# PARCEL TRACKING SYSTEM

**A PROJECT REPORT**

***Submitted by***

**JEYA SURIYA S (8115U23EC041)**

***in partial fulfillment of requirements for the award of the course***

**EGB1201 - JAVA PROGRAMMING**

***in***

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING**

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

**SAMAYAPURAM – 621 112**

**DECEMBER - 2024**

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

**SAMAYAPURAM – 621 112**

## BONAFIDE CERTIFICATE

Certified that this project report on **"VISITOR MANAGEMENT SYSTEM"** is the bonafide work of **JEYA SURIYA S (8115U23EC041)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

[illegible]

**SIGNATURE**
Dr. T. M. NITHYA, M.E.,Ph.D.,
**HEAD OF THE DEPARTMENT**
ASSOCIATE PROFESSOR
Department of CSE
K.Ramakrishnan College of Engineering (Autonomous)
Samayapuram–621112.

[illegible]

**SIGNATURE**
Mr.V.KUMARARAJA, M.E.,(Ph.D.,),
**SUPERVISOR**
ASSISTANT PROFESSOR
Department of CSE
K.Ramakrishnan College of Engineering (Autonomous)
Samayapuram–621112.

Submitted for the viva-voce examination held on 06/12/24

[illegible]
INTERNAL EXAMINER

[illegible]
EXTERNAL EXAMINER

# DECLARATION

I declare that the project report on **"PARCEL TRACKING SYSTEM"** 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 **EGB1201 - JAVA PROGRAMMING.**

.

**Signature**

JEYA SURIYA S

Place: Samayapuram

Date:

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Engineering (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. D. SRINIVASAN, B.E, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. M. NITHYA, 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.V.KUMARARAJA, M.E., (Ph.D.,),** 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 achieve a prominent position among the top technical institutions.

## MISSION OF THE INSTITUTION

- M1: To bestow standard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.
- M3: To provide education for developing high-quality professionals to transform the society.

## VISION OF DEPARTMENT

To create eminent professionals of Computer Science and Engineering by imparting quality education.

## MISSION OF DEPARTMENT

**M1**: To provide technical exposure in the field of Computer Science and Engineering through state of the art infrastructure and ethical standards.

**M2**: To engage the students in research and development activities in the field of Computer Science and Engineering.

**M3**: To empower the learners to involve in industrial and multi-disciplinary projects for addressing the societal needs.

## PROGRAM EDUCATIONAL OBJECTIVES

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

## PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.

- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

## 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

The **Package Tracking System** is a Java-based desktop application designed to simplify and automate the management of package details. Using the Abstract Window Toolkit (AWT), the system provides a user-friendly interface to add, view, and search for packages by their tracking ID.

The application ensures data accuracy through input validation for critical fields like pincode (six digits), mobile number (starting with '91'), and email address (basic format validation). Package details are stored efficiently using a HashMap, enabling quick retrieval and seamless management.

This project demonstrates the application of Java programming principles, such as modularity and error handling, to create a reliable and scalable solution. Designed for small-scale organizations, the system offers potential for future enhancements, including database integration and real-time tracking support.

## ABSTRACT WITH POs AND PSOs MAPPING

| ABSTRACT | POs MAPPED | PSOs MAPPED |
|---|---|---|
| The **Package Tracking System** is a Java-based desktop application designed to manage package tracking efficiently. Developed using Java AWT, the system provides a user-friendly interface for performing tasks such as adding, viewing, and searching packages. The project incorporates modules for handling essential details such as tracking ID, pincode, mobile number, email, and address.<br><br>This system aims to simplify package management while ensuring data validation for fields like mobile numbers (starting with '91'), email addresses, and pincodes. Real-time data updates and error handling enhance its robustness. The project demonstrates the application of object-oriented programming principles and modularity in addressing real-world challenges. | PO 1 - 3<br>PO 2 - 3<br>PO 3 - 3<br>PO 5 - 3<br>PO 6 - 2<br>PO 8 - 3<br>PO 9 - 2<br>PO 12 - 3 | PSO 1 - 3<br>PSO 2 - 3 |

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

# TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

## 1.1 Objective

The primary objective of the **Package Tracking System** is to create an efficient, user-friendly application for managing package details in real-time. The system provides functionalities for adding, viewing, and searching for packages using their unique tracking IDs.

- Streamline the management of package details, including tracking IDs, pincodes, mobile numbers, emails, and addresses.
- Ensure the accuracy and integrity of the entered data through strict validation rules.
- Provide an intuitive platform that can be scaled and customized to meet the needs of users.

The system addresses the challenges of manual errors, inefficient data handling, and lack of quick retrieval mechanisms by offering a structured and reliable solution.

## 1.2 Overview

The **Package Tracking System** is a standalone Java application designed for organizations dealing with internal or local deliveries. Built using Java AWT for its user interface, the system offers a graphical interface that simplifies user interaction. The application features three primary modules:

1. **Add Package**: Allows users to input package details, including tracking ID, pincode, mobile number, email, and delivery address.
2. **View Packages**: Displays a list of all tracked packages with their associated details.
3. **Search Package**: Enables users to retrieve package details based on a tracking ID.

Data integrity is maintained through input validation mechanisms that enforce specific formats for fields like mobile numbers and email addresses. The system uses a `HashMap` for efficient storage and retrieval of data, ensuring fast access even with a large dataset.

The modular design of the application makes it adaptable for future enhancements, such as integration with external databases, APIs for real-time tracking, and multi-user support.

## 1.3 Java Programming Concepts

The Pharmacy Management System leverages key Java programming concepts to ensure its design is efficient, scalable, and user-friendly.

1. **Object-Oriented Programming (OOP)**:

- **Encapsulation**: Prescription details, patient data, and medicine inventory are encapsulated within classes to restrict direct access and maintain data integrity.
- **Inheritance**: Common properties and behaviours across different components (e.g., user interface elements) are inherited to promote code reuse.

## 1.4 Java Programming Concepts

### Object-Oriented Programming (OOP) Principles

#### Encapsulation:

1. All package details, such as tracking ID, pincode, mobile number, and email, are encapsulated within specific classes and methods.
2. This hides the internal implementation details and protects data from unintended interference.

#### Modularity:

1. Each core functionality (e.g., adding, viewing, and searching packages) is implemented as a separate module or method, ensuring reusability and ease of maintenance.

#### Abstraction:

1. The graphical user interface (GUI) abstracts the complexity of backend operations from the user, allowing them to interact seamlessly with the system.

#### Polymorphism:

1. Event handling methods, such as actionPerformed, demonstrate polymorphism by enabling different responses to user interactions.

## 2. Java Collections Framework

- **HashMap:**
  - Used to store package details with the tracking ID as the key and the package information as the value.
  - Provides efficient data storage and retrieval, ensuring quick access to package information.

## 3. Graphical User Interface (GUI) with Abstract Window Toolkit (AWT)

**Components:**

- AWT components such as Frame, Label, TextField, Button, and TextArea are used to create the graphical interface.
- These components allow users to interact with the system visually.

**Event Handling:**

- Event-driven programming is implemented using ActionListener and WindowAdapter.
- Example: Button clicks trigger specific actions, such as saving package details or searching for a package.

## 4. Input Validation

- Regular expressions and logical checks are used to validate user inputs, such as:
  - **Pincode:** Must be six numeric digits.
  - **Mobile Number:** Must start with '91' and have exactly ten digits following.
  - **Email:** Checked for the presence of '@' to ensure correct formatting.

# CHAPTER-2
# PROJECT METHODOLOGY

## 2.1 Proposed Work :

The **Package Tracking System** is designed to address the challenges of managing package details efficiently and accurately. This project aims to provide a user-friendly solution that automates the process of tracking packages by leveraging the robust features of Java programming. The system includes modules for adding, viewing, and searching packages, each designed to ensure seamless functionality and scalability.

The proposed work focuses on creating a graphical user interface (GUI) using Java's Abstract Window Toolkit (AWT) to make the system accessible and intuitive for users. The application uses a HashMap to store package details, enabling quick data retrieval based on tracking IDs. Input validation is a critical component of the system, ensuring that essential fields such as pincode, mobile number, and email adhere to predefined formats, thereby reducing errors and enhancing data reliability.

Additionally, the modular design of the system facilitates scalability, allowing for future enhancements such as database integration, real-time tracking, and multi-user access. Error handling mechanisms are incorporated to provide meaningful feedback to users in case of invalid inputs or system errors. Overall, the proposed work aims to deliver a comprehensive, efficient, and reliable package tracking solution that meets the needs of small-scale organizations or internal delivery systems.

## 2.2 Block Diagram

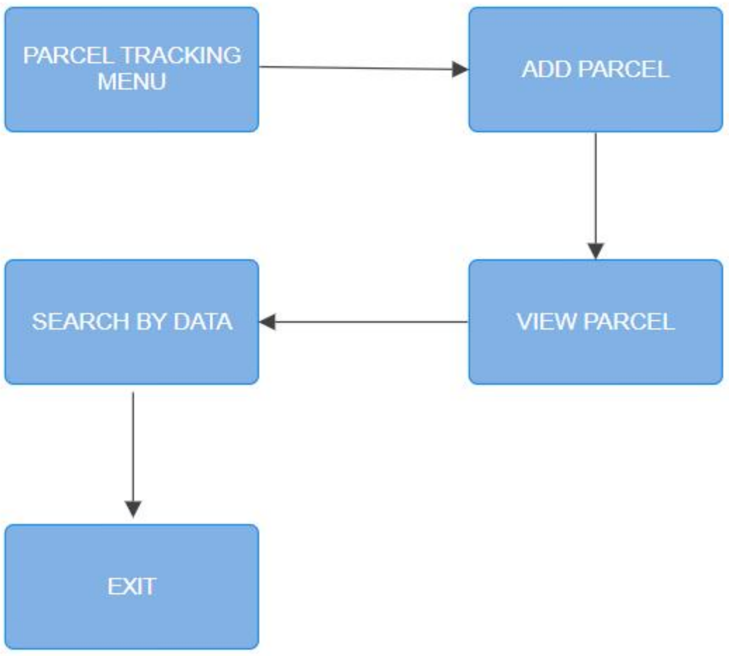


**Fig 2.1 Pharmacy management system**

# CHAPTER 3
# MODULE DESCRIPTION

The **Package Tracking System** is structured into distinct modules to ensure functionality, maintainability, and scalability. Each module is designed to handle a specific aspect of the system, with a focus on user interaction, data management, and validation. Below is a detailed description of each module:

## User Interface Module

This module provides an intuitive graphical user interface (GUI) built using Java's Abstract Window Toolkit (AWT). It allows users to interact with the system seamlessly, using components such as:

- **Buttons:** For initiating actions like adding, viewing, or searching packages.
- **TextFields:** For entering package details such as tracking ID, pincode, mobile number, email, and address.
- **TextArea:** For displaying output such as package lists or search results. The layout is carefully designed to ensure usability and accessibility, making the system user-friendly for all.

## 3.1 Add Package Module

This module enables users to add new package details to the system. It collects the following data:

- **Tracking ID:** A unique identifier for each package.
- **Pincode:** A six-digit numeric code for the destination.

- **Mobile Number:** Starting with '91' and followed by ten digits.
- **Email Address:** Ensures a valid format (e.g., must contain '@').
- **Address:** Includes delivery address details. Input validation is enforced to ensure the accuracy and completeness of the data. Any invalid input is flagged, and a polite error message is displayed to the user. Successfully added packages are stored in a HashMap for quick retrieval.

## 3.2 View Packages Module

This module displays a list of all tracked packages. Each package's details, including the tracking ID, pincode, mobile number, email, and address, are presented in a readable format using a TextArea. If no packages are available, the system informs the user accordingly. This module is crucial for providing an overview of all active packages in the system.

## 3.3 Search Package Module

The search module allows users to find specific packages by entering their tracking ID. If the tracking ID exists in the system, all associated details are displayed. If the ID is not found, the system displays a polite error message indicating that the package does not exist. This module ensures quick and accurate retrieval of package information, reducing the time spent on manual searches.

## 3.4 Input Validation Module

To maintain data integrity, the system incorporates a robust validation mechanism. This module checks:

- **Pincode:** Ensures it is exactly six numeric digits.
- **Mobile Number:** Validates the format as starting with '91' and containing ten digits.
- **Email Address:** Confirms the presence of '@' for a basic email format. Invalid inputs trigger error messages, guiding users to correct their entries and preventing the system from storing erroneous data.

## 3.5 Error Handling Module

This module ensures the application remains robust and user-friendly. Common errors, such as invalid inputs or missing fields, are caught and handled gracefully. Error messages are displayed in the output area, helping users understand and rectify issues without disrupting the overall workflow.

# CHAPTER 4
# CONCLUSION AND FUTURE SCOPE

## 4.1 Conclusion

The **Package Tracking System** is an efficient and user-friendly application designed to simplify the management of package details. By leveraging Java programming and the Abstract Window Toolkit (AWT), the system provides a seamless experience for users to add, view, and search for packages. The use of **input validation** ensures that only accurate data is entered, minimizing errors. The **HashMap** data structure is employed to ensure quick and efficient retrieval of package information, making it ideal for small-scale organizations or internal delivery systems.

The system's modular design allows for easy updates and future enhancements, such as database integration or real-time tracking. Its implementation of **object-oriented principles**, **error handling**, and **event-driven programming** demonstrates a clear understanding of key Java concepts. Through this project, we have successfully built a scalable and reliable package management solution that can be extended further as required.

Overall, the **Package Tracking System** achieves its objective of improving operational efficiency, reducing manual errors, and providing a streamlined solution for package tracking, demonstrating the practical application of Java programming in solving real-world problems.

## 4.2 Future Scope

The **Package Tracking System** has significant potential for future development and scalability. Some of the possible enhancements include:

**Database Integration**

1. Transition from using a HashMap to integrating a database (e.g., MySQL or MongoDB) for persistent data storage, allowing the system to handle large volumes of data and provide secure storage.

**Real-Time Tracking**

Integration with real-time tracking systems via APIs or GPS, allowing users to track packages dynamically with updated locations.

**Automated Notifications**

Adding functionality to send automated email or SMS notifications to users about package status updates, such as when a package is dispatched, in transit, or delivered.

**Multi-User Support**

Implement multi-user functionality with role-based access control, allowing different users (e.g., admins, users) to access and manage package information based on their roles.

**Cloud Integration**

Migrate the application to a cloud platform for centralized storage, scalability, and access from multiple devices, enabling remote monitoring and management.

**Mobile Application Development**

Develop a mobile version of the application to make the package tracking system more accessible for users on smartphones or tablets.

**Advanced Analytics and Reporting**

Introduce data analytics features to generate insights on delivery patterns, user behavior, and operational efficiencies, aiding decision-making for logistics teams.

**Security Features**

Implement security measures like encryption for sensitive data and multi-factor authentication to protect user and package information.

**Multi-Language Support**

Include support for multiple languages to make the system more inclusive and adaptable to different regions.

**Integration with External APIs**

Integration with other logistics and delivery service APIs for enhanced functionality and broader package management capabilities.

By implementing these enhancements, the **Package Tracking System** can evolve into a comprehensive solution for large-scale logistics and e-commerce operations, making it adaptable to meet the growing needs of the industry.

## APPENDIX A (SOURCE CODE)

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

public class PackageTrackingSystem extends Frame {
 private HashMap<String, String> packages = new HashMap<>(); // Stores trackingand
  package details

    public PackageTrackingSystem() {
        setTitle("Package Tracking Management System");
        setSize(800, 500);
        setLayout(null);

        Label titleLabel = new Label("Package Tracking Management System",
Label.CENTER);
        titleLabel.setBounds(200, 50, 400, 30);
        add(titleLabel);

        Button addPackageButton = new Button("Add Package");
        addPackageButton.setBounds(100, 100, 150, 30);
        add(addPackageButton);

        Button viewPackagesButton = new Button("View Packages");
        viewPackagesButton.setBounds(325, 100, 150, 30);
        add(viewPackagesButton);

        Button searchPackageButton = new Button("Search Package");
        searchPackageButton.setBounds(550, 100, 150, 30);
        add(searchPackageButton);

        TextArea displayArea = new TextArea();
        displayArea.setBounds(50, 150, 700, 300);
        add(displayArea);

        // Add Package
```

```
addPackageButton.addActionListener(e -> {
   Frame addFrame = new Frame("Add Package");
   addFrame.setSize(500, 500);
   addFrame.setLayout(null);

   Label trackingLabel = new Label("Tracking ID:");
   trackingLabel.setBounds(50, 50, 80, 30);
   addFrame.add(trackingLabel);

   TextField trackingField = new TextField();
   trackingField.setBounds(150, 50, 250, 30);
   addFrame.add(trackingField);

   Label pincodeLabel = new Label("Pincode:");
   pincodeLabel.setBounds(50, 100, 80, 30);
   addFrame.add(pincodeLabel);

   TextField pincodeField = new TextField();
   pincodeField.setBounds(150, 100, 250, 30);
   addFrame.add(pincodeField);

   Label mobileLabel = new Label("Mobile:");
   mobileLabel.setBounds(50, 150, 80, 30);
   addFrame.add(mobileLabel);

   TextField mobileField = new TextField();
   mobileField.setBounds(150, 150, 250, 30);
   addFrame.add(mobileField);

   Label emailLabel = new Label("Email:");
   emailLabel.setBounds(50, 200, 80, 30);
   addFrame.add(emailLabel);

   TextField emailField = new TextField();
   emailField.setBounds(150, 200, 250, 30);
   addFrame.add(emailField);
```

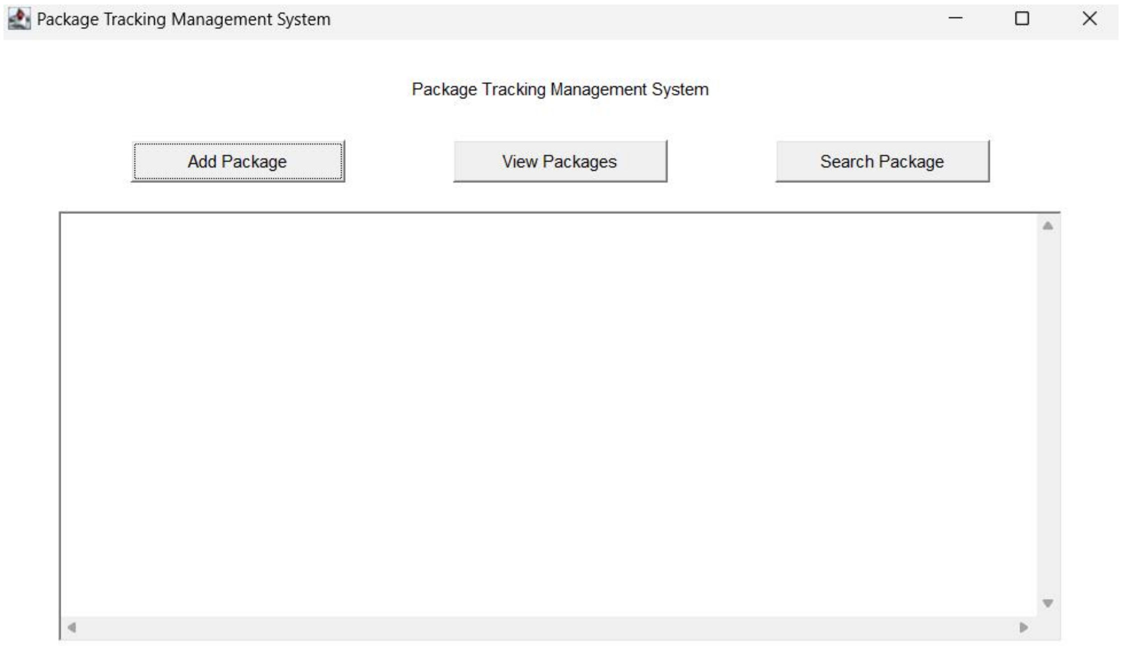
```
        Label detailsLabel = new Label("Address:");
        detailsLabel.setBounds(50, 250, 80, 30);
        addFrame.add(detailsLabel);

        TextField detailsField = new TextField();
        detailsField.setBounds(150, 250, 250, 30);
        addFrame.add(detailsField);

        Button saveButton = new Button("Save");
        saveButton.setBounds(200, 300, 80, 30);
        addFrame.add(saveButton);

        saveButton.addActionListener(ev -> {
          String trackingID = trackingField.getText();
          String pincode = pincodeField.getText();
          String mobile = mobileField.getText();
          String email = emailField.getText();
          String details = detailsField.getText();

          if (trackingID.isEmpty() || pincode.isEmpty() || mobile.isEmpty() ||
email.isEmpty() || details.isEmpty()) {
            displayArea.setText("All fields are mandatory!");
          } else if (!pincode.matches("\\d{6}")) {
            displayArea.setText("Invalid Pincode. It should be 6 digits.");
          } else if (!mobile.matches("91\\d{10}")) {
            displayArea.setText("Invalid Mobile Number. It should start with '91'
followed by 10 digits.");
          } else if (!email.contains("@")) {
            displayArea.setText("Invalid Email Address.");
          } else {
            String packageDetails = "Pincode: " + pincode + ", Mobile: " + mobile + ",
Email: " + email + ", Details: " + details;
            packages.put(trackingID, packageDetails);
            displayArea.setText("Package added successfully!");
          }
          addFrame.dispose();
        });
```

```
            addFrame.setVisible(true);
        });
        // View All Packages
        viewPackagesButton.addActionListener(e -> {
            displayArea.setText("Tracked Packages:\n");
            if (packages.isEmpty()) {
                displayArea.append("No packages available.\n");
            } else {
                for (String trackingID : packages.keySet()) {
                    displayArea.append("Tracking ID: " + trackingID + ", " +
packages.get(trackingID) + "\n");
                }
            }
        });
        // Search for a Package
        searchPackageButton.addActionListener(e -> {
            Frame searchFrame = new Frame("Search Package");
            searchFrame.setSize(400, 200);
            searchFrame.setLayout(null);

            Label trackingLabel = new Label("Tracking ID:");
            trackingLabel.setBounds(50, 50, 80, 30);
            searchFrame.add(trackingLabel);

            TextField trackingField = new TextField();
            trackingField.setBounds(150, 50, 200, 30);
            searchFrame.add(trackingField);

            Button searchButton = new Button("Search");
            searchButton.setBounds(150, 100, 80, 30);
            searchFrame.add(searchButton);

            searchButton.addActionListener(ev -> {
                String trackingID = trackingField.getText();
                if (packages.containsKey(trackingID)) {
                    displayArea.setText("Details for Tracking ID " + trackingID + ": " +
packages.get(trackingID));
```

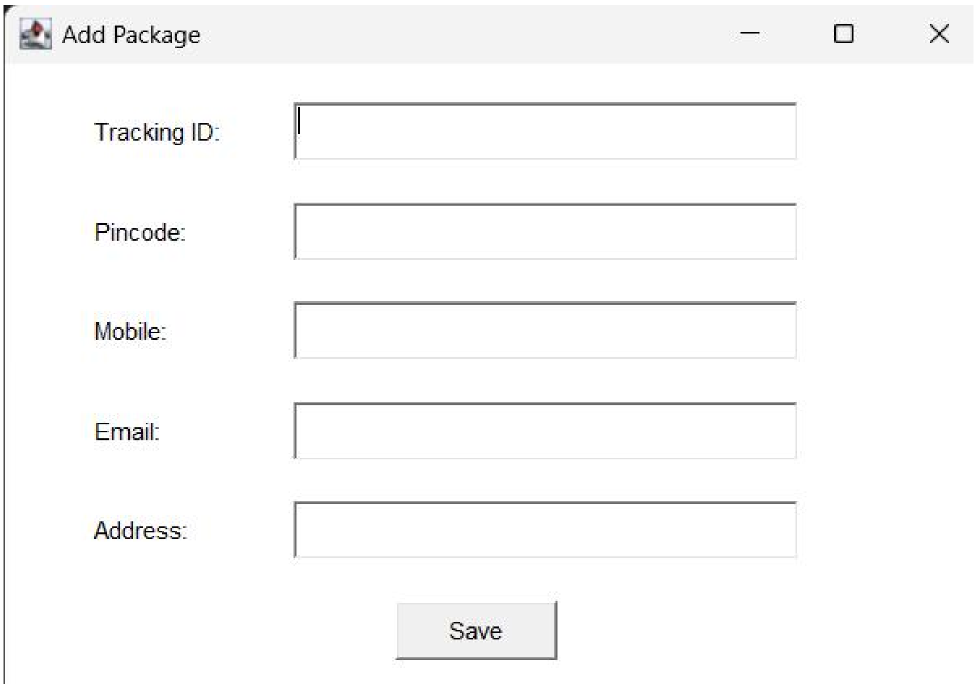
```
                } else {
                    displayArea.setText("No package found with Tracking ID: " + trackingID);
                }
                searchFrame.dispose();
            });

            searchFrame.setVisible(true);
        });

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });

        setVisible(true);
    }

    public static void main(String[] args) {
        new PackageTrackingSystem();
    }
}
```

## APPENDIX B (SCREENSHOTS)

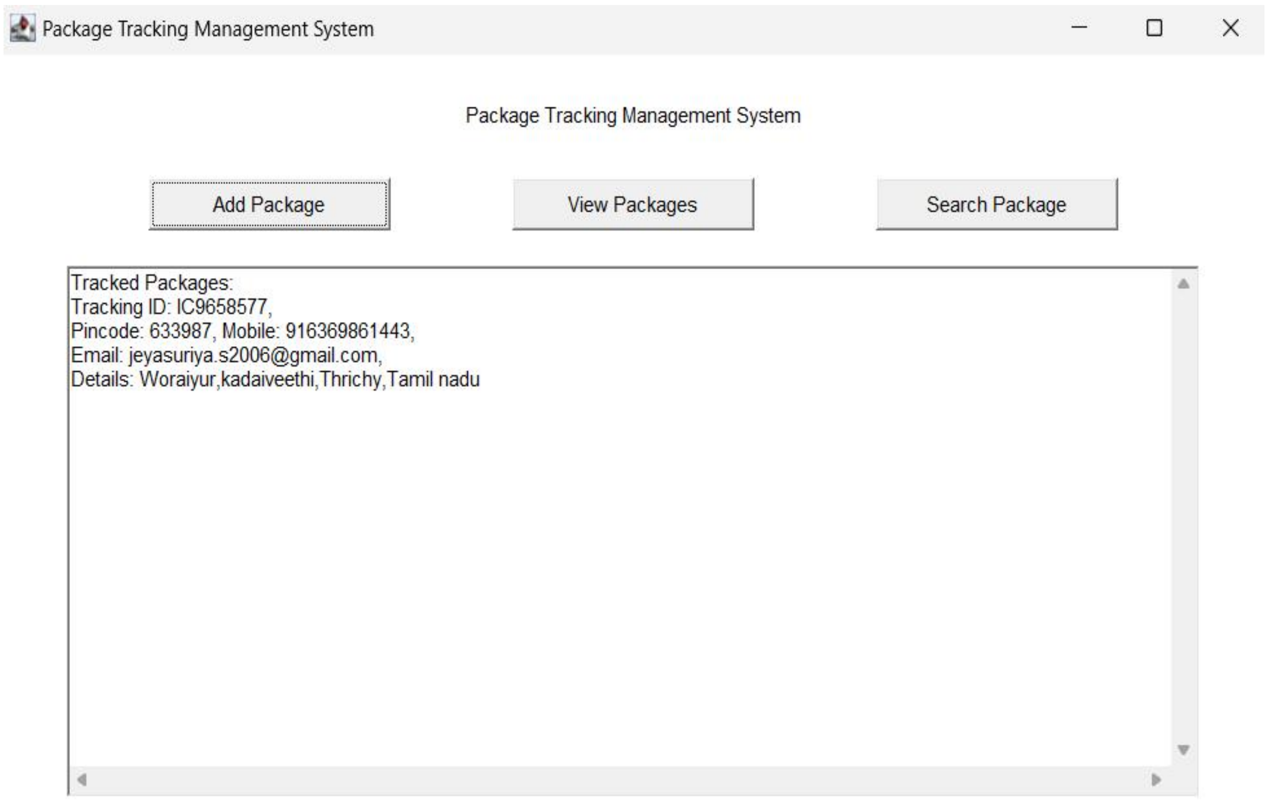
### HOME PAGE:


Package Tracking Management System
Package Tracking Management System
Add Package
View Packages
Search Package

### ADD PACKAGE:

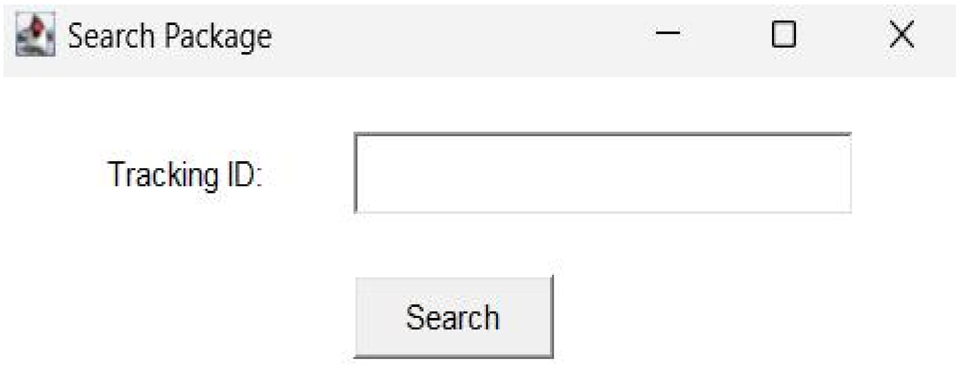


## VIEW PACKAGE:

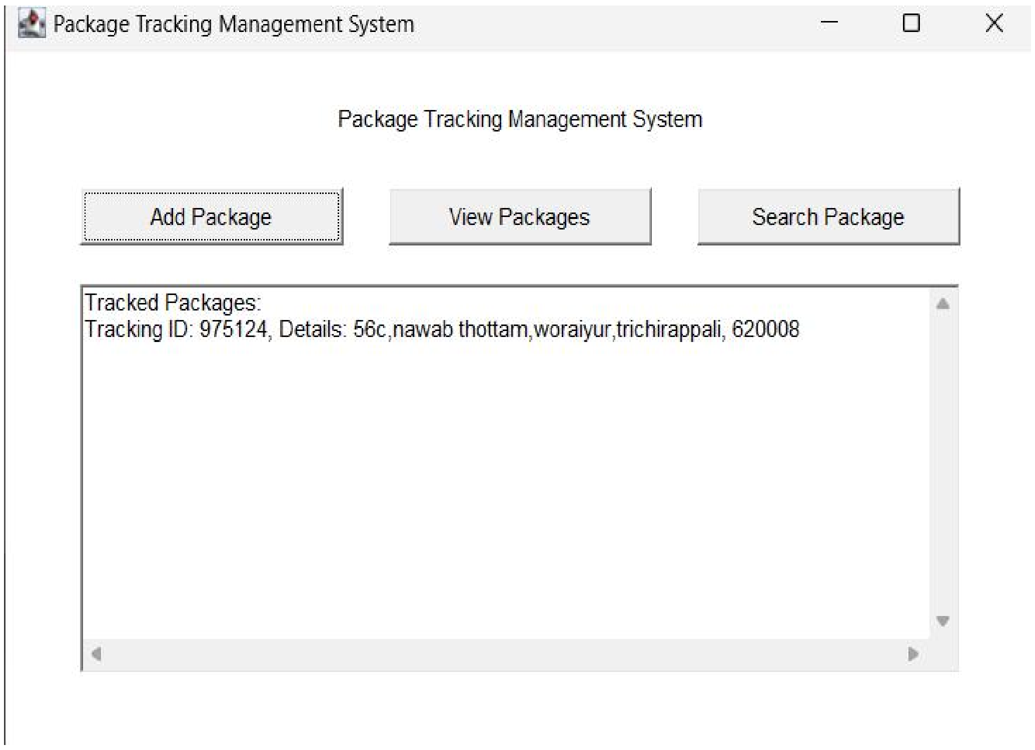

Package Tracking Management System
Package Tracking Management System
Add Package
View Packages
Search Package

```
Tracked Packages:
Tracking ID: IC9658577,
Pincode: 633987, Mobile: 916369861443,
Email: jeyasuriya.s2006@gmail.com,
Details: Woraiyur,kadaiveethi,Thrichy,Tamil nadu
```

## SEARCH PACKAGE:



**SEARCH  PACKAGES :**


Package Tracking Management System
Package Tracking Management System
Add Package
View Packages
Search Package

```
Tracked Packages:
Tracking ID: 975124, Details: 56c,nawab thottam,woraiyur,trichirappali, 620008
```

## REFERENCES:

**Schildt, H.** (2018). *Java: The Complete Reference* (10th ed.). McGraw-Hill Education.
This book provides comprehensive coverage of Java programming, including the use of AWT for building graphical user interfaces and various Java programming concepts used in this project.

**Flanagan, D.** (2005). *Java Examples in a Nutshell* (4th ed.). O'Reilly Media.
This reference covers many Java features, including error handling, event handling, and AWT, which were essential to the development of the **Package Tracking System**.

**Horstmann, C. S., & Cornell, G.** (2013). *Core Java Volume I–Fundamentals* (9th ed.). Prentice Hall.
This book covers core Java concepts, such as object-oriented programming, which were applied in the development of the **Package Tracking System**.

**Eckel, B.** (2006). *Thinking in Java* (4th ed.). Prentice Hall.
This reference provides in-depth knowledge of Java programming, including object-oriented principles, which were used to create a modular and maintainable design for the

**Budd, T. A.** (2005). *Understanding Object-Oriented Programming with Java* (3rd ed.). Addison-Wesley.
This textbook provides insights into object-oriented programming principles, including encapsulation, polymorphism, and inheritance, which were integral to the design of the system.

**GeeksforGeeks.** (2024). *Java AWT Tutorial*. Retrieved from https://www.geeksforgeeks.org/java-awt-abstract-window-toolkit/
A helpful resource for understanding AWT components and their usage in Java, which guided the development of the system's graphical interface.