allows for easy discovery, categorization, and favoriting of various locations in Jammu.

**1.3 OBJECTIVE**

The primary objective of Jammu Gems is to simplify the exploration and management of notable places in Jammu city for both residents and visitors. This will be achieved by centralizing comprehensive and reliable information on local attractions such as hotels, restaurants, bakeries, and temples. The application aims to enhance the user experience through a user-friendly interface that allows for easy discovery, categorization, and favoriting of various locations. By leveraging MySQL for efficient data management, Jammu Gems ensures that information is easily accessible and up-to-date. Additionally, the application promotes personalization by enabling users to maintain a personalized list of favorite spots with detailed descriptions and images, thereby supporting local tourism and businesses by making them more accessible to a broader audience.

**1.4 OVERVIEW OF THE TECHNOLOGIES USED**

1. **MySQL**: MySQL is the core database management system used in Jammu Gems. It is responsible for storing and managing data related to various places, including hotels, restaurants, bakeries, and temples. MySQL ensures efficient data retrieval and manipulation, enabling the application to provide up-to-date information quickly.

2. **C# and Windows Forms**: The application is built using C# and Windows Forms, which provide a robust framework for developing desktop applications on Windows. This technology facilitates the creation of a user-friendly interface, allowing users to interact with the application seamlessly. Windows Forms provides various controls like text boxes, buttons, list boxes, and picture boxes, which are essential for the application's functionality.

**3. MySql.Data.MySqlClient**: The MySql.Data.MySqlClient library is used to establish a connection between the application and the MySQL database. It provides classes for connecting to a MySQL database, executing queries, and retrieving results. This library is crucial for performing CRUD (Create, Read, Update, Delete) operations within the application.

**4. System.Drawing**: The System.Drawing namespace is utilized for handling images within the application. This includes loading images from the database, converting them to a format that can be displayed in the PictureBox control, and managing image rendering.

5. **Windows Forms Controls:**

Various Windows Forms controls are used to create the graphical user interface. These include:

ListBox: To display the list of places.

TextBox: For searching places.

PictureBox: To display images associated with selected places.

Buttons: For navigation and executing commands like searching and favoriting places.

Labels: To display text and information to the user.

CheckBox: To manage and display favorite status.

**2. LITERATURE OVERVIEW**

The "JammuGems" application is a C# Windows Forms application designed to assist users in exploring, managing, and favoriting various places in Jammu city. It connects to a MySQL database to store and retrieve information about places, ensuring efficient data management. The main form, Form1, establishes the initial database connection and provides navigation to Form3, which is responsible for displaying a list of places categorized into hotels, restaurants, bakeries, and temples. Form3 also includes a search functionality that allows users to filter places by name, dynamically updating the list based on the search query.

When users select a place from the list, Form3 retrieves and displays detailed information, including images. It features a list box for place selection, a picture box for displaying images, and text boxes for search input. Additionally, Form3 allows users to navigate to Form4, which categorizes places and enables more focused browsing. Form4 displays the names of places within a selected category and allows further interaction by linking to Form5, where users can see more detailed information about a specific place, including its image and description.

Form5 provides functionality for marking places as favorites. It checks if a place is already a favorite, displays its details, and allows users to add or remove it from their favorites list. The favorites are stored in a separate database, accessed by Form6, which lists all favorite places and provides functionality to delete them. This separation of concerns ensures that users can easily manage their preferred locations without cluttering the primary database.

Form7 offers an informative section about the application, detailing its name, version, description, and features. It serves as a user guide, explaining how to browse places, add them to favorites, and manage these favorites. The use of multiple forms and database integration in "JammuGems" showcases a modular approach to application development, promoting maintainability and scalability. Each form has a specific role, contributing to a cohesive user experience while leveraging MySQL for robust data handling.

**3. SYSTEM DESIGN**

System design ensures the effective implementation of the Jammu Gems Desktop Application, providing users to identify the famous places in Jammu.

* 1. **ARCHITECTURE OF THE APPLICATION**

The architecture of the JammuGems application follows a modular and layered approach, ensuring maintainability, scalability, and efficient data management. The primary components of the architecture are:

**Presentation Layer:** The Presentation Layer is responsible for the user interface and user interaction. It comprises multiple Windows Forms, each dedicated to specific functionalities.

**Business Logic Layer:** The Business Logic Layer handles the core functionality and application logic, including data processing, validation, and user actions. This layer ensures that the data retrieved from the Data Access Layer is correctly processed and presented to the user.

**Data Access Layer:** The Data Access Layer is responsible for communication with the database. It contains methods for querying the database, retrieving data, and performing CRUD operations.

* **Database Connections**: Establishes connections to the “**places”** and “**fav”** databases using MySQL.
* **Data Retrieval**: Executes SQL queries to fetch data for places, categories, and favorites.
* **Data Manipulation**: Handles insertion and deletion of favorite places in the database.

**Database Layer:** The Database Layer consists of two MySQL databases:

* **places Database**: Contains tables for different categories of places (hotels, restaurants, bakeries, temples). Each table stores information such as name, image, and place description.
* **fav Database**: Contains a table **areas** to store the names of favorite places.
  1. **DATABASE SCHEMA DESIGN**

The database schema is designed to efficiently store and manage information about various places in Jammu, as well as user favorites. The schema consists of tables in two databases: **places** and **fav**.

* + 1. places Database:

The **places** database contains four tables, each representing a category of places:

* **Hotels**
* **Restaurants**
* **Bakery**
* **Temples**

**Each table has the following columns:**

* **Name (VARCHAR): Stores the name of the place.**
* **Place (VARCHAR): Stores the location or address of the place.**
* **Image (BLOB): Stores the image of the place as a binary large object.**
  + 1. fav Database

The **fav** database contains one table named **areas** to store user favorite places:

The table has the following column:

* Name (VARCHAR): Stores the name of the favorite place.

**3.3 GUI DESIGN**

The GUI design of the Jammu Gems application emphasizes user-friendliness and intuitive navigation, incorporating various elements to enhance usability. The main interface comprises:

**3.3.1 ListBox Control**

Utilized to display a list of places in each category, allowing users to browse through hotels, restaurants, bakeries, and temples easily.

**3.3.2 PictureBox Control**

Used to display images of selected places, providing a visual representation that enhances user engagement.

**3.3.3 User Interaction Elements**

Buttons:

* Add to Favourites: Allows users to mark a place as a favorite.
* View Favourites: Enables users to navigate to a list of their favorite places.
* Back: Lets users navigate back to the previous form or main menu.
* Search: Triggers a search operation based on user input.
* Exit: Closes the application.

Textboxes:

* Search Field: Input field for users to enter search queries, such as the name of a place they are looking for.

Labels:

* Descriptions: Static text elements used to provide descriptions or instructions to users, such as category names or search instructions.
* Status Messages: Informational messages to indicate actions like "Added to favourites" or "Error occurred".

CheckBox:

* Favourite Indicator: A checkbox to show whether a place is marked as a favorite. Checking the box adds the place to the favorites list, while unchecking it removes the place.

3.3.4 MenuStrip Control

* Provides a structured and easy-to-navigate menu for users to access different categories of places (hotels, restaurants, bakeries, temples) and other features like viewing favorites or exiting the application.

3.3.5 Search Functionality

* A search bar allows users to search for places by entering keywords in the search field. The application then displays matching results in the ListBox, providing quick access to specific places.

**4. IMPLEMENTATION**

**4.1 Code**

**4.1.1**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MySql.Data.MySqlClient;

using System.Windows.Forms;

namespace Assignment1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

string mysqlCon = "datasource=localhost;port=3306;username=root;password=root;database=places";

MySqlConnection mysqlconnection = new MySqlConnection(mysqlCon);

}

private void button1\_Click(object sender, EventArgs e)

{

Form3 db =new Form3();

db.Show();

this.Hide();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}

**4.1.2**

using MySql.Data.MySqlClient;

using MySqlX.XDevAPI.Relational;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Assignment1

{

public partial class Form3 : Form

{

string connectionString = "datasource=localhost;port=3306;username=root;password=root;database=places";

MySqlConnection connection;

MySqlDataAdapter adapter;

DataTable table;

public Form3()

{

InitializeComponent();

connection = new MySqlConnection(connectionString);

adapter = new MySqlDataAdapter();

table = new DataTable();

LoadData();

}

private void LoadData(string searchQuery = "")

{

try

{

connection.Open();

string query = "SELECT name, image FROM hotels " +

"UNION ALL SELECT name, image FROM restaurants " +

"UNION ALL SELECT name, image FROM bakery " +

"UNION ALL SELECT name, image FROM temples";

if (!string.IsNullOrEmpty(searchQuery))

{

query = "SELECT name, image FROM hotels WHERE name LIKE @searchQuery " +

"UNION ALL SELECT name, image FROM restaurants WHERE name LIKE @searchQuery " +

"UNION ALL SELECT name, image FROM bakery WHERE name LIKE @searchQuery " +

"UNION ALL SELECT name, image FROM temples WHERE name LIKE @searchQuery";

}

MySqlCommand command = new MySqlCommand(query, connection);

if (!string.IsNullOrEmpty(searchQuery))

{

command.Parameters.AddWithValue("@searchQuery", "%" + searchQuery + "%");

}

adapter.SelectCommand = command;

table.Clear();

adapter.Fill(table);

listBox1.Items.Clear();

foreach (DataRow row in table.Rows)

{

listBox1.Items.Add(row["name"].ToString());

}

}

catch (Exception ex)

{

MessageBox.Show("An error occurred: " + ex.Message);

}

finally

{

connection.Close();

}

}

private void Form3\_Load(object sender, EventArgs e)

{

LoadData();

}

private void menuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBox1.SelectedIndex >= 0)

{

string selectedName = listBox1.SelectedItem.ToString();

DataRow[] selectedRows = table.Select("name = '" + selectedName + "'");

if (selectedRows.Length > 0)

{

byte[] imageData = (byte[])selectedRows[0]["image"];

if (imageData != null)

{

using (MemoryStream ms = new MemoryStream(imageData))

{

pictureBox1.Image = Image.FromStream(ms);

}

}

}

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

LoadData(textBox1.Text);

}

private void button1\_Click(object sender, EventArgs e)

{

LoadData(textBox1.Text);

}

private void pLACESToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void hOTELSToolStripMenuItem\_Click(object sender, EventArgs e)

{

OpenForm4WithItemName("HOTELS", "hotels");

this.Hide();

}

private void bAKERYToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

OpenForm4WithItemName("BAKERY", "bakery");

this.Hide();

}

private void rESTAURANTSToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

OpenForm4WithItemName("RESTAURANTS", "restaurants");

this.Hide();

}

private void tEMPLESToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

OpenForm4WithItemName("TEMPLES", "temples");

this.Hide();

}

private void OpenForm4WithItemName(string itemName, string tablename)

{

Form4 form4 = new Form4(tablename);

form4.SetItemName(itemName);

form4.Show();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void fAVOURITEPLACESToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form6 form6 = new Form6();

form6.Show();

this.Hide();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

int exitCode = 0;

Environment.Exit(exitCode);

}

private void button2\_Click(object sender, EventArgs e)

{

Form7 form7=new Form7();

form7.Show();

this.Hide();

}

}

}

**4.1.3**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Assignment1

{

public partial class Form4 : Form

{

private string tableName;

private string connectionString = "datasource=localhost;port=3306;username=root;password=root;database=places";

public Form4(string tableName)

{

InitializeComponent();

this.tableName = tableName;

LoadData();

}

private void LoadData()

{

try

{

MySqlConnection connection = new MySqlConnection(connectionString);

string query = $"SELECT name FROM {tableName}";

MySqlCommand command = new MySqlCommand(query, connection);

connection.Open();

MySqlDataReader reader = command.ExecuteReader();

listBox1.Items.Clear();

while (reader.Read())

{

listBox1.Items.Add(reader["name"].ToString());

}

}

catch (Exception ex)

{

MessageBox.Show($"An error occurred while loading data from {tableName}: " + ex.Message);

}

}

public void label1\_Click(object sender, EventArgs e)

{

}

public void fun(string label)

{

label1.Text = label;

}

public void SetItemName(string itemName)

{

label1.Text = itemName;

Form5 form5 = new Form5(itemName);

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBox1.SelectedItem != null)

{

string selectedItem = listBox1.SelectedItem.ToString();

Form5 form5 = new Form5(tableName);

form5.SetItemName(selectedItem);

form5.Show();

this.Hide();

}

}

private void button1\_Click(object sender, EventArgs e)

{

Form3 fom3=new Form3();

fom3.Show();

this.Hide();

}

private void Form4\_Load(object sender, EventArgs e)

{

}

}

}

**4.1.4**

using MySql.Data.MySqlClient;

using Org.BouncyCastle.Crypto.Generators;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Assignment1

{

public partial class Form5 : Form

{

private string table;

private string connectionString = "datasource=localhost;port=3306;username=root;password=root;database=places";

private string favConnectionString = "datasource=localhost;port=3306;username=root;password=root;database=fav";

public Form5(string data)

{

InitializeComponent();

this.table = data;

}

private void LoadData()

{

try

{

MySqlConnection connection = new MySqlConnection(connectionString);

string query = $"SELECT name FROM {table}";

MySqlCommand command = new MySqlCommand(query, connection);

connection.Open();

}

catch (Exception ex)

{

MessageBox.Show($"An error occurred while loading data from {table}: " + ex.Message);

}

}

private void Form5\_Load(object sender, EventArgs e)

{

string itemName = label1.Text;

if (!string.IsNullOrEmpty(itemName))

{

bool isFavourite = CheckIfFavourite(itemName);

checkBox1.Checked = isFavourite;

if (isFavourite)

{

label4.Text = "Added to favourite";

}

}

}

public void SetItemName(string itemName)

{

label1.Text = itemName;

var (place, image) = GetPlaceForItem(itemName);

if (!string.IsNullOrEmpty(place))

{

label3.Text = place;

checkBox1.Checked = CheckIfFavourite(itemName);

if (image != null)

{

using (var ms = new System.IO.MemoryStream(image))

{

pictureBox1.Image = Image.FromStream(ms);

}

}

else

{

pictureBox1.Image = null;

}

}

else

{

MessageBox.Show($"Item '{itemName}' not found in table '{table}'.");

}

}

private (string place, byte[] image) GetPlaceForItem(string itemName)

{

string place = null;

byte[] image = null;

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = $"SELECT place, image FROM `{table}` WHERE `name` = @itemName";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@itemName", itemName);

using (MySqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

place = reader["place"].ToString();

image = reader["image"] as byte[];

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Error retrieving place and image for item '{itemName}': " + ex.Message);

}

}

return (place, image);

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

string itemName = label1.Text;

string place = label3.Text;

if (checkBox1.Checked)

{

label4.Text = "Added to favourites ✓";

if (!string.IsNullOrEmpty(itemName) && !CheckIfFavourite(itemName))

{

AddToFavourite(itemName);

}

}

else

{

label4.Text = "";

}

}

private bool CheckIfFavourite(string itemName)

{

using (MySqlConnection connection = new MySqlConnection(favConnectionString))

{

try

{

connection.Open();

string query = "SELECT COUNT(\*) FROM areas WHERE name = @itemName";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@itemName", itemName);

return Convert.ToInt32(command.ExecuteScalar()) > 0;

}

catch (Exception ex)

{

MessageBox.Show($"Error checking favourite status for item '{itemName}': " + ex.Message);

return false;

}

}

}

private void AddToFavourite(string itemName)

{

using (MySqlConnection connection = new MySqlConnection(favConnectionString))

{

try

{

connection.Open();

string query = "INSERT INTO areas (name) VALUES (@itemName)";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@itemName", itemName);

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show($"Error adding item '{itemName}' to favourites: " + ex.Message);

}

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

Form4 form4 = new Form4(table); // Create an instance of Form4

form4.fun(table.ToUpper());

form4.Show(); // Call Show on the instance

this.Hide(); // Hide the current form

}

private void groupBox1\_Enter(object sender, EventArgs e)

{

}

}

}

**4.1.5**

using MySql.Data.MySqlClient;

using System;

using System.Windows.Forms;

namespace Assignment1

{

public partial class Form6 : Form

{

string table;

private string connectionString1 = "datasource=localhost;port=3306;username=root;password=root;database=fav";

public Form6()

{

InitializeComponent();

LoadFavourites();

}

// Load favourite places from the database into the ListBox

private void LoadFavourites()

{

listBox1.Items.Clear();

using (MySqlConnection connection = new MySqlConnection(connectionString1))

{

try

{

connection.Open();

string query = "SELECT name FROM areas";

MySqlCommand command = new MySqlCommand(query, connection);

using (MySqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

listBox1.Items.Add(reader["name"].ToString());

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Error loading favourites: {ex.Message}");

}

}

}

// Event handler for delete button click

private void buttonDelete\_Click(object sender, EventArgs e)

{

string selectedItem = listBox1.SelectedItem?.ToString();

if (!string.IsNullOrEmpty(selectedItem))

{

DialogResult result = MessageBox.Show($"Do you want to delete '{selectedItem}' from your favourites?", "Delete Favourite", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

DeleteFromFavourites(selectedItem);

listBox1.Items.Remove(selectedItem);

}

}

else

{

MessageBox.Show("Please select an item to delete.");

}

}

// Delete selected favourite place from the database

private void DeleteFromFavourites(string itemName)

{

using (MySqlConnection connection = new MySqlConnection(connectionString1))

{

try

{

connection.Open();

string query = "DELETE FROM areas WHERE name = @itemName";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@itemName", itemName);

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show($"Error deleting from favourites: {ex.Message}");

}

}

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

// Optionally, handle label click events here

}

private void button3\_Click(object sender, EventArgs e)

{

Update\_list.Visible = true;

}

private void Update\_list\_SelectedIndexChanged(object sender, EventArgs e)

{

string itemname = Update\_list.SelectedItem.ToString();

string tablename= Update\_list.SelectedItem.ToString().ToLower();

Form3 form3 = new Form3();

OpenForm4WithItemName(itemname,tablename);

}

private void OpenForm4WithItemName(string itemName, string tablename)

{

Form4 form4 = new Form4(tablename);

form4.SetItemName(itemName);

form4.Show();

this.Hide();

}

private void Form6\_Load(object sender, EventArgs e)

{

}

private void delete\_all()

{

using (MySqlConnection connection = new MySqlConnection(connectionString1))

{

try

{

connection.Open();

string query = "DELETE FROM areas";

MySqlCommand command = new MySqlCommand(query, connection);

int rowsAffected = command.ExecuteNonQuery();

MessageBox.Show($"{rowsAffected} items deleted from favourites.");

listBox1.Items.Clear(); // Clear the ListBox after deletion

}

catch (Exception ex)

{

MessageBox.Show($"Error deleting from favourites: {ex.Message}");

}

}

}

private void button4\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("Do you want to delete all items from your favourites?", "Delete All Favourites", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

delete\_all();

}

}

private void button2\_Click(object sender, EventArgs e)

{

Form3 form3=new Form3();

form3.Show();

this.Hide();

}

}

}

*4.1.6*

using System;

using System.Windows.Forms;

namespace Assignment1

{

public partial class Form7 : Form

{

public Form7()

{

InitializeComponent();

SetInfoText();

}

private void button1\_Click(object sender, EventArgs e)

{

Form3 form3 = new Form3();

form3.Show();

this.Hide();

}

private void SetInfoText()

{

labelInfo.Text = "Application Name\n" +

"JammuGems\n\n" +

"Version\n" +

"Version 1.0\n\n" +

"Description\n" +

"JammuGems is a user-friendly application designed to help you explore, manage, and favorite places in Jammu\n city. " +

"Whether you are a resident or a visitor, JammuGems offers an easy way to discover the gems of Jammu,\n from popular attractions to hidden spots.\n\n" +

"Features\n" +

"●Explore Places: Browse through a variety of places categorized for easy navigation.\n" +

"●Favorite Management: Add places to your favorites for quick access and future visits.\n" +

"●Database Integration: Powered by MySQL, ensuring reliable and efficient data management.\n\n" +

"How to Use\n" +

"●Browsing Places: Use the main menu to navigate through different categories of places.\n" +

"●Adding to Favorites: Select a place and mark it as a favorite to add it to your personal list.\n" +

"●Viewing Favorites: Access your list of favorite places from the 'Favorite Places' section.\n" +

"●Deleting Favorites: Remove places from your favorites list easily if you change your preferences.";

}

private void groupBox1\_Enter(object sender, EventArgs e)

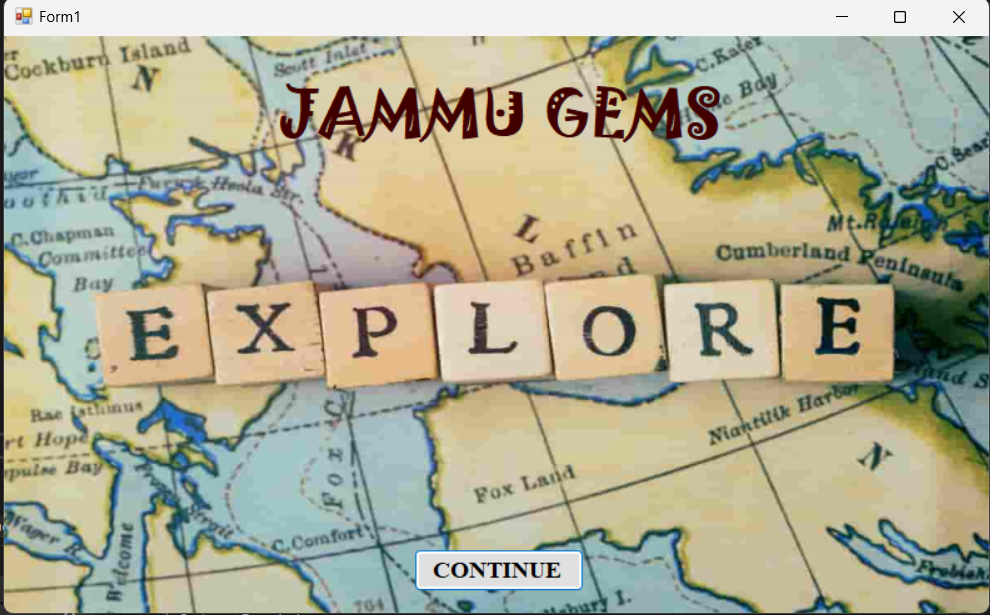
{

}

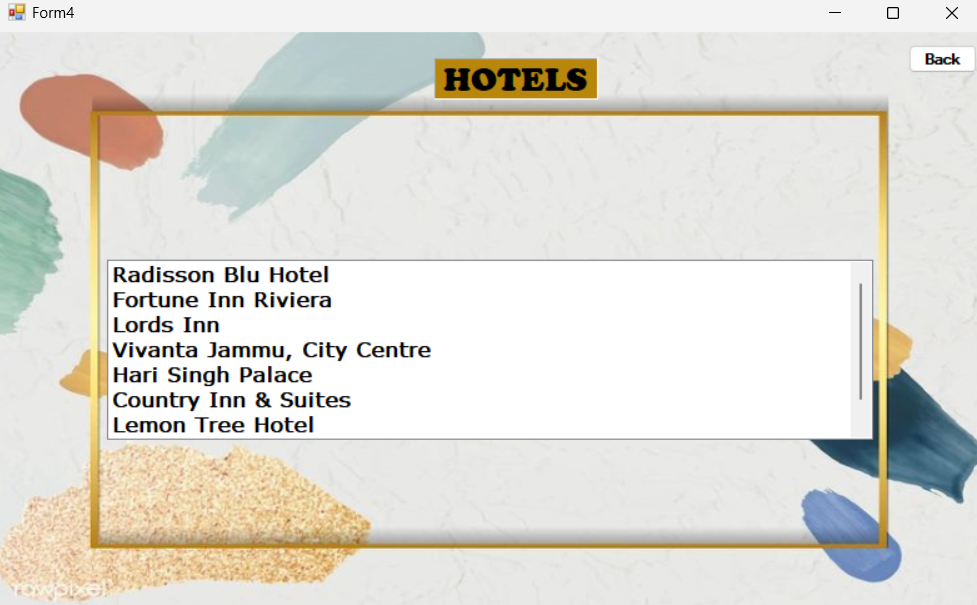
}

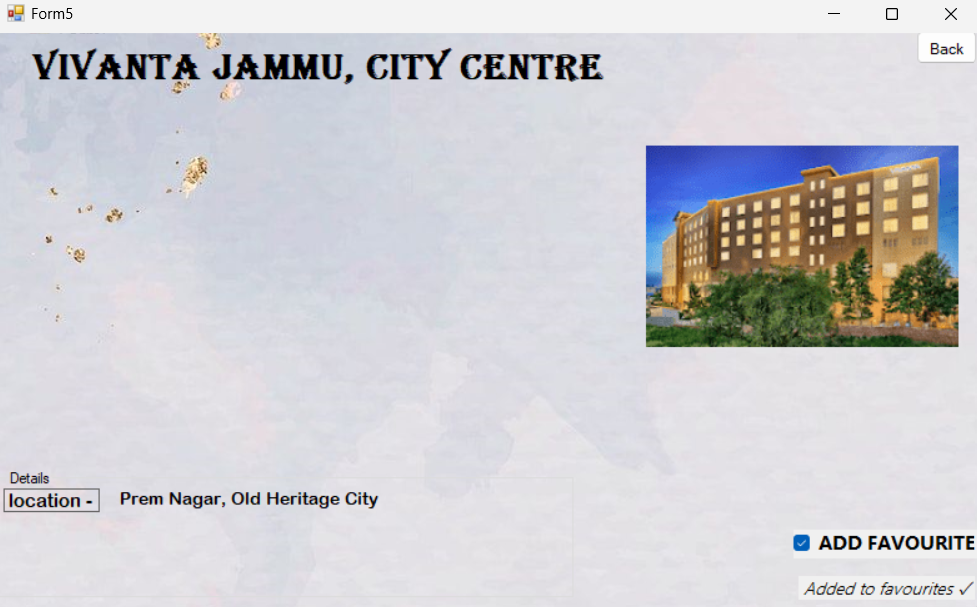
}

**4.5 Testing**



**

**

**

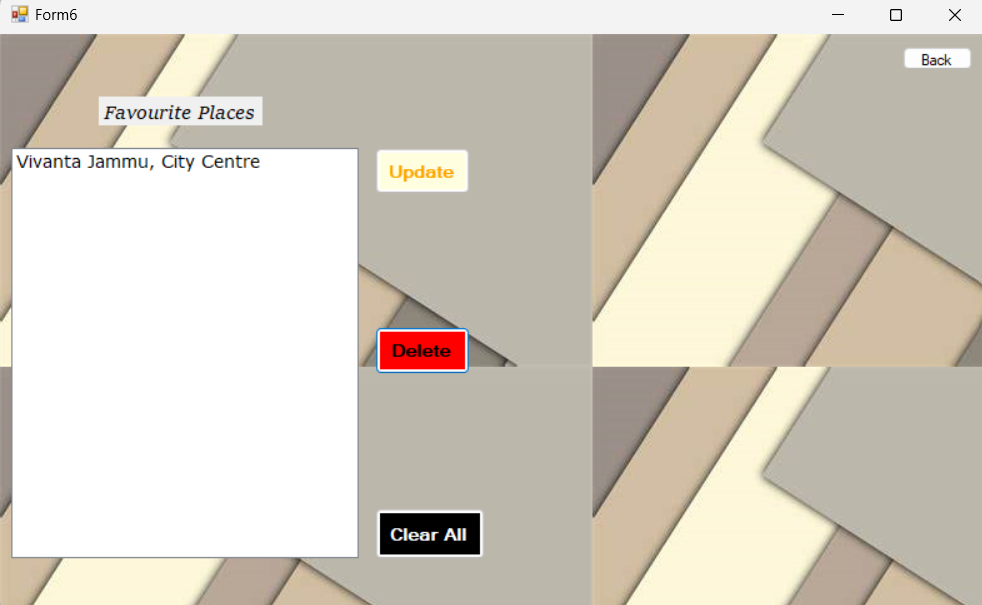
**

Figure 7: Searching data

**6. CONCLUSION**

In conclusion, the Jammu Gems project represents a significant advancement in simplifying and enhancing the exploration and management of noteworthy places within Jammu city. By leveraging a user-friendly desktop application developed using C# for the frontend and MySQL for efficient data management, this project offers a seamless and enriching experience for users to discover, categorize, and favorite various locations such as hotels, restaurants, bakeries, and temples.

One of the project's key strengths lies in its ability to cater to both residents and visitors, providing them with a comprehensive platform to explore and engage with the city's diverse offerings. This approach not only enhances the user experience but also reflects a commitment to providing a reliable and efficient platform for discovering the hidden gems of Jammu.

Overall, Jammu Gems stands as a testament to the potential of technology to enrich our lives and enhance our understanding of the world around us. Through its innovative approach to city exploration and management, this project sets a new benchmark for how cities can be experienced, inviting users to discover, appreciate, and engage with the rich tapestry of Jammu city in a meaningful and immersive manner.

**7. REFERENCES**

* 1. **"C# in Depth" by Jon Skeet**

In-depth exploration of C# features and best practices, suitable for both beginners and advanced developers.

* 1. **"Essential C# 8.0" by Mark Michaelis and Eric Lippert**

Detailed coverage of C# 8.0 features and development practices, with examples and explanations.

* 1. **"Learning MySQL: Get a Handle on Your Data" by Seyed M.M. Tahaghoghi and Hugh E. Williams**

A practical guide to learning MySQL, covering everything from basic queries to advanced data management.

* 1. **"Windows Forms Programming in C#" by Chris Sells and Michael Weinhardt**

Provides a comprehensive guide to developing Windows Forms applications using C#. It covers topics such as controls, data binding, and deployment.

* 1. **"C# GUI Programming with Windows Forms" by Eric Brown**

Developing GUI applications using Windows Forms. It covers topics such as controls, event handling, and data binding.

* **Assignment 1 -- SQL as a Data Manipulation Language:**

**Using MySQL:**

1. Create the tables for the Company database in your text, and populate with data.

2.  Create a simple desktop app to load, add and delete the data from database. [Use any language Python tk, c#, .net etc.]

3.  Create a Mini Project report for the application you have created.

**GITHUB LINK FOR PROJECT VIDEO**

http:

* **Assignment 2 -- Practice writing Queries:**

**Q1. Show the name, sales, and quota of Bill Adams**

SELECT name, sales, quota

FROM SalesReps

WHERE name = 'Bill Adams';

**Q2. List the company names and the product description of all the products each has ordered. Arrange descending by company.**

SELECT company, p.DESCRIPTION

FROM CUSTOMERS c

INNER JOIN ORDERS o ON c.CUST\_NUM = o.CUST

INNER JOIN PRODUCTS p ON o.PRODUCT = p.PRODUCT\_ID

ORDER BY c.COMPANY DESC;

**Q3. Show the total value of the inventory on hand for each product. Arrange in descending order by total value.**

SELECT PRODUCT\_ID, DESCRIPTION, PRICE \* QTY\_ON\_HAND AS

TOTAL\_VALUE

FROM PRODUCTS

BY TOTAL\_VALUE DESC;

**Q4. How many customers are there?**

SELECT COUNT(\*) AS total\_customers

FROM CUSTOMERS;

**Q5. List the offices with a target over $600,000.**

SELECT \*

FROM OFFICES

WHERE TARGET > '$600000';

**Q6. What is the average of all the salespeople?**

SELECT AVG(sales) AS average\_sales

FROM SalesReps;

**Q7. List orders over $25,000, including the name of the salesperson who took the order and the name of the customer who placed it.**

SELECT o.ORDER\_NUM, o.AMOUNT, s.name AS salesperson, c. SELECT name, sales, quota

FROM SalesReps

WHERE name = 'Bill Adams';

SELECT c.COMPANY, p.DESCRIPTION

FROM CUSTOMERS c

INNER JOIN ORDERS o ON c.CUST\_NUM = o.CUST

INNER JOIN PRODUCTS p ON o.PRODUCT = p.PRODUCT\_ID

ORDER BY c.COMPANY DESC;

SELECT PRODUCT\_ID, DESCRIPTION, PRICE \* QTY\_ON\_HAND AS

TOTAL\_VALUE

FROM PRODUCTS

BY TOTAL\_VALUE DESC;

SELECT COUNT(\*) AS total\_customers

FROM CUSTOMERS;

SELECT \*

FROM OFFICES

WHERE TARGET > '$600000';

SELECT AVG(sales) AS average\_sales

FROM SalesReps;

customer

FROM ORDERS o

INNER JOIN SalesReps s ON o.REP = s.emp\_num

COMPANY AS customer

FROM ORDERS o

INNER JOIN SalesReps s ON o.REP = s.emp\_num

INNER JOIN CUSTOMERS c ON o.CUST = c.CUST\_NUM

WHERE o.AMOUNT > '$25000';

**Q8. How many sales offices have salespeople who are over quota?**

SELECT COUNT(DISTINCT OFFICE) AS num\_offices

FROM SalesReps

WHERE sales > quota;

**Q9. Show the name, sales, and office for each salesperson. Order by increasing sales.**

SELECT name, sales, rep\_office AS office

FROM SalesReps

ORDER BY sales;

**Q10. List all the companies who have ordered any size widget, and the widget they ordered**

SELECT DISTINCT c.COMPANY, p.DESCRIPTION AS widget\_ordered

FROM CUSTOMERS c

INNER JOIN ORDERS o ON c.CUST\_NUM = o.CUST

INNER JOIN PRODUCTS p ON o.PRODUCT = p.PRODUCT\_ID

WHERE p.DESCRIPTION LIKE '%Widget%';

**Q11. List the city, region, and amount that sales are over/under target for each office.**

SELECT CITY, REGION, (SALES - TARGET) AS variance

FROM OFFICES;

**Q12. What is the total number of each part that has been ordered?**

SELECT PRODUCT\_ID, SUM(QTY) AS total\_ordered

FROM ORDERS

GROUP BY PRODUCT\_ID;

**Q13. List the salespeople, the city they work in, and the manager of the office in which they work**

SELECT s.name AS salesperson, o.CITY, o.MGR AS manager

FROM SalesReps s

INNER JOIN OFFICES o ON s.rep\_office = o.OFFICE;

**Q14. List all orders showing order number, amount, customer name, and the customer’s credit limit where the order was greater than $20,000.**

SELECT o.ORDER\_NUM, o.AMOUNT, c.COMPANY AS customer, c.CREDIT\_LIMIT

FROM ORDERS o

INNER JOIN CUSTOMERS c ON o.CUST = c.CUST\_NUM

WHERE o.AMOUNT > '$20000';

**Q15. Are there any customers who are over their credit limit? If so, list the customer, the total amount the customer has on order, and the credit lim**

SELECT c.COMPANY AS customer, SUM(o.AMOUNT) AS total\_amount\_on\_order, c.CREDIT\_LIMIT

FROM ORDERS o

INNER JOIN CUSTOMERS c ON o.CUST = c.CUST\_NUM

GROUP BY c.COMPANY, c.CREDIT\_LIMIT

HAVING SUM(o.AMOUNT) > c.CREDIT\_LIMIT;

**Q16. List the salespeople with a higher quota than their manager.**

SELECT s.name AS salesperson, s.quota AS salesperson\_quota, o.MGR AS manager, sr.quota AS manager\_quota

FROM SalesReps s

INNER JOIN OFFICES o ON s.rep\_office = o.OFFICE

INNER JOIN SalesReps sr ON o.MGR = sr.emp\_num

WHERE s.quota > sr.quota;

**Q17. List salespeople who work in different offices than their managers, show the name and office where each work.**

SELECT s.name AS salesperson, s.rep\_office AS salesperson\_office, o.OFFICE AS manager\_office

FROM SalesReps s

INNER JOIN OFFICES o ON s.rep\_office != o.OFFICE AND s.manager = o.MGR;

**Q18. What is the total order size for each salesperson? Order by increasing sales.**

SELECT s.name AS salesperson, SUM(o.QTY) AS total\_order\_size

FROM ORDERS o

INNER JOIN SalesReps s ON o.REP = s.emp\_num

GROUP BY s.name

ORDER BY total\_order\_size;

**Q19. List all the customers whose sales representative is a manager. Arrange increasing by company.**

SELECT c.COMPANY AS customer, s.name AS sales\_rep\_name

FROM CUSTOMERS c

INNER JOIN SalesReps s ON c.CUST\_REP = s.emp\_num AND s.title LIKE '%Manager%'

ORDER BY c.COMPANY;

**Q20. What is the total order size for each salesperson whose orders total more than $30,000?**

SELECT s.name AS salesperson, SUM(o.QTY) AS total\_order\_size

FROM ORDERS o

INNER JOIN SalesReps s ON o.REP = s.emp\_num

GROUP BY s.name

HAVING SUM(o.AMOUNT) > '$30000';

**Q21. List the offices where the sales target for the office exceeds the sum of the individual salespeople’s quotas.**

SELECT o.OFFICE, o.TARGET AS office\_target, SUM(s.quota) AS total\_quotas

FROM OFFICES o

INNER JOIN SalesReps s ON o.OFFICE = s.rep\_office

GROUP BY o.OFFICE, o.TARGET

HAVING SUM(s.quota) > o.TARGET;

**Q22. List the salespeople whose quotas are equal to or higher than the target of the Atlanta sales office.**

SELECT s.name AS salesperson, s.quota, o.TARGET AS atlanta\_target

FROM OFFICES o

INNER JOIN SalesReps s ON o.OFFICE = s.rep\_office AND o.CITY = 'Atlanta'

WHERE s.quota >= o.TARGET;

**Q23. List the salespeople who do not work in offices managed by Larry Fitch (employee 108).**

SELECT s.name AS salesperson, o.MGR AS office\_manager

FROM SalesReps s

INNER JOIN OFFICES o ON s.rep\_office = o.OFFICE

WHERE o.MGR != '108';

**Q24. List the products for which an order of $25,000 or more has been received.**

SELECT DISTINCT p.DESCRIPTION

FROM ORDERS o

INNER JOIN PRODUCTS p ON o.PRODUCT = p.PRODUCT\_ID

WHERE o.AMOUNT >= '$25000';

**Q25. List the companies who placed an order with a sales rep that is not the sales rep that usually calls on them. Include the names of the salesreps, indicating by attribute name who took the order.**

SELECT DISTINCT c.COMPANY, s1.name AS usual\_sales\_rep, s2.name AS sales\_rep\_for\_order

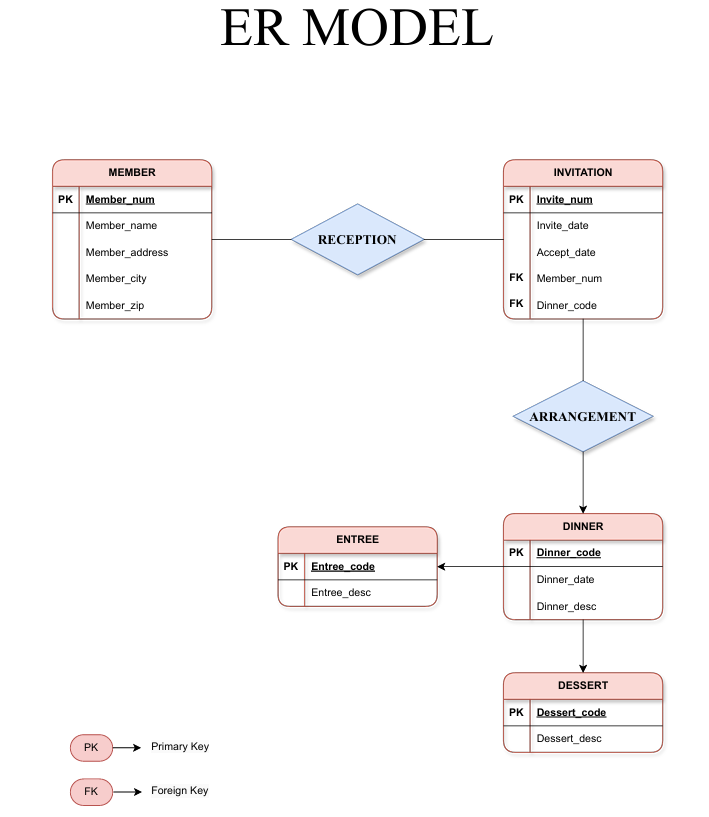
FROM CUSTOMERS c

INNER JOIN SalesReps s1 ON c.CUST\_REP = s1.emp\_num

INNER JOIN ORDERS o ON c.CUST\_NUM = o.CUST

INNER JOIN SalesReps s2 ON o.REP = s2.emp\_num AND s1.emp\_num != s2.emp\_num;

* **Assignment 3 -- Database Design Assignment:**
* **Part I -- The ER Diagram**
* **The ER Diagram:**



* Relationships:

1. **Member - Invitation**1 member = multiple invitations.Each invitation = 1 member.Cardinality: 1 Member: N Invitations (1: N)Participation: Total for Invitations, Partial for Members (Every invitation must be associated with a member, but not every member might have an invitation.)
2. **Dinner - Entree**1 dinner = 1 entree.Each entree = multiple dinners.Cardinality: 1 Entree: N Dinners (1: N)Participation: Total for Dinners, Partial for Entrees (Every dinner must have an entree, but not every entree must be part of a dinner.)
3. **Dinner - Dessert**1 dinner = 1 dessert.Each dessert = multiple dinners.Cardinality: 1 Dessert: N Dinners (1: N)Participation: Total for Dinners, Partial for Desserts (Every dinner must have a dessert, but not every dessert must be part of a dinner.)
4. **Invitation - Dinner**1 invitation = 1 dinner.Each dinner = multiple invitations.Cardinality: 1 Dinner: N Invitations (1: N)Participation: Total for Invitations, Partial for Dinners (Every invitation must be associated with a dinner, but not every dinner might have an invitation.)

* Mapping relational schemas:

1. **MEMBER**

CREATE TABLE Member (

Member\_num INT PRIMARY KEY,

Member\_name VARCHAR(50),

Member\_address VARCHAR(100),

Member\_city VARCHAR(50),

Member\_zip CHAR(5)

);

1. **INVITATION**

CREATE TABLE Invitation (

Invite\_num INT PRIMARY KEY,

Invite\_date DATE,

Accept\_date DATE,

Dinner\_date DATE,

Dinner\_attend CHAR(1),

Member\_num INT,

Dinner\_code INT,

FOREIGN KEY (Member\_num) REFERENCES Member(Member\_num),

FOREIGN KEY (Dinner\_code) REFERENCES Dinner(Dinner\_code)

);

1. **DINNER**

CREATE TABLE Dinner (

Dinner\_code INT PRIMARY KEY,

Dinner\_desc VARCHAR(50)

);

1. **ENTRÉE**

CREATE TABLE Entree (

Entree\_code INT PRIMARY KEY,

Entree\_desc VARCHAR(50)

);

1. **DESSERT**

CREATE TABLE Dessert (

Dessert\_code INT PRIMARY KEY,

Dessert\_desc VARCHAR(50)

);

1. **DINNER\_ENTREE**

CREATE TABLE Dinner\_Entree (

Dinner\_code INT,

Entree\_code INT,

PRIMARY KEY (Dinner\_code, Entree\_code),

FOREIGN KEY (Dinner\_code) REFERENCES Dinner(Dinner\_code),

FOREIGN KEY (Entree\_code) REFERENCES Entree(Entree\_code)

);

1. **DINNER\_DESSERT**

CREATE TABLE Dinner\_Dessert (

Dinner\_code INT,

Dessert\_code INT,

PRIMARY KEY (Dinner\_code, Dessert\_code),

FOREIGN KEY (Dinner\_code) REFERENCES Dinner(Dinner\_code),

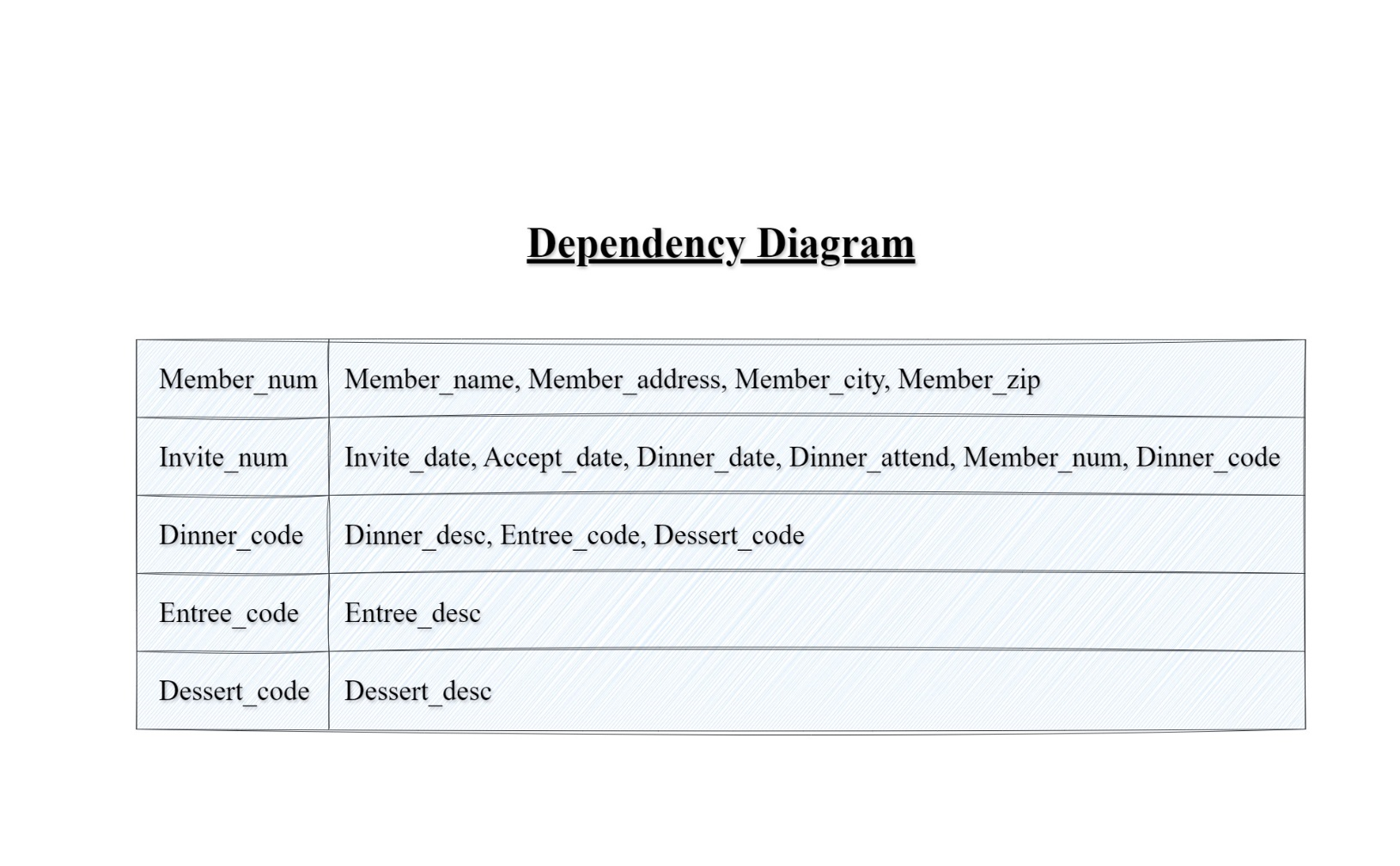
FOREIGN KEY (Dessert\_code) REFERENCES Dessert(Dessert\_code)

);

* **Part II -- Normalization of the Universal Relation**
* **Given the above structure, draw its dependency diagram. Label all transitive and/or partial dependencies.**

**Functional Dependencies**

**1. Member\_num → Member\_name, Member\_address, Member\_city, Member\_zip2. Invite\_num → Invite\_date, Accept\_date, Dinner\_date, Dinner\_attend, Member\_num, Dinner\_code3. Dinner\_code → Dinner\_desc, Entree\_code, Dessert\_code4. Entree\_code → Entree\_desc5. Dessert\_code → Dessert\_desc**



* **Partial Dependencies:**Invite\_num → Invite\_date, Accept\_date, Dinner\_date, Dinner\_attend, Member\_num, Dinner\_codeDinner\_code → Dinner\_desc, Entree\_code, Dessert\_code
* **Transitive Dependencies:**Dinner\_code → Entree\_code → Entree\_descDinner\_code → Dessert\_code → Dessert\_desc
* **Normalize the diagram above to produce dependency diagrams in are in 3NF.**

**1.** **First Normal Form (1NF)**

The relation is already in 1NF as all attributes have atomic values.**2. Second Normal Form (2NF)**

A relation is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the primary key.

* Identify the primary keys and remove partial dependencies:

CREATE TABLE Member (

Member\_num INT PRIMARY KEY,

Member\_name VARCHAR(50),

Member\_address VARCHAR(100),

Member\_city VARCHAR(50),

Member\_zip CHAR(5)

);

CREATE TABLE Invitation (

Invite\_num INT PRIMARY KEY,

Invite\_date DATE,

Accept\_date DATE,

Dinner\_date DATE,

Dinner\_attend CHAR(1),

Member\_num INT,

Dinner\_code INT,

FOREIGN KEY (Member\_num) REFERENCES Member(Member\_num),

FOREIGN KEY (Dinner\_code) REFERENCES Dinner(Dinner\_code)

);

CREATE TABLE Dinner (

Dinner\_code INT PRIMARY KEY,

Dinner\_desc VARCHAR(50)

);

CREATE TABLE Entree (

Entree\_code INT PRIMARY KEY,

Entree\_desc VARCHAR(50)

);

CREATE TABLE Dessert (

Dessert\_code INT PRIMARY KEY,

Dessert\_desc VARCHAR(50)

);

**3. Third Normal Form (3NF)**

A relation is in 3NF if it is in 2NF and there are no transitive dependencies. Remove transitive dependencies by creating separate tables for Entree and Dessert.

CREATE TABLE Dinner (

Dinner\_code INT PRIMARY KEY,

Dinner\_desc VARCHAR(50)

);

CREATE TABLE Dinner\_Entree (

Dinner\_code INT,

Entree\_code INT,

PRIMARY KEY (Dinner\_code, Entree\_code),

FOREIGN KEY (Dinner\_code) REFERENCES Dinner(Dinner\_code),

FOREIGN KEY (Entree\_code) REFERENCES Entree(Entree\_code)

);

CREATE TABLE Dinner\_Dessert (

Dinner\_code INT,

Dessert\_code INT,

PRIMARY KEY (Dinner\_code, Dessert\_code),

FOREIGN KEY (Dinner\_code) REFERENCES Dinner(Dinner\_code),

FOREIGN KEY (Dessert\_code) REFERENCES Dessert(Dessert\_code)

);

