import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import java.util.\*;

import javax.swing.table.DefaultTableModel;

public class BookReservationSystem {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new LoginFrame());

}

static Connection connect() {

try {

return DriverManager.getConnection(

"jdbc:mysql://localhost:3306/library", "root", "Sathwik@2718");

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "Database connection failed!");

return null;

}

}

// ---------- LOGIN FRAME ----------

static class LoginFrame extends JFrame {

JTextField usernameField;

JPasswordField passwordField;

LoginFrame() {

setTitle("Login");

setSize(350, 250); // Increased height for admin button

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

JLabel userLabel = new JLabel("Username:");

JLabel passLabel = new JLabel("Password:");

usernameField = new JTextField(15);

passwordField = new JPasswordField(15);

JButton loginBtn = new JButton("Login");

JButton registerBtn = new JButton("Register");

JButton adminBtn = new JButton("Admin Login");

gbc.insets = new Insets(5, 5, 5, 5);

gbc.gridx = 0; gbc.gridy = 0; add(userLabel, gbc);

gbc.gridx = 1; add(usernameField, gbc);

gbc.gridx = 0; gbc.gridy = 1; add(passLabel, gbc);

gbc.gridx = 1; add(passwordField, gbc);

gbc.gridx = 1; gbc.gridy = 2; add(loginBtn, gbc);

gbc.gridx = 1; gbc.gridy = 3; add(registerBtn, gbc);

gbc.gridx = 1; gbc.gridy = 4; add(adminBtn, gbc);

loginBtn.addActionListener(e -> authenticate());

registerBtn.addActionListener(e -> new RegisterFrame());

adminBtn.addActionListener(e -> authenticateAdmin());

setLocationRelativeTo(null);

setVisible(true);

}

void authenticateAdmin() {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement stmt = conn.prepareStatement(

"SELECT \* FROM admin WHERE username=? AND password=?");

stmt.setString(1, username);

stmt.setString(2, password);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

JOptionPane.showMessageDialog(this, "Admin login successful!");

dispose();

new AdminFrame();

} else {

JOptionPane.showMessageDialog(this, "Invalid admin credentials!");

}

} catch (SQLException e) {

e.printStackTrace();

}

}

void authenticate() {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement stmt = conn.prepareStatement(

"SELECT user\_id FROM users1 WHERE username=? AND password=?");

stmt.setString(1, username);

stmt.setString(2, password);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

int userId = rs.getInt("user\_id");

JOptionPane.showMessageDialog(this, "Login successful!");

dispose();

new ReservationFrame(userId);

} else {

JOptionPane.showMessageDialog(this, "Invalid credentials!");

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

// ---------- REGISTER FRAME ----------

static class RegisterFrame extends JFrame {

JTextField usernameField;

JPasswordField passwordField;

RegisterFrame() {

setTitle("Register");

setSize(350, 220);

setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

setLayout(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

usernameField = new JTextField(15);

passwordField = new JPasswordField(15);

JButton registerBtn = new JButton("Register");

gbc.insets = new Insets(5, 5, 5, 5);

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

gbc.gridx = 1; add(usernameField, gbc);

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

gbc.gridx = 1; add(passwordField, gbc);

gbc.gridx = 1; gbc.gridy = 2; add(registerBtn, gbc);

registerBtn.addActionListener(e -> register());

setLocationRelativeTo(null);

setVisible(true);

}

void register() {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement checkStmt = conn.prepareStatement(

"SELECT \* FROM users1 WHERE username = ?");

checkStmt.setString(1, username);

ResultSet rs = checkStmt.executeQuery();

if (rs.next()) {

JOptionPane.showMessageDialog(this, "User already exists.");

return;

}

PreparedStatement maxIdStmt = conn.prepareStatement("SELECT MAX(user\_id) FROM users1");

ResultSet maxRs = maxIdStmt.executeQuery();

int newId = 1;

if (maxRs.next() && maxRs.getInt(1) != 0) {

newId = maxRs.getInt(1) + 1;

}

PreparedStatement insertStmt = conn.prepareStatement(

"INSERT INTO users1 (user\_id, username, password) VALUES (?, ?, ?)");

insertStmt.setInt(1, newId);

insertStmt.setString(2, username);

insertStmt.setString(3, password);

insertStmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Registration successful!");

dispose();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

// ---------- RESERVATION FRAME ----------

static class ReservationFrame extends JFrame {

JComboBox<String> bookDropdown;

JTextField daysField;

int userId;

Map<String, Integer> bookMap = new HashMap<>();

JTable userBooksTable;

DefaultTableModel tableModel;

JScrollPane scrollPane;

JLabel userInfoLabel;

JLabel bookCountLabel;

JPanel mainPanel;

CardLayout cardLayout;

JPanel reservationPanel;

JPanel viewBooksPanel;

JPanel welcomePanel;

JPanel returnBooksPanel;

ReservationFrame(int userId) {

this.userId = userId;

setTitle("Book Reservation");

setSize(800, 600);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

// Create user info panel

JPanel userInfoPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

userInfoLabel = new JLabel();

bookCountLabel = new JLabel();

userInfoPanel.add(userInfoLabel);

userInfoPanel.add(new JLabel(" | "));

userInfoPanel.add(bookCountLabel);

loadUserInfo();

// Create navigation buttons panel

JPanel navPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));

JButton reserveBtn = new JButton("Reserve a Book");

JButton viewBooksBtn = new JButton("View My Reservations");

JButton returnBtn = new JButton("Return Books");

JButton logoutBtn = new JButton("Logout");

navPanel.add(reserveBtn);

navPanel.add(viewBooksBtn);

navPanel.add(returnBtn);

navPanel.add(logoutBtn);

// Create card layout for switching between panels

cardLayout = new CardLayout();

mainPanel = new JPanel(cardLayout);

// Create welcome panel

welcomePanel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(20, 20, 20, 20);

JLabel welcomeLabel = new JLabel("Welcome to Book Reservation System");

welcomeLabel.setFont(new Font("Arial", Font.BOLD, 24));

welcomePanel.add(welcomeLabel, gbc);

// Create other panels

reservationPanel = createReservationPanel();

viewBooksPanel = createViewBooksPanel();

returnBooksPanel = createReturnBooksPanel();

// Add panels to card layout

mainPanel.add(welcomePanel, "WELCOME");

mainPanel.add(reservationPanel, "RESERVE");

mainPanel.add(viewBooksPanel, "VIEW");

mainPanel.add(returnBooksPanel, "RETURN");

// Add action listeners for navigation

reserveBtn.addActionListener(e -> cardLayout.show(mainPanel, "RESERVE"));

viewBooksBtn.addActionListener(e -> {

loadUserBooks();

cardLayout.show(mainPanel, "VIEW");

});

returnBtn.addActionListener(e -> {

loadReturnableBooks();

cardLayout.show(mainPanel, "RETURN");

});

logoutBtn.addActionListener(e -> {

dispose();

new LoginFrame();

});

// Add all panels to frame

add(userInfoPanel, BorderLayout.NORTH);

add(navPanel, BorderLayout.CENTER);

add(mainPanel, BorderLayout.SOUTH);

// Show welcome panel by default

cardLayout.show(mainPanel, "WELCOME");

setLocationRelativeTo(null);

setVisible(true);

}

private JPanel createReservationPanel() {

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(5, 5, 5, 5);

bookDropdown = new JComboBox<>();

daysField = new JTextField(10);

JButton submitReserveBtn = new JButton("Submit Reservation");

loadAvailableBooks();

gbc.gridx = 0; gbc.gridy = 0; panel.add(new JLabel("Select Book:"), gbc);

gbc.gridx = 1; panel.add(bookDropdown, gbc);

gbc.gridx = 0; gbc.gridy = 1; panel.add(new JLabel("Number of Days:"), gbc);

gbc.gridx = 1; panel.add(daysField, gbc);

gbc.gridx = 1; gbc.gridy = 2; panel.add(submitReserveBtn, gbc);

submitReserveBtn.addActionListener(e -> reserveBook());

return panel;

}

private JPanel createViewBooksPanel() {

JPanel panel = new JPanel(new BorderLayout());

// Create table for user's books

tableModel = new DefaultTableModel();

userBooksTable = new JTable(tableModel);

scrollPane = new JScrollPane(userBooksTable);

panel.add(new JLabel("Your Reserved Books:"), BorderLayout.NORTH);

panel.add(scrollPane, BorderLayout.CENTER);

return panel;

}

private JPanel createReturnBooksPanel() {

JPanel panel = new JPanel(new BorderLayout());

// Create table for returnable books

DefaultTableModel returnTableModel = new DefaultTableModel();

JTable returnTable = new JTable(returnTableModel);

JScrollPane returnScrollPane = new JScrollPane(returnTable);

// Add columns

returnTableModel.addColumn("Book Title");

returnTableModel.addColumn("Author");

returnTableModel.addColumn("Due Date");

returnTableModel.addColumn("Days Overdue");

returnTableModel.addColumn("Fine Amount");

// Create return button

JButton returnButton = new JButton("Return Selected Book");

returnButton.addActionListener(e -> {

int selectedRow = returnTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a book to return.");

return;

}

String bookTitle = (String) returnTable.getValueAt(selectedRow, 0);

returnBook(bookTitle, returnTableModel, selectedRow);

});

panel.add(new JLabel("Select a book to return:"), BorderLayout.NORTH);

panel.add(returnScrollPane, BorderLayout.CENTER);

panel.add(returnButton, BorderLayout.SOUTH);

return panel;

}

void loadUserInfo() {

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement stmt = conn.prepareStatement(

"SELECT username FROM users1 WHERE user\_id = ?");

stmt.setInt(1, userId);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

String username = rs.getString("username");

userInfoLabel.setText("Welcome, " + username);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

void loadUserBooks() {

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement stmt = conn.prepareStatement(

"SELECT b.title, b.author, r.reserve\_days, r.reservation\_date, " +

"DATE\_ADD(r.reservation\_date, INTERVAL r.reserve\_days DAY) as due\_date " +

"FROM reservations r " +

"JOIN books b ON r.book\_id = b.book\_id " +

"WHERE r.user\_id = ? " +

"ORDER BY r.reservation\_date DESC");

stmt.setInt(1, userId);

ResultSet rs = stmt.executeQuery();

// Clear existing table data

tableModel.setRowCount(0);

tableModel.setColumnCount(0);

// Add columns

tableModel.addColumn("Book Title");

tableModel.addColumn("Author");

tableModel.addColumn("Reserved Days");

tableModel.addColumn("Reservation Date");

tableModel.addColumn("Due Date");

// Add rows and count them

int rowCount = 0;

while (rs.next()) {

rowCount++;

Object[] row = {

rs.getString("title"),

rs.getString("author"),

rs.getInt("reserve\_days"),

rs.getDate("reservation\_date"),

rs.getDate("due\_date")

};

tableModel.addRow(row);

}

// Update book count label

bookCountLabel.setText("Total Books Reserved: " + rowCount);

if (rowCount == 0) {

// Add a message row if no books are found

Object[] messageRow = {"No books reserved", "", "", "", ""};

tableModel.addRow(messageRow);

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading user's books: " + e.getMessage());

}

}

void loadAvailableBooks() {

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement stmt = conn.prepareStatement(

"SELECT book\_id, title FROM books WHERE is\_checked\_out = FALSE");

ResultSet rs = stmt.executeQuery();

bookDropdown.removeAllItems();

bookMap.clear();

while (rs.next()) {

String title = rs.getString("title");

int id = rs.getInt("book\_id");

bookDropdown.addItem(title);

bookMap.put(title, id);

}

if (bookDropdown.getItemCount() == 0) {

bookDropdown.addItem("No books available");

bookDropdown.setEnabled(false);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

void reserveBook() {

String selectedTitle = (String) bookDropdown.getSelectedItem();

if (selectedTitle == null || !bookMap.containsKey(selectedTitle)) {

JOptionPane.showMessageDialog(this, "Please select a valid book.");

return;

}

int bookId = bookMap.get(selectedTitle);

int days;

try {

days = Integer.parseInt(daysField.getText());

if (days <= 0) {

throw new NumberFormatException();

}

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "Please enter a valid number of days (greater than 0).");

return;

}

try (Connection conn = BookReservationSystem.connect()) {

conn.setAutoCommit(false);

// Double check availability

PreparedStatement check = conn.prepareStatement(

"SELECT is\_checked\_out FROM books WHERE book\_id = ?");

check.setInt(1, bookId);

ResultSet rs = check.executeQuery();

if (rs.next() && rs.getBoolean("is\_checked\_out")) {

JOptionPane.showMessageDialog(this, "Book is already reserved!");

return;

}

// Reserve the book

PreparedStatement insert = conn.prepareStatement(

"INSERT INTO reservations (user\_id, book\_id, reserve\_days, reservation\_date) " +

"VALUES (?, ?, ?, CURRENT\_DATE)");

insert.setInt(1, userId);

insert.setInt(2, bookId);

insert.setInt(3, days);

insert.executeUpdate();

// Update book status

PreparedStatement update = conn.prepareStatement(

"UPDATE books SET is\_checked\_out = TRUE WHERE book\_id = ?");

update.setInt(1, bookId);

update.executeUpdate();

conn.commit();

// Get book info for display

PreparedStatement info = conn.prepareStatement(

"SELECT author FROM books WHERE book\_id = ?");

info.setInt(1, bookId);

ResultSet infoRs = info.executeQuery();

String author = "Unknown";

if (infoRs.next()) {

author = infoRs.getString("author");

}

JOptionPane.showMessageDialog(this,

"Reservation Successful!\n" +

"Book: " + selectedTitle + "\nAuthor: " + author + "\nDays: " + days);

// Refresh the available books and user's books

loadAvailableBooks();

loadUserBooks();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Reservation failed.");

}

}

void loadReturnableBooks() {

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement stmt = conn.prepareStatement(

"SELECT b.title, b.author, " +

"DATE\_ADD(r.reservation\_date, INTERVAL r.reserve\_days DAY) as due\_date, " +

"DATEDIFF(CURRENT\_DATE, DATE\_ADD(r.reservation\_date, INTERVAL r.reserve\_days DAY)) as days\_overdue " +

"FROM reservations r " +

"JOIN books b ON r.book\_id = b.book\_id " +

"WHERE r.user\_id = ? AND b.is\_checked\_out = TRUE " +

"ORDER BY due\_date");

stmt.setInt(1, userId);

ResultSet rs = stmt.executeQuery();

DefaultTableModel model = (DefaultTableModel) ((JTable)((JScrollPane)((JPanel)mainPanel.getComponent(3)).getComponent(1)).getViewport().getView()).getModel();

model.setRowCount(0);

while (rs.next()) {

String title = rs.getString("title");

String author = rs.getString("author");

java.sql.Date dueDate = rs.getDate("due\_date");

int daysOverdue = rs.getInt("days\_overdue");

double fine = Math.max(0, daysOverdue \* 10.0); // 10 rupees per day

Object[] row = {

title,

author,

dueDate,

Math.max(0, daysOverdue),

String.format("₹%.2f", fine)

};

model.addRow(row);

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading returnable books: " + e.getMessage());

}

}

void returnBook(String bookTitle, DefaultTableModel model, int selectedRow) {

try (Connection conn = BookReservationSystem.connect()) {

conn.setAutoCommit(false);

// Get book\_id and calculate fine

PreparedStatement stmt = conn.prepareStatement(

"SELECT b.book\_id, " +

"DATEDIFF(CURRENT\_DATE, DATE\_ADD(r.reservation\_date, INTERVAL r.reserve\_days DAY)) as days\_overdue " +

"FROM reservations r " +

"JOIN books b ON r.book\_id = b.book\_id " +

"WHERE b.title = ? AND r.user\_id = ?");

stmt.setString(1, bookTitle);

stmt.setInt(2, userId);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

int bookId = rs.getInt("book\_id");

int daysOverdue = rs.getInt("days\_overdue");

double fine = Math.max(0, daysOverdue \* 10.0);

// Update book status

PreparedStatement updateBook = conn.prepareStatement(

"UPDATE books SET is\_checked\_out = FALSE WHERE book\_id = ?");

updateBook.setInt(1, bookId);

updateBook.executeUpdate();

// Remove reservation

PreparedStatement deleteReservation = conn.prepareStatement(

"DELETE FROM reservations WHERE book\_id = ? AND user\_id = ?");

deleteReservation.setInt(1, bookId);

deleteReservation.setInt(2, userId);

deleteReservation.executeUpdate();

conn.commit();

// Show success message with fine

String message = "Book returned successfully!";

if (fine > 0) {

message += String.format("\nFine amount: ₹%.2f", fine);

} else {

message += "\nNo fine charged.";

}

JOptionPane.showMessageDialog(this, message);

// Remove row from table

model.removeRow(selectedRow);

// Refresh other views

loadUserBooks();

loadAvailableBooks();

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error returning book: " + e.getMessage());

}

}

}

// ---------- ADMIN FRAME ----------

static class AdminFrame extends JFrame {

JTable dataTable;

DefaultTableModel tableModel;

JScrollPane scrollPane;

AdminFrame() {

setTitle("Admin Panel");

setSize(800, 600);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

// Create buttons panel

JPanel buttonPanel = new JPanel();

JButton showUsersBtn = new JButton("Show Users");

JButton showReservationsBtn = new JButton("Show Reservations");

JButton showBooksBtn = new JButton("Show Available Books");

JButton logoutBtn = new JButton("Logout");

buttonPanel.add(showUsersBtn);

buttonPanel.add(showReservationsBtn);

buttonPanel.add(showBooksBtn);

buttonPanel.add(logoutBtn);

// Create table

tableModel = new DefaultTableModel();

dataTable = new JTable(tableModel);

scrollPane = new JScrollPane(dataTable);

// Add components to frame

add(buttonPanel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

// Add action listeners

showUsersBtn.addActionListener(e -> showUsers());

showReservationsBtn.addActionListener(e -> showReservations());

showBooksBtn.addActionListener(e -> showAvailableBooks());

logoutBtn.addActionListener(e -> {

dispose();

new LoginFrame();

});

setLocationRelativeTo(null);

setVisible(true);

}

void showUsers() {

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement stmt = conn.prepareStatement("SELECT user\_id, username FROM users1");

ResultSet rs = stmt.executeQuery();

// Clear existing table data

tableModel.setRowCount(0);

tableModel.setColumnCount(0);

// Add columns

ResultSetMetaData metaData = rs.getMetaData();

int columnCount = metaData.getColumnCount();

for (int i = 1; i <= columnCount; i++) {

tableModel.addColumn(metaData.getColumnName(i));

}

// Add rows

while (rs.next()) {

Object[] row = new Object[columnCount];

for (int i = 1; i <= columnCount; i++) {

row[i-1] = rs.getObject(i);

}

tableModel.addRow(row);

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading users data!");

}

}

void showReservations() {

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement stmt = conn.prepareStatement(

"SELECT r.reservation\_id, u.username, b.title, r.reserve\_days " +

"FROM reservations r " +

"JOIN users1 u ON r.user\_id = u.user\_id " +

"JOIN books b ON r.book\_id = b.book\_id");

ResultSet rs = stmt.executeQuery();

// Clear existing table data

tableModel.setRowCount(0);

tableModel.setColumnCount(0);

// Add columns

ResultSetMetaData metaData = rs.getMetaData();

int columnCount = metaData.getColumnCount();

for (int i = 1; i <= columnCount; i++) {

tableModel.addColumn(metaData.getColumnName(i));

}

// Add rows

while (rs.next()) {

Object[] row = new Object[columnCount];

for (int i = 1; i <= columnCount; i++) {

row[i-1] = rs.getObject(i);

}

tableModel.addRow(row);

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading reservations data!");

}

}

void showAvailableBooks() {

try (Connection conn = BookReservationSystem.connect()) {

PreparedStatement stmt = conn.prepareStatement(

"SELECT book\_id, title, author, is\_checked\_out FROM books");

ResultSet rs = stmt.executeQuery();

// Clear existing table data

tableModel.setRowCount(0);

tableModel.setColumnCount(0);

// Add columns

ResultSetMetaData metaData = rs.getMetaData();

int columnCount = metaData.getColumnCount();

for (int i = 1; i <= columnCount; i++) {

tableModel.addColumn(metaData.getColumnName(i));

}

// Add rows

while (rs.next()) {

Object[] row = new Object[columnCount];

for (int i = 1; i <= columnCount; i++) {

row[i-1] = rs.getObject(i);

}

tableModel.addRow(row);

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading books data!");

}

}

}

}