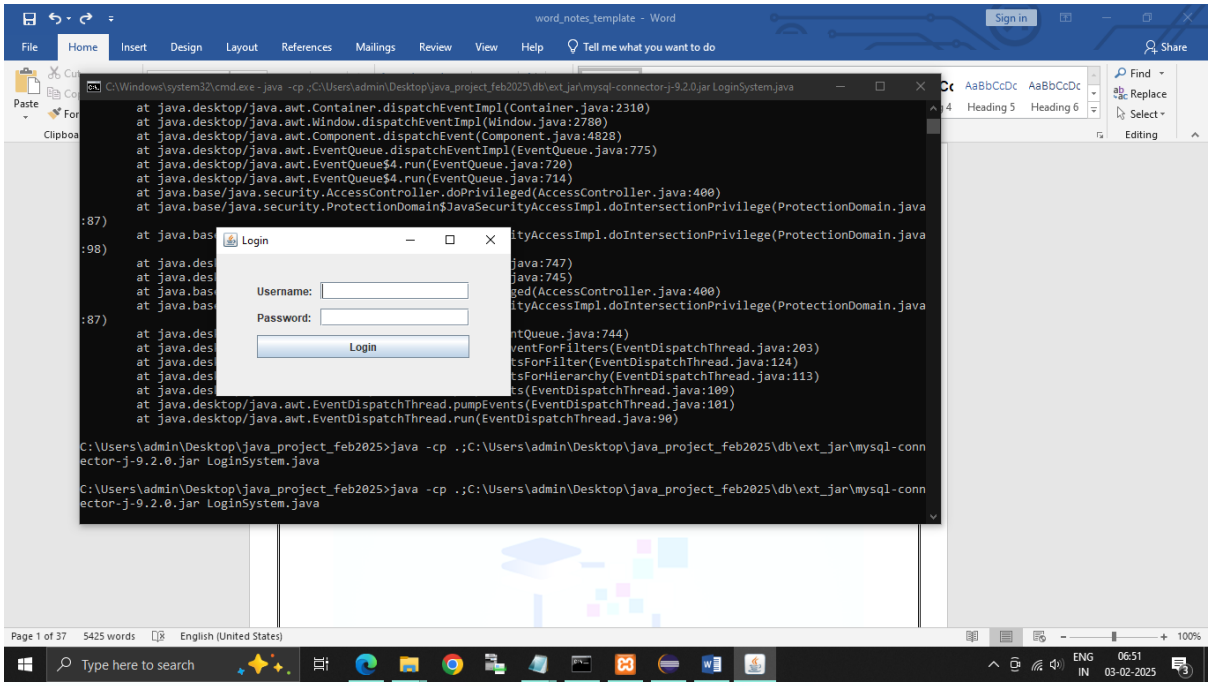
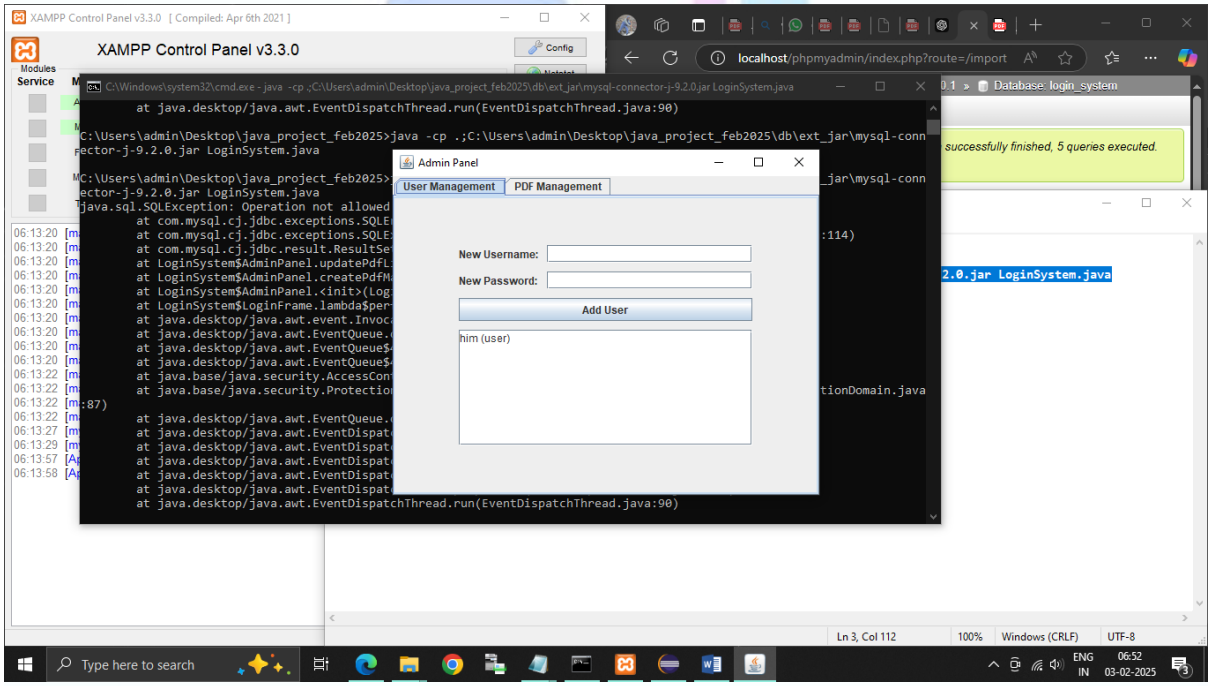
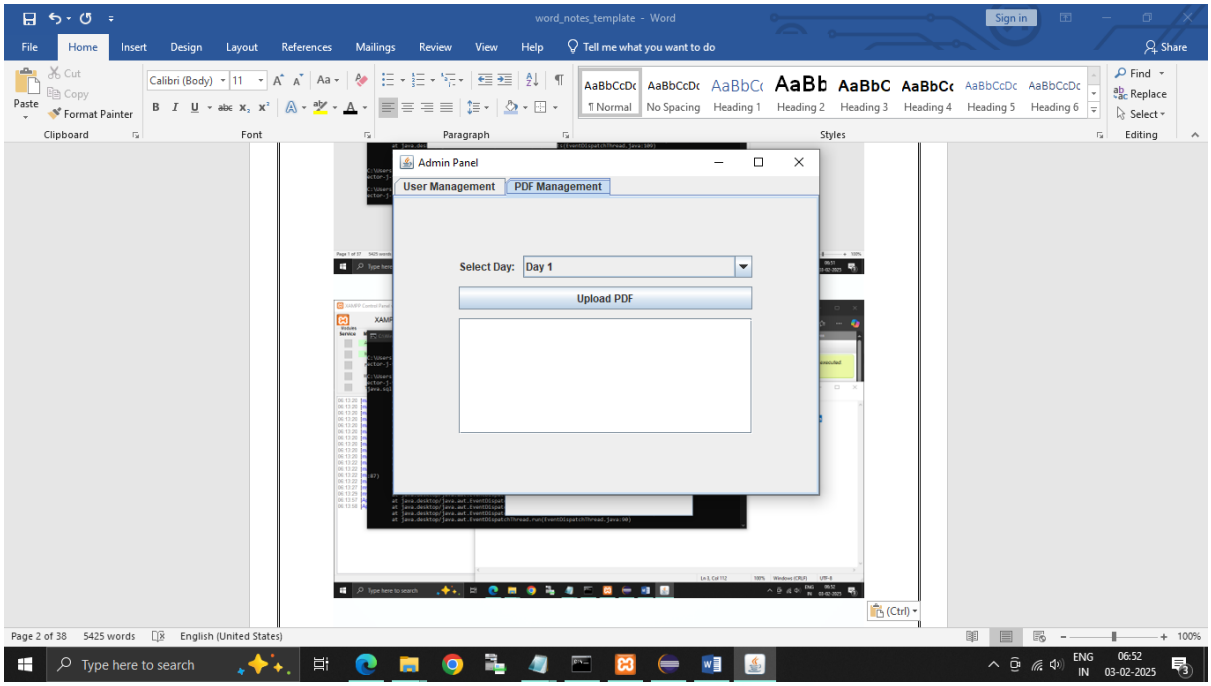
JAVA SWING DESKTOP APPLICATION RESOURCE MANAGEMENT SYSTEM

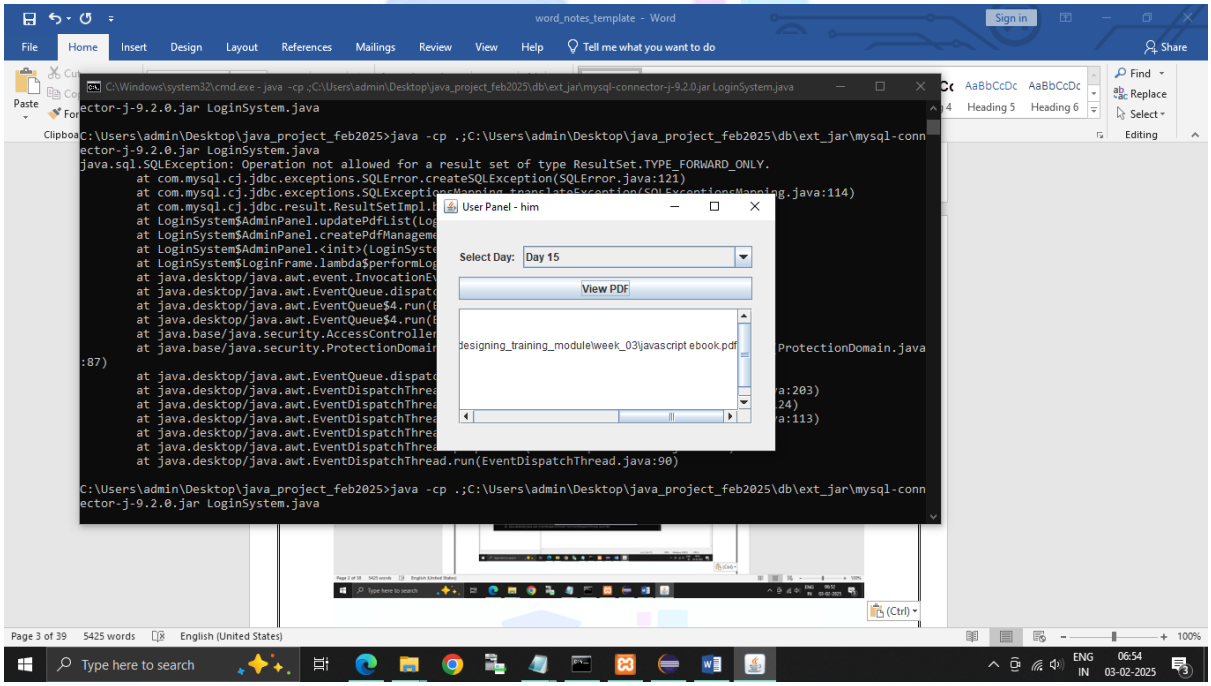
By Himanshu Singh

Himanshu Singh
himanshusingh1814@gmail.com


word_notes_template - Word
Login
Username:
Password:
Login
Type here to search


XAMPP Control Panel v3.3.0
Admin Panel
User Management
PDF Management
New Username:
New Password:
Add User
him (user)
localhost/phpmyadmin/index.php?route=/import
Database: login_system
successfully finished, 5 queries executed.
Type here to search


word_notes_template - Word
Admin Panel
User Management
PDF Management
Select Day:
Day 1
Upload PDF
Page 2 of 38
5425 words
English (United States)
Type here to search


word_notes_template - Word
C:\Windows\system32\cmd.exe - java -cp .;C:\Users\admin\Desktop\java_project_feb2025\db\ext_jar\mysql-connector-j-9.2.0.jar LoginSystem.java
C:\Users\admin\Desktop\java_project_feb2025>java -cp .;C:\Users\admin\Desktop\java_project_feb2025\db\ext_jar\mysql-connector-j-9.2.0.jar LoginSystem.java
java.sql.SQLException: Operation not allowed for a result set of type ResultSet.TYPE_FORWARD_ONLY.
User Panel - him
Select Day:
Day 15
View PDF
designing_training_module\week_03\javascript ebook.pdf
Page 3 of 39
5425 words
English (United States)
Type here to search

make a login page java swing

Below is a simple example of a Java Swing login page. This example creates a window with fields for a username and password and a "Login" button. When the button is pressed, it checks if the username and password match a hardcoded pair (in this example, "admin" for the username and "password" for the password). You can customize the logic as needed.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class LoginPage extends JFrame {

    // Components of the login form
    private JTextField userTextField;
    private JPasswordField passwordField;
    private JButton loginButton;
    private JLabel statusLabel;

    public LoginPage() {
        // Set up the frame
        setTitle("Login Page");
        setSize(350, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);  // Center the window

        // Create panel for the form
        JPanel panel = new JPanel();
        panel.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(5, 5, 5, 5);

        // Username label and text field
        gbc.gridx = 0;
        gbc.gridy = 0;
        panel.add(new JLabel("Username:"), gbc);

        gbc.gridx = 1;
```

```
        userTextField = new JTextField(15);
        panel.add(userTextField, gbc);

        // Password label and password field
        gbc.gridx = 0;
        gbc.gridy = 1;
        panel.add(new JLabel("Password:"), gbc);

        gbc.gridx = 1;
        passwordField = new JPasswordField(15);
        panel.add(passwordField, gbc);

        // Login button
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 2;
        gbc.anchor = GridBagConstraints.CENTER;
        loginButton = new JButton("Login");
        panel.add(loginButton, gbc);

        // Status label for messages
        gbc.gridy = 3;
        statusLabel = new JLabel("");
        panel.add(statusLabel, gbc);

        // Add panel to frame
        add(panel);

        // Add action listener for the login button
        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                performLogin();
            }
        });
    }

    private void performLogin() {
        String username = userTextField.getText();
        char[] passwordChars = passwordField.getPassword();
        String password = new String(passwordChars);
```

```
        // Simple hardcoded check; in a real application, you would query a database or
another secure source
        if ("admin".equals(username) && "password".equals(password)) {
            statusLabel.setForeground(Color.GREEN);
            statusLabel.setText("Login successful!");
            // Optionally, open a new window or perform other actions
        } else {
            statusLabel.setForeground(Color.RED);
            statusLabel.setText("Invalid username or password.");
        }

        // Clear the password for security
        passwordField.setText("");
    }

    public static void main(String[] args) {
        // Ensure the UI is created on the Event Dispatch Thread
        SwingUtilities.invokeLater(() -> {
            LoginPage loginPage = new LoginPage();
            loginPage.setVisible(true);
        });
    }
}
```

How to Run

1. **Save the Code**: Save the code in a file named LoginPage.java.
2. **Compile**: Open a terminal or command prompt, navigate to the directory where you saved the file, and run:

```
javac LoginPage.java
```

3. **Run**: After successful compilation, run the application with:

```
java LoginPage
```

This will open a window with a login form. You can try entering "admin" as the username and "password" as the password to see a successful login message. Otherwise, an error message will be shown. Feel free to modify the design and logic to fit your application needs.

Setting up panels

OK, let me see. I'm thinking through the setup of distinct admin and user panels, each equipped with specific functionalities like adding credentials and organizing PDFs by days from 1 to 30 for user access.

Mapping the interface

I'm piecing together a Java Swing application with login functionality. Admins can upload PDFs and add user credentials, while users can view PDFs sorted by days 1 to 30.

Mapping storage paths

I'm thinking through where to store PDF file paths. Options include in-memory storage or a more structured approach like a database, to support both admin and user panel functionality.

Mapping data

Thinking through using an in-memory map to link days with PDF paths, and a HashMap for storing admin credentials for efficient user login and management.

Setting up login logic

I'm thinking about using static maps for user credentials and PDF mappings by day. The login page will check username and password, guiding users to their respective panels.

Organizing the admin panel

Using JTabbedPane, the admin panel divides into "User Management" and "PDF Management" sections. New users can be added and PDFs for each day can be uploaded through a JFileChooser.

Crafting the user interface

I'm developing a list or combo box for users to view PDFs by day. Selecting a day triggers a "View PDF" button, opening the PDF with Desktop.open() if available.

Constructing the framework

I'm mapping out a Java Swing-based login system, with nested classes for the login page, admin panel, and user panel, all housed within a single file.

Mapping out the system

I'm working on a Java Swing-based login system, integrating nested classes for login, admin, and user panels, all unified in a single file with static maps for a comprehensive structure.

Integrating functionality

I'm including user and password verification, managing a user list, and enabling PDF uploads with progress bars. The admin panel is meticulously designed, showcasing a comprehensive interface for data handling.

Below is an example of a simple Java Swing application that demonstrates a basic multi-user login system with one admin and multiple users. In this example, the admin can:

- **Add new user credentials** (which are stored in memory), and
- **Upload PDF files** for "Day 1" to "Day 30" (by selecting a file using a file chooser).

Once logged in, a user sees a panel where they can select a day and view the corresponding PDF (if one has been uploaded). For simplicity, this example stores user credentials and PDF file references in static maps in memory. In a production application, you would likely persist such data in a database or file system.

Note: This is a simple example. You may wish to enhance error checking, security (e.g. secure password storage), and file handling (e.g. verifying file types) for a real-world application.

Below is the complete code. Save it as **LoginSystem.java.**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class LoginSystem {
    // In-memory storage for user credentials (username -> password)
    public static Map<String, String> userCredentials = new HashMap<>();
    // In-memory storage for PDF files (day label -> File)
    public static Map<String, File> pdfFiles = new HashMap<>();

    // Hardcoded admin credentials
```

```
    public static final String ADMIN_USERNAME = "admin";
    public static final String ADMIN_PASSWORD = "admin123";

    public static void main(String[] args) {
        // Ensure the UI is created on the Event Dispatch Thread
        SwingUtilities.invokeLater(() -> {
            new LoginFrame();
        });
    }

    // ================== Login Frame ==================
    static class LoginFrame extends JFrame {
        private JTextField usernameField;
        private JPasswordField passwordField;
        private JButton loginButton;
        private JLabel statusLabel;

        public LoginFrame() {
            setTitle("Login");
            setSize(350, 200);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setLocationRelativeTo(null);  // Center the window

            JPanel panel = new JPanel(new GridBagLayout());
            GridBagConstraints gbc = new GridBagConstraints();
            gbc.insets = new Insets(5, 5, 5, 5);
            gbc.fill = GridBagConstraints.HORIZONTAL;

            // Username label and field
            gbc.gridx = 0;
            gbc.gridy = 0;
            panel.add(new JLabel("Username:"), gbc);

            gbc.gridx = 1;
            usernameField = new JTextField(15);
            panel.add(usernameField, gbc);

            // Password label and field
            gbc.gridx = 0;
            gbc.gridy = 1;
            panel.add(new JLabel("Password:"), gbc);
```

```
            gbc.gridx = 1;
            passwordField = new JPasswordField(15);
            panel.add(passwordField, gbc);

            // Login button
            gbc.gridx = 0;
            gbc.gridy = 2;
            gbc.gridwidth = 2;
            gbc.anchor = GridBagConstraints.CENTER;
            loginButton = new JButton("Login");
            panel.add(loginButton, gbc);

            // Status label for messages
            gbc.gridy = 3;
            statusLabel = new JLabel("");
            panel.add(statusLabel, gbc);

            add(panel);

            // Handle login button click
            loginButton.addActionListener(e -> {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());

                if (username.equals(ADMIN_USERNAME) &&
password.equals(ADMIN_PASSWORD)) {
                    statusLabel.setForeground(Color.GREEN);
                    statusLabel.setText("Admin login successful.");
                    SwingUtilities.invokeLater(() -> new AdminPanel());
                    dispose();  // Close the login window
                } else if (userCredentials.containsKey(username) &&
userCredentials.get(username).equals(password)) {
                    statusLabel.setForeground(Color.GREEN);
                    statusLabel.setText("User login successful.");
                    SwingUtilities.invokeLater(() -> new UserPanel(username));
                    dispose();
                } else {
                    statusLabel.setForeground(Color.RED);
                    statusLabel.setText("Invalid credentials.");
                }
```

```
            });

            setVisible(true);
        }
    }

    // ================== Admin Panel ==================
    static class AdminPanel extends JFrame {
        private JTabbedPane tabbedPane;
        private JPanel userManagementPanel;
        private JPanel pdfManagementPanel;

        // Components for user management
        private JTextField newUsernameField;
        private JPasswordField newPasswordField;
        private JButton addUserButton;
        private JTextArea userListArea;

        // Components for PDF management
        private JComboBox<String> dayComboBox;
        private JButton uploadPdfButton;
        private JTextArea pdfListArea;

        public AdminPanel() {
            setTitle("Admin Panel");
            setSize(500, 400);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setLocationRelativeTo(null);

            tabbedPane = new JTabbedPane();
            userManagementPanel = new JPanel(new GridBagLayout());
            pdfManagementPanel = new JPanel(new GridBagLayout());

            createUserManagementPanel();
            createPdfManagementPanel();

            tabbedPane.addTab("User Management", userManagementPanel);
            tabbedPane.addTab("PDF Management", pdfManagementPanel);

            add(tabbedPane);
            setVisible(true);
```

```
        }

        // Panel for adding new user credentials
        private void createUserManagementPanel() {
            GridBagConstraints gbc = new GridBagConstraints();
            gbc.insets = new Insets(5,5,5,5);
            gbc.fill = GridBagConstraints.HORIZONTAL;

            // New Username
            gbc.gridx = 0;
            gbc.gridy = 0;
            userManagementPanel.add(new JLabel("New Username:"), gbc);

            gbc.gridx = 1;
            newUsernameField = new JTextField(15);
            userManagementPanel.add(newUsernameField, gbc);

            // New Password
            gbc.gridx = 0;
            gbc.gridy = 1;
            userManagementPanel.add(new JLabel("New Password:"), gbc);

            gbc.gridx = 1;
            newPasswordField = new JPasswordField(15);
            userManagementPanel.add(newPasswordField, gbc);

            // Add User Button
            gbc.gridx = 0;
            gbc.gridy = 2;
            gbc.gridwidth = 2;
            gbc.anchor = GridBagConstraints.CENTER;
            addUserButton = new JButton("Add User");
            userManagementPanel.add(addUserButton, gbc);

            // Area to list existing users
            gbc.gridy = 3;
            userListArea = new JTextArea(8, 30);
            userListArea.setEditable(false);
            JScrollPane scrollPane = new JScrollPane(userListArea);
            userManagementPanel.add(scrollPane, gbc);
```

```
            addUserButton.addActionListener(e -> {
                String newUser = newUsernameField.getText().trim();
                String newPass = new String(newPasswordField.getPassword()).trim();
                if (newUser.isEmpty() || newPass.isEmpty()) {
                    JOptionPane.showMessageDialog(this, "Username and password
cannot be empty.");
                    return;
                }
                if (userCredentials.containsKey(newUser) ||
newUser.equals(ADMIN_USERNAME)) {
                    JOptionPane.showMessageDialog(this, "User already exists or invalid
username.");
                    return;
                }
                userCredentials.put(newUser, newPass);
                JOptionPane.showMessageDialog(this, "User added successfully.");
                newUsernameField.setText("");
                newPasswordField.setText("");
                updateUserList();
            });

            updateUserList();
        }

        // Update the list of users displayed in the admin panel
        private void updateUserList() {
            StringBuilder sb = new StringBuilder();
            for (String user : userCredentials.keySet()) {
                sb.append(user).append("\n");
            }
            userListArea.setText(sb.toString());
        }

        // Panel for uploading PDFs by day
        private void createPdfManagementPanel() {
            GridBagConstraints gbc = new GridBagConstraints();
            gbc.insets = new Insets(5,5,5,5);
            gbc.fill = GridBagConstraints.HORIZONTAL;

            // Day selection combo box
            gbc.gridx = 0;
```

```
            gbc.gridy = 0;
            pdfManagementPanel.add(new JLabel("Select Day:"), gbc);

            gbc.gridx = 1;
            String[] days = new String[30];
            for (int i = 0; i < 30; i++) {
                days[i] = "Day " + (i + 1);
            }
            dayComboBox = new JComboBox<>(days);
            pdfManagementPanel.add(dayComboBox, gbc);

            // Upload PDF Button
            gbc.gridx = 0;
            gbc.gridy = 1;
            gbc.gridwidth = 2;
            gbc.anchor = GridBagConstraints.CENTER;
            uploadPdfButton = new JButton("Upload PDF");
            pdfManagementPanel.add(uploadPdfButton, gbc);

            // Area to list uploaded PDFs
            gbc.gridy = 2;
            pdfListArea = new JTextArea(8, 30);
            pdfListArea.setEditable(false);
            JScrollPane scrollPane = new JScrollPane(pdfListArea);
            pdfManagementPanel.add(scrollPane, gbc);

            uploadPdfButton.addActionListener(e -> {
                JFileChooser fileChooser = new JFileChooser();
                int result = fileChooser.showOpenDialog(this);
                if (result == JFileChooser.APPROVE_OPTION) {
                    File selectedFile = fileChooser.getSelectedFile();
                    // Optionally, check that the file has a ".pdf" extension
                    String day = (String) dayComboBox.getSelectedItem();
                    pdfFiles.put(day, selectedFile);
                    JOptionPane.showMessageDialog(this, "PDF uploaded for " + day);
                    updatePdfList();
                }
            });

            updatePdfList();
        }
```

```
        // Update the list of PDF uploads shown in the admin panel
        private void updatePdfList() {
            StringBuilder sb = new StringBuilder();
            for (int i = 1; i <= 30; i++) {
                String day = "Day " + i;
                sb.append(day).append(": ");
                if (pdfFiles.containsKey(day)) {
                    sb.append(pdfFiles.get(day).getName());
                } else {
                    sb.append("No PDF uploaded");
                }
                sb.append("\n");
            }
            pdfListArea.setText(sb.toString());
        }
    }

    // ================== User Panel ==================
    static class UserPanel extends JFrame {
        private JComboBox<String> dayComboBox;
        private JButton viewPdfButton;
        private JTextArea pdfInfoArea;
        private String username;

        public UserPanel(String username) {
            this.username = username;
            setTitle("User Panel - " + username);
            setSize(400, 300);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setLocationRelativeTo(null);

            JPanel panel = new JPanel(new GridBagLayout());
            GridBagConstraints gbc = new GridBagConstraints();
            gbc.insets = new Insets(5,5,5,5);
            gbc.fill = GridBagConstraints.HORIZONTAL;

            // Day selection combo box
            gbc.gridx = 0;
            gbc.gridy = 0;
            panel.add(new JLabel("Select Day:"), gbc);
```

```
            gbc.gridx = 1;
            String[] days = new String[30];
            for (int i = 0; i < 30; i++) {
                days[i] = "Day " + (i + 1);
            }
            dayComboBox = new JComboBox<>(days);
            panel.add(dayComboBox, gbc);

            // Button to view PDF
            gbc.gridx = 0;
            gbc.gridy = 1;
            gbc.gridwidth = 2;
            viewPdfButton = new JButton("View PDF");
            panel.add(viewPdfButton, gbc);

            // Text area to display PDF info
            gbc.gridy = 2;
            pdfInfoArea = new JTextArea(8, 30);
            pdfInfoArea.setEditable(false);
            JScrollPane scrollPane = new JScrollPane(pdfInfoArea);
            panel.add(scrollPane, gbc);

            viewPdfButton.addActionListener(e -> {
                String day = (String) dayComboBox.getSelectedItem();
                if (pdfFiles.containsKey(day)) {
                    File pdfFile = pdfFiles.get(day);
                    pdfInfoArea.setText("PDF for " + day + ":\n" + pdfFile.getName() +
"\nPath: " + pdfFile.getAbsolutePath());

                    // Attempt to open the PDF with the default desktop application
                    if (Desktop.isDesktopSupported()) {
                        try {
                            Desktop.getDesktop().open(pdfFile);
                        } catch (IOException ex) {
                            JOptionPane.showMessageDialog(this, "Error opening PDF
file.");
                        }
                    }
                } else {
                    pdfInfoArea.setText("No PDF uploaded for " + day);
```

```
                }
            });

            add(panel);
            setVisible(true);
        }
    }
}
```

How the Code Works

1. **Login Frame:**
   When the program starts, a login window appears.
   - If you enter the admin credentials (admin / admin123), the Admin Panel is displayed.
   - If you enter credentials that match a user stored in the userCredentials map, the User Panel opens.
   - Otherwise, an error message is displayed.
2. **Admin Panel:**
   The admin panel uses a **JTabbedPane** with two tabs:
   - **User Management:**
     Allows the admin to add new users by entering a username and password. The current user list is displayed in a text area.
   - **PDF Management:**
     Allows the admin to select a day (from Day 1 to Day 30) and upload a PDF file for that day using a file chooser. The list of days and their PDF upload status is shown.
3. **User Panel:**
   Users can select a day from a combo box and click the "View PDF" button.
   - If a PDF was uploaded for that day, its name and path are shown and an attempt is made to open it using the system's default PDF viewer.
   - Otherwise, a message indicates that no PDF is available for that day.

Running the Code

1. **Save the Code:** Save the code in a file named LoginSystem.java.
2. **Compile:**
   Open a terminal, navigate to the directory containing the file, and compile:

```
javac LoginSystem.java
```

3. **Run:**
   Run the application with:

```
java LoginSystem
```

This will open the login window. You can log in as admin using:

- **Username:** admin
- **Password:** admin123

Then use the Admin Panel to add user credentials and upload PDFs. Users can later log in with the created credentials to view the PDFs.

Feel free to modify and enhance the code as needed for your project!

connect it to a db mysql (i have xampp))make it dynamic

Enhancing database access

I'm planning to integrate MySQL with Java Swing using JDBC. This involves setting up a connection, tweaking user login steps, adding users to the DB, and synchronizing PDF information.

Laying out the options

I'm thinking about structure: three tables for admin, users, and PDFs. We could either keep admin hardcoded or use database storage. I'll also provide SQL for these tables.

Crafting database structure

I'm developing SQL queries for a login system, establishing tables for users and PDFs, and crafting Java Swing code to dynamically link to the database for various user functionalities.

Below is an example of how you can convert the previous Swing login system into a dynamic application that uses a MySQL database (for example, using XAMPP). In this example, we assume you have created a database named login_system with two tables: one for users (including the admin) and one for PDF file records. The application now performs the following actions dynamically via JDBC:

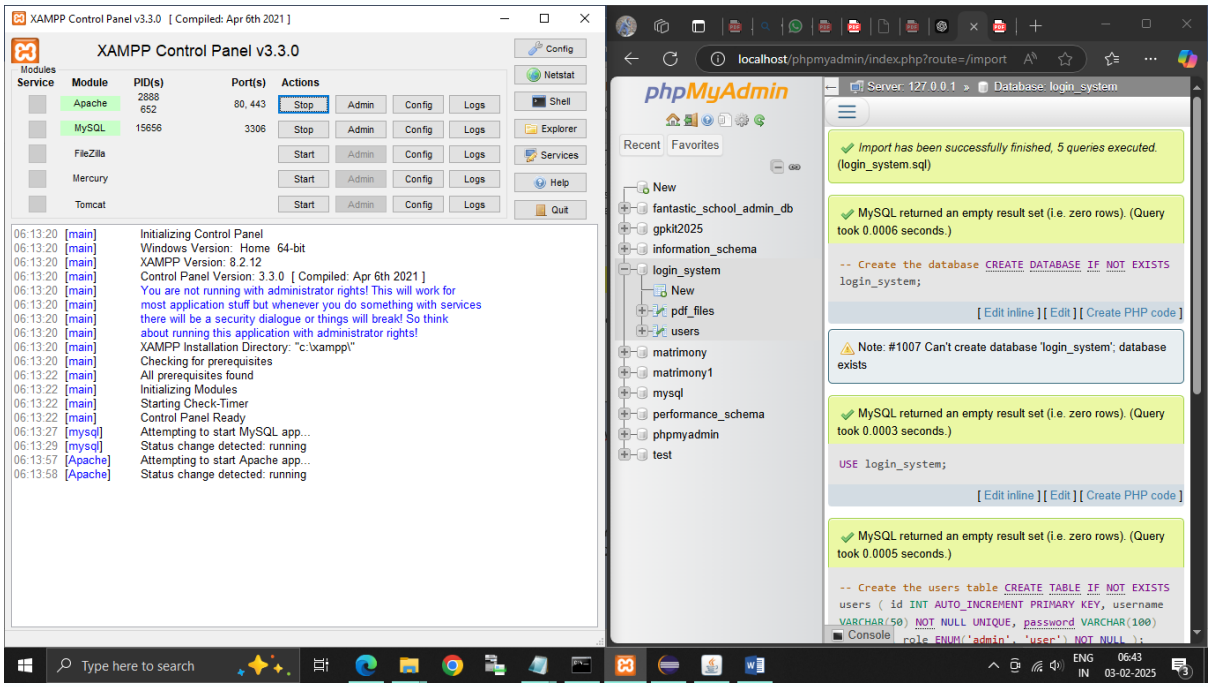
- **Login:** Verifies credentials against the database.
- **Admin Panel:**
  - **User Management:** Adds new user records and displays the list of users from the database.
  - **PDF Management:** Uploads (registers) a PDF file for a given day by storing the file path in the database.
- **User Panel:** Allows a user to select a day and view the PDF file path (and attempts to open the file) if it was uploaded by the admin.

Important:

- Make sure you have the MySQL Connector/J JDBC driver in your project's classpath.
- Update the connection parameters (URL, username, password) if they differ in your XAMPP setup.
- This example uses plain-text passwords for simplicity. In a production system, you would hash passwords and follow other security best practices.

Step 1. Create the Database and Tables

Using a MySQL client (such as phpMyAdmin included with XAMPP), run the following SQL commands to set up your database:




```
-- Create the database
CREATE DATABASE IF NOT EXISTS login_system;
USE login_system;

-- Create the users table
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(100) NOT NULL,
    role ENUM('admin', 'user') NOT NULL
);
```

```
-- Insert the admin user (username: admin, password: admin123)
INSERT IGNORE INTO users (username, password, role) VALUES ('admin', 'admin123',
'admin');

-- Create the pdf_files table
CREATE TABLE IF NOT EXISTS pdf_files (
    id INT AUTO_INCREMENT PRIMARY KEY,
    day VARCHAR(10) NOT NULL UNIQUE,
    file_path VARCHAR(255) NOT NULL
);
```

Create a new database name it >> create it and import above sql file.

Step 2. The Java Code

Save the following code as LoginSystem.java. It contains a helper class for database connectivity and the full Swing application. (You can refactor the code into separate files as needed.)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.io.IOException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class LoginSystem {

    // ================== DB Utility ==================
    public static class DBUtil {
        // Update these parameters if needed
        private static final String DB_URL = "jdbc:mysql://localhost:3306/login_system";
        private static final String DB_USER = "root";
        private static final String DB_PASSWORD = "";
```

```
        static {
            try {
                // Load MySQL JDBC driver
                Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException e) {
                JOptionPane.showMessageDialog(null, "MySQL JDBC Driver not found.");
                System.exit(1);
            }
        }

        public static Connection getConnection() throws SQLException {
            return DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
        }
    }

    // ================== Main ==================
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new LoginFrame());
    }

    // ================== Login Frame ==================
    static class LoginFrame extends JFrame {
        private JTextField usernameField;
        private JPasswordField passwordField;
        private JButton loginButton;
        private JLabel statusLabel;

        public LoginFrame() {
            setTitle("Login");
            setSize(350, 200);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setLocationRelativeTo(null);

            JPanel panel = new JPanel(new GridBagLayout());
            GridBagConstraints gbc = new GridBagConstraints();
            gbc.insets = new Insets(5, 5, 5, 5);
            gbc.fill = GridBagConstraints.HORIZONTAL;

            // Username
            gbc.gridx = 0;
            gbc.gridy = 0;
```

```
            panel.add(new JLabel("Username:"), gbc);
            gbc.gridx = 1;
            usernameField = new JTextField(15);
            panel.add(usernameField, gbc);

            // Password
            gbc.gridx = 0;
            gbc.gridy = 1;
            panel.add(new JLabel("Password:"), gbc);
            gbc.gridx = 1;
            passwordField = new JPasswordField(15);
            panel.add(passwordField, gbc);

            // Login button
            gbc.gridx = 0;
            gbc.gridy = 2;
            gbc.gridwidth = 2;
            gbc.anchor = GridBagConstraints.CENTER;
            loginButton = new JButton("Login");
            panel.add(loginButton, gbc);

            // Status label
            gbc.gridy = 3;
            statusLabel = new JLabel("");
            panel.add(statusLabel, gbc);

            add(panel);

            loginButton.addActionListener(e -> performLogin());

            setVisible(true);
        }

        private void performLogin() {
            String username = usernameField.getText().trim();
            String password = new String(passwordField.getPassword()).trim();

            try (Connection conn = DBUtil.getConnection()) {
                String sql = "SELECT role FROM users WHERE username = ? AND
password = ?";
                try (PreparedStatement stmt = conn.prepareStatement(sql)) {
```

```
                    stmt.setString(1, username);
                    stmt.setString(2, password);
                    try (ResultSet rs = stmt.executeQuery()) {
                        if (rs.next()) {
                            String role = rs.getString("role");
                            if ("admin".equals(role)) {
                                statusLabel.setForeground(Color.GREEN);
                                statusLabel.setText("Admin login successful.");
                                SwingUtilities.invokeLater(() -> new AdminPanel());
                                dispose();
                            } else if ("user".equals(role)) {
                                statusLabel.setForeground(Color.GREEN);
                                statusLabel.setText("User login successful.");
                                SwingUtilities.invokeLater(() -> new UserPanel(username));
                                dispose();
                            }
                        } else {
                            statusLabel.setForeground(Color.RED);
                            statusLabel.setText("Invalid credentials.");
                        }
                    }
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Database error: " +
ex.getMessage());
            }
        }
    }

    // ================== Admin Panel ==================
    static class AdminPanel extends JFrame {
        private JTabbedPane tabbedPane;
        private JPanel userManagementPanel;
        private JPanel pdfManagementPanel;

        // User Management components
        private JTextField newUsernameField;
        private JPasswordField newPasswordField;
        private JButton addUserButton;
        private JTextArea userListArea;
```

```
    // PDF Management components
    private JComboBox<String> dayComboBox;
    private JButton uploadPdfButton;
    private JTextArea pdfListArea;

    public AdminPanel() {
        setTitle("Admin Panel");
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        tabbedPane = new JTabbedPane();
        userManagementPanel = new JPanel(new GridBagLayout());
        pdfManagementPanel = new JPanel(new GridBagLayout());

        createUserManagementPanel();
        createPdfManagementPanel();

        tabbedPane.addTab("User Management", userManagementPanel);
        tabbedPane.addTab("PDF Management", pdfManagementPanel);
        add(tabbedPane);

        setVisible(true);
    }

    // ----- User Management Panel -----
    private void createUserManagementPanel() {
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        // New Username
        gbc.gridx = 0;
        gbc.gridy = 0;
        userManagementPanel.add(new JLabel("New Username:"), gbc);
        gbc.gridx = 1;
        newUsernameField = new JTextField(15);
        userManagementPanel.add(newUsernameField, gbc);

        // New Password
```

```
            gbc.gridx = 0;
            gbc.gridy = 1;
            userManagementPanel.add(new JLabel("New Password:"), gbc);
            gbc.gridx = 1;
            newPasswordField = new JPasswordField(15);
            userManagementPanel.add(newPasswordField, gbc);

            // Add User Button
            gbc.gridx = 0;
            gbc.gridy = 2;
            gbc.gridwidth = 2;
            gbc.anchor = GridBagConstraints.CENTER;
            addUserButton = new JButton("Add User");
            userManagementPanel.add(addUserButton, gbc);

            // User list area
            gbc.gridy = 3;
            userListArea = new JTextArea(8, 30);
            userListArea.setEditable(false);
            JScrollPane scrollPane = new JScrollPane(userListArea);
            userManagementPanel.add(scrollPane, gbc);

            addUserButton.addActionListener(e -> addUser());
            updateUserList();
        }

        private void addUser() {
            String newUser = newUsernameField.getText().trim();
            String newPass = new String(newPasswordField.getPassword()).trim();

            if (newUser.isEmpty() || newPass.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Username and password cannot
be empty.");
                return;
            }
            if ("admin".equalsIgnoreCase(newUser)) {
                JOptionPane.showMessageDialog(this, "Invalid username.");
                return;
            }

            try (Connection conn = DBUtil.getConnection()) {
```

```
                // Check if user exists
                String checkSql = "SELECT id FROM users WHERE username = ?";
                try (PreparedStatement checkStmt = conn.prepareStatement(checkSql)) {
                    checkStmt.setString(1, newUser);
                    try (ResultSet rs = checkStmt.executeQuery()) {
                        if (rs.next()) {
                            JOptionPane.showMessageDialog(this, "User already exists.");
                            return;
                        }
                    }
                }

                // Insert new user (role 'user')
                String insertSql = "INSERT INTO users (username, password, role)
VALUES (?, ?, 'user')";
                try (PreparedStatement stmt = conn.prepareStatement(insertSql)) {
                    stmt.setString(1, newUser);
                    stmt.setString(2, newPass);
                    int rows = stmt.executeUpdate();
                    if (rows > 0) {
                        JOptionPane.showMessageDialog(this, "User added
successfully.");
                        newUsernameField.setText("");
                        newPasswordField.setText("");
                        updateUserList();
                    }
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Database error: " +
ex.getMessage());
            }
        }

        private void updateUserList() {
            try (Connection conn = DBUtil.getConnection()) {
                String sql = "SELECT username, role FROM users WHERE role =
'user'";
                try (Statement stmt = conn.createStatement();
                     ResultSet rs = stmt.executeQuery(sql)) {
```

```
                    StringBuilder sb = new StringBuilder();
                    while (rs.next()) {
                        sb.append(rs.getString("username")).append(" (")
                          .append(rs.getString("role")).append(")\n");
                    }
                    userListArea.setText(sb.toString());
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Database error: " +
ex.getMessage());
            }
        }

        // ----- PDF Management Panel -----
        private void createPdfManagementPanel() {
            GridBagConstraints gbc = new GridBagConstraints();
            gbc.insets = new Insets(5, 5, 5, 5);
            gbc.fill = GridBagConstraints.HORIZONTAL;

            // Day selection
            gbc.gridx = 0;
            gbc.gridy = 0;
            pdfManagementPanel.add(new JLabel("Select Day:"), gbc);
            gbc.gridx = 1;
            String[] days = new String[30];
            for (int i = 0; i < 30; i++) {
                days[i] = "Day " + (i + 1);
            }
            dayComboBox = new JComboBox<>(days);
            pdfManagementPanel.add(dayComboBox, gbc);

            // Upload PDF button
            gbc.gridx = 0;
            gbc.gridy = 1;
            gbc.gridwidth = 2;
            gbc.anchor = GridBagConstraints.CENTER;
            uploadPdfButton = new JButton("Upload PDF");
            pdfManagementPanel.add(uploadPdfButton, gbc);

            // PDF list area
```

```
            gbc.gridy = 2;
            pdfListArea = new JTextArea(8, 30);
            pdfListArea.setEditable(false);
            JScrollPane scrollPane = new JScrollPane(pdfListArea);
            pdfManagementPanel.add(scrollPane, gbc);

            uploadPdfButton.addActionListener(e -> uploadPdf());
            updatePdfList();
        }

        private void uploadPdf() {
            JFileChooser fileChooser = new JFileChooser();
            int result = fileChooser.showOpenDialog(this);
            if (result == JFileChooser.APPROVE_OPTION) {
                File selectedFile = fileChooser.getSelectedFile();
                String day = (String) dayComboBox.getSelectedItem();

                // You might want to verify the file extension here.
                try (Connection conn = DBUtil.getConnection()) {
                    // Check if a record for the day exists
                    String checkSql = "SELECT id FROM pdf_files WHERE day = ?";
                    try (PreparedStatement checkStmt =
conn.prepareStatement(checkSql)) {
                        checkStmt.setString(1, day);
                        try (ResultSet rs = checkStmt.executeQuery()) {
                            if (rs.next()) {
                                // Update existing record
                                String updateSql = "UPDATE pdf_files SET file_path = ?
WHERE day = ?";
                                try (PreparedStatement updateStmt =
conn.prepareStatement(updateSql)) {
                                    updateStmt.setString(1,
selectedFile.getAbsolutePath());
                                    updateStmt.setString(2, day);
                                    updateStmt.executeUpdate();
                                }
                            } else {
                                // Insert new record
                                String insertSql = "INSERT INTO pdf_files (day, file_path)
VALUES (?, ?)";
```

```
                                try (PreparedStatement insertStmt =
conn.prepareStatement(insertSql)) {
                                    insertStmt.setString(1, day);
                                    insertStmt.setString(2, selectedFile.getAbsolutePath());
                                    insertStmt.executeUpdate();
                                }
                            }
                        }
                    }
                    JOptionPane.showMessageDialog(this, "PDF uploaded for " + day);
                    updatePdfList();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                    JOptionPane.showMessageDialog(this, "Database error: " +
ex.getMessage());
                }
            }
        }

        private void updatePdfList() {
            try (Connection conn = DBUtil.getConnection()) {
                String sql = "SELECT day, file_path FROM pdf_files ORDER BY id";
                try (Statement stmt = conn.createStatement();
                     ResultSet rs = stmt.executeQuery(sql)) {
                    // Build list for all 30 days
                    StringBuilder sb = new StringBuilder();
                    for (int i = 1; i <= 30; i++) {
                        String day = "Day " + i;
                        sb.append(day).append(": ");
                        boolean found = false;
                        rs.beforeFirst(); // Reset cursor to beginning
                        while (rs.next()) {
                            if (day.equals(rs.getString("day"))) {
                                sb.append(new File(rs.getString("file_path")).getName());
                                found = true;
                                break;
                            }
                        }
                        if (!found) {
                            sb.append("No PDF uploaded");
                        }
```

```
                        sb.append("\n");
                    }
                    pdfListArea.setText(sb.toString());
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Database error: " +
ex.getMessage());
            }
        }
    }

    // ================== User Panel ==================
    static class UserPanel extends JFrame {
        private JComboBox<String> dayComboBox;
        private JButton viewPdfButton;
        private JTextArea pdfInfoArea;
        private String username;

        public UserPanel(String username) {
            this.username = username;
            setTitle("User Panel - " + username);
            setSize(400, 300);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setLocationRelativeTo(null);

            JPanel panel = new JPanel(new GridBagLayout());
            GridBagConstraints gbc = new GridBagConstraints();
            gbc.insets = new Insets(5, 5, 5, 5);
            gbc.fill = GridBagConstraints.HORIZONTAL;

            // Day selection
            gbc.gridx = 0;
            gbc.gridy = 0;
            panel.add(new JLabel("Select Day:"), gbc);
            gbc.gridx = 1;
            String[] days = new String[30];
            for (int i = 0; i < 30; i++) {
                days[i] = "Day " + (i + 1);
            }
            dayComboBox = new JComboBox<>(days);
```

```
            panel.add(dayComboBox, gbc);

            // View PDF button
            gbc.gridx = 0;
            gbc.gridy = 1;
            gbc.gridwidth = 2;
            viewPdfButton = new JButton("View PDF");
            panel.add(viewPdfButton, gbc);

            // PDF info area
            gbc.gridy = 2;
            pdfInfoArea = new JTextArea(8, 30);
            pdfInfoArea.setEditable(false);
            JScrollPane scrollPane = new JScrollPane(pdfInfoArea);
            panel.add(scrollPane, gbc);

            viewPdfButton.addActionListener(e -> viewPdf());

            add(panel);
            setVisible(true);
        }

        private void viewPdf() {
            String day = (String) dayComboBox.getSelectedItem();
            try (Connection conn = DBUtil.getConnection()) {
                String sql = "SELECT file_path FROM pdf_files WHERE day = ?";
                try (PreparedStatement stmt = conn.prepareStatement(sql)) {
                    stmt.setString(1, day);
                    try (ResultSet rs = stmt.executeQuery()) {
                        if (rs.next()) {
                            String filePath = rs.getString("file_path");
                            File pdfFile = new File(filePath);
                            pdfInfoArea.setText("PDF for " + day + ":\n" +
pdfFile.getName() + "\nPath: " + pdfFile.getAbsolutePath());
                            if (Desktop.isDesktopSupported()) {
                                try {
                                    Desktop.getDesktop().open(pdfFile);
                                } catch (IOException ex) {
                                    JOptionPane.showMessageDialog(this, "Error opening
PDF file.");
                                }
```

```
                            }
                        } else {
                            pdfInfoArea.setText("No PDF uploaded for " + day);
                        }
                    }
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Database error: " +
ex.getMessage());
            }
        }
    }
}
```

Step 3. Running the Application

1. **Ensure the MySQL JDBC Driver is in the Classpath:**
   For example, if you are compiling from the command line, add the connector JAR:

```
javac -cp .;mysql-connector-java-8.0.xx.jar LoginSystem.java
```

```
javac -cp .;C:\Users\admin\Desktop\java_project_feb2025\db\ext_jar\mysql-connector-j-9.2.0.jar LoginSystem.java
```

   And run it with:

```
java -cp .;mysql-connector-java-8.0.xx.jar LoginSystem
```

   (Replace mysql-connector-java-8.0.xx.jar with the actual file name.)

```
java -cp .;C:\Users\admin\Desktop\java_project_feb2025\db\ext_jar\mysql-connector-j-9.2.0.jar LoginSystem.java
```

2. **Start XAMPP and MySQL:**
   Make sure MySQL is running and your database and tables have been created.
3. **Run the Application:**
   When you launch the application, the login window appears. You can log in as:
   - **Admin:**
     - Username: admin
     - Password: admin123
   - **User:**
     First, add a user via the admin panel or insert a record directly into the database.
4. **Use the Admin and User Panels:**
   - The **Admin Panel** allows you to add users and upload (register) PDFs for a specific day.
   - The **User Panel** lets users select a day and view the PDF (attempting to open it using the system's default application).

This example demonstrates a dynamic Java Swing application connected to a MySQL database. You can further enhance this by adding features such as password hashing, file storage (instead of just the file path), and additional error handling as needed.

To use the MySQL JDBC driver in your Java project, you must ensure that the connector JAR file is available on your classpath when compiling and running your application. Here are several common methods:

---

1. Using the Command Line

a. Download the Connector

1. Download the latest MySQL Connector/J from the MySQL Downloads page.
2. Extract the ZIP file and locate the JAR file (e.g., mysql-connector-java-8.0.xx.jar).

b. Compile and Run Your Code

On Windows:

- **Compile:**
  Open a Command Prompt, navigate to your source folder, and run:

  ```
  javac -cp .;path\to\mysql-connector-java-8.0.xx.jar YourMainClass.java
  ```

  (Replace path\to\mysql-connector-java-8.0.xx.jar with the actual path to the JAR file.)

- **Run:**

  ```
  java -cp .;path\to\mysql-connector-java-8.0.xx.jar YourMainClass
  ```

On macOS/Linux:

- **Compile:**
  Open a Terminal, navigate to your source folder, and run:

  ```
  javac -cp .:path/to/mysql-connector-java-8.0.xx.jar YourMainClass.java
  ```

- **Run:**

  ```
  java -cp .:path/to/mysql-connector-java-8.0.xx.jar YourMainClass
  ```

Note: The separator in the classpath is a semicolon (;) on Windows and a colon (:) on Unix-like systems.

---

2. Using an IDE

Most Integrated Development Environments (IDEs) like **Eclipse**, **IntelliJ IDEA**, or **NetBeans** allow you to add external JARs to your project.

In Eclipse:

1. **Right-click** your project in the **Package Explorer** and select **Build Path** > **Configure Build Path**.
2. Click the **Libraries** tab.
3. Click **Add External JARs...** and browse to the location of mysql-connector-java-8.0.xx.jar.
4. Select the JAR and click **Open**.
5. Click **Apply and Close**.

In IntelliJ IDEA:

1. Open **File > Project Structure**.
2. Under **Modules**, select your module.
3. Go to the **Dependencies** tab.
4. Click the **+** button and choose **JARs or directories**.
5. Browse to and select the MySQL connector JAR.
6. Click **OK** to add it to your project dependencies.

In NetBeans:

1. Right-click the **Libraries** folder under your project.
2. Choose **Add JAR/Folder...**.
3. Browse and select the MySQL connector JAR.
4. Click **Open** to add it.

---

3. Using Build Tools (Maven or Gradle)

Maven:

If you are using Maven, add the dependency in your pom.xml:

```
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.xx</version>
    </dependency>
</dependencies>
```

Replace 8.0.xx with the actual version number.

Gradle:

If you are using Gradle, add the dependency in your build.gradle file:

```
dependencies {
    implementation 'mysql:mysql-connector-java:8.0.xx'
}
```

Replace 8.0.xx with the actual version number.

Summary

- **Command Line:** Use the -cp option with the path to the JAR file.
- **IDE:** Add the connector JAR as an external library in your project's build path.
- **Build Tools:** Define the dependency in your pom.xml (Maven) or build.gradle (Gradle).

Once the connector JAR is on your classpath, your application will be able to load the MySQL JDBC driver (for example, using Class.forName("com.mysql.cj.jdbc.Driver")), and you can establish connections to your MySQL database.

2. Compile and Run Using the Command Line

Assume that:

- Your Java source file (MyApp.java) is in the current directory.
- The MySQL Connector/J JAR file is located at:
  C:\libs\mysql-connector-java-8.0.26.jar
  (Replace 8.0.26 with your actual version.)

a. Compiling

Open a Command Prompt, navigate to the directory containing MyApp.java, and run:

```
javac -cp .;C:\libs\mysql-connector-java-8.0.26.jar MyApp.java
```

Explanation:

- -cp .;C:\libs\mysql-connector-java-8.0.26.jar tells the compiler to include the current directory (.) and the specified MySQL Connector JAR file on the classpath.

b. Running

After a successful compilation, run the application with:

```
java -cp .;C:\libs\mysql-connector-java-8.0.26.jar MyApp
```

Explanation:

- Again, -cp .;C:\libs\mysql-connector-java-8.0.26.jar includes both the current directory and the JAR on the classpath.
- MyApp is the name of the class with the main method.

3. Summary

- **Compile Command (Windows):**

  ```
  javac -cp .;C:\libs\mysql-connector-java-8.0.26.jar MyApp.java
  ```

- **Run Command (Windows):**

```
java -cp .;C:\libs\mysql-connector-java-8.0.26.jar MyApp
```

This example shows how to include the MySQL JDBC driver in your classpath when compiling and running your Java application from the command line on Windows. If you're using macOS or Linux, remember to replace the semicolon (;) with a colon (:) in your classpath:

```
javac -cp .:path/to/mysql-connector-java-8.0.26.jar MyApp.java
java -cp .:path/to/mysql-connector-java-8.0.26.jar MyApp
```

o3-mini

This error typically occurs when you try to perform an operation that requires a scrollable result set (such as calling rs.beforeFirst()) on a result set that is, by default, forward-only. By default, when you create a statement (or a prepared statement), the result set is of type ResultSet.TYPE_FORWARD_ONLY, which does not support methods like beforeFirst(), afterLast(), or absolute(int row).

There are two main ways to resolve this:

---

1. Create a Scrollable ResultSet

If you need to call methods such as beforeFirst(), you can create your statement with a scrollable result set. For example:

```
try (Connection conn = DBUtil.getConnection()) {
    String sql = "SELECT day, file_path FROM pdf_files ORDER BY id";
    // Create a scrollable result set
    Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
    ResultSet rs = stmt.executeQuery(sql);

    // Now you can call rs.beforeFirst(), etc.
    // Example: iterate and then reset cursor:
    while (rs.next()) {
        System.out.println("Day: " + rs.getString("day"));
    }
    rs.beforeFirst(); // Reset cursor to before the first row, if needed

    // Process the result set again if needed
    while (rs.next()) {
```

```
            System.out.println("File: " + rs.getString("file_path"));
        }

        rs.close();
        stmt.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
```

Key Points:

- **ResultSet.TYPE_SCROLL_INSENSITIVE** allows you to scroll forwards and backwards.
- **ResultSet.CONCUR_READ_ONLY** indicates that the result set is read-only.

---

2. Avoid Needing a Scrollable ResultSet

If you don't actually need to scroll back in the result set, you can avoid using methods such as rs.beforeFirst(). For example, if you need to iterate over the rows more than once, consider storing the data in a collection or re-running the query. Here's one way to store data in a list:

```
List<String> pdfInfo = new ArrayList<>();
try (Connection conn = DBUtil.getConnection()) {
    String sql = "SELECT day, file_path FROM pdf_files ORDER BY id";
    try (Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {

        // Store results in a list for later processing
        while (rs.next()) {
            String day = rs.getString("day");
            String filePath = rs.getString("file_path");
            pdfInfo.add(day + ": " + filePath);
        }
    }
} catch (SQLException ex) {
    ex.printStackTrace();
}

// Now you can iterate over pdfInfo as many times as needed.
```

```
for (String info : pdfInfo) {
    System.out.println(info);
}
```

Which Approach to Choose?

- **Scrollable ResultSet:**
  Choose this if you need to move the cursor backwards or reset it within the same query result. Keep in mind that scrollable result sets may have slightly more overhead compared to forward-only ones.
- **Storing Data in a Collection:**
  If you only need to process the results multiple times (or in a non-sequential manner) and the data set is reasonably small, you can simply store the results in a collection like a List or Map.

Summary

java.sql.SQLException: Operation not allowed for a result set of type ResultSet.TYPE_FORWARD_ONLY.

is resolved by either:

1. **Creating a scrollable result set** when you execute the query:

   ```
   Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
   ```

2. **Avoiding methods that require scrolling** (like beforeFirst()) on a forward-only result set and instead processing the results in a single pass or storing them for later use.

Choose the approach that best fits your needs and application design.