**PROTOTYPE DOCUMENTATION**

**Back-end**

-- Table: Attachment

CREATE TABLE IF NOT EXISTS Attachment (

AttachementID INT NOT NULL AUTO\_INCREMENT,

AttachementType VARCHAR(45),

PRIMARY KEY (AttachementID)

);

-- Table: Device

CREATE TABLE IF NOT EXISTS Device (

DeviceID INT NOT NULL AUTO\_INCREMENT,

DeviceName VARCHAR(45),

Paltform\_PaltformID INT NOT NULL,

PRIMARY KEY (DeviceID),

FOREIGN KEY (Paltform\_PaltformID) REFERENCES Paltform (PaltformID)

);

-- Table: Folder

CREATE TABLE IF NOT EXISTS Folder (

FolderID INT NOT NULL AUTO\_INCREMENT,

FolderName VARCHAR(45) NOT NULL,

DateCreated DATE NOT NULL,

DateModified DATE NOT NULL,

User\_UserID INT NOT NULL,

PRIMARY KEY (FolderID),

FOREIGN KEY (User\_UserID) REFERENCES User (UserID)

);

-- Table: NoteAttachment

CREATE TABLE IF NOT EXISTS NoteAttachment (

NoteAttachmentID INT NOT NULL AUTO\_INCREMENT,

Notes\_NoteID INT NOT NULL,

Notes\_Folder\_FolderID INT NOT NULL,

Attachment\_AttachementID INT NOT NULL,

PRIMARY KEY (NoteAttachmentID),

FOREIGN KEY (Notes\_NoteID, Notes\_Folder\_FolderID) REFERENCES Notes (NoteID, Folder\_FolderID),

FOREIGN KEY (Attachment\_AttachementID) REFERENCES Attachment (AttachementID)

);

-- Table: NoteFormatting

CREATE TABLE IF NOT EXISTS NoteFormatting (

formatingID INT NOT NULL AUTO\_INCREMENT,

StyleName VARCHAR(45),

PRIMARY KEY (formatingID)

);

-- Table: NoteTags

CREATE TABLE IF NOT EXISTS NoteTags (

NoteTagID INT NOT NULL AUTO\_INCREMENT,

Tags\_TagID INT NOT NULL,

Notes\_NoteID INT NOT NULL,

Notes\_Folder\_FolderID INT NOT NULL,

PRIMARY KEY (NoteTagID),

FOREIGN KEY (Tags\_TagID) REFERENCES Tags (TagID),

FOREIGN KEY (Notes\_NoteID, Notes\_Folder\_FolderID) REFERENCES Notes (NoteID, Folder\_FolderID)

);

-- Table: Notes

CREATE TABLE IF NOT EXISTS Notes (

NoteID INT NOT NULL AUTO\_INCREMENT,

Title VARCHAR(45) NOT NULL,

Content VARCHAR(100),

CreationDate DATE,

LastModifiedDate DATE NOT NULL,

isPublic TINYINT NOT NULL,

Folder\_FolderID INT NOT NULL,

NoteFormatting\_formatingID INT NOT NULL,

PRIMARY KEY (NoteID),

FOREIGN KEY (Folder\_FolderID) REFERENCES Folder (FolderID),

FOREIGN KEY (NoteFormatting\_formatingID) REFERENCES NoteFormatting (formatingID)

);

-- Table: Paltform

CREATE TABLE IF NOT EXISTS Paltform (

PaltformID INT NOT NULL AUTO\_INCREMENT,

PaltformName VARCHAR(45),

PlatformType VARCHAR(45),

PRIMARY KEY (PaltformID)

);

-- Table: SearchHistory

CREATE TABLE IF NOT EXISTS SearchHistory (

SearchID INT NOT NULL AUTO\_INCREMENT,

SearchQuery VARCHAR(100),

SearchTimestamp DATETIME,

User\_UserID INT NOT NULL,

PRIMARY KEY (SearchID),

FOREIGN KEY (User\_UserID) REFERENCES User (UserID)

);

-- Table: Tags

CREATE TABLE IF NOT EXISTS Tags (

TagID INT NOT NULL AUTO\_INCREMENT,

TagName VARCHAR(45),

PRIMARY KEY (TagID)

);

-- Table: User

CREATE TABLE IF NOT EXISTS User (

UserID INT NOT NULL AUTO\_INCREMENT,

Username VARCHAR(45) NOT NULL,

Email VARCHAR(45) NOT NULL,

Password VARCHAR(45) NOT NULL,

PRIMARY KEY (UserID)

);

**Middle-tier**

**Procedures**

CREATE DEFINER=`root`@`localhost` PROCEDURE `createNewNote`(IN p\_username VARCHAR(45), IN p\_selectedFolder VARCHAR(45), IN p\_newNoteTitle VARCHAR(45))

BEGIN

DECLARE v\_folderId INT;

SELECT FolderID INTO v\_folderId FROM Folder WHERE FolderName = p\_selectedFolder;

IF v\_folderId IS NOT NULL THEN

INSERT INTO Notes (Title, Content, CreationDate, LastModifiedDate, isPublic, Folder\_FolderID, NoteFormatting\_formatingID) VALUES (p\_newNoteTitle, '', CURDATE(), CURDATE(), 1, v\_folderId, 1);

END IF;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `insertFolder`(IN p\_username VARCHAR(45), IN p\_folderName VARCHAR(45))

BEGIN

DECLARE v\_userId INT;

SELECT UserID INTO v\_userId FROM User WHERE Username = p\_username;

IF v\_userId IS NOT NULL THEN

INSERT INTO Folder (FolderName, DateCreacted, DateModified, User\_UserID) VALUES (p\_folderName, CURDATE(), CURDATE(), v\_userId);

END IF;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `InsertUser`(IN p\_username VARCHAR(45), IN p\_email VARCHAR(45), IN p\_password VARCHAR(45))

BEGIN

INSERT INTO User (Username, Email, Password) VALUES (p\_username, p\_email, p\_password);

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `isValidLogin`(IN p\_username VARCHAR(45), IN p\_password VARCHAR(45))

BEGIN

SELECT EXISTS(SELECT 1 FROM User WHERE Username = p\_username AND Password = p\_password) AS isValid;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `LoadNoteContent`(IN p\_username VARCHAR(255), IN p\_folderName VARCHAR(255), IN p\_noteTitle VARCHAR(255))

BEGIN

SELECT n.Content

FROM Notes n

JOIN Folder f ON n.Folder\_FolderID = f.FolderID

JOIN User u ON f.User\_UserID = u.UserID

WHERE f.FolderName = p\_folderName

AND n.Title = p\_noteTitle

AND u.Username = p\_username;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `populateFolderList`(IN p\_username VARCHAR(45))

BEGIN

SELECT CONCAT(FolderName, ' (Created: ', DateCreacted, ', Modified: ', DateModified, ')') AS folderInfo

FROM Folder f

JOIN User u ON f.User\_UserID = u.UserID

WHERE u.Username = p\_username;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `SaveNoteContent`(IN p\_username VARCHAR(255), IN p\_folderName VARCHAR(255), IN p\_noteTitle VARCHAR(255), IN p\_content TEXT)

BEGIN

UPDATE Notes n

JOIN Folder f ON n.Folder\_FolderID = f.FolderID

JOIN User u ON f.User\_UserID = u.UserID

SET n.Content = p\_content

WHERE f.FolderName = p\_folderName

AND n.Title = p\_noteTitle

AND u.Username = p\_username;

END

**Triggers**

CREATE TRIGGER after\_note\_insert

AFTER INSERT ON Notes

FOR EACH ROW

BEGIN

UPDATE Folder

SET DateModified = CURDATE()

WHERE FolderID = NEW.Folder\_FolderID;

END;

**Java Program**

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import java.sql.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

public class App {

    private static JFrame loginFrame;

    private static JFrame folderFrame;

    private static JFrame notesFrame;

    private static JTextField usernameField;

    private static JPasswordField passwordField;

    private static JLabel userInfoLabel;

    private static JList<String> folderList;

    private static DefaultListModel<String> noteListModel;

    public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> {

            createAndShowLoginGUI();

        });

    }

    private static void createAndShowLoginGUI() {

        loginFrame = new JFrame("Login Page");

        loginFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        loginFrame.setMinimumSize(new Dimension(500, 500));

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(4, 2));

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

        usernameField = new JTextField(20);

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

        passwordField = new JPasswordField(20);

        JButton loginButton = new JButton("Login");

        loginButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String username = usernameField.getText();

                char[] passwordChars = passwordField.getPassword();

                String password = new String(passwordChars);

                if (isValidLogin(username, password)) {

                    showFolders(username);

                } else {

                    JOptionPane.showMessageDialog(loginFrame, "Login failed. Please check your credentials.");

                }

            }

        });

        JButton signUpButton = new JButton("Sign Up");

        signUpButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                openSignUpWindow();

            }

        });

        panel.add(usernameLabel);

        panel.add(usernameField);

        panel.add(passwordLabel);

        panel.add(passwordField);

        panel.add(loginButton);

        panel.add(signUpButton);

        loginFrame.getContentPane().add(panel);

        loginFrame.pack();

        loginFrame.setVisible(true);

    }

    private static boolean isValidLogin(String username, String password) {

        String dbURL = "jdbc:mysql://localhost:3306/dpgprototype";

        try (Connection dbConnect = DriverManager.getConnection(dbURL, "root", "James!DPG123")) {

            String SQL = "SELECT \* FROM User WHERE Username = ? AND Password = ?";

            try (PreparedStatement preparedStatement = dbConnect.prepareStatement(SQL)) {

                preparedStatement.setString(1, username);

                preparedStatement.setString(2, password);

                try (ResultSet resultSet = preparedStatement.executeQuery()) {

                    return resultSet.next();

                }

            }

        } catch (SQLException ex) {

            Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);

            return false;

        }

    }

    private static void openSignUpWindow() {

        JFrame signUpFrame = new JFrame("Sign Up");

        signUpFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

        JPanel signUpPanel = new JPanel();

        signUpPanel.setLayout(new GridLayout(4, 2));

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

        JTextField newUsernameField = new JTextField(20);

        JLabel emailLabel = new JLabel("Email:");

        JTextField emailField = new JTextField(20);

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

        JPasswordField newPasswordField = new JPasswordField(20);

        JButton signUpButton = new JButton("Sign Up");

        signUpButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String newUsername = newUsernameField.getText();

                String email = emailField.getText();

                char[] newPasswordChars = newPasswordField.getPassword();

                String newPassword = new String(newPasswordChars);

                if (insertUser(newUsername, email, newPassword)) {

                    JOptionPane.showMessageDialog(signUpFrame, "Sign up successful. You can now log in.");

                    signUpFrame.dispose();

                } else {

                    JOptionPane.showMessageDialog(signUpFrame, "Sign up failed. Please try again.");

                }

            }

        });

        signUpPanel.add(usernameLabel);

        signUpPanel.add(newUsernameField);

        signUpPanel.add(emailLabel);

        signUpPanel.add(emailField);

        signUpPanel.add(passwordLabel);

        signUpPanel.add(newPasswordField);

        signUpPanel.add(signUpButton);

        signUpFrame.getContentPane().add(signUpPanel);

        signUpFrame.pack();

        signUpFrame.setVisible(true);

    }

    private static boolean insertUser(String username, String email, String password) {

        String dbURL = "jdbc:mysql://localhost:3306/dpgprototype";

        try (Connection dbConnect = DriverManager.getConnection(dbURL, "root", "James!DPG123")) {

            String SQL = "INSERT INTO User (Username, Email, Password) VALUES (?, ?, ?)";

            try (PreparedStatement preparedStatement = dbConnect.prepareStatement(SQL)) {

                preparedStatement.setString(1, username);

                preparedStatement.setString(2, email);

                preparedStatement.setString(3, password);

                int rowsAffected = preparedStatement.executeUpdate();

                return rowsAffected > 0;

            }

        } catch (SQLException ex) {

            Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);

            return false;

        }

    }

    private static void showFolders(String username) {

        folderFrame = new JFrame("User Folders");

        folderFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        folderFrame.setMinimumSize(new Dimension(500, 500));

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(3, 1));

        userInfoLabel = new JLabel("Folders for user: " + username);

        DefaultListModel<String> folderListModel = new DefaultListModel<>();

        folderList = new JList<>(folderListModel);

        JScrollPane scrollPane = new JScrollPane(folderList);

        populateFolderList(username, folderListModel);

        panel.add(userInfoLabel);

        panel.add(scrollPane);

        JButton createFolderButton = new JButton("Create Folder");

        createFolderButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String newFolderName = JOptionPane.showInputDialog(folderFrame, "Enter the name for the new folder:");

                if (newFolderName != null && !newFolderName.isEmpty()) {

                    if (insertFolder(username, newFolderName)) {

                        JOptionPane.showMessageDialog(folderFrame, "New folder created successfully.");

                        folderListModel.addElement(newFolderName);

                    } else {

                        JOptionPane.showMessageDialog(folderFrame, "Failed to create a new folder.");

                    }

                }

            }

        });

        panel.add(createFolderButton);

        folderFrame.getContentPane().add(panel);

        folderFrame.pack();

        folderFrame.setVisible(true);

        folderList.addListSelectionListener(new ListSelectionListener() {

            @Override

            public void valueChanged(ListSelectionEvent e) {

                if (!e.getValueIsAdjusting()) {

                    String selectedFolder = folderList.getSelectedValue();

                    if (selectedFolder != null) {

                        // Extract the folder name without the date

                        String folderName = selectedFolder.split(" ")[0];

                        showNotesForFolder(username, folderName);

                    }

                }

            }

        });

    }

    private static boolean insertFolder(String username, String folderName) {

        String dbURL = "jdbc:mysql://localhost:3306/dpgprototype";

        try (Connection dbConnect = DriverManager.getConnection(dbURL, "root", "James!DPG123")) {

            // Retrieve the user ID for the given username

            String getUserIdSQL = "SELECT UserID FROM User WHERE Username = ?";

            try (PreparedStatement getUserIdStatement = dbConnect.prepareStatement(getUserIdSQL)) {

                getUserIdStatement.setString(1, username);

                try (ResultSet userIdResult = getUserIdStatement.executeQuery()) {

                    if (userIdResult.next()) {

                        int userId = userIdResult.getInt("UserID");

                        // Insert the new folder with the correct user ID

                        String insertFolderSQL = "INSERT INTO Folder (FolderName, DateCreacted, DateModified, User\_UserID) VALUES (?, CURDATE(), CURDATE(), ?)";

                        try (PreparedStatement insertFolderStatement = dbConnect.prepareStatement(insertFolderSQL)) {

                            insertFolderStatement.setString(1, folderName);

                            insertFolderStatement.setInt(2, userId);

                            int rowsAffected = insertFolderStatement.executeUpdate();

                            return rowsAffected > 0;

                        }

                    }

                }

            }

        } catch (SQLException ex) {

            Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);

            return false;

        }

        return false;

    }

    private static void populateFolderList(String username, DefaultListModel<String> folderListModel) {

        String dbURL = "jdbc:mysql://localhost:3306/dpgprototype";

        try (Connection dbConnect = DriverManager.getConnection(dbURL, "root", "James!DPG123")) {

            String SQL = "SELECT FolderName, DateCreacted, DateModified FROM Folder f " +

                    "JOIN User u ON f.User\_UserID = u.UserID " +

                    "WHERE u.Username = ?";

            try (PreparedStatement preparedStatement = dbConnect.prepareStatement(SQL)) {

                preparedStatement.setString(1, username);

                try (ResultSet resultSet = preparedStatement.executeQuery()) {

                    while (resultSet.next()) {

                        String folderName = resultSet.getString("FolderName");

                        Date dateCreated = resultSet.getDate("DateCreacted");

                        Date dateModified = resultSet.getDate("DateModified");

                        String folderInfo = folderName + " (Created: " + dateCreated + ", Modified: " + dateModified + ")";

                        folderListModel.addElement(folderInfo);

                    }

                }

            }

        } catch (SQLException ex) {

            Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);

            JOptionPane.showMessageDialog(folderFrame, "Error retrieving folder information.");

        }

    }

    private static void showNotesForFolder(String username, String selectedFolder) {

        notesFrame = new JFrame("Notes for " + selectedFolder);

        notesFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

        notesFrame.setMinimumSize(new Dimension(500, 500));

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(3, 1));

        noteListModel = new DefaultListModel<>();

        JList<String> noteList = new JList<>(noteListModel);

        JScrollPane scrollPane = new JScrollPane(noteList);

        noteListModel.clear(); // Clear the notes list

        populateNotesForFolder(username, selectedFolder, noteListModel);

        JButton createNoteButton = new JButton("Create New Note");

        createNoteButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String newNoteTitle = JOptionPane.showInputDialog(notesFrame, "Enter the title for the new note:");

                if (newNoteTitle != null && !newNoteTitle.isEmpty()) {

                    createNewNote(username, selectedFolder, newNoteTitle);

                }

            }

        });

        panel.add(new JLabel("Notes for " + selectedFolder));

        panel.add(scrollPane);

        panel.add(createNoteButton);

        notesFrame.getContentPane().add(panel);

        notesFrame.pack();

        notesFrame.setVisible(true);

        noteList.addListSelectionListener(new ListSelectionListener() {

            @Override

            public void valueChanged(ListSelectionEvent e) {

                if (!e.getValueIsAdjusting()) {

                    String selectedNote = noteList.getSelectedValue();

                    if (selectedNote != null) {

                        // Extract the note title from the selectedNote string

                        String noteTitle = selectedNote.split(" ")[0];

                        openNoteEditingWindow(username, selectedFolder, noteTitle);

                    }

                }

            }

        });

    }

    private static void populateNotesForFolder(String username, String selectedFolder, DefaultListModel<String> noteListModel) {

        System.out.println("Fetching notes for folder: " + selectedFolder);

        String dbURL = "jdbc:mysql://localhost:3306/dpgprototype";

        // Initialize noteInfo to a default message

        String noteInfo = "No notes found for the selected folder.";

        try (Connection dbConnect = DriverManager.getConnection(dbURL, "root", "James!DPG123")) {

            String SQL = "SELECT Title, DateCreacted, LastModifiedDate FROM Notes n " +

                    "INNER JOIN Folder f ON n.Folder\_FolderID = f.FolderID " +

                    "WHERE f.FolderName = ?";

            try (PreparedStatement preparedStatement = dbConnect.prepareStatement(SQL)) {

                preparedStatement.setString(1, selectedFolder);

                try (ResultSet resultSet = preparedStatement.executeQuery()) {

                    if (resultSet.next()) {

                        // If there is at least one result, set noteInfo

                        String noteTitle = resultSet.getString("Title");

                        Date creationDate = resultSet.getDate("DateCreacted");

                        Date modificationDate = resultSet.getDate("LastModifiedDate");

                        noteInfo = noteTitle + " (Created: " + creationDate + ", Modified: " + modificationDate + ")";

                    }

                    // Add the noteInfo to the list model

                    if (!noteInfo.equals("No notes found for the selected folder.")){

                        noteListModel.addElement(noteInfo);

                        System.out.println("Note Info: " + noteInfo);

                    }

                }

            }

        } catch (SQLException ex) {

            Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);

            JOptionPane.showMessageDialog(notesFrame, "Error retrieving note information.");

        }

    }

    private static void openNoteEditingWindow(String username, String folderName, String noteTitle) {

        JFrame noteEditFrame = new JFrame("Note Editing: " + noteTitle);

        noteEditFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

        noteEditFrame.setMinimumSize(new Dimension(500, 500));

        // You can add text areas, labels, and buttons for editing the note content here

        JTextArea noteTextArea = new JTextArea();

        JButton saveButton = new JButton("Save");

        // Load the note content from the database based on username, folderName, and noteTitle

        String noteContent = loadNoteContent(username, folderName, noteTitle);

        noteTextArea.setText(noteContent);

        saveButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Save the updated note content to the database

                String updatedNoteContent = noteTextArea.getText();

                saveNoteContent(username, folderName, noteTitle, updatedNoteContent);

            }

        });

        JPanel panel = new JPanel();

        panel.setLayout(new BorderLayout());

        panel.add(new JScrollPane(noteTextArea), BorderLayout.CENTER);

        panel.add(saveButton, BorderLayout.SOUTH);

        noteEditFrame.getContentPane().add(panel);

        noteEditFrame.pack();

        noteEditFrame.setVisible(true);

    }

    private static String loadNoteContent(String username, String folderName, String noteTitle) {

        String dbURL = "jdbc:mysql://localhost:3306/dpgprototype";

        try (Connection dbConnect = DriverManager.getConnection(dbURL, "root", "James!DPG123")) {

            String SQL = "SELECT Content FROM Notes n " +

                    "INNER JOIN Folder f ON n.Folder\_FolderID = f.FolderID " +

                    "WHERE f.FolderName = ? AND n.Title = ?";

            try (PreparedStatement preparedStatement = dbConnect.prepareStatement(SQL)) {

                preparedStatement.setString(1, folderName);

                preparedStatement.setString(2, noteTitle);

                try (ResultSet resultSet = preparedStatement.executeQuery()) {

                    if (resultSet.next()) {

                        return resultSet.getString("Content");

                    }

                }

            }

        } catch (SQLException ex) {

            Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);

            JOptionPane.showMessageDialog(notesFrame, "Error retrieving note content.");

        }

        return "";

    }

    private static void saveNoteContent(String username, String folderName, String noteTitle, String content) {

        String dbURL = "jdbc:mysql://localhost:3306/dpgprototype";

        try (Connection dbConnect = DriverManager.getConnection(dbURL, "root", "James!DPG123")) {

            String SQL = "UPDATE Notes SET Content = ? " +

                    "WHERE Title = ? AND Folder\_FolderID = " +

                    "(SELECT FolderID FROM Folder WHERE FolderName = ?)";

            try (PreparedStatement preparedStatement = dbConnect.prepareStatement(SQL)) {

                preparedStatement.setString(1, content);

                preparedStatement.setString(2, noteTitle);

                preparedStatement.setString(3, folderName);

                int rowsAffected = preparedStatement.executeUpdate();

                if (rowsAffected > 0) {

                    JOptionPane.showMessageDialog(notesFrame, "Note content saved successfully.");

                }

            }

        } catch (SQLException ex) {

            Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);

            JOptionPane.showMessageDialog(notesFrame, "Error saving note content.");

        }

    }

    private static void createNewNote(String username, String selectedFolder, String newNoteTitle) {

        String dbURL = "jdbc:mysql://localhost:3306/dpgprototype";

        try (Connection dbConnect = DriverManager.getConnection(dbURL, "root", "James!DPG123")) {

            // Retrieve the folder ID for the selected folder

            String folderIdQuery = "SELECT FolderID FROM Folder WHERE FolderName = ?";

            try (PreparedStatement folderIdStatement = dbConnect.prepareStatement(folderIdQuery)) {

                folderIdStatement.setString(1, selectedFolder);

                try (ResultSet folderIdResult = folderIdStatement.executeQuery()) {

                    if (folderIdResult.next()) {

                        int folderId = folderIdResult.getInt("FolderID");

                        // Insert the new note with the correct folder ID

                        String insertSQL = "INSERT INTO Notes (Title, Content, CreationDate, LastModifiedDate, isPublic, Folder\_FolderID, NoteFormatting\_formatingID) VALUES (?, '', CURDATE(), CURDATE(), 1, ?, 1)";

                        try (PreparedStatement preparedStatement = dbConnect.prepareStatement(insertSQL)) {

                            preparedStatement.setString(1, newNoteTitle);

                            preparedStatement.setInt(2, folderId); // Set the folder ID

                            int rowsAffected = preparedStatement.executeUpdate();

                            if (rowsAffected > 0) {

                                JOptionPane.showMessageDialog(notesFrame, "New note created successfully.");

                                noteListModel.addElement(newNoteTitle);

                            } else {

                                JOptionPane.showMessageDialog(notesFrame, "Failed to create a new note.");

                            }

                        }

                    } else {

                        JOptionPane.showMessageDialog(notesFrame, "Folder not found.");

                    }

                }

            }

        } catch (SQLException ex) {

            Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);

            JOptionPane.showMessageDialog(notesFrame, "Error creating a new note.");

        }

    }

}

**Front end**

**System screenshots**

**Login Page**

A screenshot of a login page

Description automatically generated

**Sign-Up Page**

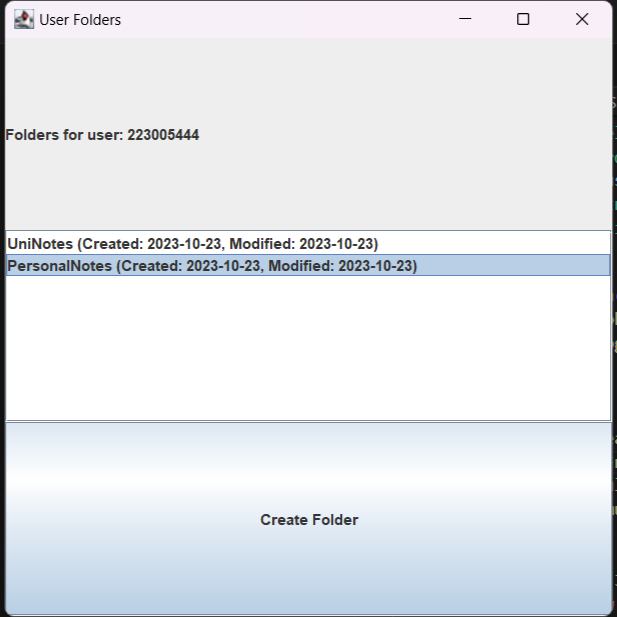
A screenshot of a computer screen

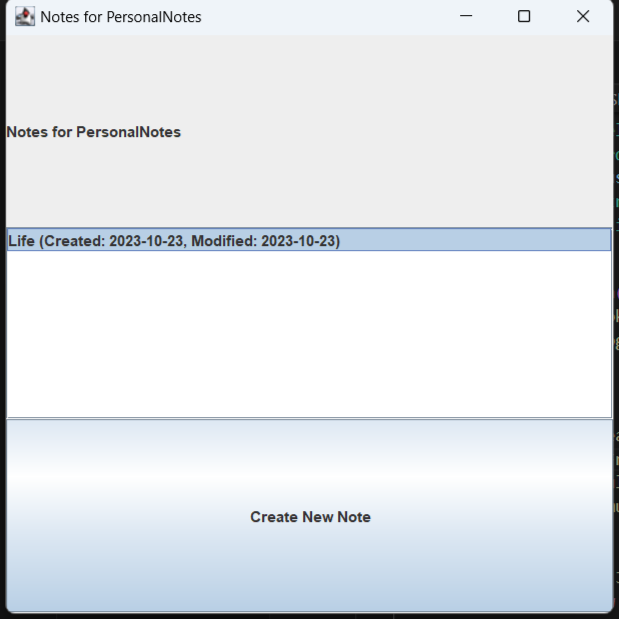
Description automatically generated

**Pop-up if user enters false credentials Page.**

A screenshot of a computer

Description automatically generated

**Folders Page**

**Notes Page**

**Note Editing Page**

