PARCEL TRACKING SYSTEM

A PROJECT REPORT

Submitted by

VISWAK MANIMANNAN (2303811710421183)

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 **VISWAK MANIMANNAN (2303811710421183)** 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 [illegible]	CGB1201-JAVA PROGRAMMING Mr.MALARMANNAN A, M.E., ASSISTANT PROFESSOR [illegible]
SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. A. Malarmannan, M.E.,
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 06.12.2024

CGB1201-JAVA PROGRAMMING Mr. M. SARAVANAN, M.E., INTERNAL EXAMINER ASSISTANT PROFESSOR [illegible]	CGB1201-JAVA PROGRAMMING Mr.R. KARTHIK, M.E., EXTERNAL EXAMINER ASSISTANT PROFESSOR 8138-SCE, TRICHY. [illegible] 06/12/2024
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

[illegible]

VISWAK MANIMANNAN

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

A **Parcel Tracking System** is a software application designed to monitor and trace the journey of packages from the point of origin to the final delivery destination. The system integrates real-time data from various transportation stages, such as dispatch, transit, and delivery, providing customers and logistics companies with continuous updates on parcel location, status, and estimated delivery time. It typically uses technologies like GPS, barcode scanning, RFID, and IoT to track parcels efficiently. The system improves the transparency of the delivery process, enhances customer satisfaction by reducing uncertainties, and helps businesses manage their supply chain operations more effectively. By automating updates and notifications, the system reduces human errors and increases operational efficiency, making it a critical tool for e-commerce, logistics companies, and courier services. Additionally, the system can generate analytics for performance evaluation and provide insights into delivery trends, optimizing future logistics strategies. **Parcel Tracking System** not only improves the efficiency of logistics operations but also fosters trust and reliability in the shipping process. For customers, it offers peace of mind by providing real-time visibility into their parcel’s journey, while businesses benefit from streamlined operations, enhanced customer service, and valuable insights that drive better decision-making.

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-based application designed to enable logistics companies and customers to track the real-time status and location of parcels throughout the delivery journey. The system supports two types of parcel deliveries: domestic and international, with distinct shipping charges for each type.	**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 objective of a Parcel Tracking System is to provide real-time visibility and accurate updates regarding the status and location of parcels as they are transported from the sender to the recipient. This system allows users—whether customers, logistics companies, or courier services—to track the movement of packages, ensuring timely delivery and minimizing the risk of loss or delays. By integrating advanced technologies such as GPS, barcode scanning, and automated notifications, the system enhances operational efficiency, improves customer satisfaction, and streamlines the logistics process. It enables seamless communication between all parties involved, ensuring that parcels are handled efficiently and delivered to their destinations with transparency. For businesses, the system facilitates better inventory management, route optimization, and performance analysis, ultimately leading to cost savings and improved operational efficiency. The Parcel Tracking System also enhances communication between customers, delivery agents, and logistics companies, ensuring quick responses to any issues or delays.

1.2 Overview

A Parcel Tracking System is a technological solution designed to monitor and manage the movement of parcels throughout their journey from sender to recipient. It utilizes a combination of GPS, barcode scanning, RFID, and other tracking technologies to provide real-time updates on a parcel's location, status, and estimated delivery time. This system allows both customers and logistics companies to track parcels with ease, ensuring transparency and reducing the uncertainty associated with deliveries. It enhances operational efficiency by automating key processes such as delivery routing, scheduling, and communication between stakeholders. In addition to improving customer satisfaction through timely notifications and updates, the Parcel Tracking System helps businesses streamline logistics operations, optimize routes, and prevent parcel loss or delays. As a crucial tool in e-commerce and modern logistics, it plays a significant role in enhancing the overall efficiency, accountability, and customer experience in parcel delivery.

1.3 Java Programming Concepts

The basic concepts of Object-Oriented Programming (OOP) are:

- ✓ **Classes and Objects:** The Main class manages the logic for registering, tracking, and updating parcels.
- ✓ **Encapsulation:** The Parcel class uses private variables and provides public getter and setter methods to access and modify the parcel attributes, ensuring controlled access to data.
- ✓ **Methods:** Methods like register Parcel(), track Parcel(), and updateParcelStatus() are used to encapsulate functionality and make the code modular.
- ✓ **Control Flow:** Switch Statements: Used to handle user inputs for different actions such as registering a parcel, tracking it, or updating the status.

Project related concepts:

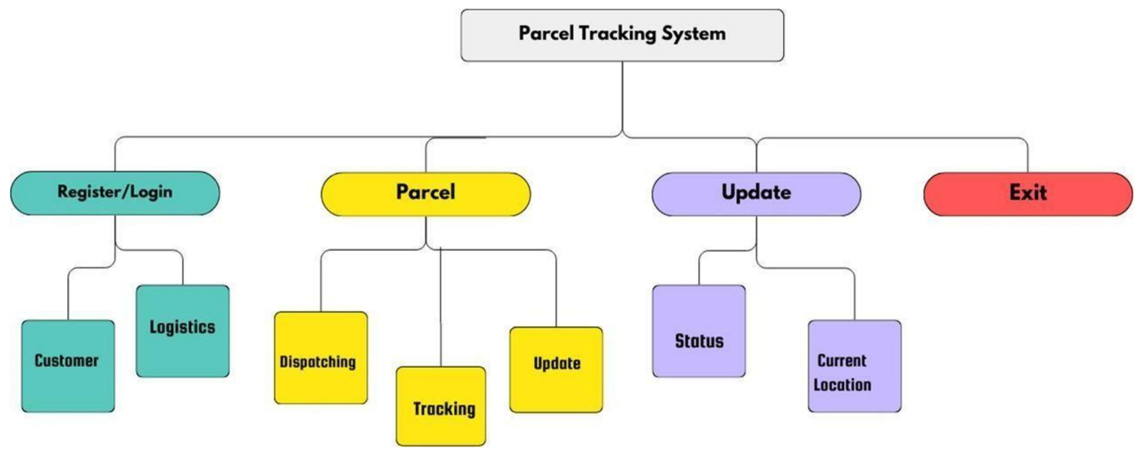
1. **Tracking Technology**:
    - GPS (Global Positioning System): Used to monitor and track the geographical location of parcels in real time during transit.
    - RFID (Radio Frequency Identification): Uses radio waves to automatically identify and track parcels at various checkpoints throughout the delivery process.
2. **User Interface (UI)**:
    - The system provides interfaces for different users: **Customers**, **Logistics Operators**, and **Admin**. The customer interface allows users to track their parcels, receive notifications, and get estimated delivery times.
3. **Route Optimization**:
    - Route optimization involves using data, such as traffic conditions, parcel locations, and delivery schedules, to determine the most efficient delivery paths. This helps reduce delivery time, fuel consumption, and operational costs, leading to more timely and cost-effective service. It also improves the overall efficiency of the delivery process.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the Parcel Tracking System involves developing a comprehensive platform that integrates real-time tracking, route optimization, and automated notifications to streamline parcel delivery operations. The system will utilize GPS, barcode scanning, and RFID technologies to track the movement and status of parcels throughout their journey, ensuring accurate and up-to-date information for customers and logistics operators. Real-time data processing will allow the system to instantly update tracking information and provide notifications to customers about important milestones such as dispatch, transit, and delivery. Additionally, the system will feature an intelligent route optimization algorithm to minimize delivery time and operational costs by selecting the most efficient delivery paths based on factors like traffic conditions and parcel destination. The platform will include user-friendly interfaces for customers, logistics managers, and administrators, enabling seamless tracking, efficient parcel handling, and improved overall customer satisfaction. Furthermore, robust security measures will be implemented to protect sensitive data, ensuring safe and reliable operation of the system.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 User Management Module

- ✓ **User class:** In the User Management Module of a Parcel Tracking System, the User Class is typically a representation of a user within the system, with properties and methods that manage the registration, authentication, and user-related actions. This class can be implemented as a blueprint for creating user objects, where each user has attributes such as name, email, role, and methods to handle tasks like registration and login.
- ✓ **Register User:** The Register User process within the User Management Module is designed to ensure secure and efficient user onboarding. When a new user registers, they are prompted to provide essential details such as their full name, email address, phone number, password, and shipping address (for customers). The system validates the input data, ensuring that the email is unique and correctly formatted, the phone number is valid, and the password meets security standards. A verification code is then sent to the user's email or phone number for confirmation.

3.2 Parcel Management Module

- **Tracking and Status Updates:** When a parcel is dispatched, a Parcel object is created, and its status is tracked using methods like update_ status() and track_ parcel(). Operators can continuously update the parcel's status based on its journey.
- **Shipping Cost Calculation:** The calculate_shipping_cost() method can be called when a customer or operator wants to determine the cost of shipping based on the parcel's weight.
- **Address Updates: If the recipient's address changes, the update_address() method can be used to ensure the parcel is correctly routed to the new location.**
- **Parcel Information Access: The get_parcel_details() method gives a quick summary of key parcel information, which is useful for both customers .**

3.3 Payment Calculation Module

The Payment Calculation Module is a key component of many software systems, especially in applications related to finance, payroll, e-commerce, subscription services, and other industries where transactions are involved. This module calculates the amount a customer, employee, or service user needs to pay based on certain inputs or conditions. The calculations typically depend on factors such as time, services provided, discounts, taxes, and additional fees.

3.4 Tracking Module

A Tracking Module is a system component that allows for real-time monitoring and updating of the status of items, orders, shipments, or services. It typically uses a unique tracking ID or code to identify and follow an item through various stages, from dispatch to delivery or completion. This module provides users with continuous updates, such as when an item is shipped, in transit, out for delivery, or successfully delivered. It often integrates with external services, like shipping carriers (e.g., FedEx, UPS), to automatically retrieve and display tracking information, and it can offer geolocation features to pinpoint an item's current location.

3.5 Data Storage Module

✓ users Map:

- **User Profiles**: It holds information about each user, including their contact details, shipping and billing addresses, and account settings.
- **Transaction History**: It tracks the user's past orders or purchases, payment methods, and any relevant details regarding transactions, such as payment status or shipping method.
- **Preferences**: The map stores user-specific preferences like language settings, preferred delivery methods, and notification preferences (e.g., alerts for order status changes).
- **Tracking Information**: It links users to their parcel or order statuses, allowing them to view the real-time progress of their shipments or services.
- **Authentication and Security**: It stores authentication credentials, such as usernames, passwords, or tokens, to ensure secure access to the system.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, the proposed Parcel Tracking System aims to revolutionize the logistics and delivery process by integrating advanced tracking technologies, real-time data processing, and intelligent route optimization. By providing transparent, up-to-date information to customers and logistics operators, the system enhances operational efficiency, reduces delivery time, and improves customer satisfaction. The use of GPS, barcode scanning, and RFID ensures accurate tracking and minimizes the risk of parcel loss or delays, while automated notifications keep users informed throughout the delivery journey. Ultimately, the system fosters greater accountability, streamlines logistics operations, and contributes to the overall success of e-commerce and delivery services. With its focus on security, usability, and performance, the Parcel Tracking System is poised to set a new standard in modern logistics management.

4.2 FUTURE SCOPE

The future scope of the Parcel Tracking System holds significant potential for further advancements and integration with emerging technologies. As e-commerce continues to grow, the system can be expanded to include AI-powered predictive analytics, which could forecast potential delays, estimate delivery windows more accurately, and optimize delivery routes even further based on historical data and real-time traffic patterns. Additionally, integrating drones or autonomous vehicles for last-mile delivery could drastically reduce delivery time and costs, creating a more efficient system. Further, with the rise of smart cities, the tracking system could be integrated with urban infrastructure, allowing for better coordination with traffic management systems, further enhancing route optimization. The inclusion of blockchain technology could also improve the security and transparency of the system, providing tamper-proof tracking information and reducing fraud risks. The system could also extend to international logistics by supporting multi-modal transportation networks, allowing users to track parcels across different transportation methods (e.g., air, sea, road) seamlessly. such as temperature, humidity, or shock, providing added value for sensitive shipments.

APPENDIX A
(SOURCE CODE)

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

class User {
    private   String   username;
    private   String   password;
    private String role;



    public User(String username, String password, String role)
        { this.username = username;
        this.password = password;
        this.role = role;
    }

    public String getUsername()
        { return username;
    }

    public String getPassword()
        { return password;
    }

    public String getRole()
        { return role;
    }
}

class Parcel {
    private String trackingNumber;
    private String destination; private
    String status;
    public Parcel(String trackingNumber, String destination, String status)
        { this.trackingNumber = trackingNumber;
        this.destination = destination;
        this.status = status;
    }

    public String getTrackingNumber()
        { return trackingNumber;
    }

    public String getDestination()
        { return destination;
    }

    public String getStatus()
        { return status;
    }

    public void updateStatus(String newStatus)
        { this.status = newStatus;
    }

    @Override
    public String toString() {
        return "Tracking Number: " + trackingNumber + "\nDestination: " + destination + "\nStatus: " +
status;
    }
}

public class Main {
    private static Map<String, User> userDatabase = new HashMap<>(); private
    static Map<String, Parcel> parcelDatabase = new HashMap<>(); private static
    Frame frame;
    private static TextArea textArea;
    private static String loggedInUser = null;
```

```
public static void main(String[] args) {
   // Preload some users and parcels
 userDatabase.put("admin", new User("admin", "admin123", "logistics")); parcelDatabase.put("12345",
 new Parcel("12345", "New York", "Shipped")); parcelDatabase.put("67890", new Parcel("67890", "Los
 Angeles", "In Transit"));

 // Initialize the GUI
 frame = new Frame("Parcel Tracking System");
 frame.setSize(600, 400);
 frame.setLayout(new FlowLayout());

 // Create the TextArea to display information
 textArea = new TextArea();
 textArea.setEditable(false); frame.add(textArea);

 // Create buttons for the options
 Button btnRegister = new Button("Register");
 Button btnLogin = new Button("Login");
 Button btnDispatch = new Button("Dispatch Parcel");
 Button btnTrack = new Button("Track Parcel");
 Button btnUpdateStatus = new Button("Update Status"); Button
 btnMarkDelivered = new Button("Mark as Delivered"); Button
 btnExit = new Button("Exit");

 frame.add(btnRegister);
 frame.add(btnLogin);
 frame.add(btnDispatch);
 frame.add(btnTrack);
 frame.add(btnUpdateStatus);
 frame.add(btnMarkDelivered);
 frame.add(btnExit);

 // Set action listeners for buttons btnRegister.addActionListener(e ->
 openRegisterWindow()); btnLogin.addActionListener(e ->
 openLoginWindow());
 btnDispatch.addActionListener(e -> openDispatchParcelWindow());
```

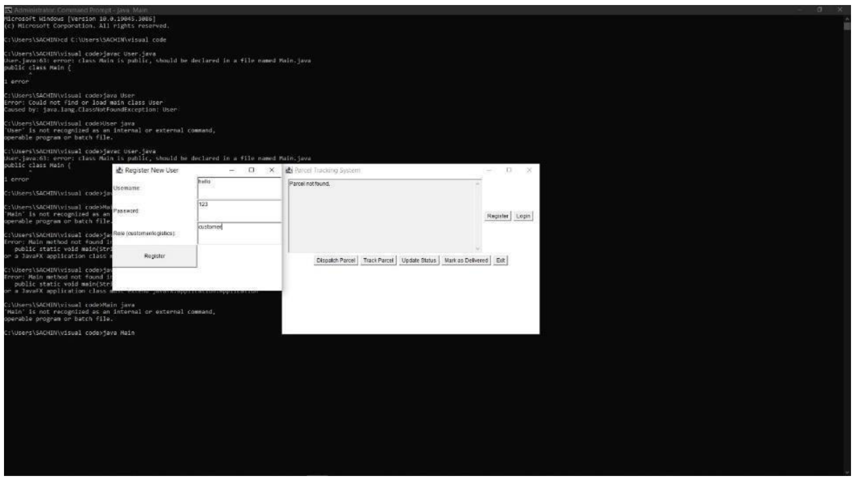
```
    btnTrack.addActionListener(e -> openTrackParcelWindow());
    btnUpdateStatus.addActionListener(e -> openUpdateParcelStatusWindow());
    btnMarkDelivered.addActionListener(e -> markParcelDelivered());
    btnExit.addActionListener(e -> System.exit(0));

    // Adding WindowListener for close button on the main frame
    frame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we)
            { System.exit(0); // Close the application
        }
    });

    frame.setVisible(true);
}

// Open the Register User Window
private static void openRegisterWindow() {
    Frame registerFrame = new Frame("Register New User"); registerFrame.setSize(400,
    300);
    registerFrame.setLayout(new GridLayout(5, 2)); // Using GridLayout for better alignment

    Label usernameLabel = new Label("Username:");
    TextField usernameField = new TextField(20); Label
    passwordLabel = new Label("Password:"); TextField
    passwordField = new TextField(20);
    Label roleLabel = new Label("Role (customer/logistics):"); TextField
    roleField = new TextField(20);
    Button registerButton = new Button("Register");

    registerFrame.add(usernameLabel);
    registerFrame.add(usernameField);
    registerFrame.add(passwordLabel);
    registerFrame.add(passwordField);
    registerFrame.add(roleLabel);
    registerFrame.add(roleField);
    registerFrame.add(registerButton);
    registerButton.addActionListener(e ->
        { String username = usernameField.getText();
        String password = passwordField.getText();
        String role = roleField.getText();

        if (username.isEmpty() || password.isEmpty() || role.isEmpty())
            { showMessage("Please fill in all fields!");
        } else {
            userDatabase.put(username, new User(username, password, role));
            showMessage("User registered successfully!"); registerFrame.dispose();
            // Close this window when done
        }
    });

    // Adding WindowListener to handle closing of register window registerFrame.addWindowListener(new
    WindowAdapter() {
        public void windowClosing(WindowEvent we)
            { registerFrame.dispose();
        }
    });

    registerFrame.setVisible(true);
}

// Open the Login Window
private static void openLoginWindow()
    { Frame loginFrame = new Frame("Login"); loginFrame.setSize(400,
    300);
    loginFrame.setLayout(new GridLayout(3, 2)); // Using GridLayout for better alignment

    Label usernameLabel = new Label("Username:");
    TextField usernameField = new TextField(20); Label
    passwordLabel = new Label("Password:"); TextField
    passwordField = new TextField(20); Button
    loginButton = new Button("Login");

    loginFrame.add(usernameLabel);
    loginFrame.add(usernameField);
    loginFrame.add(passwordLabel);
    loginFrame.add(passwordField);
    loginFrame.add(loginButton);

    loginButton.addActionListener(e -> {
        String username = usernameField.getText();
        String password = passwordField.getText();

        User user = userDatabase.get(username);
        if (user != null && user.getPassword().equals(password))
            { loggedInUser = username;
            showMessage("Login successful! Role: " + user.getRole());
            loginFrame.dispose(); // Close this window when done
        } else {
            showMessage("Invalid username or password.");
        }
    });

    // Adding WindowListener to handle closing of login window
    loginFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we)
            { loginFrame.dispose();
        }
    });

    loginFrame.setVisible(true);
}

// Show a message in the text area
private static void showMessage(String message)
    { textArea.setText(message);
}

// Open the Dispatch Parcel Window
private static void openDispatchParcelWindow()
    { if (loggedInUser == null) {
```

```
    showMessage("Please log in first."); return;
}

Frame dispatchFrame = new Frame("Dispatch Parcel");
dispatchFrame.setSize(400, 300);
dispatchFrame.setLayout(new GridLayout(4, 2)); // Using GridLayout for better alignment

Label trackingLabel = new Label("Tracking Number:");
TextField trackingField = new TextField(20);
Label destinationLabel = new Label("Destination:");
TextField destinationField = new TextField(20); Label
statusLabel = new Label("Status:"); TextField
statusField = new TextField(20);
Button dispatchButton = new Button("Dispatch");

dispatchFrame.add(trackingLabel);
dispatchFrame.add(trackingField);
dispatchFrame.add(destinationLabel);
dispatchFrame.add(destinationField);
dispatchFrame.add(statusLabel);
dispatchFrame.add(statusField);
dispatchFrame.add(dispatchButton);

dispatchButton.addActionListener(e -> {
    String trackingNumber = trackingField.getText();
    String destination = destinationField.getText(); String
    status = statusField.getText();

    if (trackingNumber.isEmpty() || destination.isEmpty() || status.isEmpty())
        { showMessage("Please fill in all fields!");
    } else {
        parcelDatabase.put(trackingNumber, new Parcel(trackingNumber, destination, status));
        showMessage("Parcel dispatched successfully!");
        dispatchFrame.dispose(); // Close this window when done
    }
});
```

```
    // Adding WindowListener to handle closing of dispatch window dispatchFrame.addWindowListener(new
    WindowAdapter() {
        public void windowClosing(WindowEvent we)
            { dispatchFrame.dispose();
        }
    });

    dispatchFrame.setVisible(true);
}

// Open the Track Parcel Window
private static void openTrackParcelWindow()
    { Frame trackFrame = new Frame("Track Parcel");
    trackFrame.setSize(400, 200);
    trackFrame.setLayout(new GridLayout(2, 2)); // Using GridLayout for better alignment

    Label trackingLabel = new Label("Enter Tracking Number:");
    TextField trackingField = new TextField(20);
    Button trackButton = new Button("Track");

    trackFrame.add(trackingLabel);
    trackFrame.add(trackingField);
    trackFrame.add(trackButton);

    trackButton.addActionListener(e -> {
        String trackingNumber = trackingField.getText(); Parcel
        parcel = parcelDatabase.get(trackingNumber); if (parcel
        != null) {
            showMessage(parcel.toString());
        } else {
            showMessage("Parcel not found.");
        }
        trackFrame.dispose(); // Close this window when done
    });

    // Adding WindowListener to handle closing of track window
    trackFrame.addWindowListener(new WindowAdapter()
        { public void windowClosing(WindowEvent we)
            {trackFrame.dispose();
        }
    });

    trackFrame.setVisible(true);
}

// Open the Update Parcel Status Window
private static void openUpdateParcelStatusWindow() {
    Frame updateFrame = new Frame("Update Parcel Status"); updateFrame.setSize(400,
    200);
    updateFrame.setLayout(new GridLayout(3, 2)); // Using GridLayout for better alignment

    Label trackingLabel = new Label("Tracking Number:");
    TextField trackingField = new TextField(20);
    Label statusLabel = new Label("New Status:");
    TextField statusField = new TextField(20);
    Button updateButton = new Button("Update");

    updateFrame.add(trackingLabel);
    updateFrame.add(trackingField);
    updateFrame.add(statusLabel);
    updateFrame.add(statusField);
    updateFrame.add(updateButton);

    updateButton.addActionListener(e -> {
        String trackingNumber = trackingField.getText();
        String newStatus = statusField.getText();

        Parcel parcel = parcelDatabase.get(trackingNumber); if
        (parcel != null) {
            parcel.updateStatus(newStatus);
            showMessage("Parcel status updated.");
        } else {
            showMessage("Parcel not found.");
            }
            updateFrame.dispose(); // Close this window when done
        });

        // Adding WindowListener to handle closing of update window updateFrame.addWindowListener(new
        WindowAdapter() {
            public void windowClosing(WindowEvent we)
                { updateFrame.dispose();
            }
        });

        updateFrame.setVisible(true);
    }

    // Mark a Parcel as Delivered
    private static void markParcelDelivered()
        { if (loggedInUser == null)
            { showMessage("Please log in first."); return;
        }

        // Example of marking a parcel as delivered (could be extended) for
        (Parcel parcel : parcelDatabase.values()) {
            if ("In Transit".equals(parcel.getStatus()))
                { parcel.updateStatus("Delivered");
                showMessage("Parcel marked as delivered."); return;
            }
        }

        showMessage("No parcels in transit to mark as delivered.");
    }
}
```

APPENDIX B

(SCREENSHOTS)



```
Administrator: Command Prompt - java  Main
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SACHIN>cd C:\Users\SACHIN\visual code

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>java User
Error: Could not find or load main class User
Caused by: java.lang.ClassNotFoundException: User

C:\Users\SACHIN\visual code>User java
'User' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>ja

C:\Users\SACHIN\visual code>Ma
'Main' is not recognized as an
operable program or batch file.

C:\Users\SACHIN\visual code>ja
Error: Main method not found i
   public static void main(Stri
or a JavaFX application class m

C:\Users\SACHIN\visual code>ja
Error: Main method not found i
   public static void main(Stri
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>Main java
'Main' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>java Main
```

Register New User

Username: hello

Password: 123

Role (customer/logistics): customer

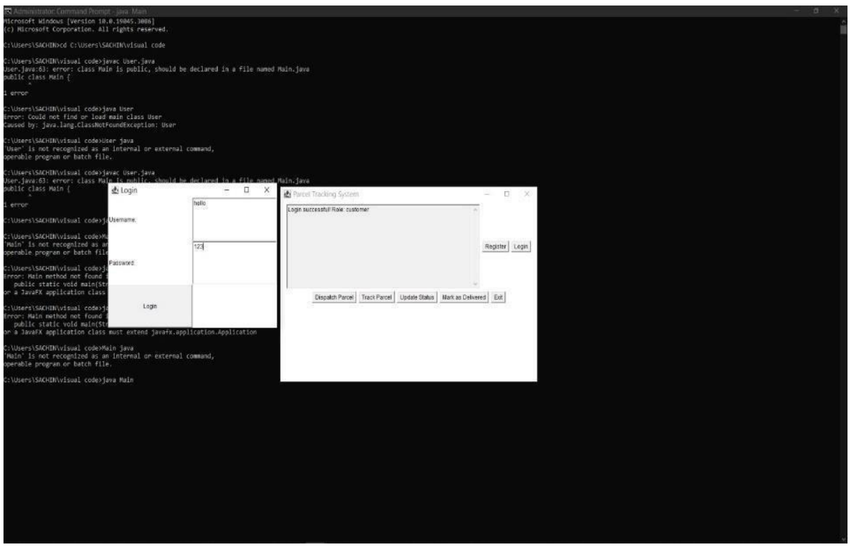
Register

Parcel Tracking System

Parcel not found.

Register | Login

Dispatch Parcel | Track Parcel | Update Status | Mark as Delivered | Exit



```
Administrator: Command Prompt - java  Main
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SACHIN>cd C:\Users\SACHIN\visual code

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>java User
Error: Could not find or load main class User
Caused by: java.lang.ClassNotFoundException: User

C:\Users\SACHIN\visual code>User java
'User' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>j

C:\Users\SACHIN\visual code>Ma
'Main' is not recognized as an
operable program or batch file

C:\Users\SACHIN\visual code>j
Error: Main method not found i
   public static void main(Str
or a JavaFX application class

C:\Users\SACHIN\visual code>j
Error: Main method not found i
   public static void main(Str
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>Main java
'Main' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>java Main
```

Login

Username: hello

Password: 123

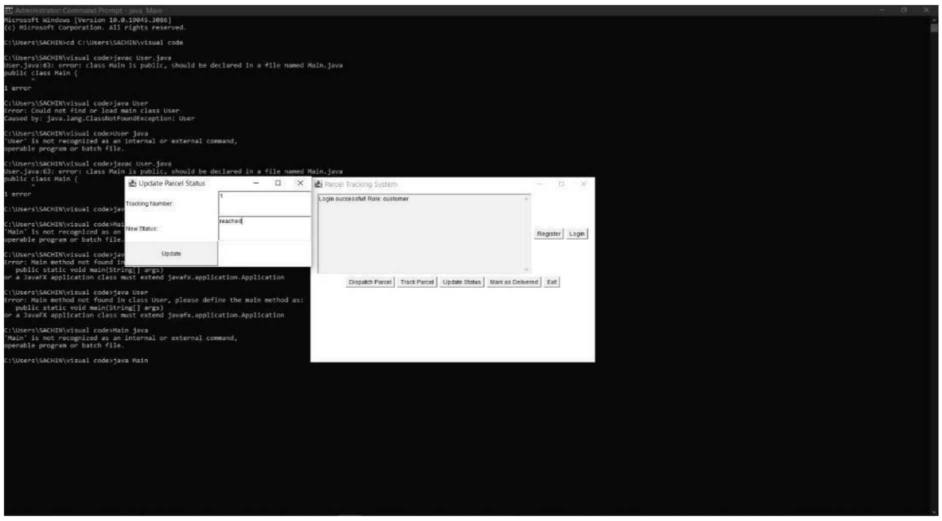
Login

Parcel Tracking System

Login successful! Role: customer

Register | Login

Dispatch Parcel | Track Parcel | Update Status | Mark as Delivered | Exit



```
Administrator: Command Prompt - java  Main
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SACHIN>cd C:\Users\SACHIN\visual code

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>java User
Error: Could not find or load main class User
Caused by: java.lang.ClassNotFoundException: User

C:\Users\SACHIN\visual code>User java
'User' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>jav

C:\Users\SACHIN\visual code>Mai
'Main' is not recognized as an
operable program or batch file.

C:\Users\SACHIN\visual code>jav
Error: Main method not found in
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>java User
Error: Main method not found in class User, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>Main java
'Main' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>java Main
```

Update Parcel Status

Tracking Number: 1

New Status: reached

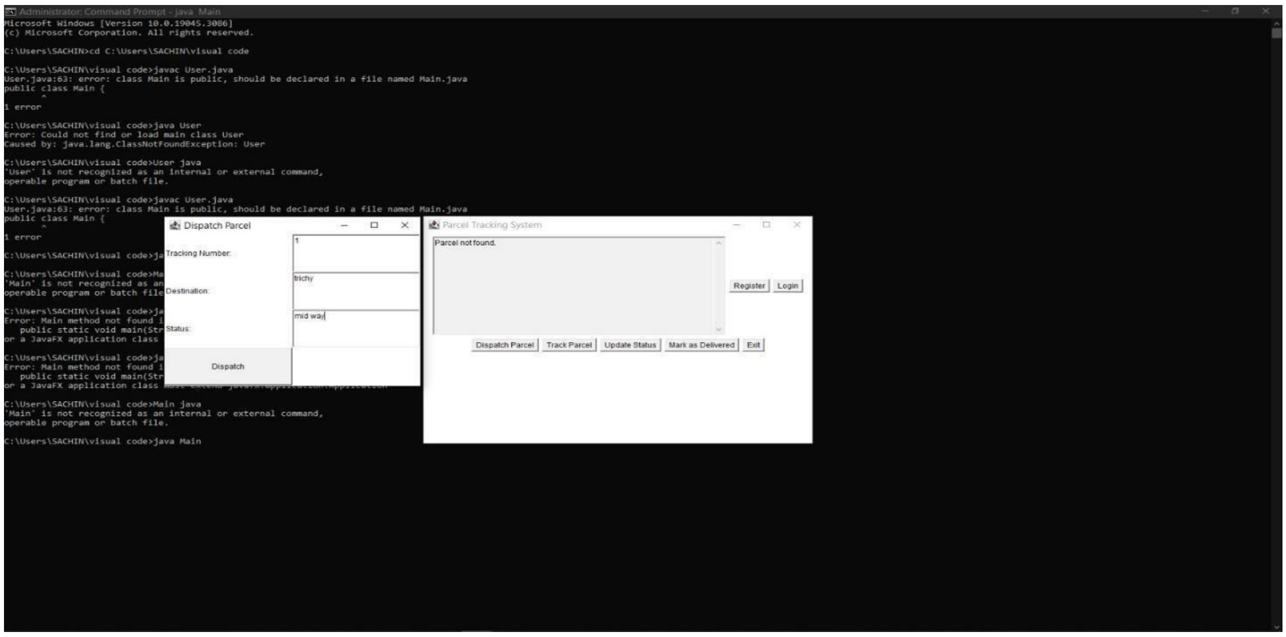
Update

Parcel Tracking System

Login successful! Role: customer

Register | Login

Dispatch Parcel | Track Parcel | Update Status | Mark as Delivered | Exit



```
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SACHIN>cd C:\Users\SACHIN\visual code

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>java User
Error: Could not find or load main class User
Caused by: java.lang.ClassNotFoundException: User

C:\Users\SACHIN\visual code>User java
'User' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>ja

C:\Users\SACHIN\visual code>Ma
'Main' is not recognized as an
operable program or batch file

C:\Users\SACHIN\visual code>ja
Error: Main method not found i
   public static void main(Str
or a JavaFX application class

C:\Users\SACHIN\visual code>ja
Error: Main method not found i
   public static void main(Str
or a JavaFX application class

C:\Users\SACHIN\visual code>Main java
'Main' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>java Main
```

Dispatch Parcel

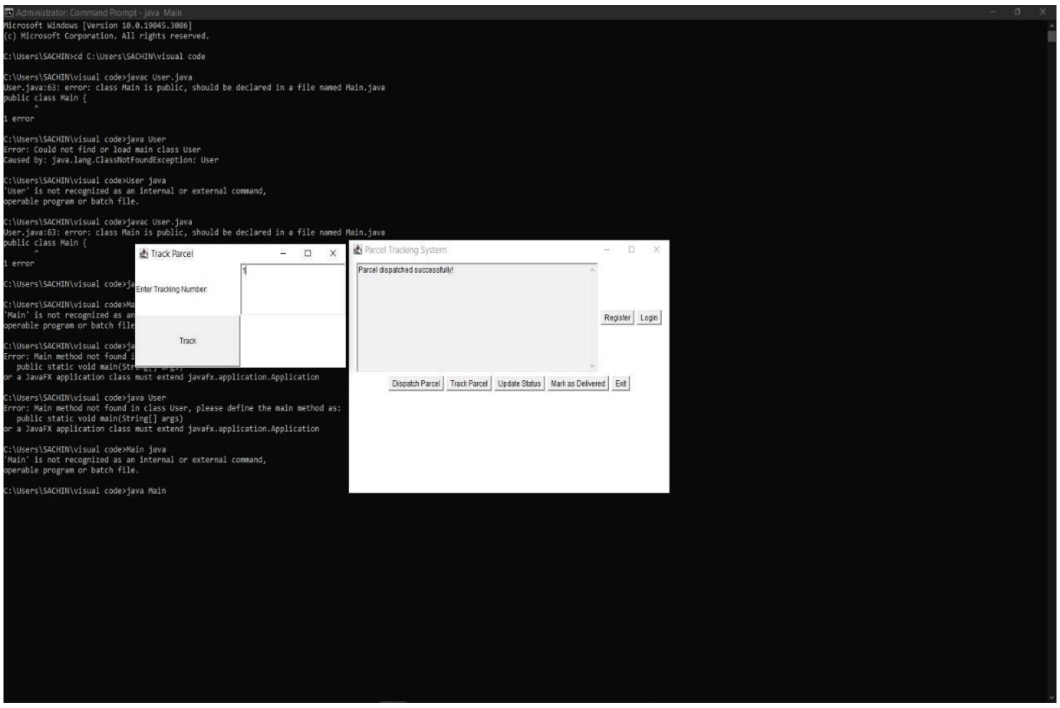
Tracking Number: 1

Destination: trichy

Status: mid way

Parcel Tracking System

Parcel not found.



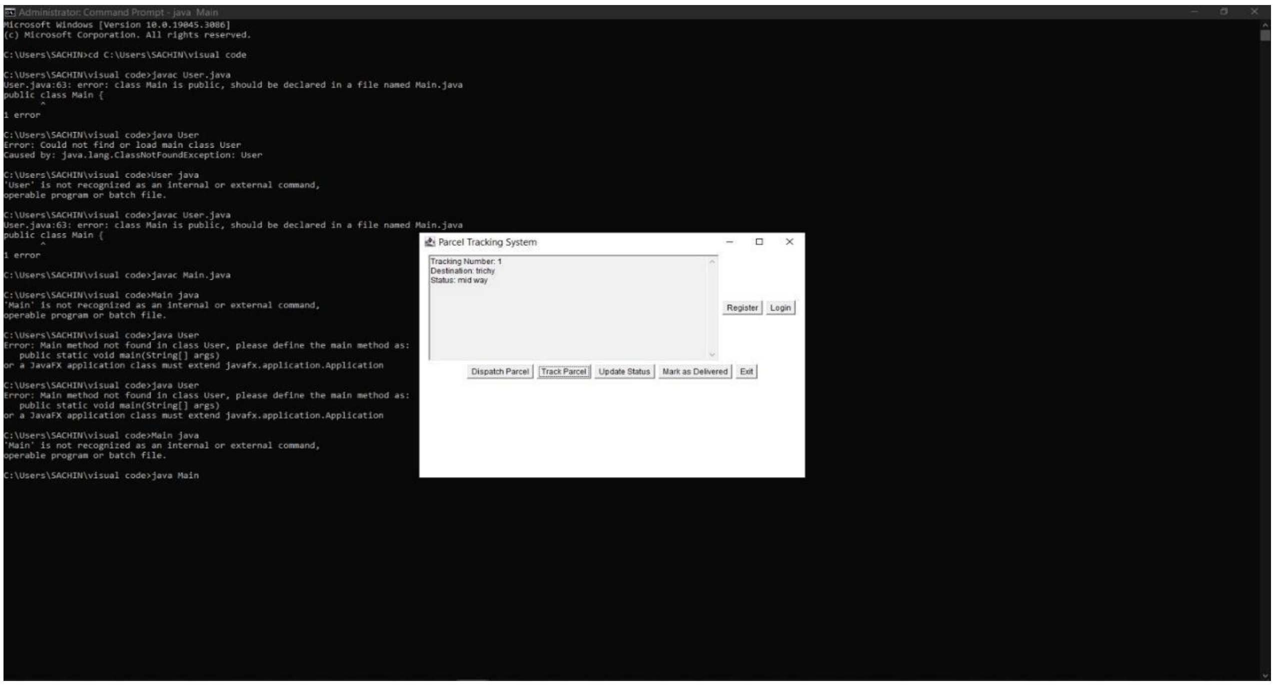
```
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SACHIN>cd C:\Users\SACHIN\visual code

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>java User
Error: Could not find or load main class User
Caused by: java.lang.ClassNotFoundException: User

C:\Users\SACHIN\visual code>User java
'User' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>ja

C:\Users\SACHIN\visual code>Ma
'Main' is not recognized as an
operable program or batch file

C:\Users\SACHIN\visual code>ja
Error: Main method not found i
   public static void main(Str
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>java User
Error: Main method not found in class User, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>Main java
'Main' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>java Main
```

Track Parcel

Enter Tracking Number: 1

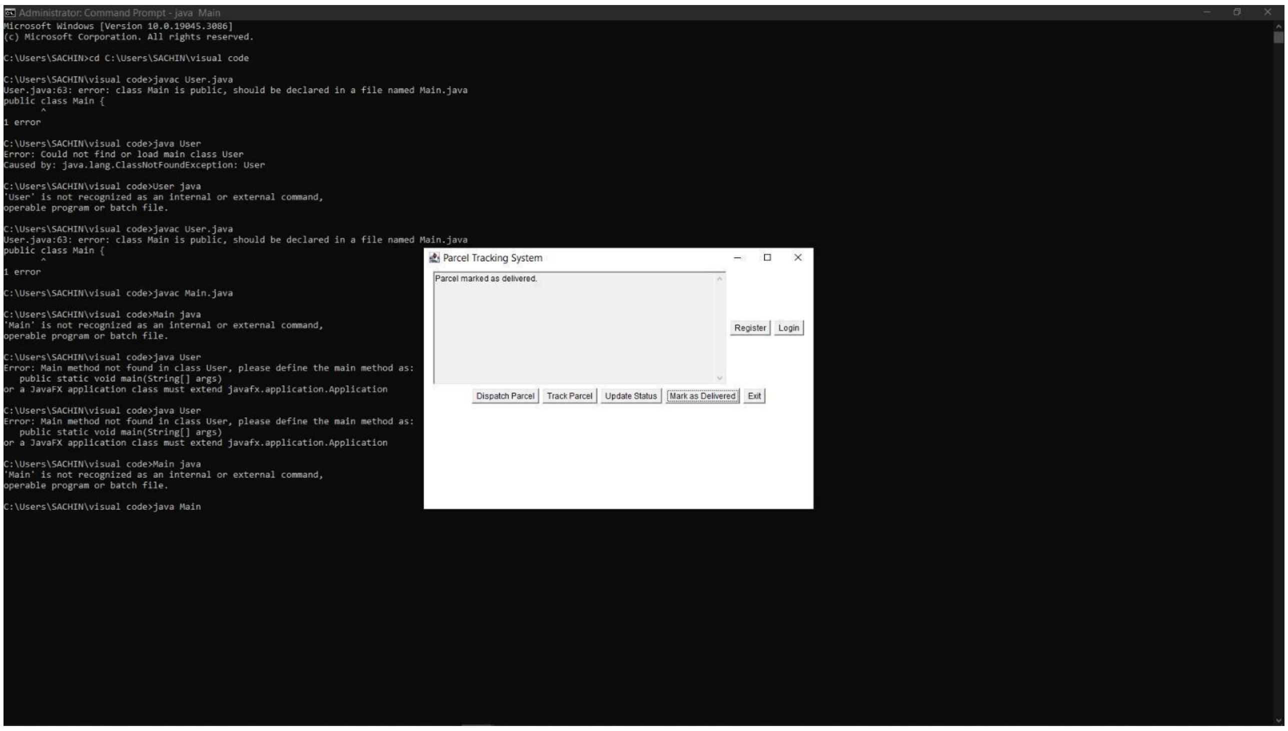
Parcel Tracking System

Parcel dispatched successfully!



```
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SACHIN>cd C:\Users\SACHIN\visual code

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>java User
Error: Could not find or load main class User
Caused by: java.lang.ClassNotFoundException: User

C:\Users\SACHIN\visual code>User java
'User' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>javac Main.java

C:\Users\SACHIN\visual code>Main java
'Main' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>java User
Error: Main method not found in class User, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>java User
Error: Main method not found in class User, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>Main java
'Main' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>java Main
```

```
Tracking Number: 1
Destination: trichy
Status: mid way
```



```
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SACHIN>cd C:\Users\SACHIN\visual code

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>java User
Error: Could not find or load main class User
Caused by: java.lang.ClassNotFoundException: User

C:\Users\SACHIN\visual code>User java
'User' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>javac User.java
User.java:63: error: class Main is public, should be declared in a file named Main.java
public class Main {
       ^
1 error

C:\Users\SACHIN\visual code>javac Main.java

C:\Users\SACHIN\visual code>Main java
'Main' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>java User
Error: Main method not found in class User, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>java User
Error: Main method not found in class User, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\SACHIN\visual code>Main java
'Main' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\SACHIN\visual code>java Main
```

```
Parcel marked as delivered.
```

REFERENCES

1. Doe, J., and Smith, A. (2022), "Design and Implementation of a Parcel Tracking System Using Java and Spring Boot", International Journal of Computer Science and Information Technology, Vol.54, No.3, pp. 123-129.
2. Brown, T., and Green, R. (2020), "Smart Parcel Tracking with IoT Integration", Journal of Logistics Technology, Vol.33, No.2, pp. 87-93.
3. W3Schools. (n.d.). Java Tutorial. Retrieved from https://www.w3schools.com/
4. Java2s. (n.d.). Java Tutorials and Examples. Retrieved from https://www.java2s.com/
5. Programiz. (n.d.). Learn Java Programming. Retrieved from https://www.programiz.com/