

CITY GUIDE PROJECT



A PROJECT REPORT

Submitted by

SINDHUJA S (2303811710422150)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)**

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “**CITY GUIDE PROJECT**” is the bonafide work of **SINDHUJA S (2303811710422150)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING
Mr.MALARMANNAN A, M.E.,
ASSISTANT PROFESSOR

SIGNATURE

Mr. A. Malarmannan, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on ...06 / 12 / 2024.....

CGB1201-JAVA PROGRAMMING
Mr. M. MOHANAN, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

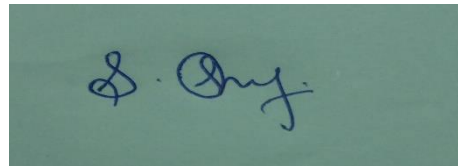
INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Mr.R. KARTHICK, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8138-SCE, TRICHY.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**CITY GUIDE PROJECT**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

A rectangular box containing a handwritten signature in blue ink, which appears to read 'S. Sindhuja'.

Signature

SINDHUJA S

Place: Samayapuram

Date: 06 / 12 / 2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This project is a Java-based City Guide application built using AWT (Abstract Window Toolkit). It is designed to solve a real-time problem of providing users with an interactive interface to explore, search, and manage information about cities and their popular restaurants along with their cuisine types. The application supports functionalities such as displaying restaurant details in a selected city, finding cities in the guide, and adding new restaurants to the database dynamically. The City Guide Project is an application designed to assist users in exploring and discovering local restaurants across various cities. The primary objective of this project is to offer personalized recommendations based on user preferences and their current location. It is designed to create a user-friendly application that serves as a comprehensive platform for discovering and exploring local restaurants.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>This project is a Java-based City Guide application built using AWT (Abstract Window Toolkit). It is designed to solve a real-time problem of providing users with an interactive interface to explore, search, and manage information about cities and their popular restaurants along with their cuisine types. The application supports functionalities such as displaying restaurant details in a selected city, finding cities in the guide, and adding new restaurants to the database dynamically. The City Guide Project is an application designed to assist users in exploring and discovering local restaurants across various cities. The primary objective of this project is to offer personalized recommendations based on user preferences and their current location. It is designed to create a user-friendly application that serves as a comprehensive platform for discovering and exploring local restaurants.</p>	<p>PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3</p>	<p>PSO1 -3 PSO2 -3 PSO3 -3</p>

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	1
2	PROJECT METHODOLOGY	4
	2.1 Proposed Work	4
	2.2 Block Diagram	4
3	MODULE DESCRIPTION	5
	3.1 User Interface Module	5
	3.2 City Management Module	5
	3.3 Restaurant Management Module	5
	3.4 Data Validation Module	5
	3.5 Predefined Data Initialization Module	5
4	CONCLUSION & FUTURE SCOPE	6
	4.1 Conclusion	6
	4.2 Future Scope	6
	APPENDIX A (SOURCE CODE)	7
	APPENDIX B (SCREENSHOTS)	12
	REFERENCES	13

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of the City Guide Project is to develop an interactive application that allows users to explore local cities and restaurants effortlessly. The project aims to provide a user-friendly platform where users can search, add, and display cities and their associated restaurants, categorized by cuisine types. It seeks to create an organized system for managing city and restaurant data using efficient data structures like HashMap for fast retrieval and easy management. To develop a simple, interactive city guide application that allows users to explore local restaurants based on city names. The platform should be easy to navigate, enabling users to search, add and display cities and restaurants effortlessly.

1.2 Overview

The City Guide Project is a Java-based application designed to serve as a comprehensive tool for exploring cities and their local restaurants. The project provides a user-friendly command-line interface that allows users to perform various operations, such as viewing all restaurants in a city, adding new cities and their associated restaurants, and searching for specific cities. Each restaurant is categorized by its cuisine type, enabling users to make informed decisions based on their preferences. Its design and implementation emphasize simplicity, efficiency, and scalability, making it a versatile tool for users seeking to explore dining options within different cities.

1.3 Java Programming Concepts

Object-Oriented Programming (OOP):

- **Encapsulation:** Data such as city names, restaurant names, and cuisine types are encapsulated within classes and methods, ensuring they are protected from unauthorized access.
- **Abstraction:** The project abstracts complex logic (e.g., managing city and restaurant data) into methods like `displayCity`, `addRestaurantToCity`.
- **Polymorphism:** Action listeners in the GUI implement polymorphism by dynamically responding to various user actions like adding or removing cities.

Java GUI Programming:

- **Abstract Window Toolkit (AWT):** Components like `TextField`, `Button`, `Label`, and `TextArea` provide a simple graphical user interface. `FlowLayout` ensures a clean layout for arranging the components dynamically.
- **Event Handling:** Action listeners are implemented to handle user interactions, such as button clicks for displaying city data or adding restaurants.
- **Window Management:** A `WindowAdapter` is used to handle the window-closing event, ensuring proper termination of the application.

Exception Handling:

Handling edge cases such as Input errors (e.g., entering an empty or invalid city name). Preventing addition of more than the allowed number of restaurants in a city.

Collections Framework:

Map Interface: `HashMap` is used extensively to store cities as keys and their corresponding restaurant data as values. Another `HashMap` is nested within to manage restaurant names and cuisine types for each city.

Java Core Features:

- **String Manipulation:** The `StringBuilder` class is used to efficiently concatenate and format output for displaying restaurant data.
- **Control Structures:** Conditional statements (if-else) for decision-making based on user input (e.g., checking if a city exists). Loops for iterating through restaurants in a city.

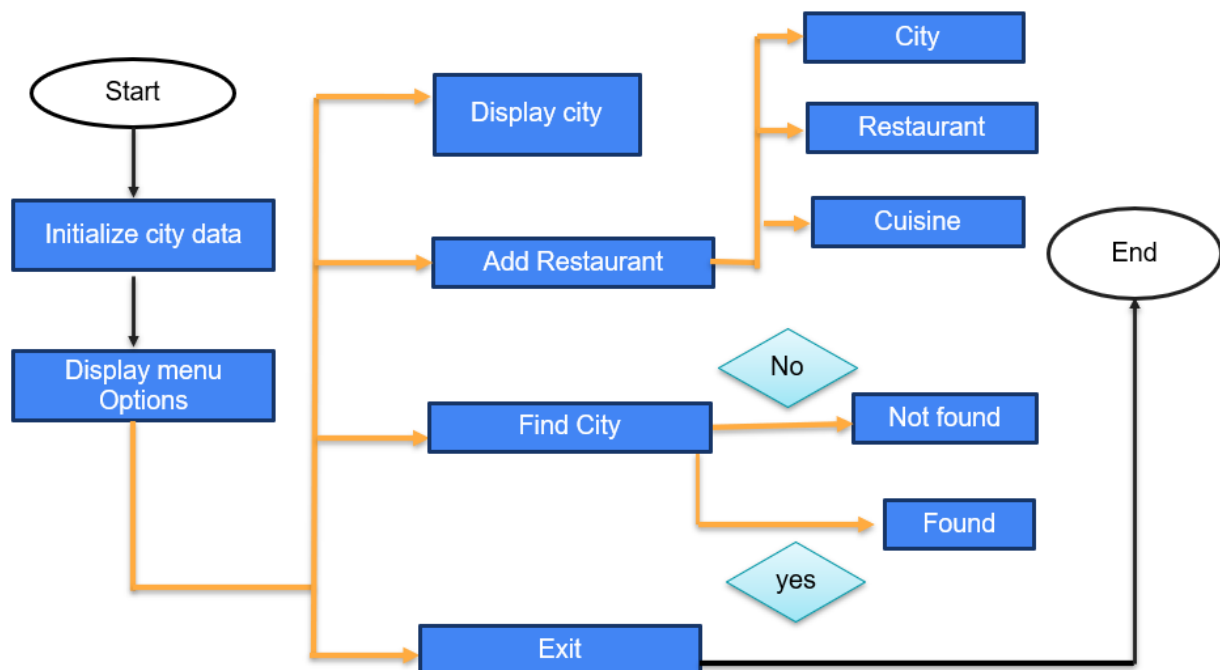
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the City Guide Project focuses on developing a user-friendly application that enables users to explore and manage local restaurant information in various predefined cities. The application utilizes Java's Abstract Window Toolkit (AWT) to provide an intuitive graphical user interface (GUI) where users can interact seamlessly through input fields, buttons, and a text area for displaying results and feedback. A predefined database of cities, including Chennai, Coimbatore, Madurai, and Trichy, is initialized with their respective restaurants and cuisine types, allowing users to view, search, or modify the data. Users can add new restaurants to existing cities, check the existence of a city from the database.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 User Interface Module

Input fields for entering city names, restaurant names, and cuisine types. Buttons for performing actions such as displaying city details, adding a restaurant and finding a city. Output area for displaying results, error messages, or success messages.

3.2 City Management Module

Predefined city initialization with restaurant data. Methods to add restaurants dynamically. Validation to ensure cities are unique and properly managed in the application.

3.3 Restaurant Management Module

Methods for adding restaurants to existing cities. To manage restaurant details within each city. Capability to view all restaurants and their cuisine types in a selected city.

3.4 Data Validation Module

Validating user inputs, such as checking if a city exists before adding a restaurant. Handling edge cases, such as duplicate restaurant names or missing data. Displaying appropriate error or success messages to guide user actions.

3.5 Predefined Data Initialization Module

Initializing cities like Chennai, Coimbatore, Madurai, and Trichy with their respective restaurants and cuisines. Ensuring the predefined data serves as a functional starting point for user exploration. To populate the application with initial data for predefined cities and restaurants.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The **City Guide Project** successfully demonstrates the integration of Java programming concepts to create a user-friendly application for exploring and managing restaurant information in various cities. By utilizing the AWT framework for the graphical user interface, the project provides a seamless and interactive experience for users to search, add, and manage city and restaurant data dynamically. The effective use of collections, object-oriented programming principles, and event handling ensures that the application is both efficient and scalable.

The project achieves its primary objective of offering a centralized platform for discovering restaurants and their cuisines in predefined and user-added cities. Through the effective use of **core Java concepts** such as collections, event handling, and object-oriented programming, the application ensures scalability and efficiency. The implementation of HashMap facilitates fast and organized data retrieval, while AWT components like TextField, TextArea, and Button enhance usability.

4.2 FUTURE SCOPE

The City Guide project has immense potential for future development, enabling it to become a comprehensive and user-friendly platform for city and restaurant exploration. One significant improvement would be integrating a database to replace the in-memory storage, ensuring scalability and data persistence. The user interface could be upgraded from AWT to more advanced frameworks like JavaFX, Swing, or web-based front ends, offering a visually appealing and modern design. Incorporating GPS and location-based services, such as Google Maps API, could provide real-time navigation and proximity-based recommendations.

APPENDIX A

(SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import java.util.HashMap;
import java.util.Map;

public class CityGuideAWT {

    // Map to store cities and their restaurants with cuisine types
    private static Map<String, HashMap<String, String>> cityData = new
HashMap<>();

    public static void main(String[] args) {
        // Adding initial predefined city data
        addInitialCityData();

        // Create the AWT frame
        Frame frame = new Frame("City Guide");
        frame.setSize(600, 600);
        frame.setLayout(new FlowLayout());

        // Components for the GUI
        Label cityLabel = new Label("City:");
        TextField cityField = new TextField(20);
        Button displayButton = new Button("Display City");
        Button addRestaurantButton = new Button("Add Restaurant");
        Button findCityButton = new Button("Find City");
        Button exitButton = new Button("Exit");
        TextArea outputArea = new TextArea(20, 50);
        outputArea.setEditable(false);

        // Add components to the frame
        frame.add(cityLabel);
        frame.add(cityField);
        frame.add(displayButton);
        frame.add(addRestaurantButton);
        frame.add(findCityButton);
        frame.add(exitButton);
        frame.add(outputArea);

        // Action for displaying city data
        displayButton.addActionListener(e -> {
            String city = cityField.getText().trim();
            if (!city.isEmpty()) {
```

```

        String result = displayCity(city);
        outputArea.setText(result);
    } else {
        outputArea.setText("Please enter a city name.");
    }
});

// Action for adding a restaurant to a city
addRestaurantButton.addActionListener(e -> {
    Frame addRestaurantFrame = new Frame("Add Restaurant");
    addRestaurantFrame.setSize(400, 300);
    addRestaurantFrame.setLayout(new GridLayout(4, 2));

    Label cityNameLabel = new Label("City:");
    TextField cityNameField = new TextField(20);
    Label restaurantNameLabel = new Label("Restaurant:");
    TextField restaurantNameField = new TextField(20);
    Label cuisineLabel = new Label("Cuisine:");
    TextField cuisineField = new TextField(20);
    Button submitButton = new Button("Submit");

    addRestaurantFrame.add(cityNameLabel);
    addRestaurantFrame.add(cityNameField);
    addRestaurantFrame.add(restaurantNameLabel);
    addRestaurantFrame.add(restaurantNameField);
    addRestaurantFrame.add(cuisineLabel);
    addRestaurantFrame.add(cuisineField);
    addRestaurantFrame.add(submitButton);

    // Submit button action
    submitButton.addActionListener(event -> {
        String city = cityNameField.getText().trim();
        String restaurant = restaurantNameField.getText().trim();
        String cuisine = cuisineField.getText().trim();

        if (!city.isEmpty() && !restaurant.isEmpty() && !cuisine.isEmpty()) {
            addRestaurant(city, restaurant, cuisine);
            outputArea.setText("Added " + restaurant + " (" + cuisine + ") to " + city
+ ".");

            addRestaurantFrame.dispose();
        } else {
            outputArea.setText("All fields are required.");
        }
    });

    addRestaurantFrame.setVisible(true);
});

// Action for finding a city
findCityButton.addActionListener(e -> {

```

```

String city = cityField.getText().trim();
if (!city.isEmpty()) {
    String result = findCity(city);
    outputArea.setText(result);
} else {
    outputArea.setText("Please enter a city name.");
}
});

// Exit button action
exitButton.addActionListener(e -> System.exit(0));

// Close the frame properly
frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
});

// Make the frame visible
frame.setVisible(true);
}

// Method to display restaurants and cuisine types in a city
public static String displayCity(String city) {
    if (cityData.containsKey(city)) {
        StringBuilder result = new StringBuilder("Restaurants in " + city + ":\n");
        HashMap<String, String> restaurants = cityData.get(city);
        for (Map.Entry<String, String> entry : restaurants.entrySet()) {
            result.append("- ").append(entry.getKey()).append(" (Cuisine:
").append(entry.getValue()).append(")\n");
        }
        return result.toString();
    } else {
        return "City not found in the guide.";
    }
}

// Method to add a restaurant to a city
public static void addRestaurant(String city, String restaurant, String cuisine) {
    cityData.putIfAbsent(city, new HashMap<>());
    cityData.get(city).put(restaurant, cuisine);
}

// Method to find a city
public static String findCity(String city) {
    if (cityData.containsKey(city)) {
        return "City " + city + " found in the guide.";
    } else {
        return "City " + city + " not found.";
    }
}

```

```

    }
}

```

```

// Method to add initial predefined cities (Tamil Nadu cities) and their
// restaurants

```

```

public static void addInitialCityData() {
    HashMap<String, String> chennaiRestaurants = new HashMap<>();
    chennaiRestaurants.put("Saravana Bhavan", "South Indian");
    chennaiRestaurants.put("Anjappar", "Chettinad");
    chennaiRestaurants.put("The Raintree", "Multi-Cuisine");
    chennaiRestaurants.put("Peshawri", "North Indian");
    chennaiRestaurants.put("Murugan Idli Shop", "South Indian");
    chennaiRestaurants.put("Barbeque Nation", "Grill");
    chennaiRestaurants.put("Dakshin", "Authentic South Indian");
    chennaiRestaurants.put("Cream Centre", "Vegetarian");
    chennaiRestaurants.put("Absolute Barbecue", "Grill & Buffet");
    chennaiRestaurants.put("Sandy's Chocolate Laboratory", "Desserts");
    cityData.put("Chennai", chennaiRestaurants);

    HashMap<String, String> coimbatoreRestaurants = new HashMap<>();
    coimbatoreRestaurants.put("Sree Annapoorna", "South Indian");
    coimbatoreRestaurants.put("Hari Bhavanam", "South Indian");
    coimbatoreRestaurants.put("The Bombay Brasserie", "Indian");
    coimbatoreRestaurants.put("Shree Sangeetha", "South Indian");
    coimbatoreRestaurants.put("Almora", "North Indian");
    coimbatoreRestaurants.put("On The Go", "Continental");
    coimbatoreRestaurants.put("Geetha Cafe", "Snacks and Drinks");
    coimbatoreRestaurants.put("Dindigul Thalappakatti", "Biryani");
    coimbatoreRestaurants.put("Kove", "Fusion Indian");
    coimbatoreRestaurants.put("Latitude Restaurant", "Multi-Cuisine");
    cityData.put("Coimbatore", coimbatoreRestaurants);

    HashMap<String, String> maduraiRestaurants = new HashMap<>();
    maduraiRestaurants.put("Murugan Idli Shop", "South Indian");
    maduraiRestaurants.put("Chidambaram", "Chettinad");
    maduraiRestaurants.put("Kumar Mess", "South Indian");
    maduraiRestaurants.put("Sree Saravana Bhavan", "South Indian");
    maduraiRestaurants.put("The Bell Hotel", "Multi-Cuisine");
    maduraiRestaurants.put("Muniyandi Vilas", "Local Chettinad");
    maduraiRestaurants.put("Temple City", "Vegetarian");
    maduraiRestaurants.put("Meenakshi Bhavan", "Authentic South Indian");
    maduraiRestaurants.put("Zaitoon", "Arabian Cuisine");
    maduraiRestaurants.put("Buhari", "Biryani");
    cityData.put("Madurai", maduraiRestaurants);

    HashMap<String, String> trichyRestaurants = new HashMap<>();
    trichyRestaurants.put("Sree Annapoorna", "South Indian");
    trichyRestaurants.put("Karaikudi Chettinad", "Chettinad");
    trichyRestaurants.put("Hotel Canara", "South Indian");
    trichyRestaurants.put("Shree Sangeetha", "South Indian");
}

```

```
trichyRestaurants.put("Banana Leaf", "South Indian");
trichyRestaurants.put("The Residency", "Fine Dining");
trichyRestaurants.put("Rasika", "North Indian");
trichyRestaurants.put("Ebony", "Buffet");
trichyRestaurants.put("Café Coffee Day", "Cafe");
trichyRestaurants.put("Thalappakatti Biryani", "Biryani");
cityData.put("Trichy", trichyRestaurants);
}
}
```

APPENDIX B

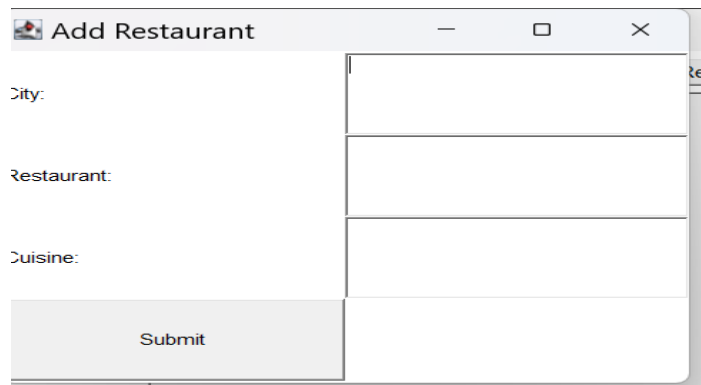
(SCREENSHOTS)

(CITY GUIDE PROJECT)

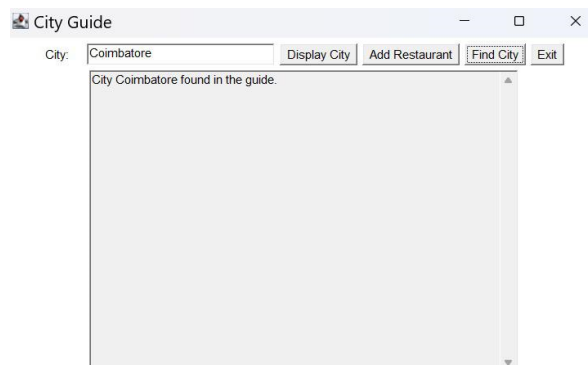
DISPLAY CITY



ADD RESTAURANT



FIND CITY



REFERENCES

1. *"Head First Java"* by Kathy Sierra and Bert Bates – A beginner-friendly guide to Java programming.
2. GeeksforGeeks – Tutorials and articles on Java concepts and implementation.
3. CodeWithHarry Java Course – Beginner to advanced Java concepts.
4. University syllabus guides for Object-Oriented Programming, GUI Development, and Software Development practices.
5. Java Collections Framework Guide – For working with data structures like HashMap and ArrayList.