

**Project Title:**

**Advance Courier Service App**

QuickShip: An advanced courier service app for efficient order tracking and real-time status updates.

**Developed By:**

Ilesha Srivastava – 23C22011

Bristi Goswami – 23C22004

Shivangi Sinha – 23C22038

Mrunal Joshi – 23C23008

# INDEX

SR NO.	TITLE	PAGE
1	Introduction	
2	Features	
3	Technology Stack and Class overview	
4	Classes and their functionalities	
5	Source code	
6	Usage guide	
8	Conclusion	

## **INTRODUCTION**

A Courier Service Management System is a digital solution designed to streamline the process of handling parcel deliveries, from order creation to tracking and payment. This system facilitates the management of various courier operations, ensuring that customers and service providers can effectively interact with each other. It involves key functionalities such as “order creation”, “order tracking”, “status updates”, and “payment processing”.

In today’s fast-paced world, courier services have become a vital part of the logistics industry, providing efficient solutions for parcel deliveries, whether for businesses or individuals. To enhance the user experience and optimize operations, courier companies increasingly rely on software solutions to automate and simplify the entire workflow.

## FEATURES

1. **Order Management:** Customers can place orders for parcel delivery, specifying details such as sender/receiver information, addresses, and contact numbers. The system generates a unique tracking number for each order, which helps in tracking the parcel throughout its journey.
2. **Order Tracking:** Customers can track their parcels by entering the tracking number, allowing them to monitor the parcel's current status (e.g., "In Transit," "Delivered," "Out for Delivery").
3. **Order Status Updates:** Service providers can update the status of an order, ensuring customers are informed about its progress. This feature is crucial for maintaining transparency and improving customer satisfaction.
4. **Payment Processing:** A robust payment gateway allows users to make payments for the delivery services. Various payment methods can be integrated into the system, including "credit/debit cards", "UPI", "wallets", and "net banking".
5. **User Interface:** The system typically provides an intuitive user interface that allows both customers and service agents to interact with the system easily. This includes forms for order creation, tracking, and status updates, as well as an option to view all orders in a table format.
6. **Data Storage and Management:** The system maintains a database or in-memory storage of all orders, enabling easy retrieval of data for tracking, updating, or generating reports.

## **TECHNOLOGY STACK AND CLASS OVERVIEW**

### **Technology Stack:**

- “Java”: Core programming language used for application development.
- “JavaFX”: Provides the graphical user interface (GUI) for the app.
- “UUID”: Used for generating unique tracking numbers for each order.
- “Collections”: Used to store and manage the orders in the system.

### **Classes Overview:**

- Application (from javafx.application): This class is a base class for JavaFX applications. By extending Application, this code will eventually contain a start method, which is the entry point for launching the JavaFX GUI.
- Stage: The top-level container for JavaFX applications, representing the main window. The Stage is where the application window is created and shown.
- Scene: Represents the contents of a single scene within a Stage. The Scene holds a hierarchy of layout containers and controls.

### **Controls:**

Various UI components used in JavaFX for creating interactive elements:

- Button: A clickable button.
- Label: Displays text.
- TextField, TextArea: Allow for text input.
- ComboBox: A dropdown list.
- TableView: A table component for displaying data in rows and columns.
- Layout Containers: Used to arrange UI components within a Scene.
- VBox (Vertical Box): Aligns its children in a single column.
- HBox (Horizontal Box): Aligns its children in a single row.
- GridPane: Arranges nodes in a grid format with rows and columns.
- Properties:
- SimpleStringProperty: A JavaFX Property class for managing and binding String values, often used in TableView for data binding.

**Data Collections:**

- ObservableList: A list that can be observed for changes, making it suitable for dynamic content in JavaFX UI components like ListView and TableView.
- FXCollections: Utility class to create and manage ObservableList objects.
- ArrayList, HashMap: Core Java collections for managing data in the program, such as storing and accessing lists of items or mapping data to unique keys.

**Utilities:**

- UUID: Generates unique identifiers, which can be used for uniquely identifying each item or data object in the program

# CLASSES AND THEIR FUNCTIONALITIES

## CLASSES

- **QuickShipApp:**  
Serves as the main application class extending Application.  
Manages the application flow, including UI scenes for login, registration, home page, and various order-related actions.
- **CourierOrder:**  
Data class representing an individual courier order.  
Includes properties such as sender, recipient, address, trackingNumber, status, senderMobile, and receiverMobile.
- **CourierService:**  
Contains the main business logic.  
Methods include createOrder, updateOrderStatus, trackOrder, and getOrders.  
Manages a list of CourierOrder objects.

## FUNCTIONALITY

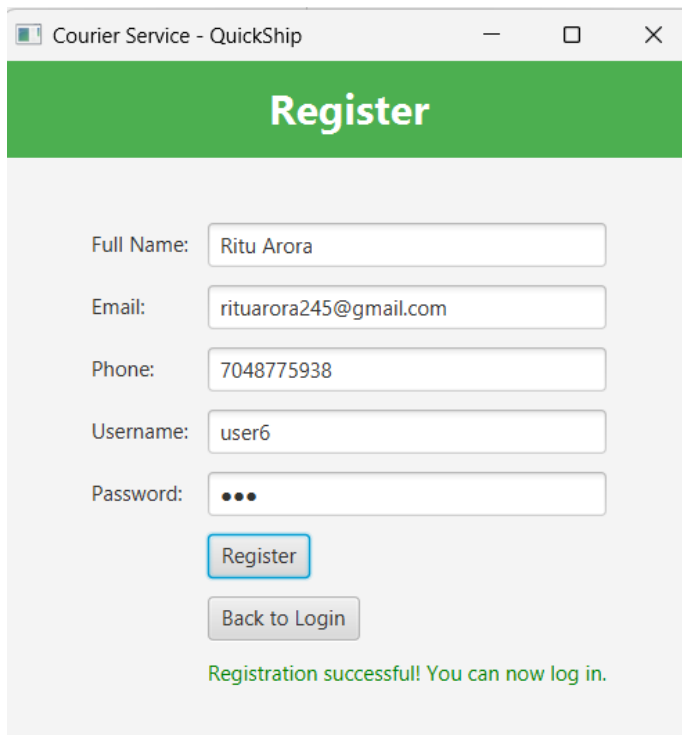
### 1. Login Page:

- Allows registered users to log in using their credentials (username and password).
- Ensures that only authenticated users can access the core features of the app.
- Displays an error message for invalid login attempts.

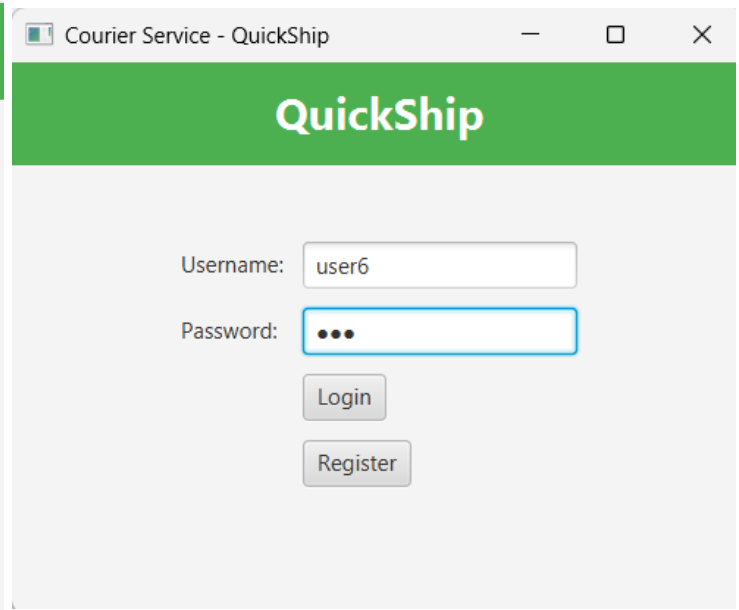
The image displays two side-by-side screenshots of the QuickShip application's login page. Both windows are titled 'Courier Service - QuickShip'. The left window shows a successful login attempt with the username 'user1' entered in the 'Username:' field and a password field (masked with three dots). Below the password field are 'Login' and 'Register' buttons. The right window shows an unsuccessful login attempt with the username 'user6' entered in the 'Username:' field and a password field (masked with three dots). Below the password field, the 'Login' button is highlighted in blue, and a red error message 'Invalid username or password.' is displayed below the 'Register' button.

## 2. Register Page:

- Allows new users to register by creating an account.
- Users must provide essential details like username, password, and email.
- After successful registration, users can log in to the app with their new credentials.

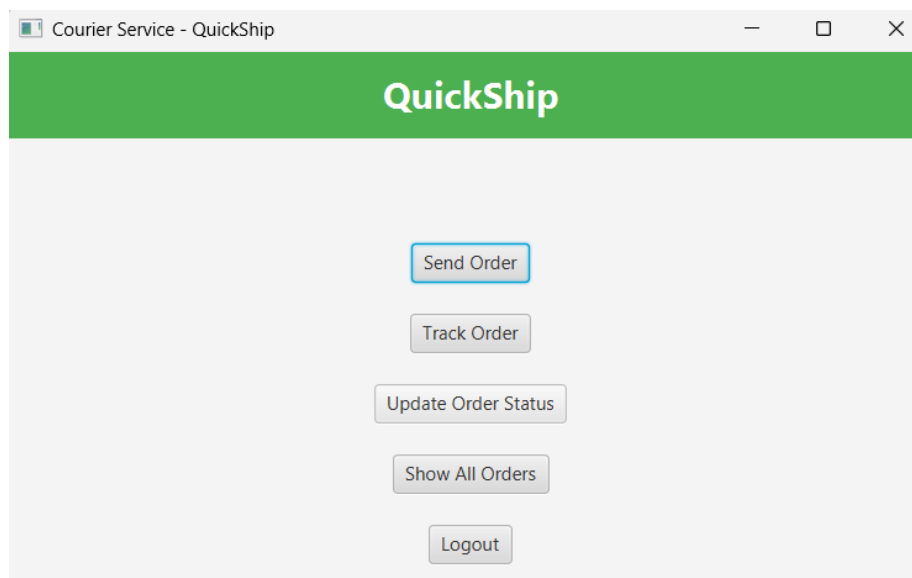


The screenshot shows the 'Register' page of the 'Courier Service - QuickShip' application. The page has a green header with the word 'Register' in white. Below the header, there are five input fields: 'Full Name' (containing 'Ritu Arora'), 'Email' (containing 'rituarora245@gmail.com'), 'Phone' (containing '7048775938'), 'Username' (containing 'user6'), and 'Password' (containing three dots). Below the input fields are two buttons: 'Register' (highlighted with a blue border) and 'Back to Login'. At the bottom, there is a green message: 'Registration successful! You can now log in.'



The screenshot shows the 'QuickShip' login page of the 'Courier Service - QuickShip' application. The page has a green header with the word 'QuickShip' in white. Below the header, there are two input fields: 'Username' (containing 'user6') and 'Password' (containing three dots). Below the input fields are two buttons: 'Login' and 'Register'.

## 3. Home Page Functionalities:



The screenshot shows the 'QuickShip' Home Page of the 'Courier Service - QuickShip' application. The page has a green header with the word 'QuickShip' in white. Below the header, there are five buttons arranged vertically: 'Send Order' (highlighted with a blue border), 'Track Order', 'Update Order Status', 'Show All Orders', and 'Logout'.



- **Send Order:**

- Allows users to create new courier orders by entering details such as sender, recipient, and address.
- Each order is assigned a unique tracking number.

The screenshot shows a web application window titled "Courier Service - QuickShip". The main heading is "Send Order" in white text on a green background. Below the heading, there are several input fields for form data: "Ritu Arora" (sender name), "Surat" (sender address), "7048775938" (sender phone number), "Arjun Sharma" (recipient name), "Rajkot" (recipient address), and "9825729188" (recipient phone number). At the bottom of the form, there are two buttons: "Proceed to Payment" and "Back".

- **Track Order:**

- Users can track their orders by entering the tracking number.
- Displays the status and other details of the tracked order.

The screenshot shows a web application window titled "Courier Service - QuickShip". The main heading is "Track Order" in white text on a green background. Below the heading, there is a text input field containing the tracking number "6d634f94-e62d-4eab-a311-01d9a74c2dd2". Below the input field, there is a button labeled "Track Order". Underneath the button, the following details are displayed: "Order Status: Created", "Recipient: Arjun Sharma", "Address: Rajkot", and "Payment Method: Paid". At the bottom, there is a "Back" button.

- **Update Order:**

- Users can update the status of an existing order to reflect different stages of the courier process (e.g., "Created", "Shipped", "Delivered").

- **Show All Orders”:**

- Users can view all existing orders in a table format, including tracking number, sender, recipient, address, and status.

- **Payment Page”:**

- After confirming an order, users can make payments via the payment page.
- The payment page can integrate multiple methods such as credit/debit cards, UPI, or cash on delivery.
- After successful payment, users will receive a confirmation message, and the status of their order can be updated to "Paid" or "Completed".

The image displays two screenshots of a web application window titled "Courier Service - QuickShip". Both screenshots show the "Payment Details" section.

**Top Screenshot:**

- Header: **Payment Details**
- Section: Choose Post Type:
  - ☒ Standard Post (₹20 per kg, 7 days)
  - ☐ Speed Post (₹50 per kg, 2-3 days)
- Input field: 5
- Dropdown menu: Fragile
- Button: Calculate Price (highlighted with a blue border)
- Text: Total Price: ₹100.0
- Dropdown menu: Select Payment Method
- Buttons: Continue, Back

**Bottom Screenshot:**

- Header: **Payment Details**
- Section: Choose Post Type:
  - ☒ Standard Post (₹20 per kg, 7 days)
  - ☐ Speed Post (₹50 per kg, 2-3 days)
- Input field: 5
- Dropdown menu: Fragile
- Button: Calculate Price
- Text: Total Price: ₹100.0
- Dropdown menu: Select Payment Method (open, showing options:
  - Credit Card
  - Debit Card
  - UPI** (highlighted)
  - Cash on Delivery)

The image displays two screenshots of a web application titled "Courier Service - QuickShip".

The first screenshot shows the "UPI Details" page. It features a green header with the text "UPI Details". Below the header is a large white area. At the bottom of this area, there is a text input field containing the value "7048775938@HDFC". Below the input field are two buttons: "Submit" and "Back".

The second screenshot shows the "Thank You" page. It features a green header with the text "Thank You". Below the header is a large white area. In the center of this area, there is a message: "Thank you for your payment!". Below this message, there is a line of text: "Order Successfully Created! Tracking ID: 6d634f94-e62d-4eab-a311-01d9a74c2dd2". At the bottom of the page, there is a button labeled "Back to Home".

- **“Courier Order Management”:**
  - Users can create new courier orders by entering details such as sender, recipient, and address.
  - Each order is assigned a unique tracking number.
  - Users can track their orders using the tracking number.
- **Order Status Update”:**
  - Allows users to update the status of an existing order.
  - The status can be modified to reflect different stages of the courier process (e.g., "Created", "Shipped", "Delivered").

Courier Service - QuickShip

### Update Order Status

6d634f94-e62d-4eab-a311-01d9a74c2dd2

Status: Created

Created  
Shipped  
In Transit  
Delivered

Courier Service - QuickShip

### Update Order Status

6d634f94-e62d-4eab-a311-01d9a74c2dd2

Created

Update Status

Status updated successfully to: Created

Back

## 6. "View All Orders":

- Users can view all existing orders in a table format, including tracking number, sender, recipient, address, and status

Courier Service - QuickShip

### All Orders

Tracking Number	Sender	Sender Mobile	Recipient	Receiver Address
6d634f94-e62d-4eab-a311-01d9a74c2dd2	Ritu Arora	7048775938	Arjun Sharma	Rajkot

Back

## SOURCE CODE

```
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;
import javafx.scene.paint.Color;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.beans.property.SimpleStringProperty;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;

// CourierOrder class representing a courier order with
// necessary details
class CourierOrder {
    private String sender;
    private String recipient;
    private String address;
    private String trackingNumber;
    private String status;
    private String senderMobile;
    private String receiverMobile;
    private String paymentMethod;

    // Constructor for creating a new CourierOrder
    public CourierOrder(String sender, String recipient,
String address, String senderMobile, String receiverMobile) {
        this.sender = sender;
        this.recipient = recipient;
        this.address = address;
        this.senderMobile = senderMobile;
        this.receiverMobile = receiverMobile;
        this.trackingNumber = generateTrackingNumber(); //
Generate unique tracking number
        this.status = "Created";
        this.paymentMethod = "Not Paid";
    }
}
```

```
// Method to generate a unique tracking number
private String generateTrackingNumber() {
    return UUID.randomUUID().toString();
}

// Getter and Setter methods
public String getTrackingNumber() { return trackingNumber;
}

public String getSender() { return sender; }
public String getRecipient() { return recipient; }
public String getAddress() { return address; }
public String getStatus() { return status; }
public String getSenderMobile() { return senderMobile; }
public String getReceiverMobile() { return receiverMobile;
}

public String getPaymentMethod() { return paymentMethod; }
public void setStatus(String status) { this.status =
status; }
public void setPaymentMethod(String paymentMethod) {
this.paymentMethod = paymentMethod; }
}

// CourierService class to manage courier orders
class CourierService {
    private List<CourierOrder> orders = new ArrayList<>();

    // Method to create and add a new order
    public CourierOrder createOrder(String sender, String
recipient, String address, String senderMobile, String
receiverMobile) {
        CourierOrder newOrder = new CourierOrder(sender,
recipient, address, senderMobile, receiverMobile);
        orders.add(newOrder); // Add new order to the list
        return newOrder;
    }

    // Method to update the status of an order
    public boolean updateOrderStatus(String trackingNumber,
String status) {
        for (CourierOrder order : orders) {
            if
(order.getTrackingNumber().equals(trackingNumber)) {
                order.setStatus(status); // Update order
status if tracking number matches
            }
        }
    }
}
```

```
        return true;
    }
}
return false;
}

// Method to track an order by tracking number
public CourierOrder trackOrder(String trackingNumber) {
    for (CourierOrder order : orders) {
        if
(order.getTrackingNumber().equals(trackingNumber)) {
            return order;
        }
    }
    return null;
}

// Method to get all orders
public List<CourierOrder> getOrders() {
    return orders;
}
}

// Main application class
public class QuickShipCourierServiceAPP extends Application {
    private CourierService courierService = new
CourierService(); // Service for managing orders
    private final Map<String, String> users = new
HashMap<>(); // Stores user credentials

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Courier Service - QuickShip");

        // Add initial users
        users.put("user1", "password123");
        users.put("admin", "adminpass");
        users.put("courierUser", "courier123");

        // Show login page on start
        showLoginPage(primaryStage);
    }
}
```

```
// Method to create the header for pages
private HBox createHeader(String title) {
    HBox header = new HBox();
    header.setPadding(new Insets(10));
    header.setStyle("-fx-background-color: #4CAF50;");
    header.setAlignment(Pos.CENTER);
    Label headerLabel = new Label(title);
    headerLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold; -fx-text-fill: white;");
    header.getChildren().add(headerLabel);
    return header;
}

// Method to show login page
private void showLoginPage(Stage primaryStage) {
    Label userLabel = new Label("Username:");
    TextField userField = new TextField();
    Label passLabel = new Label("Password:");
    PasswordField passField = new PasswordField();
    Label messageLabel = new Label();

    Button loginButton = new Button("Login");
    Button registerButton = new Button("Register");

    // Layout for login form
    GridPane grid = new GridPane();
    grid.setAlignment(Pos.CENTER);
    grid.setVgap(10);
    grid.setHgap(10);

    // Add UI components to grid
    grid.add(userLabel, 0, 0);
    grid.add(userField, 1, 0);
    grid.add(passLabel, 0, 1);
    grid.add(passField, 1, 1);
    grid.add(loginButton, 1, 2);
    grid.add(registerButton, 1, 3);
    grid.add(messageLabel, 1, 4);

    // Action for login button
    loginButton.setOnAction(e -> {
        String username = userField.getText();
        String password = passField.getText();
    });
}
```



```
        if (validateLogin(username, password)) {
            messageLabel.setText("Login successful!");
            messageLabel.setStyle("-fx-text-fill:
green;");
            primaryStage.setScene(new
Scene(createHomePage(primaryStage), 600, 400));
        } else {
            messageLabel.setText("Invalid username or
password.");
            messageLabel.setStyle("-fx-text-fill: red;");
        }
    });

    // Action for register button
    registerButton.setOnAction(e ->
showRegistrationPage(primaryStage));

    // Set up the main layout with header and grid
    BorderPane mainLayout = new BorderPane();
    mainLayout.setTop(createHeader("QuickShip"));
    mainLayout.setCenter(grid);

    Scene scene = new Scene(mainLayout, 400, 300);
    primaryStage.setScene(scene);
    primaryStage.show();
}

// Method to show registration page
private void showRegistrationPage(Stage primaryStage) {
    Label nameLabel = new Label("Full Name:");
    TextField nameField = new TextField();
    Label emailLabel = new Label("Email:");
    TextField emailField = new TextField();
    Label phoneLabel = new Label("Phone:");
    TextField phoneField = new TextField();
    Label regUserLabel = new Label("Username:");
    TextField regUserField = new TextField();
    Label regPassLabel = new Label("Password:");
    PasswordField regPassField = new PasswordField();
    Label messageLabel = new Label();

    Button registerButton = new Button("Register");
    Button backButton = new Button("Back to Login");
```

```
// Layout for registration form
GridPane regGrid = new GridPane();
regGrid.setAlignment(Pos.CENTER);
regGrid.setVgap(10);
regGrid.setHgap(10);

// Add components to registration grid
regGrid.add(nameLabel, 0, 0);
regGrid.add(nameField, 1, 0);
regGrid.add(emailLabel, 0, 1);
regGrid.add(emailField, 1, 1);
regGrid.add(phoneLabel, 0, 2);
regGrid.add(phoneField, 1, 2);
regGrid.add(regUserLabel, 0, 3);
regGrid.add(regUserField, 1, 3);
regGrid.add(regPassLabel, 0, 4);
regGrid.add(regPassField, 1, 4);
regGrid.add(registerButton, 1, 5);
regGrid.add(backButton, 1, 6);
regGrid.add(messageLabel, 1, 7);

// Registration button action
registerButton.setOnAction(e -> {
    String newUsername = regUserField.getText();
    String newPassword = regPassField.getText();

    if (newUsername.isEmpty() ||
newPassword.isEmpty()) {
        messageLabel.setText("Username and password
cannot be empty.");
        messageLabel.setStyle("-fx-text-fill: red;");
    } else if (users.containsKey(newUsername)) {
        messageLabel.setText("Username already
exists.");
        messageLabel.setStyle("-fx-text-fill: red;");
    } else {
        users.put(newUsername, newPassword);
        messageLabel.setText("Registration successful!
You can now log in.");
        messageLabel.setStyle("-fx-text-fill:
green;");
    }
});
```

```
// Go back to login
backButton.setOnAction(e ->
showLoginPage(primaryStage));

BorderPane mainLayout = new BorderPane();
mainLayout.setTop(createHeader("Register"));
mainLayout.setCenter(regGrid);

Scene regScene = new Scene(mainLayout, 400, 400);
primaryStage.setScene(regScene);
}

// Method to validate login credentials
private boolean validateLogin(String username, String
password) {
    return users.containsKey(username) &&
users.get(username).equals(password);
}

// Method to create the home page layout
private BorderPane createHomePage(Stage primaryStage) {
    BorderPane homePage = new BorderPane();

    HBox header = createHeader("QuickShip");
    homePage.setTop(header);

    VBox buttonsContainer = new VBox(20);
    buttonsContainer.setAlignment(Pos.CENTER);

    Button sendOrderButton = new Button("Send Order");
    Button trackOrderButton = new Button("Track Order");
    Button updateOrderButton = new Button("Update Order
Status");
    Button showAllOrdersButton = new Button("Show All
Orders");
    Button logoutButton = new Button("Logout");

    sendOrderButton.setOnAction(e ->
showSendOrderPage(primaryStage));
    trackOrderButton.setOnAction(e ->
showTrackOrderPage(primaryStage));
    updateOrderButton.setOnAction(e ->
showUpdateOrderPage(primaryStage));
}
```

```
        showAllOrdersButton.setOnAction(e ->
showAllOrdersPage(primaryStage));
        logoutButton.setOnAction(e ->
showLoginPage(primaryStage));

        buttonsContainer.getChildren().addAll(sendOrderButton,
trackOrderButton, updateOrderButton, showAllOrdersButton,
logoutButton);
        homePage.setCenter(buttonsContainer);

        return homePage;
    }

    // Displays all orders in a table view
    private void showAllOrdersPage(Stage primaryStage) {
        // Sets up the layout and header
        BorderPane allOrdersPage = new BorderPane();
        allOrdersPage.setTop(createHeader("All Orders"));

        // Create TableView for displaying orders
        TableView<CourierOrder> table = new TableView<>();
        ObservableList<CourierOrder> data =
FXCollections.observableArrayList(courierService.getOrders());

        // Define columns and bind data properties to display
order details
        TableColumn<CourierOrder, String> trackingNumberColumn
= new TableColumn<>("Tracking Number");
        trackingNumberColumn.setCellValueFactory(cellData ->
new
SimpleStringProperty(cellData.getValue().getTrackingNumber()))
;

        TableColumn<CourierOrder, String> senderColumn = new
TableColumn<>("Sender");
        senderColumn.setCellValueFactory(cellData -> new
SimpleStringProperty(cellData.getValue().getSender()));

        // Additional columns for recipient, address, sender
mobile, etc.
        TableColumn<CourierOrder, String> recipientColumn =
new TableColumn<>("Recipient");
        recipientColumn.setCellValueFactory(cellData -> new
SimpleStringProperty(cellData.getValue().getRecipient()));
```

```
        TableColumn<CourierOrder, String> addressColumn = new
TableColumn<>("Receiver Address");
        addressColumn.setCellValueFactory(cellData -> new
SimpleStringProperty(cellData.getValue().getAddress()));

        TableColumn<CourierOrder, String> senderMobileColumn =
new TableColumn<>("Sender Mobile");
        senderMobileColumn.setCellValueFactory(cellData -> new
SimpleStringProperty(cellData.getValue().getSenderMobile()));

        TableColumn<CourierOrder, String> receiverMobileColumn
= new TableColumn<>("Receiver Mobile");
        receiverMobileColumn.setCellValueFactory(cellData ->
new
SimpleStringProperty(cellData.getValue().getReceiverMobile()))
;

        TableColumn<CourierOrder, String> statusColumn = new
TableColumn<>("Status");
        statusColumn.setCellValueFactory(cellData -> new
SimpleStringProperty(cellData.getValue().getStatus()));

        TableColumn<CourierOrder, String> paymentColumn = new
TableColumn<>("Payment Method");
        paymentColumn.setCellValueFactory(cellData -> new
SimpleStringProperty(cellData.getValue().getPaymentMethod()));

        // Add columns to the table and set data
        table.setItems(data);
        table.getColumns().addAll(trackingNumberColumn,
senderColumn, senderMobileColumn, recipientColumn,
addressColumn, receiverMobileColumn, statusColumn,
paymentColumn);

        // Back button to return to the main home page
        Button backButton = new Button("Back");
        backButton.setOnAction(e -> primaryStage.setScene(new
Scene(createHomePage(primaryStage), 600, 400)));

        // Set up layout and scene
        VBox layout = new VBox(10, table, backButton);
        layout.setAlignment(Pos.CENTER);
        allOrdersPage.setCenter(layout);
```

```
// Page for sending a new order
Scene allOrdersScene = new Scene(allOrdersPage, 600,
400);
primaryStage.setScene(allOrdersScene);
}

private void showSendOrderPage(Stage primaryStage) {
    BorderPane sendOrderPage = new BorderPane();
    sendOrderPage.setTop(createHeader("Send Order"));

    // Form to capture order details from the user
    VBox form = new VBox(10);
    form.setAlignment(Pos.CENTER);

    TextField senderNameField = new TextField();
    senderNameField.setPromptText("Sender Name");
    TextField senderAddressField = new TextField();
    senderAddressField.setPromptText("Sender Address");
    TextField senderMobileField = new TextField();
    senderMobileField.setPromptText("Sender Mobile");

    TextField receiverNameField = new TextField();
    receiverNameField.setPromptText("Receiver Name");
    TextField receiverAddressField = new TextField();
    receiverAddressField.setPromptText("Receiver
Address");
    TextField receiverMobileField = new TextField();
    receiverMobileField.setPromptText("Receiver Mobile");

    // Button to proceed to the payment page
    Button proceedToPaymentButton = new Button("Proceed to
Payment");
    proceedToPaymentButton.setOnAction(e -> {
        // Validate form input
        String senderName = senderNameField.getText();
        String senderAddress =
senderAddressField.getText();
        String senderMobile = senderMobileField.getText();
        String receiverName = receiverNameField.getText();
        String receiverAddress =
receiverAddressField.getText();
        String receiverMobile =
receiverMobileField.getText();
```

```
// Show alert if any field is empty
if (senderName.isEmpty() ||
senderAddress.isEmpty() || senderMobile.isEmpty() ||
    receiverName.isEmpty() ||
receiverAddress.isEmpty() || receiverMobile.isEmpty()) {
    Alert alert = new
Alert(Alert.AlertType.ERROR);
    alert.setTitle("Missing Information");
    alert.setHeaderText(null);
    alert.setContentText("Please fill out all
fields before proceeding to payment.");
    alert.showAndWait();
} else {
    // Create a new order and pass it to the
payment page
    CourierOrder newOrder =
courierService.createOrder(senderName, receiverName,
receiverAddress, senderMobile, receiverMobile);
    showPaymentPage(primaryStage, newOrder); //
Navigate to payment page
}
});

// Add fields to the form and set up layout
form.getChildren().addAll(senderNameField,
senderAddressField, senderMobileField,
    receiverNameField, receiverAddressField,
receiverMobileField, proceedToPaymentButton);

sendOrderPage.setCenter(form);

Button backButton = new Button("Back");
backButton.setOnAction(e -> primaryStage.setScene(new
Scene(createHomePage(primaryStage), 600, 400)));

VBox layout = new VBox(10, form, backButton);
layout.setAlignment(Pos.CENTER);
sendOrderPage.setCenter(layout);

Scene sendOrderScene = new Scene(sendOrderPage, 600,
400);
primaryStage.setScene(sendOrderScene);
```

```
// Page for handling payment details
private void showPaymentPage(Stage primaryStage,
CourierOrder order) {
    BorderPane paymentPage = new BorderPane();
    paymentPage.setTop(createHeader("Payment Details"));

    VBox form = new VBox(10);
    form.setAlignment(Pos.CENTER);

    // Select post type
    Label postTypeLabel = new Label("Choose Post Type:");
    ToggleGroup postTypeGroup = new ToggleGroup();

    RadioButton standardPostButton = new
RadioButton("Standard Post (₹20 per kg, 7 days)");
    RadioButton speedPostButton = new RadioButton("Speed
Post (₹50 per kg, 2-3 days)");
    standardPostButton.setToggleGroup(postTypeGroup);
    speedPostButton.setToggleGroup(postTypeGroup);

    // Field to input weight and calculate price
    TextField weightField = new TextField();
    weightField.setPromptText("Enter Weight (kg)");

    ComboBox<String> itemTypeComboBox = new ComboBox<>();
    itemTypeComboBox.getItems().addAll("Fragile",
"Letters", "Confidential Documents", "Heavy Items");
    itemTypeComboBox.setPromptText("Select Item Type");

    Button calculateButton = new Button("Calculate
Price");
    Label totalPriceLabel = new Label();

    calculateButton.setOnAction(e -> {
        // Calculate total price based on weight and post
type
        try {
            double weight =
Double.parseDouble(weightField.getText());
            double pricePerKg =
standardPostButton.isSelected() ? 20 : 50;
            double totalPrice = weight * pricePerKg;
            totalPriceLabel.setText("Total Price: ₹" +
totalPrice);
        }
    });
}
```



```
        } catch (NumberFormatException ex) {
            totalPriceLabel.setText("Invalid weight
entered. Please enter a valid number.");
        }
    });

    // Payment method selection
    ComboBox<String> paymentMethodComboBox = new
    ComboBox<>();
    paymentMethodComboBox.getItems().addAll("Credit Card",
    "Debit Card", "UPI", "Cash on Delivery");
    paymentMethodComboBox.setPromptText("Select Payment
    Method");

    Button payButton = new Button("Continue");
    payButton.setOnAction(e -> {
        // Navigate to respective payment page based on
    selected method
        String paymentMethod =
    paymentMethodComboBox.getValue();

        if (paymentMethod != null) {
            if (paymentMethod.equals("Credit Card") ||
    paymentMethod.equals("Debit Card")) {
                showCardDetailsPage(primaryStage, order,
    paymentMethod); // Show card details page
            } else if (paymentMethod.equals("UPI")) {
                showUPIDetailsPage(primaryStage, order);
    // Show UPI details page
            } else {
                showThankYouPage(primaryStage, order); //
    Direct to Thank You page for COD
            }
        } else {
            Alert alert = new
    Alert(Alert.AlertType.ERROR);
            alert.setTitle("Payment Method Required");
            alert.setHeaderText(null);
            alert.setContentText("Please select a payment
    method.");
            alert.showAndWait();
        }
    });
```

```
        form.getChildren().addAll(postTypeLabel,
standardPostButton, speedPostButton, weightField,
            itemTypeComboBox, calculateButton,
totalPriceLabel, paymentMethodComboBox, payButton);

        paymentPage.setCenter(form);

        Button backButton = new Button("Back");
        backButton.setOnAction(e -> primaryStage.setScene(new
Scene(createHomePage(primaryStage), 600, 400)));

        VBox layout = new VBox(10, form, backButton);
        layout.setAlignment(Pos.CENTER);
        paymentPage.setCenter(layout);

        Scene paymentScene = new Scene(paymentPage, 600, 400);
        primaryStage.setScene(paymentScene);
    }

    // Show Credit/Debit Card Details Page
    private void showCardDetailsPage(Stage primaryStage,
CourierOrder order, String paymentMethod) {
        BorderPane cardDetailsPage = new BorderPane();
        cardDetailsPage.setTop(createHeader(paymentMethod + "
Details"));

        VBox form = new VBox(10);
        form.setAlignment(Pos.CENTER);

        TextField nameOnCardField = new TextField();
        nameOnCardField.setPromptText("Name on Card");

        TextField cardNumberField = new TextField();
        cardNumberField.setPromptText("Card Number");

        TextField cvvField = new TextField();
        cvvField.setPromptText("CVV");

        TextField expiryDateField = new TextField();
        expiryDateField.setPromptText("Expiry Date (MM/YY)");

        Button submitButton = new Button("Submit");
        submitButton.setOnAction(e -> {
```

```
        // Handle form submission (You can perform
validation here if needed)
        showThankYouPage(primaryStage, order);
        order.setPaymentMethod("Paid"); // Call the
method on the instance
    });

    form.getChildren().addAll(nameOnCardField,
cardNumberField, cvvField, expiryDateField, submitButton);

    cardDetailsPage.setCenter(form);

    Button backButton = new Button("Back");
    backButton.setOnAction(e -> primaryStage.setScene(new
Scene(createHomePage(primaryStage), 600, 400)));

    VBox layout = new VBox(10, form, backButton);
    layout.setAlignment(Pos.CENTER);
    cardDetailsPage.setCenter(layout);

    Scene cardDetailsScene = new Scene(cardDetailsPage,
600, 400);
    primaryStage.setScene(cardDetailsScene);
}

// Show UPI Details Page
private void showUPIDetailsPage(Stage primaryStage,
CourierOrder order) {
    BorderPane upiDetailsPage = new BorderPane();
    upiDetailsPage.setTop(createHeader("UPI Details"));

    VBox form = new VBox(10);
    form.setAlignment(Pos.CENTER);

    TextField upiIDField = new TextField();
    upiIDField.setPromptText("Enter UPI ID");

    Button submitButton = new Button("Submit");
    submitButton.setOnAction(e -> {
        // Handle UPI ID submission (Validation can be
added if needed)
        showThankYouPage(primaryStage, order);
        order.setPaymentMethod("Paid");
    });
}
```

```
        form.getChildren().addAll(upiIDField, submitButton);

        upiDetailsPage.setCenter(form);

        Button backButton = new Button("Back");
        backButton.setOnAction(e -> primaryStage.setScene(new
Scene(createHomePage(primaryStage), 600, 400)));

        VBox layout = new VBox(10, form, backButton);
        layout.setAlignment(Pos.CENTER);
        upiDetailsPage.setCenter(layout);

        Scene upiDetailsScene = new Scene(upiDetailsPage, 600,
400);
        primaryStage.setScene(upiDetailsScene);
    }

    // Show Thank You Page
    private void showThankYouPage(Stage primaryStage,
CourierOrder order) {
        BorderPane thankYouPage = new BorderPane();
        thankYouPage.setTop(createHeader("Thank You"));

        VBox form = new VBox(20);
        form.setAlignment(Pos.CENTER);

        Label thankYouLabel = new Label("Thank you for your
payment!");

        TextField orderStatusField = new TextField();
        orderStatusField.setEditable(false);
        orderStatusField.setStyle("-fx-font-weight: bold;");
        thankYouLabel.setStyle("-fx-font-size: 18px; -fx-font-
weight: bold;");
        orderStatusField.setText("Order Successfully Created!
Tracking ID: " + order.getTrackingNumber());

        Button backButton = new Button("Back to Home");
        backButton.setOnAction(e -> primaryStage.setScene(new
Scene(createHomePage(primaryStage), 600, 400)));

        form.getChildren().addAll(thankYouLabel,
orderStatusField, backButton);
```

```
        thankYouPage.setCenter(form);

        Scene thankYouScene = new Scene(thankYouPage, 600,
400);
        primaryStage.setScene(thankYouScene);
    }

    // Displays the page to track an order by its tracking
number
    private void showTrackOrderPage(Stage primaryStage) {
        // Set up layout and header for the tracking page
        BorderPane trackOrderPage = new BorderPane();
        trackOrderPage.setTop(createHeader("Track Order"));

        VBox form = new VBox(10);
        form.setAlignment(Pos.CENTER);

        // Text field to input tracking number
        TextField trackingNumberField = new TextField();
        trackingNumberField.setPromptText("Enter Tracking
Number");

        // Label to display order details or error message
        Label orderDetailsLabel = new Label();

        // Button to initiate tracking
        Button trackButton = new Button("Track Order");
        trackButton.setOnAction(e -> {
            // Get the tracking number entered by the user
            String trackingNumber =
trackingNumberField.getText();
            // Find the order with the given tracking number
            CourierOrder order =
courierService.trackOrder(trackingNumber);

            // Display order details if found, otherwise show
an error message
            if (order != null) {
                orderDetailsLabel.setText("Order Status: " +
order.getStatus() +
"\nRecipient: " + order.getRecipient()
+
"\nAddress: " + order.getAddress() +
```

```
        "\nPayment Method: " +
order.getPaymentMethod());
        } else {
            orderDetailsLabel.setText("Order not found for
Tracking Number: " + trackingNumber);
        }
    });

    // Add tracking number field, track button, and
details label to form
    form.getChildren().addAll(trackingNumberField,
trackButton, orderDetailsLabel);

    // Back button to return to the home page
    Button backButton = new Button("Back");
    backButton.setOnAction(e -> primaryStage.setScene(new
Scene(createHomePage(primaryStage), 600, 400)));

    // Set up the main layout for the tracking page
    VBox layout = new VBox(10, form, backButton);
    layout.setAlignment(Pos.CENTER);
    trackOrderPage.setCenter(layout);

    // Set and display the scene for tracking orders
    Scene trackOrderScene = new Scene(trackOrderPage, 600,
400);
    primaryStage.setScene(trackOrderScene);
}

// Displays the page to update the status of an order
private void showUpdateOrderPage(Stage primaryStage) {
    // Set up layout and header for the update status page
    BorderPane updateOrderPage = new BorderPane();
    updateOrderPage.setTop(createHeader("Update Order
Status"));

    VBox form = new VBox(10);
    form.setAlignment(Pos.CENTER);

    // Text field to input the tracking number for the
order to update
    TextField trackingNumberField = new TextField();
    trackingNumberField.setPromptText("Enter Tracking
Number");
```

```
// ComboBox to select the new status of the order
ComboBox<String> statusComboBox = new ComboBox<>();
statusComboBox.getItems().addAll("Created", "Shipped",
    "In Transit", "Delivered");
statusComboBox.setPromptText("Select New Status");

// Button to initiate the status update
Button updateButton = new Button("Update Status");
// Label to display success or error messages
Label updateStatusLabel = new Label();

updateButton.setOnAction(e -> {
    // Get the tracking number and new status selected
    by the user
    String trackingNumber =
trackingNumberField.getText();
    String newStatus = statusComboBox.getValue();

    // Update order status if both tracking number and
    status are valid
    if (newStatus != null &&
courierService.updateOrderStatus(trackingNumber, newStatus)) {
        updateStatusLabel.setText("Status updated
successfully to: " + newStatus);
    } else {
        updateStatusLabel.setText("Invalid Tracking
Number or Status. Please try again.");
    }
});

// Add fields, ComboBox, update button, and status
label to the form
form.getChildren().addAll(trackingNumberField,
statusComboBox, updateButton, updateStatusLabel);

// Back button to return to the home page
Button backButton = new Button("Back");
backButton.setOnAction(e -> primaryStage.setScene(new
Scene(createHomePage(primaryStage), 600, 400)));

// Set up the main layout for the update status page
VBox layout = new VBox(10, form, backButton);
layout.setAlignment(Pos.CENTER);
```

```
        updateOrderPage.setCenter(layout);

        // Set and display the scene for updating order status
        Scene updateOrderScene = new Scene(updateOrderPage,
600, 400);
        primaryStage.setScene(updateOrderScene);
    }

    // Main entry point to launch the JavaFX application
    public static void main(String[] args) {
        launch(args);
    }
}
```

## **USAGE GUIDE**

- **Login:** Start the application, enter username and password to log in. Use default users or register a new one.
- **Home Page:** Access options to create, track, update, or view orders.
- **Creating Orders:** Fill in all fields (sender, recipient, address, and phone numbers) to create an order.
- **Tracking Orders:** Enter the tracking number to see order details.
- **Updating Order Status:** Enter the tracking number and select a new status to update.
- **Viewing All Orders:** The admin can view a summary of all orders.



## **CONCLUSION**

The Courier Management System provides a streamlined and efficient solution for managing courier orders within a desktop environment. Developed using JavaFX, this application allows users to easily create, track, and update orders, while also giving administrators the tools to oversee and manage all courier transactions.

This project demonstrates the effectiveness of JavaFX for developing user-friendly graphical interfaces and emphasizes the importance of structured data handling and modular design. Through features like order tracking, user registration, and order status updates, the application models real-world courier operations, offering a practical approach to learning and implementing core programming concepts, user interface design, and business logic.

In its current form, the Courier Management System provides a functional prototype that can be further developed with additional features like database integration for persistent storage, role-based access control, and a more robust user interface. The project serves as a foundation for understanding the complexities of order management in a digital system and can be expanded for more extensive, enterprise-level use.