

LOST AND FOUND MANAGEMENT

A MINI-PROJECT REPORT

Submitted by

SIVA SABARI GANESAN A 230701321

SHANMUGANATHAN S 230701307

In partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2023

BONAFIDE CERTIFICATE

Certified that this project “LOST AND FOUND MANAGEMENT” is the bonafide work of “SIVA SABARI GANESAN A AND SHANMUGANATHAN S” who carried out the project work under my supervision.

SIGNATURE

MRS V JANANEE

ASSISTANT PROFESSOR

Dept. of Computer Science and Engg,
Engg,

Rajalakshmi Engineering College
Chennai

SIGNATURE

MR SARAVANA GOKUL

ASSISTANT PROFESSOR

Dept. of Computer Science and

Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on _____

TABLE OF CONTENTS

S.NO	TITLE	PAGE
	Abstract	
1	Introduction	05
2	Scope of Project	07
3	UI Diagrams	08
4	Code Implementation	16
5	Conclusion	50
6	Reference	50

ABSTRACT

The concept of "Lost and Found" serves as an essential social framework that addresses the emotional and practical challenges individuals face when they misplace personal belongings. Whether it's a cherished item like a family heirloom or a simple everyday object, the loss of personal possessions can evoke feelings of anxiety, frustration, and sadness. A well-organized Lost and Found system provides a structured approach for individuals to report lost items, allowing them to describe their belongings in detail while offering a means to connect with those who may have found them. This process not only aids in the recovery of lost items but also fosters a sense of community, as people come together to help each other in times of need.

Moreover, the Lost and Found mechanism promotes ethical behavior and social responsibility within a community. When individuals find items, they are encouraged to return them to their rightful owners, reinforcing trust and integrity among community members. This practice reduces waste by encouraging the return of lost items rather than their disposal, contributing to a more sustainable environment. Ultimately, the Lost and Found system embodies the values of empathy and cooperation, highlighting the human capacity for kindness and the importance of maintaining connections in a world where loss is an inevitable part of life.

INTRODUCTION

The Lost and Found Management System is a dedicated application designed to address the common issue of misplaced personal belongings, which can lead to feelings of anxiety and frustration for individuals. In an increasingly mobile and dynamic society, the loss of items such as wallets, keys, or personal electronics is a frequent occurrence. This application aims to provide a structured and user-friendly platform where users can easily report lost items and browse through a database of found belongings. By leveraging the capabilities of Java Swing, the system will offer an intuitive graphical user interface that enhances user interaction and facilitates effective communication between individuals who have lost items and those who have found them.

The introduction of this system is not only about creating a functional tool but also about fostering a sense of community and responsibility among users. By encouraging individuals to report lost items and return found belongings, the application promotes ethical behavior and social cooperation. The Lost and Found Management System will streamline the process of item recovery, reducing the emotional burden associated with loss and enhancing the overall user experience. Through its features, the application aims to create a supportive environment where users can feel confident in reclaiming their lost possessions while also contributing to a culture of trust and accountability within their community.

Key Features of a Hotel Management System Using Java Swing

1. User Registration and Authentication:

- A secure user registration process that allows individuals to create accounts and log in to the system. This ensures that only registered users can report lost items or claim found items, enhancing security and accountability.

2. Lost Item Reporting:

- A user-friendly form for users to report lost items, including fields for item description, category, date of loss, location, and the option to upload images. This feature allows users to provide detailed information to aid in the recovery of their belongings.

3. Found Item Database:

- A searchable database that displays all reported found items. Users can browse through the list, filter by categories or keywords, and view detailed information about each found item, facilitating easy identification of potential matches.

4. Communication System:

- An integrated messaging or notification system that alerts users when a matching item is reported or when there is an update regarding their lost item. This feature keeps users informed and engaged throughout the recovery process.

5. Admin Panel:

- A dedicated administrative interface for managing user accounts, monitoring reported items, and overseeing the overall functioning of the system. Admins can verify reports, handle disputes, and ensure the integrity of the database.

6. User Dashboard:

- A personalized dashboard for each user that displays their reported lost items, found items they have claimed, and notifications. This feature helps users keep track of their interactions with the system and enhances user experience.

SCOPE OF THE PROJECT

Scope of the Project: Lost and Found Management System Using Java Swing

The Lost and Found Management System aims to create a comprehensive desktop application that facilitates the reporting, tracking, and recovery of lost items while providing a platform for individuals to return found belongings. Here is a detailed breakdown of the scope of the project:

Scope of the Project: Hotel Management System Using Java Swing

The project involves creating a desktop-based Hotel Management System (HMS) using Java Swing to automate hotel operations like reservations, billing, room management, and guest information. Key features include:

- **User Management::** The system will allow users to register, log in, and manage their accounts securely.
- **Lost Item Reporting:** Structured form for users to report lost items.
- **Found Item Database:** Searchable database of reported found items.
- **Communication and Notification System:** Alerts for users when matching items are reported.
- **Admin Functionality:** Administrative panel for managing user accounts and reported items.
- **Reporting and Analytics:** Generation of reports on user engagement and item recovery rates.

The Lost and Found Management System using Java Swing aims to provide an efficient, user-friendly, and community-oriented solution for managing lost and found items, ultimately fostering a culture of cooperation and responsibility among users.

JAVA UI PICTURES

Frontend:

The image displays two side-by-side screenshots of a Java Swing application titled "Lost and Found - Rajalakshmi".

Left Window (Login):

- Title Bar:** Lost and Found - Rajalakshmi
- Header:** Lost and Found
- Section:** Welcome Back
- Text:** Sign in to continue
- Form Fields:**
 - Email Address
 - Password
- Buttons:** Sign In
- Link:** Create an account

Right Window (Create Account):

- Title Bar:** Lost and Found - Rajalakshmi
- Header:** Lost and Found
- Section:** Create Account
- Text:** Sign up to get started
- Form Fields:**
 - Email Address
 - Password
 - Confirm Password
- Buttons:** Sign Up
- Link:** Already have an account? Log in

Add New Item

Roll No:

123

Name:

Contact:

Location:

Item Name:

Description:

Status:

Lost



Upload Image

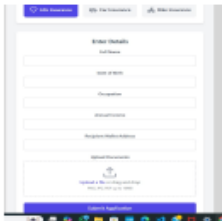
No Image Selected

Post Item

Posts

My Posts

All Posts



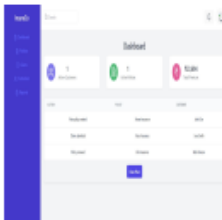
SACKJFound

Posted by: HARI

Location: K

Contact: 9360832153

Description: sajn



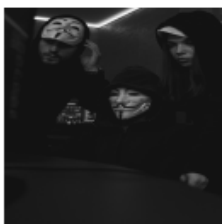
dmdfFound

Posted by: Siva

Location: slkadq,

Contact: 7846

Description: Inldk



Id cardLost

Posted by: Hari haran

Location: J Block

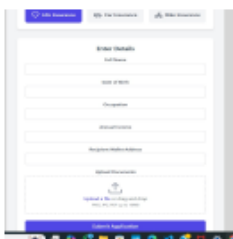
Contact: 936083213

Description: Lost

Posts

My Posts

All Posts



SACKJ Found

Posted by: HARI

Location: K

Contact: 9360832153

Description: sajn



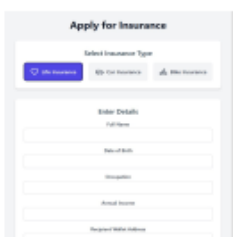
dmdf Found

Posted by: Siva

Location: slkadq,

Contact: 7846

Description: Inldk



cardLost

Posted by: Siva Sabari Ganesa

Location: hut

Contact: 8465158785

Description: lost

Add New Item

Roll No:

123

Name:

Contact:

Location:

Item Name:

Description:

Status:

Lost


Upload Image

No Image Selected

Post Item

Posts

My Posts
All Posts




SACKJFound

Posted by: HARI

Location: K

Contact: 9360832153

Description: sajn



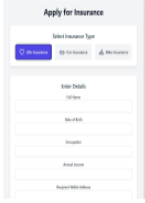
dmdfFound

Posted by: Siva

Location: silkadq,

Contact: 7846

Description: Inldk



cardLost

Posted by: Siva Sabari Ganesa

Location: hut

Contact: 8465158785

Description: lost

Backend:

localhost:27017 > lost_and_found

Open MongoDB shell

Create collection

Refresh

Sort by

Collection Name

View

lost_and_found

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	19	208.00 B	1	36.86 kB

users

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	4	75.00 B	1	36.86 kB

localhost:27017 > lost_and_found > lost_and_found

Open MongoDB shell

Documents 19

Aggregations

Schema

Indexes 1

Validation

Type a query: { field: 'value' } or [Generate query](#)

Explain

Reset

Find

Options

ADD DATA

EXPORT DATA

UPDATE

DELETE

25

1 - 19 of 19

```
_id: ObjectId('6732587c976b57115cb77109')
rollNo: "230701321"
name: "Siva "
contact: "9360832153"
location: "Idea"
itemName: "Phone"
itemDescription: "bbb"
status: "Lost"
photoPath: "C:\Users\Dell\Pictures\meta.PNG"
```

```
_id: ObjectId('67325a5b976b573be41cd098')
rollNo: "230701315"
name: "Sidhrath S"
contact: "9360832153"
location: "Rec cafe"
itemName: "Lunch boz"
itemDescription: "huah"
status: "Found"
photoPath: "C:\Users\Dell\Pictures\meta.PNG"
```

localhost:27017 > lost_and_found > users

[Open MongoDB shell](#)

Documents 4 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

Explain

Reset

Find

Options

ADD DATA

EXPORT DATA

UPDATE

DELETE

25

1 - 4 of 4

<

>

{ }

```
_id: ObjectId('67324fda976b570a9464f20c')
email: "siva@gmail"
password: "123"
```

```
_id: ObjectId('67325185976b57151846ca6e')
email: "230701320@rajalakshmi.edu.in"
password: "123"
```

```
_id: ObjectId('673b89d2976b570c10d0f4d8')
email: "230701321@rajalakshmi.edu.in"
password: "123"
```

```
_id: ObjectId('673d81bb976b5737d4bcfe90')
email: "231401032@rajalakshmi.edu.in"
password: "123"
```

CODE IMPLEMENTATION

Loginpage(frontend):

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import com.mongodb.client.MongoCollection;
import org.bson.Document;
import com.mongodb.client.model.Filters;

public class LoginPage extends JFrame {
    private JTextField emailField;
    private JPasswordField passwordField;
    private JButton loginButton;
    private JButton registerButton;
    private JLabel messageLabel;
    private Color primaryColor = new Color(30, 136, 229); // Brighter blue
    private Color backgroundColor = Color.WHITE;
    private Font mainFont = new Font("Segoe UI", Font.PLAIN, 14);
    private Font titleFont = new Font("Segoe UI", Font.BOLD, 32);
    private Font subtitleFont = new Font("Segoe UI", Font.BOLD, 24);

    public LoginPage() {
        // Basic frame setup
        setTitle("Lost and Found - Rajalakshmi");
        setSize(400, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);
        // Main panel with padding
        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y_AXIS));
        mainPanel.setBackground(backgroundColor);
        mainPanel.setBorder(BorderFactory.createEmptyBorder(40, 40, 40, 40));

        // App name
        JLabel appNameLabel = new JLabel("Lost and Found");
        appNameLabel.setFont(titleFont);
        appNameLabel.setForeground(primaryColor);
        appNameLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
```

```

// Welcome text
JLabel titleLabel = new JLabel("Welcome Back");
titleLabel.setFont(subtitleLabel);
titleLabel.setForeground(new Color(33, 33, 33));
titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

// Subtitle
JLabel subtitleLabel = new JLabel("Sign in to continue");
subtitleLabel.setFont(mainFont);
subtitleLabel.setForeground(new Color(117, 117, 117));
subtitleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

// Form container
JPanel formContainer = new JPanel();
formContainer.setLayout(new BoxLayout(formContainer,
BoxLayout.Y_AXIS));
formContainer.setBackground(backgroundColor);
formContainer.setAlignmentX(Component.CENTER_ALIGNMENT);
formContainer.setBorder(BorderFactory.createEmptyBorder(30, 0, 30, 0));

// Email field
JLabel emailLabel = new JLabel("Email Address");
emailLabel.setFont(mainFont);
emailLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
emailField = new JTextField(20);
styleTextField(emailField);

// Password field
JLabel passwordLabel = new JLabel("Password");
passwordLabel.setFont(mainFont);
passwordLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
passwordField = new JPasswordField(20);
styleTextField(passwordField);

// Login button
loginButton = new JButton("Sign In");
styleButton(loginButton);
loginButton.addActionListener(e -> loginUser());

// Register button
registerButton = new JButton("Create an account");
registerButton.setFont(mainFont);
registerButton.setForeground(primaryColor);
registerButton.setBorderPainted(false);
registerButton.setContentAreaFilled(false);

```



```

registerButton.setFocusPainted(false);
registerButton.setAlignmentX(Component.CENTER_ALIGNMENT);
registerButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
registerButton.addActionListener(e -> {
new RegisterPage();
dispose();
});

// Message label
messageLabel = new JLabel();
messageLabel.setFont(mainFont);
messageLabel.setForeground(Color.RED);
messageLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

// Add components with proper spacing
mainPanel.add(Box.createVerticalGlue());
mainPanel.add(appNameLabel);
mainPanel.add(Box.createVerticalStrut(20));
mainPanel.add(titleLabel);
mainPanel.add(Box.createVerticalStrut(10));
mainPanel.add(subtitleLabel);
mainPanel.add(Box.createVerticalStrut(40));

// Add form elements
formContainer.add(emailLabel);
formContainer.add(Box.createVerticalStrut(10));
formContainer.add(emailField);
formContainer.add(Box.createVerticalStrut(20));
formContainer.add(passwordLabel);
formContainer.add(Box.createVerticalStrut(10));
formContainer.add(passwordField);
formContainer.add(Box.createVerticalStrut(30));
formContainer.add(loginButton);
formContainer.add(Box.createVerticalStrut(15));
formContainer.add(messageLabel);
formContainer.add(Box.createVerticalStrut(15));
formContainer.add(registerButton);

mainPanel.add(formContainer);
mainPanel.add(Box.createVerticalGlue());

// Add main panel to frame
add(mainPanel);
setVisible(true);
}

```

```

private void styleTextField(JTextField field) {
    field.setFont(mainFont);
    field.setMaximumSize(new Dimension(300, 35));
    field.setPreferredSize(new Dimension(300, 35));
    field.setAlignmentX(Component.CENTER_ALIGNMENT);
    field.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(new Color(189, 189, 189)),
        BorderFactory.createEmptyBorder(5, 10, 5, 10)
    ));
}

```

```

private void styleButton(JButton button) {
    button.setFont(new Font("Segoe UI", Font.BOLD, 14));
    button.setForeground(Color.WHITE);
    button.setBackground(primaryColor);
    button.setMaximumSize(new Dimension(300, 40));
    button.setPreferredSize(new Dimension(300, 40));
    button.setAlignmentX(Component.CENTER_ALIGNMENT);
    button.setFocusPainted(false);
    button.setBorderPainted(false);
    button.setOpaque(true);
    button.setCursor(new Cursor(Cursor.HAND_CURSOR));
    button.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseEntered(java.awt.event.MouseEvent evt) {
            button.setBackground(primaryColor.darker());
        }
        public void mouseExited(java.awt.event.MouseEvent evt) {
            button.setBackground(primaryColor);
        }
    });
}

```

```

private void loginUser() {
    String email = emailField.getText();
    String password = new String(passwordField.getPassword());

    // Check if email is valid
    if (!User.isValidDomain(email)) {
        showError("Invalid email domain. Only @rajalaksh.mi.edu.in allowed.");
        return;
    }
}

```

```

MongoCollection<Document> usersCollection =
MongoDBUtil.getDatabase().getCollection("users");

```

```

Document userDoc = usersCollection.find(Filters.eq("email", email)).first();
if (userDoc == null) {
    showError("User not found.");
    return;
}

```

```

String storedPassword = userDoc.getString("password");
if (storedPassword.equals(password)) {
    messageLabel.setForeground(new Color(46, 125, 50));
    messageLabel.setText("Login successful!");
    new DashboardPage(storedPassword);
    dispose();
} else {
    showError("Incorrect password.");
}
}

```

```

private void showError(String message) {
    messageLabel.setForeground(new Color(198, 40, 40));
    messageLabel.setText(message);
}

```

```

public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (Exception e) {
        e.printStackTrace();
    }
    SwingUtilities.invokeLater(() -> new LoginPage());
}
}

```

Register Page(Frontend):

```

import javax.swing.*.*;
import javax.swing.border.*;
import java.awt.*.*;
import com.mongodb.client.MongoCollection;
import org.bson.Document;
import com.mongodb.client.model.Filters;

```

```

public class RegisterPage extends JFrame {
    private JTextField emailField;

```

```

private JPasswordField passwordField, confirmPasswordField;
private JButton registerButton, loginButton;
private JLabel messageLabel;
private Color primaryColor = new Color(30, 136, 229); // Brighter
blue
private Color backgroundColor = Color.WHITE;
private Font mainFont = new Font("Segoe UI", Font.PLAIN, 14);
private Font titleFont = new Font("Segoe UI", Font.BOLD, 32);
private Font subtitleFont = new Font("Segoe UI", Font.BOLD, 24);

public RegisterPage() {
    // Basic frame setup
    setTitle("Lost and Found - Rajalakshmi");
    setSize(400, 700);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setResizable(false);

    // Main panel with padding
    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new BoxLayout(mainPanel,
BoxLayout.Y_AXIS));
    mainPanel.setBackground(backgroundColor);
    mainPanel.setBorder(BorderFactory.createEmptyBorder(40, 40,
40, 40));

    // App name
    JLabel appNameLabel = new JLabel("Lost and Found");
    appNameLabel.setFont(titleFont);
    appNameLabel.setForeground(primaryColor);
    appNameLabel.setAlignmentX(Component.CENTER_ALIGNME
NT);

    // Welcome text
    JLabel titleLabel = new JLabel("Create Account");
    titleLabel.setFont(subtitleFont);
    titleLabel.setForeground(new Color(33, 33, 33));
    titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

```

```

// Subtitle
JLabel subtitleLabel = new JLabel("Sign up to get started");
subtitleLabel.setFont(mainFont);
subtitleLabel.setForeground(new Color(117, 117, 117));
subtitleLabel.setAlignmentX(Component.CENTER_ALIGNMENT)
;

// Form container
JPanel formContainer = new JPanel();
formContainer.setLayout(new BoxLayout(formContainer,
BoxLayout.Y_AXIS));
formContainer.setBackground(backgroundColor);
formContainer.setAlignmentX(Component.CENTER_ALIGNMEN
T);
formContainer.setBorder(BorderFactory.createEmptyBorder(30,
0, 30, 0));

// Email field
JLabel emailLabel = new JLabel("Email Address");
emailLabel.setFont(mainFont);
emailLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
emailField = new JTextField(20);
styleTextField(emailField);

// Password field
JLabel passwordLabel = new JLabel("Password");
passwordLabel.setFont(mainFont);
passwordLabel.setAlignmentX(Component.CENTER_ALIGNMEN
T);
passwordField = new JPasswordField(20);
styleTextField(passwordField);

// Confirm Password field
JLabel confirmPasswordLabel = new JLabel("Confirm
Password");
confirmPasswordLabel.setFont(mainFont);
confirmPasswordLabel.setAlignmentX(Component.CENTER_ALI
GNMENT);
confirmPasswordField = new JPasswordField(20);

```

```

styleTextField(confirmPasswordField);

// Register button
registerButton = new JButton("Sign Up");
styleButton(registerButton);
registerButton.addActionListener(e -> registerUser());

// Login button
loginButton = new JButton("Already have an account? Log in");
loginButton.setFont(mainFont);
loginButton.setForeground(primaryColor);
loginButton.setBorderPainted(false);
loginButton.setContentAreaFilled(false);
loginButton.setFocusPainted(false);
loginButton.setAlignmentX(Component.CENTER_ALIGNMENT);
loginButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
loginButton.addActionListener(e -> {
    new LoginPage();
    dispose();
});

// Message label
messageLabel = new JLabel();
messageLabel.setFont(mainFont);
messageLabel.setForeground(Color.RED);
messageLabel.setAlignmentX(Component.CENTER_ALIGNMEN
T);

// Add components with proper spacing
mainPanel.add(Box.createVerticalGlue());
mainPanel.add(appNameLabel);
mainPanel.add(Box.createVerticalStrut(20));
mainPanel.add(titleLabel);
mainPanel.add(Box.createVerticalStrut(10));
mainPanel.add(subtitleLabel);
mainPanel.add(Box.createVerticalStrut(40));

// Add form elements
formContainer.add(emailLabel);

```

```

formContainer.add(Box.createVerticalStrut(10));
formContainer.add(emailField);
formContainer.add(Box.createVerticalStrut(20));
formContainer.add(passwordLabel);
formContainer.add(Box.createVerticalStrut(10));
formContainer.add(passwordField);
formContainer.add(Box.createVerticalStrut(20));
formContainer.add(confirmPasswordLabel);
formContainer.add(Box.createVerticalStrut(10));
formContainer.add(confirmPasswordField);
formContainer.add(Box.createVerticalStrut(30));
formContainer.add(registerButton);
formContainer.add(Box.createVerticalStrut(15));
formContainer.add(messageLabel);
formContainer.add(Box.createVerticalStrut(20));
formContainer.add(loginButton);

mainPanel.add(formContainer);
mainPanel.add(Box.createVerticalGlue());

// Add main panel to frame
add(mainPanel);
setVisible(true);
}

private void styleTextField(JTextField field) {
    field.setFont(mainFont);
    field.setMaximumSize(new Dimension(300, 35));
    field.setPreferredSize(new Dimension(300, 35));
    field.setAlignmentX(Component.CENTER_ALIGNMENT);
    field.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(new Color(189, 189, 189)),
        BorderFactory.createEmptyBorder(5, 10, 5, 10)
    ));
}

private void styleButton(JButton button) {
    button.setFont(new Font("Segoe UI", Font.BOLD, 14));
    button.setForeground(Color.WHITE);
}

```

```

button.setBackground(primaryColor);
button.setMaximumSize(new Dimension(300, 40));
button.setPreferredSize(new Dimension(300, 40));
button.setAlignmentX(Component.CENTER_ALIGNMENT);
button.setFocusPainted(false);
button.setBorderPainted(false);
button.setOpaque(true);
button.setCursor(new Cursor(Cursor.HAND_CURSOR));
button.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        button.setBackground(primaryColor.darker());
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        button.setBackground(primaryColor);
    }
});
}

private void registerUser() {
    String email = emailField.getText();
    String password = new String(passwordField.getPassword());
    String confirmPassword = new
String(confirmPasswordField.getPassword());

    // Validate email domain
    if (!User.isValidDomain(email)) {
        showError("Invalid email domain. Only
@rajalaksh.mi.edu.in allowed.");
        return;
    }

    // Validate password match
    if (!password.equals(confirmPassword)) {
        showError("Passwords don't match.");
        return;
    }

    // Connect to MongoDB and check if email already exists
    MongoClient

```



```

MongoDBUtil.getDatabase().getCollection("users");

    Document existingUser = usersCollection.find(Filters.eq("email",
email)).first();
    if (existingUser != null) {
        showError("Email already registered.");
        return;
    }

    // Insert the new user
    Document userDoc = new Document("email", email)
        .append("password", password);

    usersCollection.insertOne(userDoc);
    showSuccess("Registration successful.");
}

private void showError(String message) {
    messageLabel.setForeground(new Color(198, 40, 40));
    messageLabel.setText(message);
}

private void showSuccess(String message) {
    messageLabel.setForeground(new Color(46, 125, 50));
    messageLabel.setText(message);
}

public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelFe
elClassName());
    } catch (Exception e) {
        e.printStackTrace();
    }
    SwingUtilities.invokeLater(() -> new RegisterPage());
}
}

```

Dashboard Page(Frontend):

```
import javax.swing.*.*;
```

```

import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.nio.file.*;
import com.mongodb.client.*;
import org.bson.Document;

public class DashboardPage extends JFrame {
    // Constants
    private static final Color PRIMARY_COLOR = new Color(30, 136,
229);
    private static final Color BACKGROUND_COLOR = new Color(245,
245, 250);
    private static final Color CARD_COLOR = Color.WHITE;
    private static final Color TEXT_PRIMARY = new Color(33, 33, 33);
    private static final Color TEXT_SECONDARY = new Color(117, 117,
117);
    private static final Color BORDER_COLOR = new Color(224, 224,
224);
    private static final Font TITLE_FONT = new Font("Segoe UI",
Font.BOLD, 24);
    private static final Font SUBTITLE_FONT = new Font("Segoe UI",
Font.BOLD, 18);
    private static final Font BODY_FONT = new Font("Segoe UI",
Font.PLAIN, 14);
    private static final int BORDER_RADIUS = 10;
    private static final String IMAGE_STORAGE_PATH = "images/";
    private static final Dimension FIELD_SIZE = new Dimension(300,
35);
    private static final Dimension BUTTON_SIZE = new Dimension(150,
40);

    // UI Components
    private JTextField rollNoField, nameField, contactField,
locationField, itemNameField;
    private JTextArea itemDescriptionField;
    private JComboBox<String> statusComboBox;
    private JButton uploadButton, postButton, myPostsButton,

```

```

allPostsButton;
private JLabel photoLabel;
private JPanel postsPanel;
private JScrollPane postsScrollPane;

// State
private File selectedImage;
private String loggedInUserRollNo;
private boolean showingMyPosts = true;

public DashboardPage(String loggedInUserRollNo) {
    this.loggedInUserRollNo = loggedInUserRollNo;
    createImageDirectory();
    setupFrame();
    initializeComponents();
    loadPosts(true);
    setVisible(true);
}

private void createImageDirectory() {
    File directory = new File(IMAGE_STORAGE_PATH);
    if (!directory.exists()) {
        directory.mkdirs();
    }
}

private void setupFrame() {
    setTitle("Lost and Found Dashboard");
    setSize(1200, 800);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setBackground(BACKGROUND_COLOR);

    JPanel mainContainer = new JPanel(new BorderLayout(20, 20));
    mainContainer.setBorder(new EmptyBorder(20, 20, 20, 20));
    mainContainer.setBackground(BACKGROUND_COLOR);
    setContentPane(mainContainer);

    // Add window listener for responsiveness

```

```

addWindowListener(new WindowAdapter() {
    @Override
    public void windowStateChanged(WindowEvent e) {
        updateLayoutForScreenSize();
    }
});
}

private void initializeComponents() {
    JSplitPane splitPane = new
JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
    splitPane.setDividerLocation(400);
    splitPane.setDividerSize(1);
    splitPane.setBorder(null);
    splitPane.setLeftComponent(createFormPanel());
    splitPane.setRightComponent(createPostsPanel());
    add(splitPane, BorderLayout.CENTER);
}

private JPanel createFormPanel() {
    JPanel formPanel = new JPanel();
    formPanel.setLayout(new BoxLayout(formPanel,
BoxLayout.Y_AXIS));
    formPanel.setBackground(CARD_COLOR);
    formPanel.setBorder(BorderFactory.createCompoundBorder(
        new RoundedBorder(BORDER_RADIUS),
        new EmptyBorder(20, 20, 20, 20)
    ));

    // Title
    JLabel titleLabel = new JLabel("Add New Item");
    titleLabel.setFont(TITLE_FONT);
    titleLabel.setForeground(TEXT_PRIMARY);
    titleLabel.setAlignmentX(Component.LEFT_ALIGNMENT);

    // Form fields
    rollNoField = createStyledTextField();
    rollNoField.setText(loggedInUserRollNo);
    rollNoField.setEditable(false);

```

```

nameField = createStyledTextField();
contactField = createStyledTextField();
locationField = createStyledTextField();
itemNameField = createStyledTextField();

// Description
itemDescriptionField = new JTextArea(4, 20);
itemDescriptionField.setFont(BODY_FONT);
itemDescriptionField.setLineWrap(true);
itemDescriptionField.setWrapStyleWord(true);
JScrollPane scrollPane = new JScrollPane(itemDescriptionField);
scrollPane.setBorder(BorderFactory.createLineBorder(BORDER_
COLOR));

// Status
statusComboBox = new JComboBox<>(new String[]{"Lost",
"Found"});
statusComboBox.setFont(BODY_FONT);
statusComboBox.setMaximumSize(FIELD_SIZE);

// Image upload
uploadButton = createStyledButton("Upload Image", false);
uploadButton.addActionListener(e -> uploadImage());
photoLabel = new JLabel("No Image Selected");
photoLabel.setPreferredSize(new Dimension(100, 100));
photoLabel.setBorder(BorderFactory.createLineBorder(BORDER
_COLOR));

// Post button
postButton = createStyledButton("Post Item", true);
postButton.addActionListener(e -> postItem());

// Add components
formPanel.add(titleLabel);
formPanel.add(Box.createVerticalStrut(20));
formPanel.add(createFormField("Roll No:", rollNoField));
formPanel.add(createFormField("Name:", nameField));
formPanel.add(createFormField("Contact:", contactField));
formPanel.add(createFormField("Location:", locationField));

```

```

        formPanel.add(createFormField("Item Name:", itemNameField));
        formPanel.add(createFormField("Description:", scrollPane));
        formPanel.add(createFormField("Status:", statusComboBox));

        JPanel imagePanel = new JPanel(new
FlowLayout(FlowLayout.LEFT));
        imagePanel.setBackground(CARD_COLOR);
        imagePanel.add(uploadButton);
        imagePanel.add(photoLabel);
        imagePanel.setAlignmentX(Component.LEFT_ALIGNMENT);
        formPanel.add(imagePanel);

        formPanel.add(Box.createVerticalStrut(20));
        formPanel.add(postButton);

        return formPanel;
    }

    private JTextField createStyledTextField() {
        JTextField field = new JTextField();
        field.setFont(BODY_FONT);
        field.setMaximumSize(FIELD_SIZE);
        field.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createLineBorder(BORDER_COLOR),
            new EmptyBorder(5, 10, 5, 10)
        ));
        return field;
    }

    private JButton createStyledButton(String text, boolean isPrimary) {
        JButton button = new JButton(text);
        button.setFont(BODY_FONT);
        button.setFocusPainted(false);
        button.setBorderPainted(!isPrimary);
        button.setOpaque(true);
        button.setCursor(new Cursor(Cursor.HAND_CURSOR));
        button.setMaximumSize(BUTTON_SIZE);
        button.setPreferredSize(BUTTON_SIZE);
        button.setAlignmentX(Component.LEFT_ALIGNMENT);
    }

```

```

        if (isPrimary) {
            button.setBackground(PRIMARY_COLOR);
            button.setForeground(Color.WHITE);
        } else {
            button.setBackground(Color.WHITE);
            button.setForeground(TEXT_PRIMARY);
            button.setBorder(BorderFactory.createLineBorder(BORDER_C
OLOR));
        }

        button.addMouseListener(new MouseAdapter() {
            public void mouseEntered(MouseEvent e) {
                button.setBackground(isPrimary ?
PRIMARY_COLOR.darker() : new Color(245, 245, 245));
            }
            public void mouseExited(MouseEvent e) {
                button.setBackground(isPrimary ? PRIMARY_COLOR :
Color.WHITE);
            }
        });

        return button;
    }

    private JPanel createFormField(String labelText, JComponent field)
    {
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.setBackground(CARD_COLOR);
        panel.setAlignmentX(Component.LEFT_ALIGNMENT);
        panel.setMaximumSize(new Dimension(Integer.MAX_VALUE,
70));

        JLabel label = new JLabel(labelText);
        label.setFont(BODY_FONT);
        label.setForeground(TEXT_PRIMARY);

        panel.add(label);
    }

```

```

        panel.add(Box.createVerticalStrut(5));
        panel.add(field);
        panel.add(Box.createVerticalStrut(10));

        return panel;
    }

    private JPanel createPostsPanel() {
        JPanel mainPanel = new JPanel(new BorderLayout(0, 20));
        mainPanel.setBackground(BACKGROUND_COLOR);

        // Header
        JPanel headerPanel = new JPanel(new BorderLayout());
        headerPanel.setBackground(CARD_COLOR);
        headerPanel.setBorder(BorderFactory.createCompoundBorder(
            new RoundedBorder(BORDER_RADIUS),
            new EmptyBorder(15, 20, 15, 20)
        ));

        JLabel titleLabel = new JLabel("Posts");
        titleLabel.setFont(TITLE_FONT);
        titleLabel.setForeground(TEXT_PRIMARY);

        JPanel buttonsPanel = new JPanel(new
FlowLayout(FlowLayout.RIGHT, 10, 0));
        buttonsPanel.setBackground(CARD_COLOR);

        myPostsButton = createStyledButton("My Posts", false);
        allPostsButton = createStyledButton("All Posts", false);

        myPostsButton.addActionListener(e -> {
            showingMyPosts = true;
            loadPosts(true);
        });
        allPostsButton.addActionListener(e -> {
            showingMyPosts = false;
            loadPosts(false);
        });
    }

```



```

        buttonsPanel.add(myPostsButton);
        buttonsPanel.add(allPostsButton);

        headerPanel.add(titleLabel, BorderLayout.WEST);
        headerPanel.add(buttonsPanel, BorderLayout.EAST);

        // Posts container
        postsPanel = new JPanel();
        postsPanel.setLayout(new BoxLayout(postsPanel,
BoxLayout.Y_AXIS));
        postsPanel.setBackground(BACKGROUND_COLOR);

        postsScrollPane = new JScrollPane(postsPanel);
        postsScrollPane.setBorder(null);
        postsScrollPane.getVerticalScrollBar().setUnitIncrement(16);

        mainPanel.add(headerPanel, BorderLayout.NORTH);
        mainPanel.add(postsScrollPane, BorderLayout.CENTER);

        return mainPanel;
    }

    private void uploadImage() {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setDialogTitle("Select an Image");

        fileChooser.setFileFilter(new javax.swing.filechooser.FileFilter() {
            public boolean accept(File f) {
                String name = f.getName().toLowerCase();
                return f.isDirectory() || name.endsWith(".jpg") ||
                    name.endsWith(".jpeg") || name.endsWith(".png") ||
                    name.endsWith(".gif");
            }
            public String getDescription() {
                return "Image Files (*.jpg, *.jpeg, *.png, *.gif)";
            }
        });

        int result = fileChooser.showOpenDialog(this);
    }

```

```

        if (result == JFileChooser.APPROVE_OPTION) {
            selectedImage = fileChooser.getSelectedFile();
            ImageIcon originalIcon = new
ImageIcon(selectedImage.getAbsolutePath());
            Image scaledImage =
originalIcon.getImage().getScaledInstance(100, 100,
Image.SCALE_SMOOTH);
            photoLabel.setIcon(new ImageIcon(scaledImage));
            photoLabel.setText("");
        }
    }

    private void postItem() {
        if (validateForm()) {
            String imageName = saveImage();
            if (savePost(imageName)) {
                clearForm();
                loadPosts(showingMyPosts); // Reload posts after
successful save
                JOptionPane.showMessageDialog(this,
                    "Item posted successfully!",
                    "Success",
                    JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }

    private boolean validateForm() {
        if (nameField.getText().trim().isEmpty() ||
            contactField.getText().trim().isEmpty() ||
            locationField.getText().trim().isEmpty() ||
            itemNameField.getText().trim().isEmpty() ||
            itemDescriptionField.getText().trim().isEmpty()) {

            JOptionPane.showMessageDialog(this,
                "Please fill in all fields",
                "Validation Error",
                JOptionPane.ERROR_MESSAGE);
            return false;
        }
    }

```

```

    }
    return true;
}

private String saveImage() {
    String imageName = "";
    if (selectedImage != null) {
        try {
            String timestamp =
String.valueOf(System.currentTimeMillis());
            String extension = selectedImage.getName().substring(
                selectedImage.getName().lastIndexOf("."));
            imageName = timestamp + extension;

            Path source = selectedImage.toPath();
            Path destination = Paths.get(IMAGE_STORAGE_PATH +
imageName);
            Files.copy(source, destination,
StandardCopyOption.REPLACE_EXISTING);
        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this,
                "Error saving image: " + e.getMessage(),
                "Error",
                JOptionPane.ERROR_MESSAGE);
        }
    }
    return imageName;
}

private boolean savePost(String imageName) {
    Document post = new Document()
        .append("rollNo", rollNoField.getText())
        .append("name", nameField.getText())
        .append("contact", contactField.getText())
        .append("location", locationField.getText())
        .append("itemName", itemNameField.getText())
        .append("itemDescription", itemDescriptionField.getText())
        .append("status", statusComboBox.getSelectedItem())

```

```

        .append("imageName", imageName)
        .append("timestamp", System.currentTimeMillis());

    try {
        MongoClient<Document> collection =
MongoDBUtil.getDatabase()
            .getCollection("lost_and_found");
        collection.insertOne(post);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this,
            "Error posting item: " + e.getMessage(),
            "Error",
            JOptionPane.ERROR_MESSAGE);
        return false;
    }
}

private void loadPosts(boolean filterByUser) {
    postsPanel.removeAll();

    MongoClient<Document> collection =
MongoDBUtil.getDatabase()
        .getCollection("lost_and_found");
    FindIterable<Document> iterable;

    if (filterByUser) {
        iterable = collection.find(new Document("rollNo",
loggedInUserRollNo));
    } else {
        iterable = collection.find();
    }

    iterable.sort(new Document("timestamp", -1));

    try (MongoCursor<Document> cursor = iterable.iterator()) {
        while (cursor.hasNext()) {
            Document post = cursor.next();

```

```

        postsPanel.add(createPostCard(post));
        postsPanel.add(Box.createVerticalStrut(10));
    }
}

postsPanel.revalidate();
postsPanel.repaint();
}

private JPanel createPostCard(Document post) {
    JPanel cardPanel = new JPanel(new BorderLayout(15, 15));
    cardPanel.setBackground(CARD_COLOR);
    cardPanel.setBorder(BorderFactory.createCompoundBorder(
        new RoundedBorder(BORDER_RADIUS),
        new EmptyBorder(15, 15, 15, 15)
    ));
    cardPanel.setMaximumSize(new
Dimension(Integer.MAX_VALUE, 180));

    // Image
    JPanel imagePanel =
createPostImagePanel(post.getString("imageName"));
    cardPanel.add(imagePanel, BorderLayout.WEST);

    // Details
    JPanel detailsPanel = new JPanel(new GridBagLayout());
    detailsPanel.setBackground(CARD_COLOR);
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.anchor = GridBagConstraints.WEST;
    gbc.insets = new Insets(0, 0, 8, 0);

    // Header (Item name and status)
    JPanel headerPanel = new JPanel(new BorderLayout());
    headerPanel.setBackground(CARD_COLOR);

    JLabel itemNameLabel = new
JLabel(post.getString("itemName"));

```

```

        itemNameLabel.setFont(SUBTITLE_FONT);
        headerPanel.add(itemNameLabel, BorderLayout.WEST);

        JLabel statusLabel = new JLabel(post.getString("status"));
        statusLabel.setFont(BODY_FONT);
        statusLabel.setForeground(PRIMARY_COLOR);
        headerPanel.add(statusLabel, BorderLayout.EAST);

        detailsPanel.add(headerPanel, gbc);

        // Details rows
        gbc.gridy++;
        addDetailRow(detailsPanel, "Posted by:", post.getString("name"),
gbc);
        gbc.gridy++;
        addDetailRow(detailsPanel, "Location:", post.getString("location"),
gbc);
        gbc.gridy++;
        addDetailRow(detailsPanel, "Contact:", post.getString("contact"),
gbc);
        gbc.gridy++;
        addDetailRow(detailsPanel, "Description:",
post.getString("itemDescription"), gbc);

        cardPanel.add(detailsPanel, BorderLayout.CENTER);
        return cardPanel;
    }

    private JPanel createPostImagePanel(String imageName) {
        JPanel imagePanel = new JPanel(new BorderLayout());
        imagePanel.setBackground(CARD_COLOR);
        imagePanel.setPreferredSize(new Dimension(150, 150));

        JLabel imageLabel = new JLabel();
        imageLabel.setHorizontalAlignment(JLabel.CENTER);

        if (imageName != null && !imageName.isEmpty()) {
            String imagePath = IMAGE_STORAGE_PATH + imageName;
            ImageIcon originalIcon = new ImageIcon(imagePath);

```

```

        Image scaledImage = originalIcon.getImage()
            .getScaledInstance(130, 130, Image.SCALE_SMOOTH);
        imageLabel.setIcon(new ImageIcon(scaledImage));
    } else {
        imageLabel.setText("No Image");
        imageLabel.setFont(BODY_FONT);
    }

    imagePanel.add(imageLabel, BorderLayout.CENTER);
    return imagePanel;
}

private void addDetailRow(JPanel panel, String label, String value,
GridBagConstraints gbc) {
    JLabel labelComponent = new JLabel(label);
    labelComponent.setFont(BODY_FONT);
    labelComponent.setForeground(TEXT_SECONDARY);

    JLabel valueComponent = new JLabel(value);
    valueComponent.setFont(BODY_FONT);
    valueComponent.setForeground(TEXT_PRIMARY);

    JPanel rowPanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 5, 0));
    rowPanel.setBackground(CARD_COLOR);
    rowPanel.add(labelComponent);
    rowPanel.add(valueComponent);

    panel.add(rowPanel, gbc);
}

private void clearForm() {
    nameField.setText("");
    contactField.setText("");
    locationField.setText("");
    itemNameField.setText("");
    itemDescriptionField.setText("");
    selectedImage = null;
    photoLabel.setIcon(null);
}

```

```

        photoLabel.setText("No Image Selected");
        statusComboBox.setSelectedIndex(0);
    }

    private void updateLayoutForScreenSize() {
        int width = getWidth();
        int height = getHeight();

        if (width < 800 || height < 600) {
            // Switch to a vertical layout for smaller screens
            getContentPane().removeAll();
            setLayout(new BoxLayout(getContentPane(),
BoxLayout.Y_AXIS));
            add(createFormPanel());
            add(Box.createVerticalStrut(20));
            add(createPostsPanel());
        } else {
            // Switch back to the split pane layout for larger screens
            getContentPane().removeAll();
            setLayout(new BorderLayout());
            JSplitPane splitPane = new
JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
            splitPane.setDividerLocation(400);
            splitPane.setDividerSize(1);
            splitPane.setBorder(null);
            splitPane.setLeftComponent(createFormPanel());
            splitPane.setRightComponent(createPostsPanel());
            add(splitPane, BorderLayout.CENTER);
        }

        revalidate();
        repaint();
    }

    private static class RoundedBorder extends AbstractBorder {
        private int radius;

        RoundedBorder(int radius) {
            this.radius = radius;
        }
    }

```



```

    }

    @Override
    public void paintBorder(Component c, Graphics g, int x, int y, int
width, int height) {
        Graphics2D g2 = (Graphics2D) g.create();
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        g2.setColor(BORDER_COLOR);
        g2.drawRoundRect(x, y, width - 1, height - 1, radius, radius);
        g2.dispose();
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAnd
FeelClassName());
        } catch (Exception e) {
            e.printStackTrace();
        }
        new DashboardPage("12345");
    });
}
}

```

Post view page(frontend):

```

import com.mongodb.client.MongoCollection;
import org.bson.Document;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

public class PostViewPage {

```

```
private JPanel postsPanel; // This should be the main container for displaying posts
```

```
// Constructor to initialize the posts panel
public PostViewPage() {
    postsPanel = new JPanel();
    postsPanel.setLayout(new BoxLayout(postsPanel,
BoxLayout.Y_AXIS)); // Stack the posts vertically
    JScrollPane scrollPane = new JScrollPane(postsPanel); // Add scrolling to the posts panel
    JFrame frame = new JFrame("Lost and Found Posts");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(600, 400); // Adjust the size as needed
    frame.add(scrollPane);
    frame.setVisible(true);
}
```

```
// Method to load posts from the MongoDB collection
public void loadPosts() {
    MongoCollection<Document> collection =
MongoDBUtil.getDatabase().getCollection("lost_and_found");

    for (Document post : collection.find()) {
        String rollNo = post.getString("rollNo");
        String name = post.getString("name");
        String contact = post.getString("contact");
        String location = post.getString("location");
        String itemName = post.getString("itemName");
        String itemDescription = post.getString("itemDescription");
        String status = post.getString("status");
        String photoPath = post.getString("photoPath");

        // Create a panel for each post
        JPanel postPanel = new JPanel();
        postPanel.setLayout(new BorderLayout());
        postPanel.setBorder(BorderFactory.createLineBorder(Color.BL
ACK));
    }
}
```

```

// Add the image to the top of the post panel
if (photoPath != null && !photoPath.isEmpty()) {
    File imageFile = new File(photoPath);
    if (imageFile.exists()) {
        ImageIcon imageIcon = new ImageIcon(photoPath);
        Image img =
imageIcon.getImage().getScaledInstance(200, 150,
Image.SCALE_SMOOTH);
        imageIcon = new ImageIcon(img);
        JLabel imageLabel = new JLabel(imageIcon);
        postPanel.add(imageLabel, BorderLayout.NORTH);
    }
}

// Add text details to the panel
JPanel textPanel = new JPanel();
textPanel.setLayout(new BoxLayout(textPanel,
BoxLayout.Y_AXIS));
textPanel.setBorder(BorderFactory.createEmptyBorder(10, 10,
10, 10));

StringBuilder postDetails = new StringBuilder();
postDetails.append("Item Name:
").append(itemName).append("\n")
    .append("Item Description:
").append(itemDescription).append("\n")
    .append("Roll No: ").append(rollNo).append("\n")
    .append("Name: ").append(name).append("\n")
    .append("Contact: ").append(contact).append("\n")
    .append("Location: ").append(location).append("\n")
    .append("Status: ").append(status).append("\n");

JTextArea postTextArea = new
JTextArea(postDetails.toString());
postTextArea.setEditable(false);
postTextArea.setFont(new Font("Arial", Font.PLAIN, 12));
postTextArea.setBackground(Color.WHITE);
postTextArea.setWrapStyleWord(true);
postTextArea.setLineWrap(true);

```

```

textPanel.add(postTextArea);
postPanel.add(textPanel, BorderLayout.CENTER);

// Create a "View" button for each post
JButton viewButton = new JButton("View");
viewButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Display the post details in a new window or dialog when
the "View" button is clicked
        viewPostDetails(rollNo, name, contact, location,
itemName, itemDescription, status, photoPath);
    }
});

// Add the "View" button to the post panel
postPanel.add(viewButton, BorderLayout.SOUTH);

postsPanel.add(postPanel); // Add the post panel to the main
postsPanel
}
}

// Method to display the full post details when the "View" button is
clicked
private void viewPostDetails(String rollNo, String name, String
contact, String location, String itemName,
String itemDescription, String status, String
photoPath) {
    // Create a new JFrame to display the full post details
    JFrame postDetailFrame = new JFrame("Post Details");
    postDetailFrame.setDefaultCloseOperation(JFrame.DISPOSE_O
N_CLOSE);
    postDetailFrame.setSize(400, 500); // Adjust size as needed

    // Create a JPanel for the details
    JPanel postDetailPanel = new JPanel();
    postDetailPanel.setLayout(new BoxLayout(postDetailPanel,

```

```
BoxLayout.Y_AXIS));
```

```
    // Add the image if available
    if (photoPath != null && !photoPath.isEmpty()) {
        File imageFile = new File(photoPath);
        if (imageFile.exists()) {
            ImageIcon imageIcon = new ImageIcon(photoPath);
            Image img = imageIcon.getImage().getScaledInstance(200,
150, Image.SCALE_SMOOTH);
            imageIcon = new ImageIcon(img);
            JLabel imageLabel = new JLabel(imageIcon);
            postDetailPanel.add(imageLabel); // Add image to the detail
panel
        }
    }
```

```
    // Add the text details
    JTextArea postDetailsArea = new JTextArea();
    postDetailsArea.setEditable(false);
    postDetailsArea.setFont(new Font("Arial", Font.PLAIN, 14));
    postDetailsArea.setText("Item Name: " + itemName + "\n" +
        "Item Description: " + itemDescription + "\n" +
        "Roll No: " + rollNo + "\n" +
        "Name: " + name + "\n" +
        "Contact: " + contact + "\n" +
        "Location: " + location + "\n" +
        "Status: " + status);

    postDetailPanel.add(postDetailsArea);
    postDetailFrame.add(postDetailPanel);
    postDetailFrame.setVisible(true);
}
```

```
// Main method to run the Swing application
public static void main(String[] args) {
    // Swing UI should be run on the Event Dispatch Thread
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
```

```

        PostViewPage postViewPage = new PostViewPage();
        postViewPage.loadPosts(); // Load the posts after the UI is
set up
    }
    });
}
}

```

Lostandfounditem(Backend):

```

public class LostAndFoundItem {
    private String rollNo;
    private String name;
    private String contact;
    private String location;
    private String photoPath;

    public LostAndFoundItem(String rollNo, String name, String contact,
String location, String photoPath) {
        this.rollNo = rollNo;
        this.name = name;
        this.contact = contact;
        this.location = location;
        this.photoPath = photoPath;
    }

    // Getters and Setters
    public String getRollNo() {
        return rollNo;
    }

    public void setRollNo(String rollNo) {
        this.rollNo = rollNo;
    }

    public String getName() {
        return name;
    }
}

```

```
public void setName(String name) {  
    this.name = name;  
}
```

```
public String getContact() {  
    return contact;  
}
```

```
public void setContact(String contact) {  
    this.contact = contact;  
}
```

```
public String getLocation() {  
    return location;  
}
```

```
public void setLocation(String location) {  
    this.location = location;  
}
```

```
public String getPhotoPath() {  
    return photoPath;  
}
```

```
public void setPhotoPath(String photoPath) {  
    this.photoPath = photoPath;  
}  
}
```

Users (Backend):

```
public class User {  
    private String email;  
    private String password;
```

```
public User(String email, String password) {  
    this.email = email;  
    this.password = password;
```

```
}
```

```
public String getEmail() {  
    return email;  
}
```

```
public String getPassword() {  
    return password;  
}
```

```
// Add a method to validate email domain  
public static boolean isValidDomain(String email) {  
    return email.endsWith("@rajalakshmi.edu.in");  
}  
}
```

Mongo DB Connection(backend):

```
import com.mongodb.MongoClient;  
import com.mongodb.MongoClientURI;  
import com.mongodb.client.MongoDatabase;
```

```
public class MongoDBUtil {  
    private static final String DATABASE_URL =  
        "mongodb://localhost:27017"; // Change to your MongoDB URL  
    private static final String DATABASE_NAME = "lost_and_found"; //  
    Your database name  
    private static MongoClient client;  
  
    static {  
        MongoClientURI uri = new MongoClientURI(DATABASE_URL);  
        client = new MongoClient(uri);  
    }  
  
    public static MongoDatabase getDatabase() {  
        return client.getDatabase(DATABASE_NAME);  
    }  
}
```


CONCLUSION:

The Lost and Found Management System will serve as a vital tool for individuals to recover lost items and facilitate the return of found belongings. By creating a centralized platform for reporting and tracking lost items, the application promotes community cooperation and ethical behavior. With a focus on user experience and functionality, this project exemplifies the application of object-oriented principles and event-driven programming in Java.

REFERENCE:

- <https://docs.oracle.com/javase/8/docs/api/>
- <https://docs.oracle.com/javase/tutorial/jdbc/index.html>
- <https://docs.oracle.com/javase/tutorial/uiswing/>