**ITMD 411 – Java Final Project Files**

**By: Sufyan Khan and Mbargou Gueye**

**12/09/2023**

**Extra credit GitHub Repo**

**GitHub link:** [**https://github.com/ITMD411-SkhanMgueye/ITMD411-FinalProject**](https://github.com/ITMD411-SkhanMgueye/ITMD411-FinalProject)

**Source code:**

**Dao.java**

/\*

\* Author: Sufyan Khan and Mbargou Gueye

\* Date: 12/02/2023

\* File: Dao.java

\* Description: The IIT Help Desk application's database operations are managed by the {Dao} class.

\*/

**package** javaapplication1;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Calendar;

**import** java.util.List;

**public** **class** Dao {

**static** Connection *connect* = **null**;

Statement statement = **null**;

// JDBC URL, database credentials

**static** **final** String ***DB\_URL*** = "jdbc:mysql://www.papademas.net:3307/tickets?autoReconnect=true&useSSL=false";

**static** **final** String ***USER*** = "fp411", ***PASS*** = "411";

// Establishes the database connection

**public** Connection getConnection() {

// Try to establish a connection to the database

**try** {

*connect* = DriverManager.*getConnection*(***DB\_URL***, ***USER***, ***PASS***);

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** *connect*;

}

// Creates required tables if they don't exist

**public** **void** createTables() {

// SQL statements to create tables

**final** String createTicketsTable = "CREATE TABLE IF NOT EXISTS SkhanMgueye\_tickets(ticket\_id INT AUTO\_INCREMENT PRIMARY KEY, ticket\_issuer VARCHAR(30), ticket\_description VARCHAR(200), opened DATETIME, ticket\_status VARCHAR(10))";

**final** String createUsersTable = "CREATE TABLE IF NOT EXISTS SkhanMgueye\_users(uid INT AUTO\_INCREMENT PRIMARY KEY, uname VARCHAR(30), upass VARCHAR(30), admin INT)";

**final** String createCommentsTable = "CREATE TABLE IF NOT EXISTS SkhanMgueye\_comments(comment\_id INT AUTO\_INCREMENT PRIMARY KEY, ticket\_id INT, comment\_text VARCHAR(200), FOREIGN KEY (ticket\_id) REFERENCES jpapa\_tickets(ticket\_id))";

**try** {

// Create tables in the database

statement = getConnection().createStatement();

statement.executeUpdate(createTicketsTable);

statement.executeUpdate(createUsersTable);

statement.executeUpdate(createCommentsTable);

System.***out***.println("Created tables in the given database...");

statement.close();

*connect*.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

addUsers();

}

// Adds a column 'ticket\_status' to the 'SkhanMgueye\_tickets' table

**public** **void** addColumnToTicketsTable() {

// Try adding a new column to the tickets table

**try** {

// Alter table to add a column

Connection connection = getConnection();

String alterTableQuery = "ALTER TABLE SkhanMgueye\_tickets ADD COLUMN ticket\_status VARCHAR(10)";

Statement statement = connection.createStatement();

statement.executeUpdate(alterTableQuery);

statement.close();

connection.close();

System.***out***.println("Column 'ticket\_status' added to SkhanMgueye\_tickets table successfully.");

} **catch** (SQLException e) {

System.***out***.println("Error adding column 'ticket\_status': " + e.getMessage());

}

}

// Adds user data from a CSV file to the 'SkhanMgueye\_users' table

**public** **void** addUsers() {

// Try adding users from a CSV file to the database

String sql;

Statement statement;

BufferedReader br;

List<List<String>> array = **new** ArrayList<>();

**try** {

// Read user data from CSV file

br = **new** BufferedReader(**new** FileReader(**new** File("./userlist.csv")));

String line;

**while** ((line = br.readLine()) != **null**) {

array.add(Arrays.*asList*(line.split(",")));

}

} **catch** (Exception e) {

System.***out***.println("There was a problem loading the file");

}

**try** {

// Insert user data into the database

statement = getConnection().createStatement();

**for** (List<String> rowData : array) {

sql = "INSERT INTO SkhanMgueye\_users(uname, upass, admin) VALUES('" + rowData.get(0) + "','" + rowData.get(1) + "','" + rowData.get(2) + "')";

statement.executeUpdate(sql);

}

System.***out***.println("Inserts completed in the given database...");

statement.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

// Inserts a new ticket record into the 'SkhanMgueye\_tickets' table

**public** **int** insertRecords(String ticketName, String ticketDesc) {

// Try inserting a new ticket record into the database

**int** id = 0;

**try** {

// Prepare and execute INSERT SQL query for a new ticket

String timeStamp = **new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.*getInstance*().getTime());

String query = "INSERT INTO SkhanMgueye\_tickets (ticket\_issuer, ticket\_description, opened, ticket\_status) VALUES (?, ?, ?, ?)";

PreparedStatement preparedStatement = getConnection().prepareStatement(query, Statement.***RETURN\_GENERATED\_KEYS***);

// Set parameters for the new ticket

preparedStatement.setString(1, ticketName);

preparedStatement.setString(2, ticketDesc);

preparedStatement.setString(3, timeStamp);

preparedStatement.setString(4, "OPEN");

// Execute the INSERT query and retrieve the generated ID

preparedStatement.executeUpdate();

ResultSet resultSet = preparedStatement.getGeneratedKeys();

**if** (resultSet.next()) {

id = resultSet.getInt(1);

}

preparedStatement.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** id;

}

// Retrieves all ticket records from the 'SkhanMgueye\_tickets' table

**public** ResultSet readRecords() {

// Try fetching all ticket records from the database

ResultSet results = **null**;

**try** {

statement = *connect*.createStatement();

results = statement.executeQuery("SELECT \* FROM SkhanMgueye\_tickets");

} **catch** (SQLException e1) {

e1.printStackTrace();

}

**return** results;

}

// Selects ticket records by ID from the 'SkhanMgueye\_tickets' table

**public** ResultSet selectRecords(**int** tid) {

// Try selecting ticket records by ID from the database

ResultSet results = **null**;

**try** {

statement = *connect*.createStatement();

results = statement.executeQuery("SELECT \* FROM SkhanMgueye\_tickets WHERE ticket\_id = " + tid);

} **catch** (SQLException e1) {

e1.printStackTrace();

}

**return** results;

}

// Updates ticket details in the 'SkhanMgueye\_tickets' table

**public** **void** updateRecords(**int** ticketId, String newDescription, String status) {

// Try updating ticket records in the database

**try** {

// Prepare SQL statements for updating ticket details

PreparedStatement ps = *connect*.prepareStatement("SELECT ticket\_description FROM SkhanMgueye\_tickets WHERE ticket\_id = ?");

ps.setInt(1, ticketId);

ResultSet rs = ps.executeQuery();

String currentDescription = "";

**if** (rs.next()) {

currentDescription = rs.getString("ticket\_description");

}

rs.close();

ps.close();

// Append the new description to the existing one

String updatedDescription = currentDescription + "\nUpdate: " + newDescription;

PreparedStatement updatePs = *connect*.prepareStatement("UPDATE SkhanMgueye\_tickets SET ticket\_description = ?, ticket\_status = ? WHERE ticket\_id = ?");

updatePs.setString(1, updatedDescription);

updatePs.setString(2, status);

updatePs.setInt(3, ticketId);

updatePs.executeUpdate();

updatePs.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

// Deletes a ticket record by ID from the 'SkhanMgueye\_tickets' table

**public** **int** deleteRecords(**int** tid) {

// Try deleting a ticket record by ID from the database

**try** {

statement = *connect*.createStatement();

String sql = "DELETE FROM SkhanMgueye\_tickets WHERE ticket\_id = " + tid;

statement.executeUpdate(sql);

System.***out***.println("Ticket ID : " + tid + " deleted successfully.");

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** tid;

}

// Checks if a ticket with the given ID exists in the database

**public** **boolean** checkTicketExists(**int** ticketId) {

// Try checking if a ticket exists in the database

**boolean** exists = **false**;

**try** {

// Prepare a SQL statement to check for the existence of the ticket ID

PreparedStatement ps = *connect*.prepareStatement("SELECT \* FROM SkhanMgueye\_tickets WHERE ticket\_id = ?");

ps.setInt(1, ticketId);

ResultSet rs = ps.executeQuery();

exists = rs.next(); // If there is a next record, the ticket exists

rs.close();

ps.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** exists;

}

// Updates the status of a ticket with the given ID

**public** **boolean** updateTicketStatus(**int** ticketId, String newStatus) {

// Try updating the status of a ticket in the database

**boolean** updated = **false**;

**try** {

// Prepare a SQL statement to update the ticket status

PreparedStatement ps = *connect*.prepareStatement("UPDATE SkhanMgueye\_tickets SET ticket\_status = ? WHERE ticket\_id = ?");

ps.setString(1, newStatus);

ps.setInt(2, ticketId);

**int** rowsAffected = ps.executeUpdate();

updated = rowsAffected > 0; // If rows were affected, the update was successful

ps.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** updated;

}

**public** **void** updateTicket(**int** ticketIdToUpdate, String newTicketDescription) {

// **TODO**: Implement updateTicket method

}

**public** **void** deleteTicket(**int** ticketIdToDelete) {

// **TODO**: Implement deleteTicket method

}

**public** **boolean** updateTicketDescription(**int** ticketId, String newDescription) {

**boolean** descriptionUpdated = **false**;

**try** {

// Prepare a SQL statement to update the ticket description

PreparedStatement ps = *connect*.prepareStatement("UPDATE SkhanMgueye\_tickets SET ticket\_description = ? WHERE ticket\_id = ?");

ps.setString(1, newDescription);

ps.setInt(2, ticketId);

// Execute the update query

**int** rowsAffected = ps.executeUpdate();

**if** (rowsAffected > 0) {

descriptionUpdated = **true**;

}

ps.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** descriptionUpdated;

}

}

**Tickets.java**

/\*

\* Author: Sufyan Khan and Mbargou Gueye

\* Date: 12/06/2023

\* File: Tickets.java

\*/

**package** javaapplication1;

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.WindowEvent;

**import** java.sql.SQLException;

**public** **class** Tickets **extends** JFrame **implements** ActionListener {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

Dao dao = **new** Dao();

**boolean** chkIfAdmin = **false**;

**private** JMenu mnuFile = **new** JMenu("File");

**private** JMenu mnuAdmin = **new** JMenu("Admin");

**private** JMenu mnuTickets = **new** JMenu("Tickets");

JMenuItem mnuItemExit;

JMenuItem mnuItemRefresh;

JMenuItem mnuItemDelete;

JMenuItem mnuItemOpenTicket;

JMenuItem mnuItemSelectTicket;

JMenuItem mnuItemAddTicket;

JMenuItem mnuItemCloseTicket;

JMenuItem mnuItemViewTicketStatus;

JMenuItem mnuItemUpdateDescription;

**public** Tickets(**boolean** isAdmin) {

chkIfAdmin = isAdmin;

createMenu();

prepareGUI();

**if** (chkIfAdmin) {

openTickets();

}

}

**private** **void** createMenu() {

mnuItemExit = **new** JMenuItem("Exit");

mnuFile.add(mnuItemExit);

mnuItemRefresh = **new** JMenuItem("Refresh");

mnuFile.add(mnuItemRefresh);

**if** (chkIfAdmin) {

mnuItemDelete = **new** JMenuItem("Delete Ticket");

mnuAdmin.add(mnuItemDelete);

mnuItemAddTicket = **new** JMenuItem("Add Ticket");

mnuAdmin.add(mnuItemAddTicket);

mnuItemCloseTicket = **new** JMenuItem("Close Ticket");

mnuAdmin.add(mnuItemCloseTicket);

mnuItemCloseTicket.addActionListener(**this**);

mnuItemViewTicketStatus = **new** JMenuItem("View Ticket Status");

mnuAdmin.add(mnuItemViewTicketStatus);

mnuItemViewTicketStatus.addActionListener(**this**);

mnuItemUpdateDescription = **new** JMenuItem("Update Ticket Description");

mnuAdmin.add(mnuItemUpdateDescription);

mnuItemUpdateDescription.addActionListener(**this**);

}

mnuItemOpenTicket = **new** JMenuItem("Open Ticket");

mnuTickets.add(mnuItemOpenTicket);

mnuItemSelectTicket = **new** JMenuItem("Select Ticket");

mnuTickets.add(mnuItemSelectTicket);

mnuItemExit.addActionListener(**this**);

mnuItemRefresh.addActionListener(**this**);

**if** (chkIfAdmin) {

mnuItemDelete.addActionListener(**this**);

mnuItemAddTicket.addActionListener(**this**);

}

mnuItemOpenTicket.addActionListener(**this**);

mnuItemSelectTicket.addActionListener(**this**);

}

**private** **void** updateTicketDescription() {

String ticketIdInput = JOptionPane.*showInputDialog*(**null**, "Enter the ticket ID to update the description");

**if** (ticketIdInput == **null** || ticketIdInput.isEmpty()) {

JOptionPane.*showMessageDialog*(**null**, "Ticket ID is empty.");

System.***out***.println("Ticket ID is empty.");

**return**;

}

**int** ticketId;

**try** {

ticketId = Integer.*parseInt*(ticketIdInput);

} **catch** (NumberFormatException e) {

JOptionPane.*showMessageDialog*(**null**, "Invalid ticket ID format.");

System.***out***.println("Invalid ticket ID format.");

**return**;

}

**boolean** ticketExists = dao.checkTicketExists(ticketId);

**if** (!ticketExists) {

System.***out***.println("Ticket ID: " + ticketId + " does not exist.");

JOptionPane.*showMessageDialog*(**null**, "Ticket ID: " + ticketId + " does not exist.");

**return**;

}

String newDescription = JOptionPane.*showInputDialog*(**null**, "Enter the new ticket description");

**if** (newDescription == **null** || newDescription.isEmpty()) {

JOptionPane.*showMessageDialog*(**null**, "New ticket description is empty.");

System.***out***.println("New ticket description is empty.");

**return**;

}

**boolean** descriptionUpdated = dao.updateTicketDescription(ticketId, newDescription);

**if** (descriptionUpdated) {

System.***out***.println("Ticket ID : " + ticketId + " description updated successfully.");

JOptionPane.*showMessageDialog*(**null**, "Ticket ID: " + ticketId + " description updated successfully");

refreshTicketView();

} **else** {

System.***out***.println("Failed to update ticket description.");

JOptionPane.*showMessageDialog*(**null**, "Failed to update ticket description.");

}

}

**private** **void** prepareGUI() {

JMenuBar bar = **new** JMenuBar();

bar.add(mnuFile);

**if** (chkIfAdmin) {

bar.add(mnuAdmin);

}

bar.add(mnuTickets);

setJMenuBar(bar);

addWindowListener(**new** java.awt.event.WindowAdapter() {

**public** **void** windowClosing(WindowEvent wE) {

System.*exit*(0);

}

});

setSize(400, 400);

getContentPane().setBackground(Color.***GREEN***);

setLocationRelativeTo(**null**);

setVisible(**true**);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

**if** (e.getSource() == mnuItemExit) {

System.*exit*(0);

} **else** **if** (e.getSource() == mnuItemRefresh) {

refreshTicketView();

} **else** **if** (e.getSource() == mnuItemSelectTicket) {

selectTicket();

} **else** **if** (e.getSource() == mnuItemAddTicket) {

addTicket();

} **else** **if** (e.getSource() == mnuItemDelete) {

deleteTicket();

} **else** **if** (e.getSource() == mnuItemOpenTicket) {

openTickets();

} **else** **if** (e.getSource() == mnuItemCloseTicket && chkIfAdmin) {

closeTicketStatus();

} **else** **if** (e.getSource() == mnuItemViewTicketStatus && chkIfAdmin) {

openTicketStatus();

} **else** **if** (e.getSource() == mnuItemUpdateDescription && chkIfAdmin) {

updateTicketDescription();

}

}

**private** **void** refreshTicketView() {

**try** {

JTable jt = **new** JTable(ticketsJTable.*buildTableModel*(dao.readRecords()));

jt.setBounds(30, 40, 200, 400);

jt.setBackground(Color.***green***);

jt.setForeground(Color.***black***);

jt.getTableHeader().setBackground(Color.***BLACK***);

jt.getTableHeader().setForeground(Color.***white***);

JScrollPane sp = **new** JScrollPane(jt);

getContentPane().removeAll();

add(sp);

revalidate();

repaint();

} **catch** (SQLException e1) {

System.***out***.println("Ticket view refresh failed.");

e1.printStackTrace();

}

}

**private** **void** selectTicket() {

JOptionPane.*showMessageDialog*(**null**, "Ticket selected!");

// You can implement the functionality for selecting a ticket here

}

**private** **void** addTicket() {

String ticketName = JOptionPane.*showInputDialog*(**null**, "Enter your name");

String ticketDesc = JOptionPane.*showInputDialog*(**null**, "Enter a ticket description");

**if** (ticketName == **null** || (ticketName != **null** && ("".equals(ticketName))) ||

ticketDesc == **null** || (ticketDesc != **null** && ("".equals(ticketDesc)))) {

JOptionPane.*showMessageDialog*(**null**, "Ticket creation failed: empty name / description.");

System.***out***.println("Ticket creation failed: empty name / description.");

} **else** {

**int** id = dao.insertRecords(ticketName, ticketDesc);

**if** (id != 0) {

System.***out***.println("Ticket ID : " + id + " created successfully.");

JOptionPane.*showMessageDialog*(**null**, "Ticket id: " + id + " created");

**try** {

JTable jt = **new** JTable(ticketsJTable.*buildTableModel*(dao.readRecords()));

jt.setBounds(30, 40, 200, 400);

jt.setBackground(Color.***green***);

jt.setForeground(Color.***white***);

jt.getTableHeader().setBackground(Color.***BLACK***);

jt.getTableHeader().setForeground(Color.***white***);

JScrollPane sp = **new** JScrollPane(jt);

getContentPane().removeAll();

add(sp);

revalidate();

repaint();

} **catch** (SQLException e1) {

System.***out***.println("Ticket view refresh failed.");

e1.printStackTrace();

}

} **else** {

System.***out***.println("Ticket creation failed.");

}

}

}

**private** **void** deleteTicket() {

String ticketId = JOptionPane.*showInputDialog*(**null**, "Enter the ticket id to delete the ticket");

**if** (ticketId == **null** || (ticketId != **null** && ("".equals(ticketId)))) {

JOptionPane.*showMessageDialog*(**null**, "Ticket deletion failed: empty tid.");

System.***out***.println("Ticket deletion failed: empty tid.");

} **else** {

**int** tid = Integer.*parseInt*(ticketId);

**int** reply = JOptionPane.*showConfirmDialog*(**null**, "Are you sure you want to delete ticket " + tid + "?", "Warning!", JOptionPane.***YES\_NO\_OPTION***);

**if** (reply == JOptionPane.***YES\_OPTION***) {

**int** id = dao.deleteRecords(tid);

**if** (id != 0) {

System.***out***.println("Ticket ID : " + id + " deleted successfully.");

JOptionPane.*showMessageDialog*(**null**, "Ticket id: " + id + " deleted");

**try** {

JTable jt = **new** JTable(ticketsJTable.*buildTableModel*(dao.readRecords()));

jt.setBounds(30, 40, 200, 400);

jt.setBackground(Color.***green***);

jt.setForeground(Color.***black***);

jt.getTableHeader().setBackground(Color.***BLACK***);

jt.getTableHeader().setForeground(Color.***white***);

JScrollPane sp = **new** JScrollPane(jt);

getContentPane().removeAll();

add(sp);

revalidate();

repaint();

} **catch** (SQLException e1) {

System.***out***.println("Ticket view refresh failed.");

e1.printStackTrace();

}

} **else** {

System.***out***.println("Ticket cannot be deleted!!!");

}

} **else** {

JOptionPane.*showMessageDialog*(**null**, "Ticket " + tid + " was not deleted.");

}

}

}

**private** **void** closeTicketStatus() {

String ticketIdInput = JOptionPane.*showInputDialog*(**null**, "Enter the ticket ID to close the ticket");

**if** (ticketIdInput == **null** || (ticketIdInput != **null** && ("".equals(ticketIdInput)))) {

JOptionPane.*showMessageDialog*(**null**, "Ticket ID is empty.");

System.***out***.println("Ticket ID is empty.");

} **else** {

**int** ticketId = Integer.*parseInt*(ticketIdInput);

**boolean** ticketExists = dao.checkTicketExists(ticketId);

**if** (ticketExists) {

**boolean** statusUpdated = dao.updateTicketStatus(ticketId, "CLOSE");

**if** (statusUpdated) {

System.***out***.println("Ticket ID : " + ticketId + " status updated successfully to 'CLOSE'.");

JOptionPane.*showMessageDialog*(**null**, "Ticket ID: " + ticketId + " status updated to 'CLOSE'");

refreshTicketView();

} **else** {

System.***out***.println("Failed to update ticket status.");

JOptionPane.*showMessageDialog*(**null**, "Failed to update ticket status.");

}

} **else** {

System.***out***.println("Ticket ID: " + ticketId + " does not exist.");

JOptionPane.*showMessageDialog*(**null**, "Ticket ID: " + ticketId + " does not exist.");

}

}

}

**private** **void** openTicketStatus() {

String ticketIdInput = JOptionPane.*showInputDialog*(**null**, "Enter the ticket ID to open the ticket");

**if** (ticketIdInput == **null** || (ticketIdInput != **null** && ("".equals(ticketIdInput)))) {

JOptionPane.*showMessageDialog*(**null**, "Ticket ID is empty.");

System.***out***.println("Ticket ID is empty.");

} **else** {

**int** ticketId = Integer.*parseInt*(ticketIdInput);

**boolean** ticketExists = dao.checkTicketExists(ticketId);

**if** (ticketExists) {

**boolean** statusUpdated = dao.updateTicketStatus(ticketId, "OPEN");

**if** (statusUpdated) {

System.***out***.println("Ticket ID : " + ticketId + " status updated successfully to 'OPEN'.");

JOptionPane.*showMessageDialog*(**null**, "Ticket ID: " + ticketId + " status updated to 'OPEN'");

refreshTicketView();

} **else** {

System.***out***.println("Failed to update ticket status.");

JOptionPane.*showMessageDialog*(**null**, "Failed to update ticket status.");

}

} **else** {

System.***out***.println("Ticket ID: " + ticketId + " does not exist.");

JOptionPane.*showMessageDialog*(**null**, "Ticket ID: " + ticketId + " does not exist.");

}

}

}

**private** **void** openTickets() {

**try** {

JTable jt = **new** JTable(ticketsJTable.*buildTableModel*(dao.readRecords()));

jt.setBounds(30, 40, 200, 400);

jt.setBackground(Color.***green***);

jt.setForeground(Color.***black***);

jt.getTableHeader().setBackground(Color.***BLACK***);

jt.getTableHeader().setForeground(Color.***white***);

JScrollPane sp = **new** JScrollPane(jt);

getContentPane().removeAll();

add(sp);

revalidate();

repaint();

} **catch** (SQLException e1) {

System.***out***.println("Ticket view failed for admin.");

e1.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

// Entry point - You may add relevant code if needed

}

}

**Login.java**

/\*

\* Author: Sufyan Khan and Mbargou Gueye

\* Date: 12/01/2023

\* File: Login.java

\* Description: This Java application provides a GUI-based login interface using Swing components.

\* It limits the number of login tries, validates user credentials against a database table, and, after a successful login, grants access to a help desk system.

\*/

**package** javaapplication1;

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

@SuppressWarnings("serial")

**public** **class** Login **extends** JFrame {

Dao conn;

**public** Login() {

**super**("IIT HELP DESK LOGIN");

// Initialize the database connection

conn = **new** Dao();

conn.createTables(); // Create necessary tables if they don't exist in the database

// Set the background color

getContentPane().setBackground(Color.***green***);

// Set the layout and size

setLayout(**new** BorderLayout()); // Use BorderLayout for organized positioning

setSize(400, 210); // Initial size

setLocationRelativeTo(**null**); // Set window location to the center of the screen

// Create title panel

JPanel titlePanel = **new** JPanel(**new** FlowLayout(FlowLayout.***CENTER***));

titlePanel.setBackground(Color.***green***);

JLabel titleLabel = **new** JLabel("IIT Help Desk Login", JLabel.***CENTER***);

titleLabel.setFont(**new** Font("Arial", Font.***BOLD***, 20)); // Set title font and size

titleLabel.setForeground(Color.***black***);

titlePanel.add(titleLabel);

// Create input panel for username and password fields

JPanel inputPanel = **new** JPanel(**new** GridLayout(3, 2, 5, 5)); // GridLayout for structured arrangement

inputPanel.setBackground(Color.***green***);

JLabel lblUsername = **new** JLabel("Username:", JLabel.***LEFT***);

lblUsername.setFont(lblUsername.getFont().deriveFont(Font.***BOLD***)); // Set font style to bold

JTextField txtUname = **new** JTextField(10); // Username text field

JLabel lblPassword = **new** JLabel("Password:", JLabel.***LEFT***);

lblPassword.setFont(lblPassword.getFont().deriveFont(Font.***BOLD***)); // Set font style to bold

JPasswordField txtPassword = **new** JPasswordField(); // Password field

JLabel lblStatus = **new** JLabel(" ", JLabel.***CENTER***); // Placeholder for status display

lblStatus.setForeground(Color.***red***); // Set text color to red for status label

inputPanel.add(lblUsername);

inputPanel.add(txtUname);

inputPanel.add(lblPassword);

inputPanel.add(txtPassword);

inputPanel.add(lblStatus); // Display status

// Create button panel for Login and Exit buttons

JPanel buttonPanel = **new** JPanel(**new** FlowLayout(FlowLayout.***CENTER***));

buttonPanel.setBackground(Color.***green***);

JButton btnExit = **new** JButton("Exit");

btnExit.setBackground(Color.***black***); // Set button background color

btnExit.setForeground(Color.***white***); // Set button text color

JButton btn = **new** JButton("Log In");

btn.setBackground(Color.***black***); // Set button background color

btn.setForeground(Color.***white***); // Set button text color

buttonPanel.add(btnExit);

buttonPanel.add(btn);

// Add components to the frame

add(titlePanel, BorderLayout.***NORTH***); // Add title panel to the top

add(inputPanel, BorderLayout.***CENTER***); // Add input fields panel in the center

add(buttonPanel, BorderLayout.***SOUTH***); // Add button panel at the bottom

// Button click actions

btn.addActionListener(**new** ActionListener() {

**int** count = 0; // Variable to track login attempts

@Override

**public** **void** actionPerformed(ActionEvent e) {

**boolean** isAdmin = **false**;

count = count + 1; // Increment attempt count on button click

String query = "SELECT \* FROM SkhanMgueye\_users WHERE uname = ? AND upass = ?";

**try** (PreparedStatement stmt = conn.getConnection().prepareStatement(query)) {

stmt.setString(1, txtUname.getText()); // Get username entered by user

stmt.setString(2, String.*valueOf*(txtPassword.getPassword())); // Get password entered by user

ResultSet rs = stmt.executeQuery(); // Execute the query

**if** (rs.next()) {

isAdmin = (rs.getInt("admin") == 1); // Check if user is an admin

System.***out***.println("Login success.");

**new** Tickets(isAdmin); // Create Tickets object with isAdmin flag

setVisible(**false**); // Hide the login window

dispose(); // Dispose of the login window resources

} **else** {

lblStatus.setText("Try again! " + (3 - count) + " / 3 attempts left"); // Display status message

System.***out***.println("Try again! " + (3 - count) + " / 3 attempts left");

}

} **catch** (SQLException ex) {

ex.printStackTrace(); // Print SQL exception stack trace

}

}

});

btnExit.addActionListener(e -> System.*exit*(0)); // Exit the application when Exit button is clicked

// Set visibility of the login window

setVisible(**true**);

}

**public** **static** **void** main(String[] args) {

**new** Login(); // Create an instance of the Login class to start the application

}

}

**TicketJTable.java**

/\*

\* Author: Sufyan Khan and Mbargou Gueye

\* Date: 12/05/2023

\* File: ticketsJTable.java

\*/

**package** javaapplication1;

**import** java.sql.ResultSet;

**import** java.sql.ResultSetMetaData;

**import** java.sql.SQLException;

**import** java.util.Vector;

**import** javax.swing.table.DefaultTableModel;

// Class to handle building a table model for JTable from a ResultSet

**public** **class** ticketsJTable {

// Method to create a DefaultTableModel from a ResultSet

**public** **static** DefaultTableModel buildTableModel(ResultSet rs) **throws** SQLException {

ResultSetMetaData metaData = rs.getMetaData();

// Store column names in a Vector

Vector<String> columnNames = **new** Vector<String>();

**int** columnCount = metaData.getColumnCount();

**for** (**int** column = 1; column <= columnCount; column++) {

columnNames.add(metaData.getColumnName(column));

}

// Store table data in a Vector of Vectors

Vector<Vector<Object>> data = **new** Vector<Vector<Object>>();

**while** (rs.next()) {

Vector<Object> vector = **new** Vector<Object>();

**for** (**int** columnIndex = 1; columnIndex <= columnCount; columnIndex++) {

vector.add(rs.getObject(columnIndex));

}

data.add(vector);

}

// Return the data and column names as a DefaultTableModel for JTable

**return** **new** DefaultTableModel(data, columnNames);

}

}

**TicketComment.java**

/\*

\* Author: Sufyan Khan and Mbargou Gueye

\* Date: 12/05/2023

\* File: TicketComment.java

\*/

**package** javaapplication1;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**public** **class** TicketComment {

Dao dao;

// Constructor initializing Dao object

**public** TicketComment() {

dao = **new** Dao(); // Initialize Dao object

}

// Method to insert a comment for a specific ticket

**public** **void** insertComment(**int** ticketId, String commentText) {

**try** {

String sql = "INSERT INTO SkhanMgueye\_comments(ticket\_id, comment\_text) VALUES (?, ?)";

PreparedStatement pstmt = dao.getConnection().prepareStatement(sql);

pstmt.setInt(1, ticketId);

pstmt.setString(2, commentText);

pstmt.executeUpdate();

System.***out***.println("Comment added successfully.");

} **catch** (SQLException e) {

e.printStackTrace();

}

}

// Method to retrieve comments for a specific ticket

**public** ResultSet getCommentsForTicket(**int** ticketId) {

ResultSet resultSet = **null**;

**try** {

String query = "SELECT \* FROM SkhanMgueye\_comments WHERE ticket\_id = ?";

PreparedStatement pstmt = dao.getConnection().prepareStatement(query);

pstmt.setInt(1, ticketId);

resultSet = pstmt.executeQuery();

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** resultSet;

}

// Method to update a comment based on its ID

**public** **void** updateComment(**int** commentId, String newCommentText) {

**try** {

String sql = "UPDATE SkhanMgueye\_comments SET comment\_text = ? WHERE comment\_id = ?";

PreparedStatement pstmt = dao.getConnection().prepareStatement(sql);

pstmt.setString(1, newCommentText);

pstmt.setInt(2, commentId);

pstmt.executeUpdate();

System.***out***.println("Comment updated successfully.");

} **catch** (SQLException e) {

e.printStackTrace();

}

}

// Method to delete a comment based on its ID

**public** **void** deleteComment(**int** commentId) {

**try** {

String sql = "DELETE FROM SkhanMgueye\_comments WHERE comment\_id = ?";

PreparedStatement pstmt = dao.getConnection().prepareStatement(sql);

pstmt.setInt(1, commentId);

pstmt.executeUpdate();

System.***out***.println("Comment deleted successfully.");

} **catch** (SQLException e) {

e.printStackTrace();

}

}

// Method to retrieve all comments

**public** ResultSet getAllComments() {

ResultSet resultSet = **null**;

**try** {

String query = "SELECT \* FROM SkhanMgueye\_comments";

PreparedStatement pstmt = dao.getConnection().prepareStatement(query);

resultSet = pstmt.executeQuery();

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** resultSet;

}

// Method to count the total number of comments for a specific ticket

**public** **int** countCommentsForTicket(**int** ticketId) {

**int** count = 0;

**try** {

String query = "SELECT COUNT(\*) AS total FROM SkhanMgueye\_comments WHERE ticket\_id = ?";

PreparedStatement pstmt = dao.getConnection().prepareStatement(query);

pstmt.setInt(1, ticketId);

ResultSet resultSet = pstmt.executeQuery();

**if** (resultSet.next()) {

count = resultSet.getInt("total");

}

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** count;

}

}