PARCEL TRACKING SYSTEM

A PROJECT REPORT

Submitted by

PRAGATHEESHWARAN E (2303811710421118)

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 **" PARCEL TRACKING SYSTEM"** is the bonafide work of **PRAGATHEESHWARAN E (2303811710421118)** 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

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.)
SUPERVISOR
ASSISTANTP ROFESSOR

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.,),
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 03-12-2024.

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

CGB1201-JAVA PROGRAMMING
ASST.P ARUNA PRIYA, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8104-DSEC, PERAMBALUR.

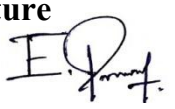
INTERNAL EXAMINER EXTERNAL EXAMINER

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 **CGB1201 - JAVA PROGRAMMING.**

.

Signature



PRAGATHEESHWARAN E

Place: Samayapuram

Date:03-12-2024

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 **Mrs. K. VALLI PRIYADHARSHINI, 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 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

The **Parcel Tracking System** is a software application designed to provide real-time parcel tracking and monitoring services for customers and logistics companies. The system leverages Java Swing for a user-friendly graphical interface, allowing seamless interaction for various stakeholders. It ensures accurate tracking of parcels at every stage of the delivery process, from dispatch to delivery. The primary objective of the system is to enhance transparency and customer satisfaction by offering reliable and up-to-date information about parcel locations and statuses. It supports functionalities such as parcel registration, tracking updates, and delivery confirmations. The system also facilitates logistics optimization by offering insights into delivery patterns and performance. This project incorporates essential features like a customer portal to check the parcel's progress, a backend for logistics providers to update parcel statuses, and alert notifications for delays or delivery confirmations. By integrating robust data handling and intuitive design, the Parcel Tracking System aims to streamline operations, reduce errors, and improve overall efficiency in the parcel delivery process.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
The **Parcel Tracking System** is a Java Swing-based application designed to provide real-time tracking of parcels throughout the delivery process. It enables customers to monitor their parcels and logistics providers to update statuses at various stages, ensuring transparency and timely updates. The system enhances customer satisfaction by providing accurate location and status information while optimizing logistics operations through efficient tracking and delivery management. With features like parcel registration, status updates, and delivery confirmations, the system aims to improve reliability, reduce delays, and streamline the delivery process.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of the Parcel Tracking System is to provide an efficient and user-friendly platform for tracking parcels in real-time. This system aims to streamline the logistics process, ensuring transparency and reliability for customers, businesses, and courier services.

The specific objectives include:

1. **Enhance Parcel Visibility:**
   Enable users to track the current location and status of their parcels throughout the delivery process.
2. **Improve Customer Experience:**
   Provide timely updates to customers, increasing trust and satisfaction with the service.
3. **Streamline Logistics Operations:**
   Assist courier companies in managing and monitoring parcels efficiently, reducing delivery errors and delays.
4. **Ensure Data Accuracy:**
   Maintain accurate records of parcel details, delivery timelines, and tracking history for better accountability.
5. **Facilitate Real-Time Notifications:**
   Deliver real-time notifications to users regarding parcel status changes, such as dispatched, in transit, and delivered.

1.2 Overview

The Parcel Tracking System is a desktop-based application developed using Java Swing, designed to automate and enhance the process of tracking parcels during their journey from sender to recipient. It offers a seamless interface for users to access real-time information about their shipments, ensuring improved communication between stakeholders in the delivery chain.

The system is intended to replace traditional manual tracking methods with a more reliable and automated solution. It integrates essential features such as parcel registration, tracking updates, and status notifications, making the process more transparent and efficient.

1.3 Java Programming Concepts

The Parcel Tracking System leverages core Java programming concepts and principles of Object-Oriented Programming (OOP) to provide a robust, scalable, and maintainable solution. Below is an overview of the essential Java programming concepts applied in this project:

1.3.1 Object-Oriented Programming Concepts

The OOPS principles are the backbone of this system, ensuring modularity, reusability, and ease of maintenance. Key OOPS concepts used include:

1. **Encapsulation:**
    - All data related to parcels, users, and tracking details are encapsulated in dedicated classes.
    - Example: A Parcel class holds attributes like trackingID, sender, recipient, and status, with private access modifiers and public getter/setter methods to manage them.
2. **Inheritance:**
    - The system uses inheritance to define relationships between classes, reducing redundancy.
    - Example: A User base class can be extended to create Admin, Sender, and Recipient subclasses, inheriting shared attributes and methods.
3. **Polymorphism:**
    - The system uses polymorphism to allow flexibility and dynamic behavior in the code.
    - Example: Overriding methods like displayDetails() for Sender and Recipient classes, providing specific implementations for each.
4. **Abstraction:**
    - Abstract classes and interfaces are employed to define essential behaviors and enforce consistency across different modules.
    - Example: An interface Trackable ensures that all trackable entities implement a getTrackingStatus() method.

1.3.2 Core Java Concepts

1. **Classes and Objects:**
    - The system is built using a class-based structure to model real-world entities like parcels, users, and courier stations.

2. **Collections Framework:**
    - Data structures such as ArrayList, HashMap, and HashSet are used to manage collections of parcels and users efficiently.
3. **Exception Handling:**
    - Proper error handling ensures the application is robust and user-friendly.
    - Example: Handling invalid tracking IDs or database connection issues using try-catch blocks.
4. **Swing Framework:**
    - Java Swing is used to design the graphical user interface (GUI), enabling interactive and responsive screens for user interaction.
5. **File I/O and Database Connectivity:**
    - File handling is used for logging purposes, and database connectivity (using JDBC) stores and retrieves user and parcel data.
6. **Multithreading (Optional):**
    - For real-time notifications and status updates, threads can be used to handle multiple tasks concurrently.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

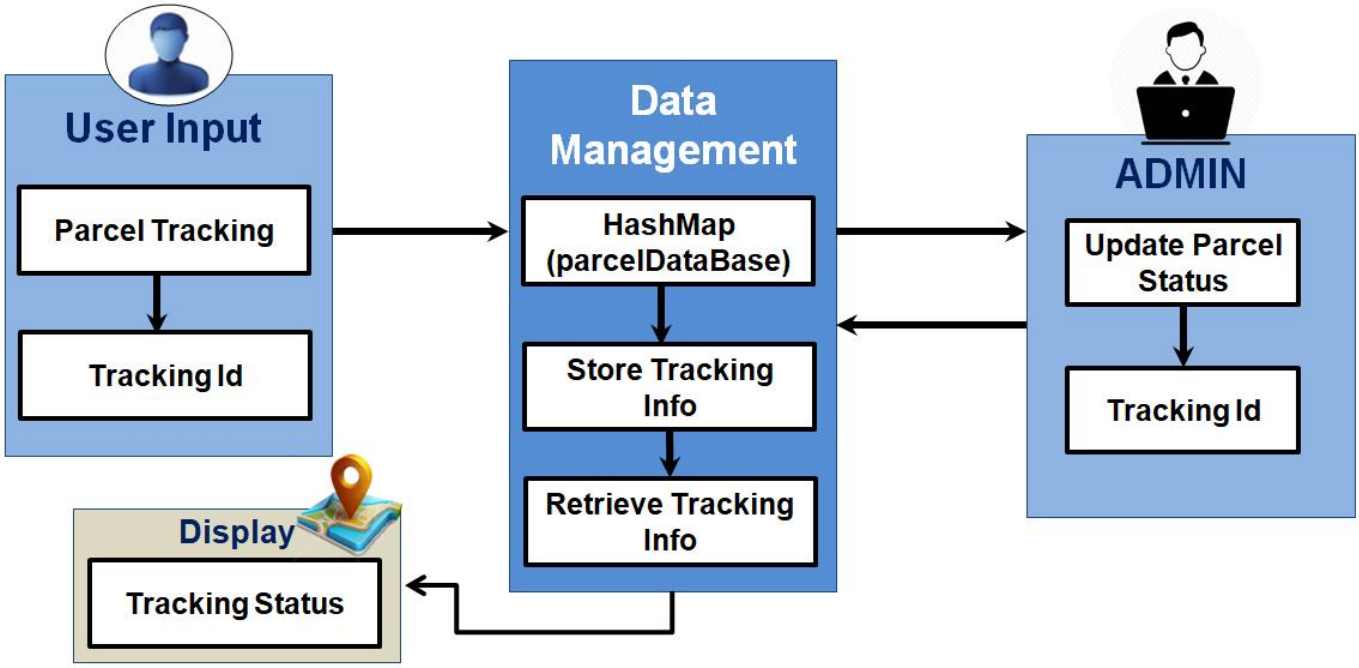
The proposed Parcel Tracking System aims to design and implement a user-friendly application that automates the process of tracking parcels. This system ensures real-time visibility and accountability throughout the shipment lifecycle, from dispatch to delivery.

The key aspects of the proposed work are as follows:

1. **System Architecture:**
    - The system will follow a modular architecture, dividing functionalities into independent modules such as User Management, Parcel Management, and Tracking Updates.
    - The backend will handle data storage and processing, while the frontend will provide an interactive GUI using Java Swing.
2. **Core Features:**
    - **Parcel Registration:** A form-based interface where senders can input parcel details and generate a unique tracking ID.
    - **Real-Time Tracking:** Updates on parcel location and status will be recorded at every checkpoint, accessible via the tracking ID.
    - **User Roles:** Separate functionalities for Admin (system management), Sender (parcel dispatch), and Recipient (status checking).
    - **Notifications:** Real-time notifications or alerts on status changes, such as "Dispatched," "In Transit," or "Delivered."
3. **Workflow:**
    - The sender registers the parcel with details such as sender name, recipient name, address, and parcel weight.
    - The system generates a unique tracking ID, which is shared with the recipient.
    - The admin updates the parcel status at different checkpoints during the delivery process.
    - The recipient can use the tracking ID to check the parcel's current location and delivery progress.

4. **Technology Stack:**
    - **Frontend:** Java Swing for creating the user interface.
    - **Backend:** Core Java for business logic and MySQL (or an equivalent database) for data storage.
    - **Integration:** JDBC for database connectivity.
5. **Advantages:**
    - Reduces manual errors in parcel tracking.
    - Provides transparency and real-time updates for users.
    - Enhances operational efficiency for courier services.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Update Module

The Update module is responsible for modifying the status and details of parcels at various checkpoints. This module is primarily used by the system administrator or logistics personnel.

Key Features:

- Allows the admin to update parcel status, such as "Dispatched," "In Transit," "Out for Delivery," and "Delivered."
- Records timestamps and locations for each status update.
- Ensures that status updates are automatically reflected in the tracking system for users to access in real-time.

Implementation Details:

- A form-based interface enables easy input of tracking IDs and status updates.
- Data is validated before being updated in the database to prevent errors.

3.2 Track Module

The Track module allows users (senders or recipients) to check the real-time status and location of their parcels.

Key Features:

- Enables tracking of parcels using a unique tracking ID.
- Displays current status, location history, and estimated delivery time.
- Provides error messages for invalid or non-existent tracking IDs.

Implementation Details:

- Retrieves parcel data from the database using the tracking ID as a search parameter.
- Displays results in an interactive GUI designed with Java Swing.

3.3 Main Module

The Main module acts as the central control point for the application, integrating all other modules. It provides the primary interface for navigation and managing user interactions.

Key Features:

- Handles user authentication and role-based access (Admin, Sender, Recipient).
- Facilitates smooth navigation between the Update, Track, and Data Management modules.
- Provides error handling and user-friendly prompts for invalid actions.

Implementation Details:

- Uses Java Swing menus and panels to provide an intuitive interface.
- Contains the primary logic for event handling and module integration.

3.4 Data Management Module

The Data Management module is responsible for handling all system data, ensuring efficient storage, retrieval, and updates. Unlike traditional database systems, this implementation uses a **HashMap** for managing data in-memory, making it suitable for smaller-scale applications or prototype versions of the system.

Key Features:

- Manages data for parcels, users, and system logs.
- Provides methods to add, update, retrieve, and delete records.
- Ensures quick data access with the HashMap's constant-time complexity for basic operations.

Implementation Details:

1. **Data Structure:**

- The module uses Java's HashMap class to store data as key-value pairs:
    - **Key:** A unique identifier, such as a tracking ID or user ID.

- **Value:** An object representing the details of a parcel, user, or log entry.

2. **Operations:**

- **Adding Data:** Use the put(key, value) method to insert new records.
- **Retrieving Data:** Use the get(key) method to fetch data for a given key.
- **Updating Data:** Retrieve the object, modify its attributes, and store it back with the same key.
- **Deleting Data:** Use the remove(key) method to delete records.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Parcel Tracking System developed using Java Swing successfully addresses the need for an efficient, user-friendly, and transparent platform to track parcels in real time. By implementing essential modules such as Update, Track, Main, and Data Management, the system ensures smooth functionality and ease of use for all stakeholders, including senders, recipients, and administrators.

The system leverages key Java concepts such as Object-Oriented Programming, HashMap-based in-memory data management, and an interactive GUI to provide a robust and modular solution. The following goals have been achieved:

- Automating parcel tracking to reduce manual errors.
- Enhancing customer satisfaction with real-time status updates.
- Simplifying the logistics process with role-based access and streamlined workflows.

Despite being a prototype or small-scale application, the system demonstrates scalability and modularity, making it a valuable foundation for further development.

4.2 FUTURE SCOPE

While the current implementation achieves the primary objectives, there is significant potential for enhancement to adapt to evolving user needs and industry standards. The proposed future developments include:

1. **Integration with a Persistent Database:**
    - Transition from HashMap to a relational database like MySQL or a NoSQL database for scalability and data persistence.
2. **Web and Mobile Versions:**
    - Develop web and mobile applications for broader accessibility and convenience.
3. **Advanced Tracking Features:**
    - Implement GPS-based real-time location tracking for greater accuracy.

- Introduce predictive analytics for estimating delivery times.

4. **Enhanced Notification System:**
    - Enable SMS and email notifications for parcel status updates.
    - Integrate push notifications for mobile users.
5. **Secure Data Handling:**
    - Implement encryption for sensitive data such as user credentials and parcel details.
6. **Integration with External Systems:**
    - Connect the system with e-commerce platforms and third-party logistics providers for seamless operation.
7. **User Feedback and Support:**
    - Add a module for users to provide feedback and track resolution of their issues.

The Parcel Tracking System thus lays the groundwork for a versatile and scalable solution, with the potential to evolve into a comprehensive logistics management tool.

APPENDIX A
(SOURCE CODE)

```
import java.awt.*;
import java.util.HashMap;
import javax.swing.*;

public class ParcelTrackingSystem {

    HashMap<String, String[]> parcelDatabase; // Using an array to store Location and Status
    String adminPassword = "admin123"; // Predefined admin password for simplicity

    public ParcelTrackingSystem() {
        parcelDatabase = new HashMap<>(); // Initialize the dynamic database
        createLoginWindow();
    }

    // Login Window to choose Admin or Customer
    void createLoginWindow() {
        JFrame loginFrame = new JFrame("Parcel Tracking System - Login");
        loginFrame.setLayout(new GridBagLayout());
        loginFrame.setSize(400, 200);
        loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel lblTitle = new JLabel("Parcel Tracking System");
        lblTitle.setFont(new Font("Arial", Font.BOLD, 18));
        JButton btnAdmin = new JButton("Login as Admin");
        JButton btnCustomer = new JButton("Login as Customer");

        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.gridx = 0;
        gbc.gridy = 0;
        loginFrame.add(lblTitle, gbc);
```

```
    gbc.gridy = 1;
    loginFrame.add(btnAdmin, gbc);

    gbc.gridy = 2;
    loginFrame.add(btnCustomer, gbc);

    btnAdmin.addActionListener(e -> createAdminLoginWindow());
    btnCustomer.addActionListener(e -> createCustomerWindow());

    loginFrame.setVisible(true);
}

// Admin Login Window
void createAdminLoginWindow() {
    JFrame adminLoginFrame = new JFrame("Admin Login");
    adminLoginFrame.setLayout(new GridBagLayout());
    adminLoginFrame.setSize(400, 200);
    adminLoginFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JLabel lblPassword = new JLabel("Enter Admin Password:");
    JPasswordField txtPassword = new JPasswordField(15);
    JButton btnLogin = new JButton("Login");
    JTextArea txtAreaOutput = new JTextArea(2, 20);
    txtAreaOutput.setEditable(false);

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(10, 10, 10, 10);
    gbc.gridx = 0;
    gbc.gridy = 0;
    adminLoginFrame.add(lblPassword, gbc);

    gbc.gridx = 1;
    adminLoginFrame.add(txtPassword, gbc);
```

```
    gbc.gridy = 1;
    gbc.gridx = 0;
    gbc.gridwidth = 2;
    adminLoginFrame.add(btnLogin, gbc);

    gbc.gridy = 2;
    adminLoginFrame.add(txtAreaOutput, gbc);

    btnLogin.addActionListener(e -> {
        String enteredPassword = new String(txtPassword.getPassword());
        if (enteredPassword.equals(adminPassword)) {
            adminLoginFrame.dispose();
            createAdminWindow();
        } else {
            txtAreaOutput.setText("Error: Incorrect password!");
        }
    });

    adminLoginFrame.setVisible(true);
}

// Admin Main Window
void createAdminWindow() {
    JFrame adminFrame = new JFrame("Admin Dashboard");
    adminFrame.setLayout(new GridBagLayout());
    adminFrame.setSize(400, 200);
    adminFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JLabel lblTitle = new JLabel("Admin Dashboard");
    lblTitle.setFont(new Font("Arial", Font.BOLD, 18));
    JButton btnUpdateParcel = new JButton("Update Parcel Status");

    GridBagConstraints gbc = new GridBagConstraints();
```

```
    gbc.insets = new Insets(10, 10, 10, 10);
    gbc.gridx = 0;
    gbc.gridy = 0;
    adminFrame.add(lblTitle, gbc);

    gbc.gridy = 1;
    adminFrame.add(btnUpdateParcel, gbc);

    btnUpdateParcel.addActionListener(e -> createRegisterWindow());

    adminFrame.setVisible(true);
}

// Customer Main Window
void createCustomerWindow() {
    JFrame customerFrame = new JFrame("Customer Dashboard");
    customerFrame.setLayout(new GridBagLayout());
    customerFrame.setSize(400, 200);
    customerFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JLabel lblTitle = new JLabel("Customer Dashboard");
    lblTitle.setFont(new Font("Arial", Font.BOLD, 18));
    JButton btnTrackParcel = new JButton("Track Parcel");

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(10, 10, 10, 10);
    gbc.gridx = 0;
    gbc.gridy = 0;
    customerFrame.add(lblTitle, gbc);

    gbc.gridy = 1;
    customerFrame.add(btnTrackParcel, gbc);

    btnTrackParcel.addActionListener(e -> createTrackWindow());
```

```
    customerFrame.setVisible(true);
}

// Register Parcel Window (Admin functionality)
void createRegisterWindow() {
    JFrame registerFrame = new JFrame("Update Parcel");
    registerFrame.setLayout(new GridBagLayout());
    registerFrame.setSize(400, 300);
    registerFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JLabel lblTitle = new JLabel("Update a Parcel");
    lblTitle.setFont(new Font("Arial", Font.BOLD, 16));
    JLabel lblTrackingID = new JLabel("Tracking ID:");
    JLabel lblLocation = new JLabel("Location:");
    JLabel lblStatus = new JLabel("Status:");
    JTextField txtTrackingID = new JTextField(15);
    JTextField txtLocation = new JTextField(15);
    JTextField txtStatus = new JTextField(15);
    JButton btnRegister = new JButton("Update");

    JTextArea txtAreaOutput = new JTextArea(5, 20);
    txtAreaOutput.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(txtAreaOutput);

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(10, 10, 10, 10);
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    registerFrame.add(lblTitle, gbc);

    gbc.gridwidth = 1;
    gbc.gridy = 1;
```

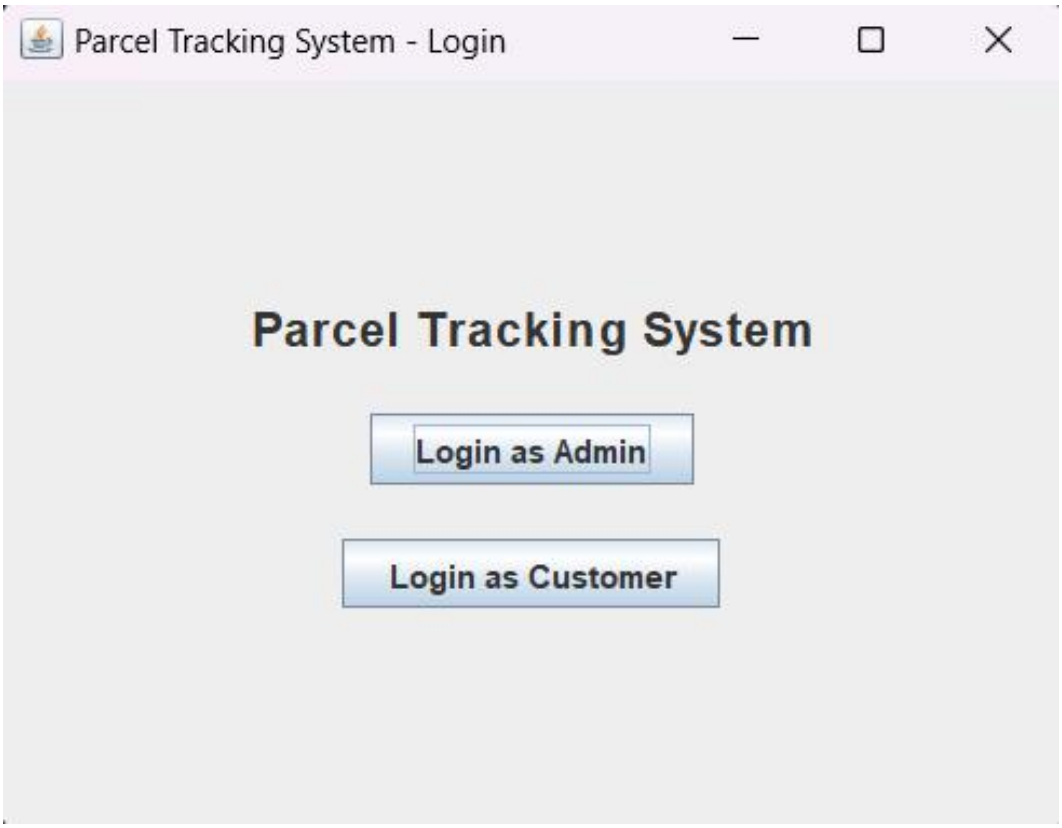
```
gbc.gridx = 0;
registerFrame.add(lblTrackingID, gbc);
gbc.gridx = 1;
registerFrame.add(txtTrackingID, gbc);

gbc.gridy = 2;
gbc.gridx = 0;
registerFrame.add(lblLocation, gbc);
gbc.gridx = 1;
registerFrame.add(txtLocation, gbc);

gbc.gridy = 3;
gbc.gridx = 0;
registerFrame.add(lblStatus, gbc);
gbc.gridx = 1;
registerFrame.add(txtStatus, gbc);

gbc.gridy = 4;
gbc.gridx = 0;
gbc.gridwidth = 2;
registerFrame.add(btnRegister, gbc);

gbc.gridy = 5;
registerFrame.add(scrollPane, gbc);

btnRegister.addActionListener(e -> {
    String trackingID = txtTrackingID.getText().trim();
    String location = txtLocation.getText().trim();
    String status = txtStatus.getText().trim();
    if (trackingID.isEmpty() || location.isEmpty() || status.isEmpty()) {
        txtAreaOutput.setText("Error: All fields must be filled out!");
    } else {
        parcelDatabase.put(trackingID, new String[]{location, status});
        txtAreaOutput.setText("Parcel updated successfully!\nTracking ID: " + trackingID);
```

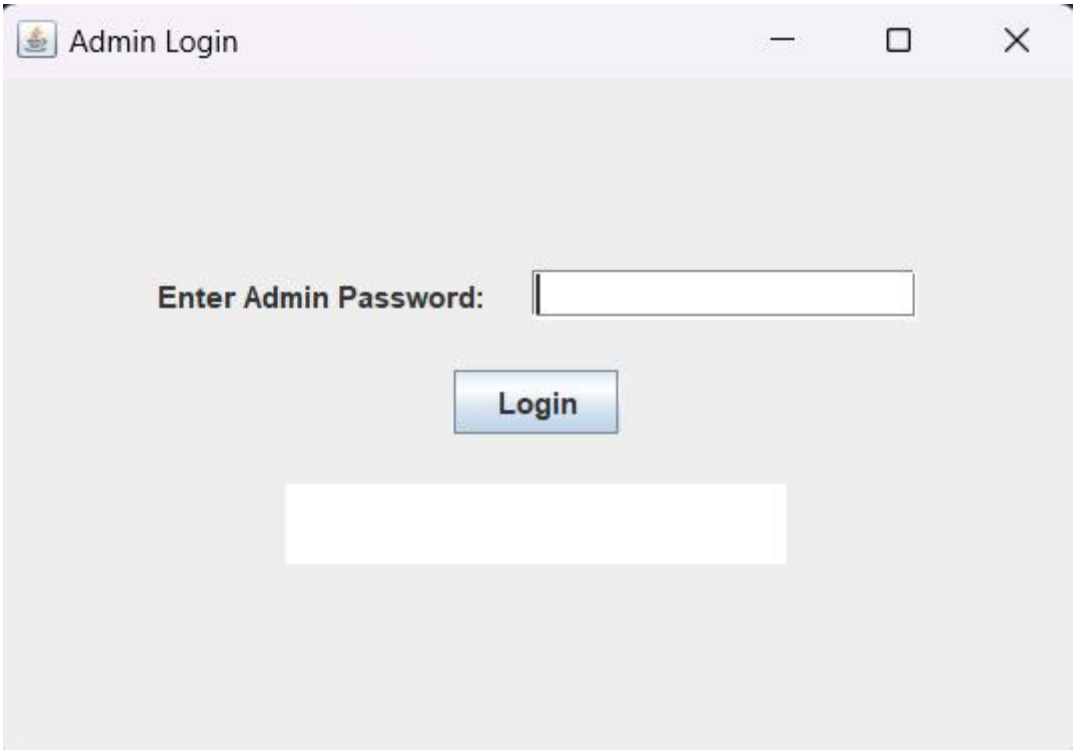
```
        }
    });

    registerFrame.setVisible(true);
}

// Track Parcel Window (Customer functionality)
void createTrackWindow() {
    JFrame trackFrame = new JFrame("Track Parcel");
    trackFrame.setLayout(new GridBagLayout());
    trackFrame.setSize(400, 300);
    trackFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JLabel lblTitle = new JLabel("Track a Parcel");
    lblTitle.setFont(new Font("Arial", Font.BOLD, 16));
    JLabel lblTrackingID = new JLabel("Tracking ID:");
    JTextField txtTrackingID = new JTextField(15);
    JButton btnTrack = new JButton("Track");

    JTextArea txtAreaOutput = new JTextArea(5, 20);
    txtAreaOutput.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(txtAreaOutput);

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(10, 10, 10, 10);
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    trackFrame.add(lblTitle, gbc);

    gbc.gridwidth = 1;
    gbc.gridy = 1;
    gbc.gridx = 0;
    trackFrame.add(lblTrackingID, gbc);
```

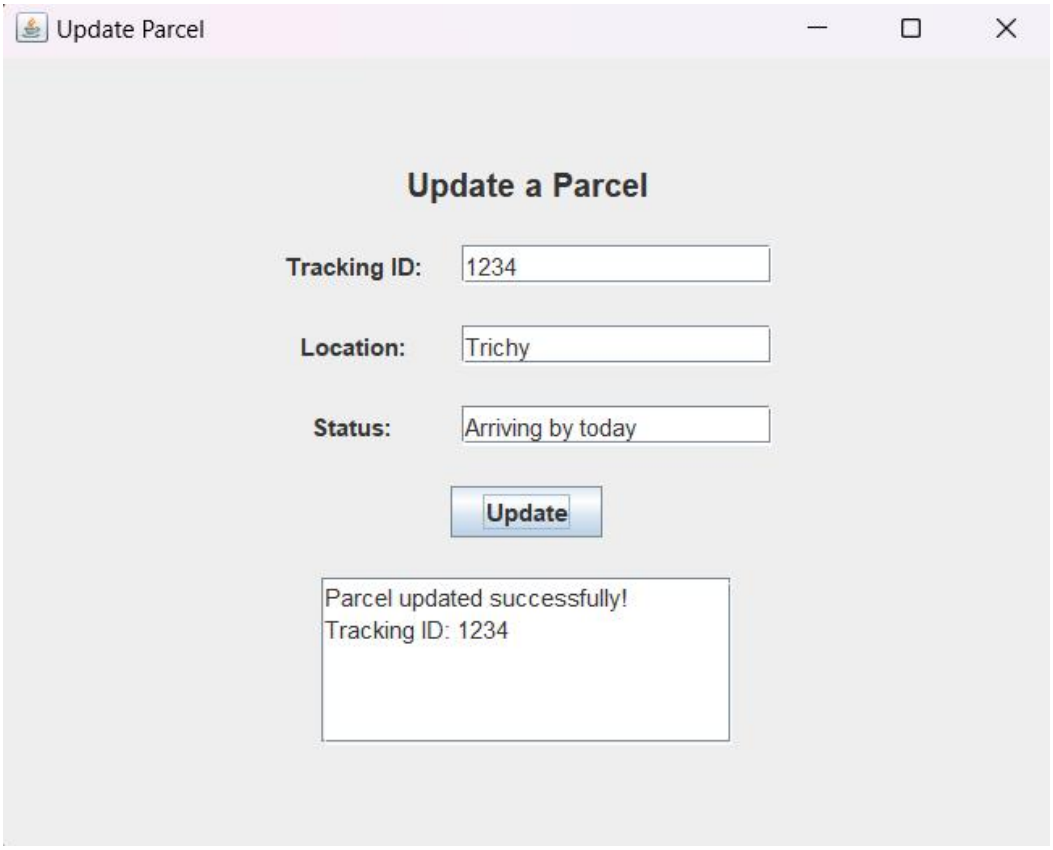
```
        gbc.gridx = 1;
        trackFrame.add(txtTrackingID, gbc);

        gbc.gridy = 2;
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        trackFrame.add(btnTrack, gbc);

        gbc.gridy = 3;
        trackFrame.add(scrollPane, gbc);

        btnTrack.addActionListener(e -> {
            String trackingID = txtTrackingID.getText().trim();
            if (trackingID.isEmpty()) {
                txtAreaOutput.setText("Error: Please enter a Tracking ID.");
            } else if (parcelDatabase.containsKey(trackingID)) {
                String[] details = parcelDatabase.get(trackingID);
                txtAreaOutput.setText("Tracking ID: " + trackingID + "\nLocation: " + details[0] +
"\nStatus: " + details[1]);
            } else {
                txtAreaOutput.setText("Error: Tracking ID not found!");
            }
        });

        trackFrame.setVisible(true);
    }

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

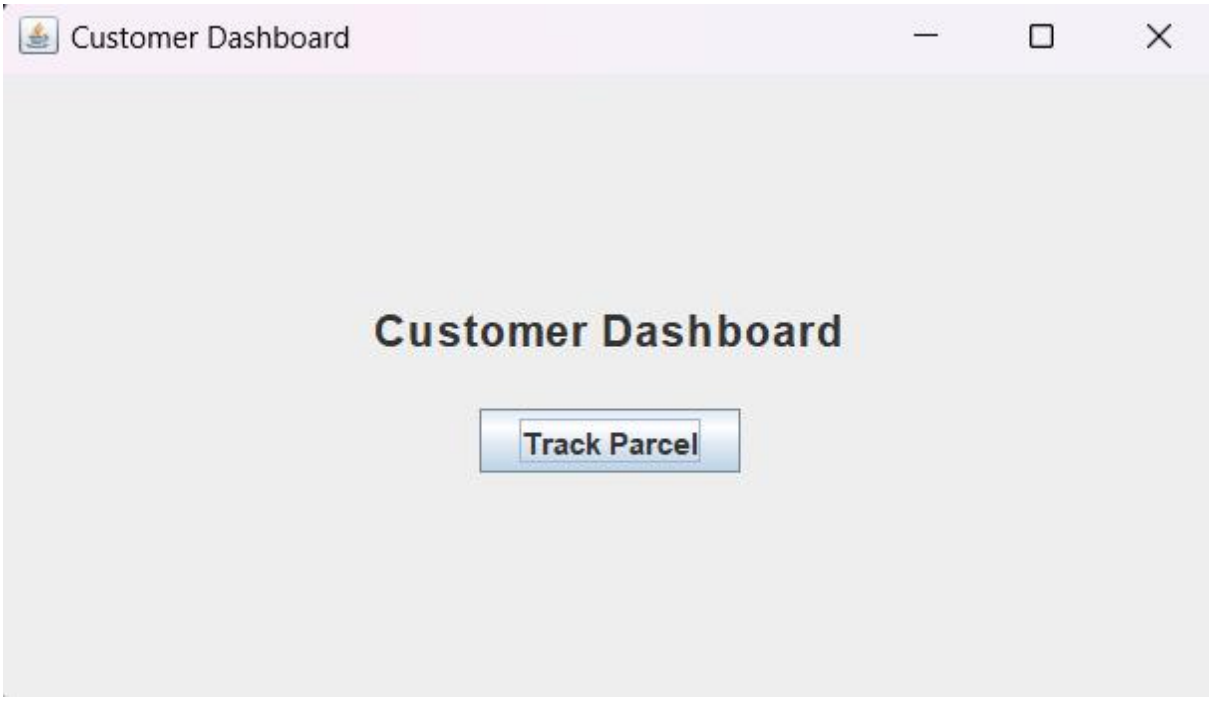
APPENDIX B

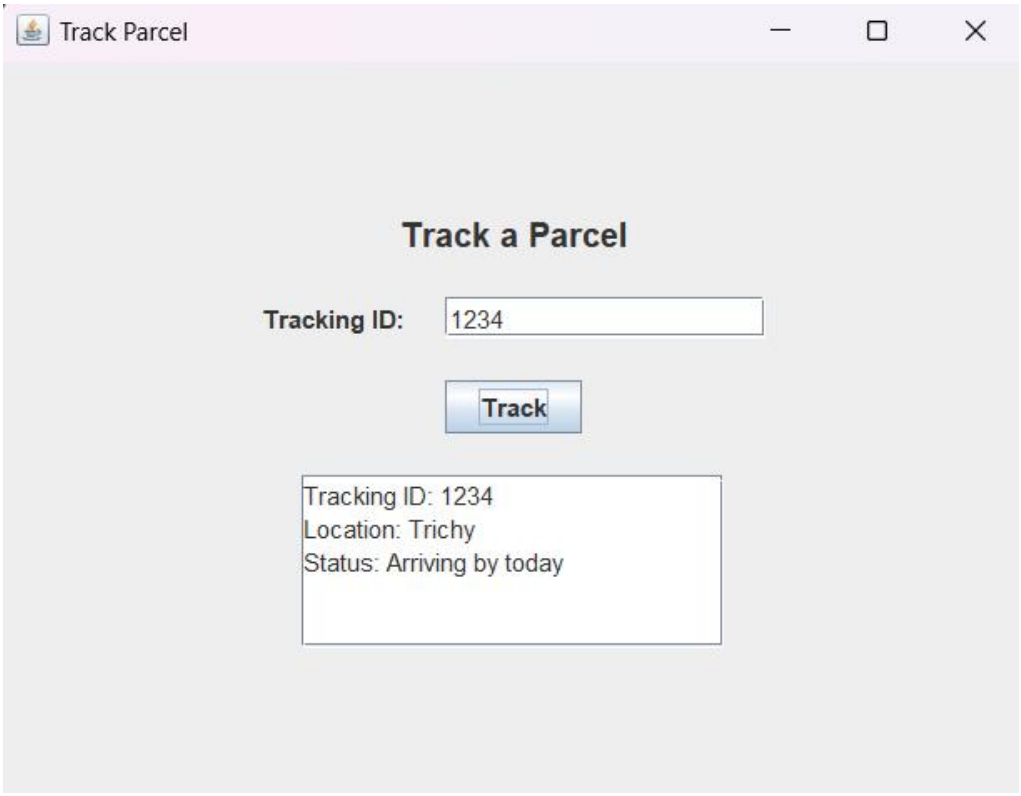
(SCREENSHOTS)

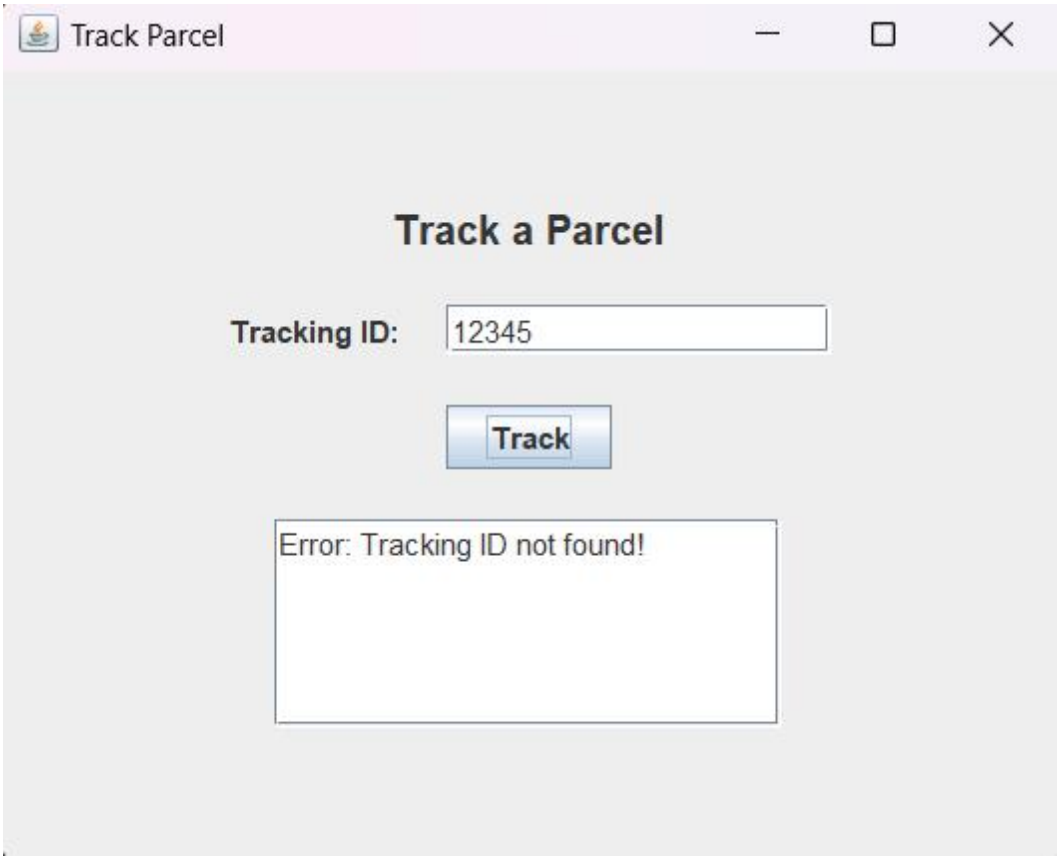





Update Parcel
Update a Parcel
Tracking ID:
1234
Location:
Trichy
Status:
Arriving by today
Update
Parcel updated successfully!
Tracking ID: 1234


Customer Dashboard
Customer Dashboard
Track Parcel


Track Parcel
Track a Parcel
Tracking ID:
1234
Track
Tracking ID: 1234
Location: Trichy
Status: Arriving by today


Track Parcel
Track a Parcel
Tracking ID:
12345
Track
Error: Tracking ID not found!

REFERENCES

WEBSITES

- https://chatgpt.com/share/674abf45-c51c-8002-b583-06902b0d7585
- https://github.com/Skentir/Courier-Delivery-App/blob/master/Parcel.java

YOUTUBE LINK

- https://youtu.be/Ha6TAOCP6yc?si=jf-Odus7SVpZ16H3
- https://youtu.be/yLIEhdJKmv4?si=-PkasusQC91JTK0S

BOOKS

- "Java: The Complete Reference" by Herbert Schildt
- "Head First Java" by Kathy Sierra and Bert Bates