

**Assignment 04**

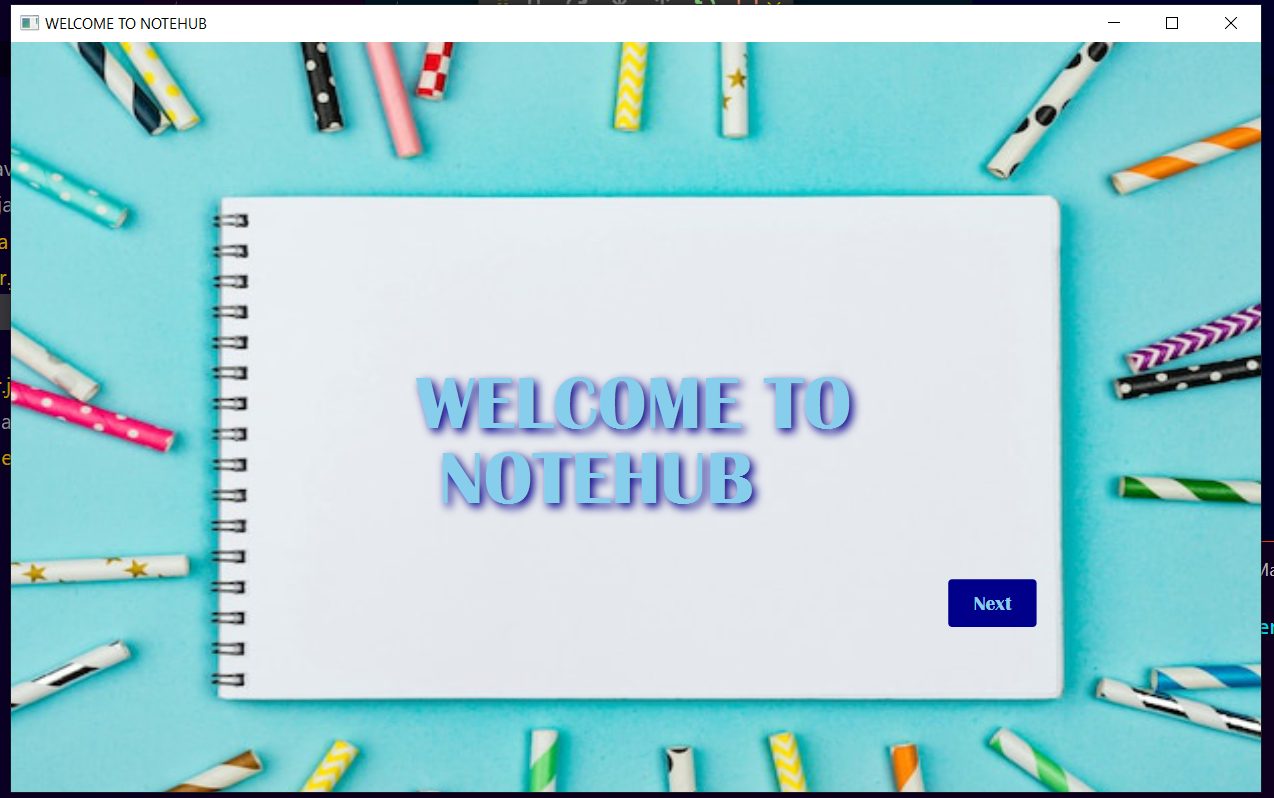
**Project Name: NoteHub**

**Group Members: Abeera (SP24-BSE-004)**

**Nafeesa Mushtaq (SP24-BSE-096)**

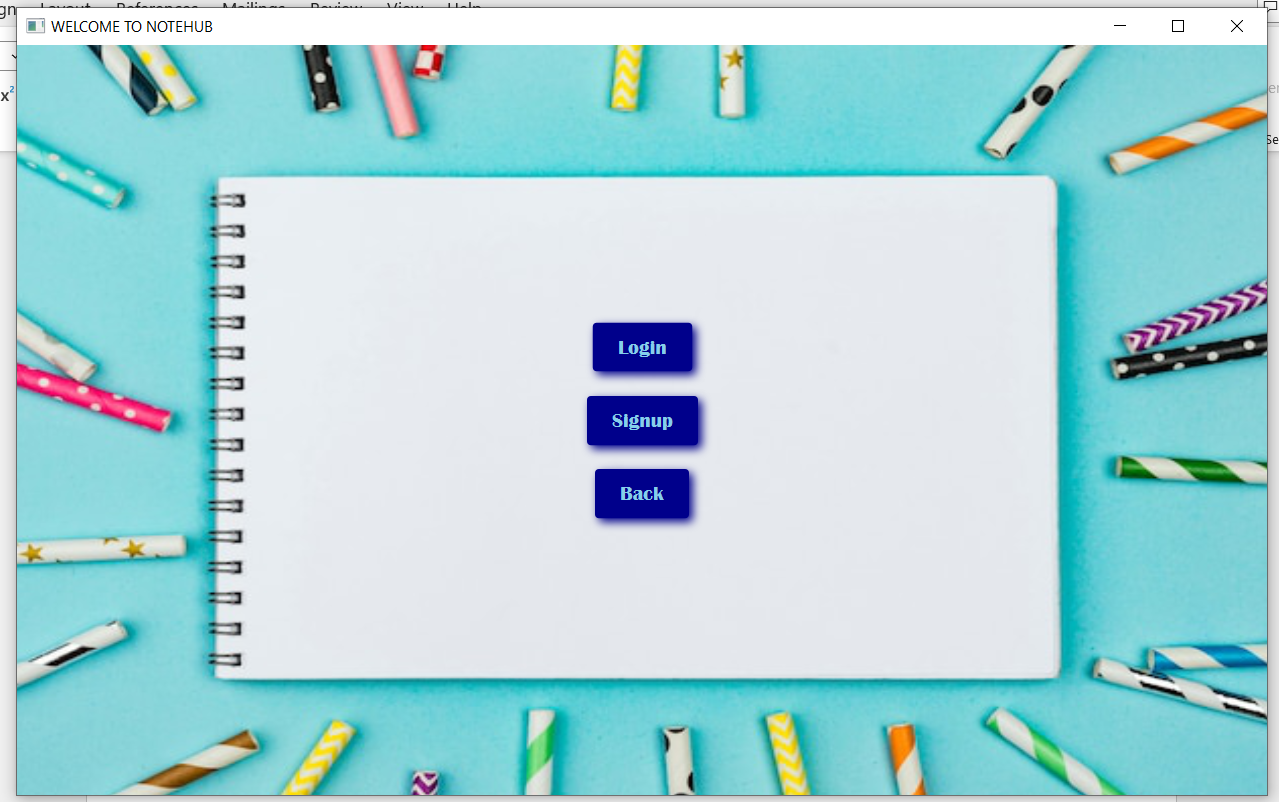
**Section : A**

**Home Page**

****

This is the home page of our NoteHub app. It contains a next button through which we can navigate to the next page.

**Menu Page**

****

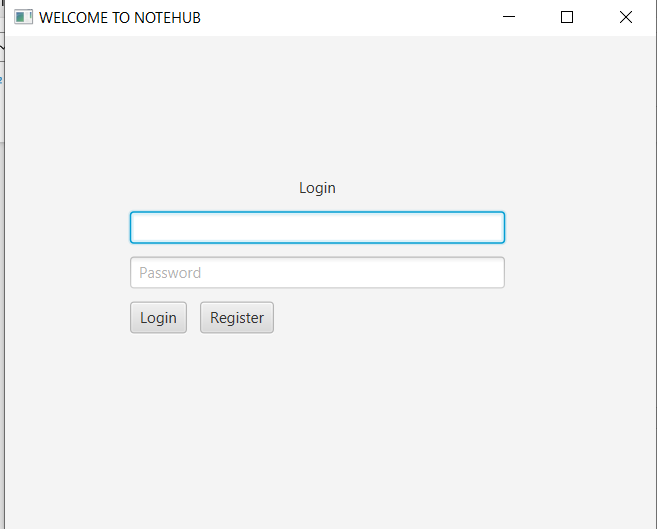
This page have login sign-up and back buttons.

On clicking the login button it will navigate the user to the login form through which user can login to the app.

On clicking the sign-up button it will navigate the user to the sign-up form thorough which users can sign-up.

On clicking the back button it will navigate us to the Home page of the NoteHub app.

**Login Form**

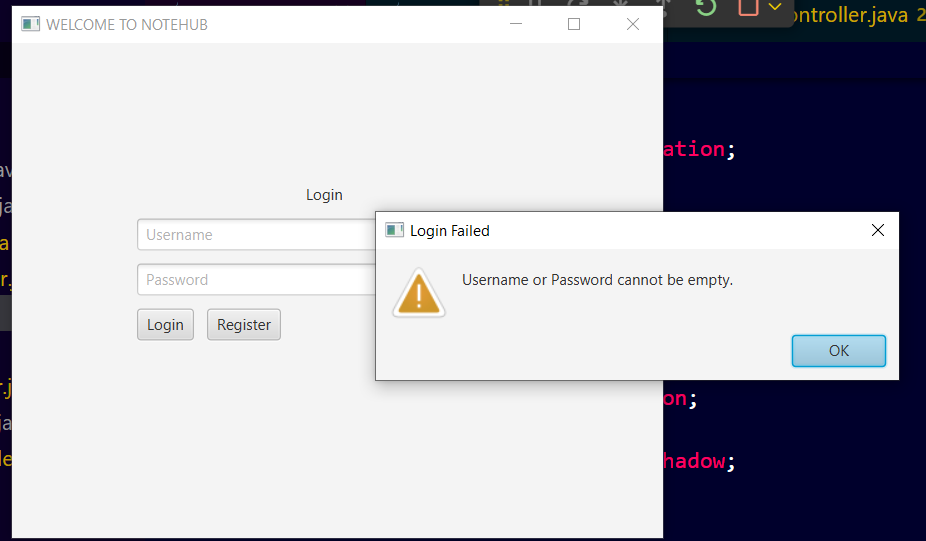
****

This login form contains textfields to enter the username and passwords.

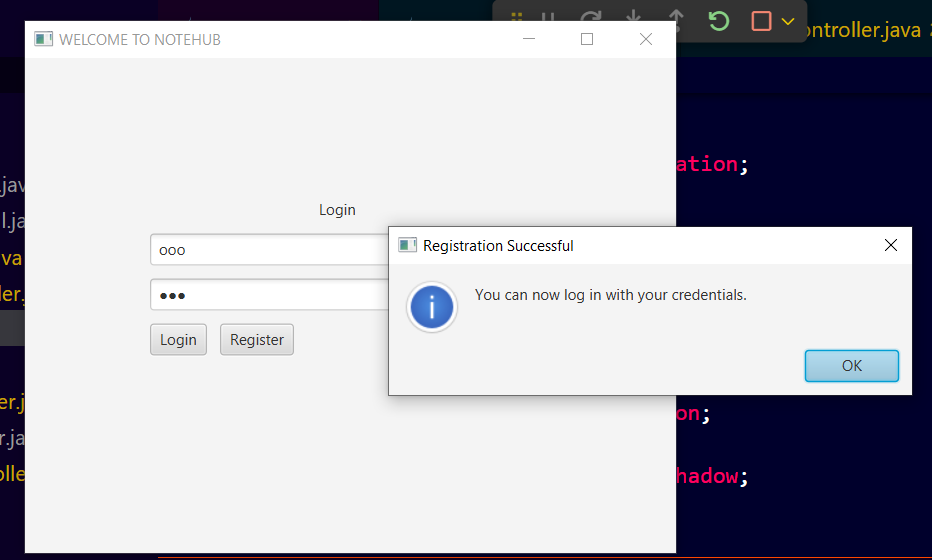
It contains two buttons named as login and register. After loging in it will take us to add the notes.

**Alerts**

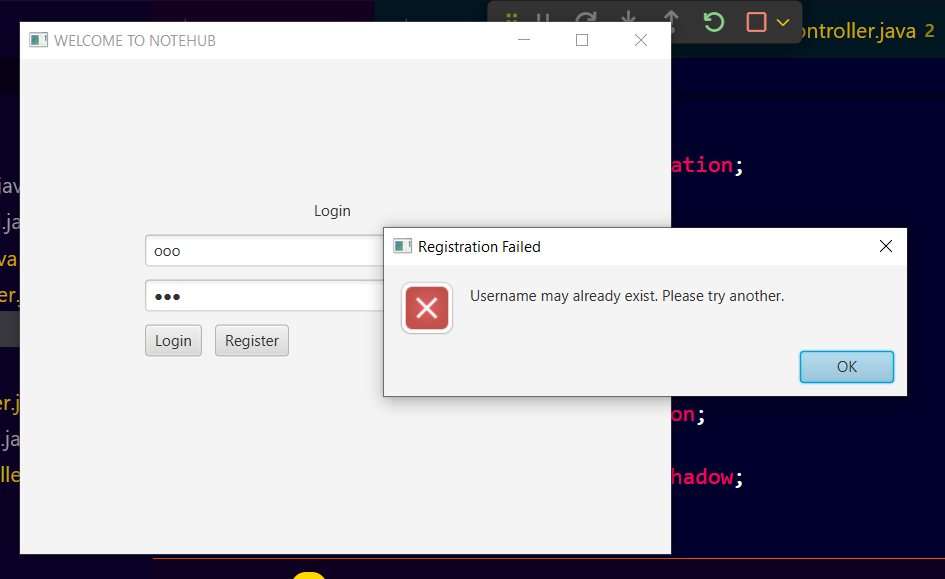
If these textfeilds are empty, and the user will try to login, It will show the following alerts.



If they are filled properly, it will show the following pop-ups.

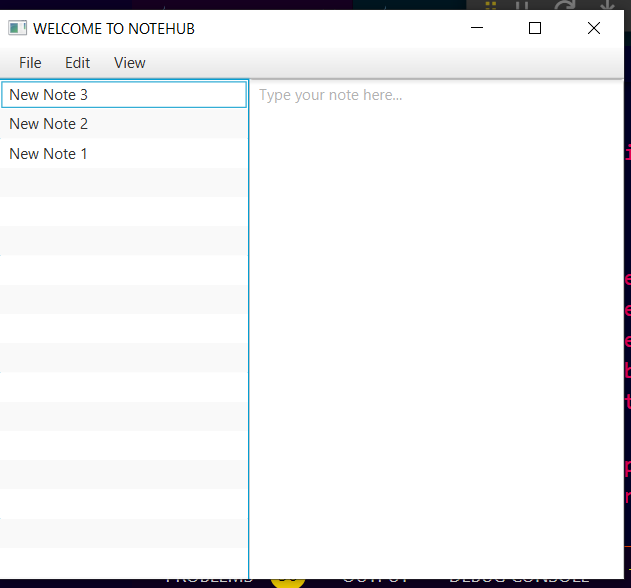


If the user with the same name and password already exists.

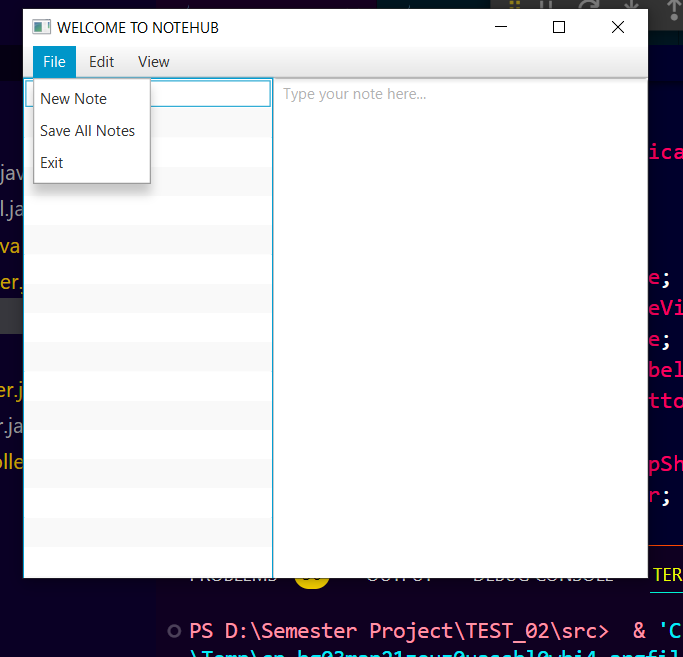


**Welcome to NoteHub**

If the login is successful. It will lead us to the main functionality of the app.



**The Menu Bar**

****

It contains the functionality to add up the new note, save the existing notes and existing from the app.

Moreover, it has the functionality to edit the notes through edit button.

Through View button we can change the view/ Theme of the app.

**The Whole Code**

*package* com.noteapp;

*import* javafx.application.Application;

*import* javafx.fxml.FXMLLoader;

*import* javafx.geometry.Pos;

*import* javafx.scene.Scene;

*import* javafx.scene.image.Image;

*import* javafx.scene.image.ImageView;

*import* javafx.scene.layout.Pane;

*import* javafx.scene.control.Label;

*import* javafx.scene.control.Button;

*import* javafx.scene.text.Font;

*import* javafx.scene.effect.DropShadow;

*import* javafx.scene.paint.Color;

*import* javafx.stage.Stage;

*import* javafx.scene.layout.StackPane;

*import* javafx.scene.layout.VBox;

*import* java.io.IOException;

public class Main extends Application {

    private final FileHandler fileHandler = new FileHandler();

    private final AppManager appManager = new AppManager();

    public static void main(String[] args) {

        launch(args);

    }

    @Override

    public void start(Stage stage) {

        showWelcomePage(stage);

    }

    private void showWelcomePage(Stage stage) {

        Image backgroundImg = new Image("img.jpg");

        ImageView backgroundView = new ImageView(backgroundImg);

        backgroundView.fitWidthProperty().bind(stage.widthProperty());

        backgroundView.fitHeightProperty().bind(stage.heightProperty());

        Label label = new Label("WELCOME TO \n NOTEHUB");

        label.setFont(new Font("Britannic Bold", 60));

        label.setTextFill(Color.SKYBLUE);

        DropShadow dropShadow = new DropShadow();

        dropShadow.setColor(Color.DARKBLUE);

        dropShadow.setOffsetX(3);

        dropShadow.setOffsetY(3);

        label.setEffect(dropShadow);

        label.layoutXProperty().bind(stage.widthProperty().subtract(label.widthProperty()).divide(2));

        label.layoutYProperty().bind(stage.heightProperty().subtract(label.heightProperty()).divide(2));

        Button nextButton = new Button("Next");

        nextButton.setFont(new Font("Britannic Bold", 16));

        nextButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: skyblue; -fx-padding: 10px 20px;");

        nextButton.setOnMouseEntered(e -> nextButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: white; -fx-padding: 10px 20px;"));

        nextButton.setOnMouseExited(e -> nextButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: skyblue; -fx-padding: 10px 20px;"));

        nextButton.setLayoutX(750);

        nextButton.setLayoutY(430);

        nextButton.setOnAction(e -> showLoginPage(stage));

        Pane root = new Pane();

        root.getChildren().addAll(backgroundView, label, nextButton);

        Scene scene = new Scene(root, 1000, 600); *// Set initial window size*

        stage.setTitle("WELCOME TO NOTEHUB");

        stage.setScene(scene);

        stage.show();

    }

    private void showLoginPage(Stage stage) {

        Image backgroundImg = new Image("img.jpg");

        ImageView backgroundView = new ImageView(backgroundImg);

        backgroundView.fitWidthProperty().bind(stage.widthProperty());

        backgroundView.fitHeightProperty().bind(stage.heightProperty());

        VBox root = new VBox(20); *// Root layout with spacing between buttons*

        DropShadow dropShadow = new DropShadow();

        dropShadow.setColor(Color.DARKBLUE);

        dropShadow.setOffsetX(3);

        dropShadow.setOffsetY(3);

        Button loginButton = new Button("Login");

        loginButton.setFont(new Font("Britannic Bold", 16));

        loginButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: skyblue; -fx-padding: 10px 20px;");

        loginButton.setOnMouseEntered(e -> loginButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: white; -fx-padding: 10px 20px;"));

        loginButton.setOnMouseExited(e -> loginButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: skyblue; -fx-padding: 10px 20px;"));

        Button signupButton = new Button("Signup");

        signupButton.setFont(new Font("Britannic Bold", 16));

        signupButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: skyblue; -fx-padding: 10px 20px;");

        signupButton.setOnMouseEntered(e -> signupButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: white; -fx-padding: 10px 20px;"));

        signupButton.setOnMouseExited(e -> signupButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: skyblue; -fx-padding: 10px 20px;"));

        Button backButton = new Button("Back");

        backButton.setFont(new Font("Britannic Bold", 16));

        backButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: skyblue; -fx-padding: 10px 20px;");

        backButton.setOnMouseEntered(e -> backButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: white; -fx-padding: 10px 20px;"));

        backButton.setOnMouseExited(e -> backButton.setStyle("-fx-background-color: darkblue; -fx-text-fill: skyblue; -fx-padding: 10px 20px;"));

        backButton.setOnAction(e -> showWelcomePage(stage));

        loginButton.setOnAction(e -> showLoginForm(stage));

        signupButton.setOnAction(e -> showSignupForm(stage));

        root.getChildren().addAll(loginButton, signupButton,backButton);

        backButton.setEffect(dropShadow);

        signupButton.setEffect(dropShadow);

        loginButton.setEffect(dropShadow);

        root.setAlignment(Pos.CENTER);

        StackPane mainRoot = new StackPane();

        mainRoot.getChildren().addAll(backgroundView, root);

        Scene loginScene = new Scene(mainRoot,1000, 600);

        stage.setScene(loginScene);

        stage.show();

    }

    private void showLoginForm(Stage stage) {

        try {

            FXMLLoader loader = new FXMLLoader(getClass().getResource("/resources/login.fxml"));

            Scene loginFormScene = new Scene(loader.load(),1000,600);

            LoginController controller = loader.getController();

*//  controller.setFileHandler(fileHandler); // Pass the FileHandler to the controller*

            stage.setScene(loginFormScene);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    private void showSignupForm(Stage stage) {

        try {

            FXMLLoader loader = new FXMLLoader(getClass().getResource("/resources/signupForm.fxml"));

            Scene signupFormScene = new Scene(loader.load(),1000,600);

            SignupController controller = loader.getController();

            controller.setFileHandler(fileHandler); *// Pass the FileHandler to the controller*

            stage.setScene(signupFormScene);

*//  stage.setResizable(false);*

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

*package* com.noteapp;

*import* javafx.fxml.FXML;

*import* javafx.scene.control.ListView;

*import* javafx.scene.control.TextArea;

public class MainController {

    @FXML

    private ListView<String> notesList;

    @FXML

    private TextArea noteContent;

    private final FileHandler fileHandler = new FileHandler();

    @FXML

    public void initialize() {

        notesList.getItems().addAll(fileHandler.getAllNotes().keySet());

        notesList.getSelectionModel().selectedItemProperty().addListener((observable, oldValue, newValue) -> {

            if (newValue != null) {

                noteContent.setText(fileHandler.getNote(newValue));

            }

        });

    }

    @FXML

    private void createNote() {

        String newNoteTitle = "New Note " + (notesList.getItems().size() + 1);

        notesList.getItems().add(newNoteTitle);

        fileHandler.saveNote(newNoteTitle, "");

    }

    @FXML

    private void saveNotes() {

        String selectedNote = notesList.getSelectionModel().getSelectedItem();

        if (selectedNote != null) {

            fileHandler.saveNote(selectedNote, noteContent.getText());

        }

    }

    @FXML

    private void searchNotes() {

        System.out.println("Search feature not implemented yet.");

    }

    @FXML

    private void changeTheme() {

        System.out.println("Theme change feature not implemented yet.");

    }

    @FXML

    private void exitApp() {

        System.exit(0);

    }

}

*package* com.noteapp;

*import* javafx.fxml.FXMLLoader;

*import* javafx.scene.Scene;

*import* javafx.stage.Stage;

*import* java.io.IOException;

public class AppManager {

    public Scene getLoginScene(Stage stage) {

        try {

            FXMLLoader loader = new FXMLLoader(getClass().getResource("/resources/login.fxml"));

            return new Scene(loader.load());

        } catch (IOException e) {

            e.printStackTrace();

            return null;

        }

    }

    public Scene getMainScene(Stage stage) {

        try {

            FXMLLoader loader = new FXMLLoader(getClass().getResource("/resources/main.fxml"));

            return new Scene(loader.load());

        } catch (IOException e) {

            e.printStackTrace();

            return null;

        }

    }

}

*// package com.noteapp;*

*// import javafx.fxml.FXML;*

*// import javafx.scene.control.Alert;*

*// import javafx.scene.control.Button;*

*// import javafx.scene.control.PasswordField;*

*// import javafx.scene.control.TextField;*

*// import javafx.stage.Stage;*

*// public class LoginController {*

*//     @FXML*

*//     private TextField usernameField;*

*//     @FXML*

*//     private PasswordField passwordField;*

*//     @FXML*

*//     private Button loginButton;*

*//     @FXML*

*//     private Button registerButton;*

*//     private final FileHandler fileHandler = new FileHandler();*

*//     private final EncryptionUtil encryptionUtil = new EncryptionUtil();*

*//      @FXML*

*//     public void initialize() {*

*//         loginButton.setOnAction(event -> handleLogin());*

*//         registerButton.setOnAction(event -> handleRegistration());*

*//     }*

*//     private void handleLogin() {*

*//         String username = usernameField.getText();*

*//         String password = passwordField.getText();*

*//         String encryptedPassword = encryptionUtil.encrypt(password);*

*//         if (fileHandler.verifyUser(username, encryptedPassword)) {*

*//             Stage stage = (Stage) loginButton.getScene().getWindow();*

*//             AppManager appManager = new AppManager();*

*//             stage.setScene(appManager.getMainScene(stage));*

*//         } else {*

*//             System.out.println("Login failed. Please check your credentials.");*

*//         }*

*//     }*

*//     private void handleRegistration() {*

*//         String username = usernameField.getText();*

*//         String password = passwordField.getText();*

*//         String encryptedPassword = encryptionUtil.encrypt(password);*

*//         if (fileHandler.saveUser(username, encryptedPassword)) {*

*//             System.out.println("Registration successful. Please log in.");*

*//         } else {*

*//             System.out.println("Registration failed. Username may already exist.");*

*//         }*

*//     }*

*// }*

*package* com.noteapp;

*import* javafx.fxml.FXML;

*import* javafx.scene.control.Alert;

*import* javafx.scene.control.Alert.AlertType;

*import* javafx.scene.control.Button;

*import* javafx.scene.control.PasswordField;

*import* javafx.scene.control.TextField;

*import* javafx.stage.Stage;

public class LoginController {

    @FXML

    private TextField usernameField;

    @FXML

    private PasswordField passwordField;

    @FXML

    private Button loginButton;

    @FXML

    private Button registerButton;

    private final FileHandler fileHandler = new FileHandler();

    private final EncryptionUtil encryptionUtil = new EncryptionUtil();

    @FXML

    public void initialize() {

        loginButton.setOnAction(event -> handleLogin());

        registerButton.setOnAction(event -> handleRegistration());

    }

    private void handleLogin() {

        String username = usernameField.getText();

        String password = passwordField.getText();

*// Check if username or password is empty*

        if (username.isEmpty() || password.isEmpty()) {

            showAlert("Login Failed", "Username or Password cannot be empty.", AlertType.WARNING);

            return;

        }

        String encryptedPassword = encryptionUtil.encrypt(password);

        if (fileHandler.verifyUser(username, encryptedPassword)) {

            Stage stage = (Stage) loginButton.getScene().getWindow();

            AppManager appManager = new AppManager();

            stage.setScene(appManager.getMainScene(stage));

        } else {

            showAlert("Login Failed", "Invalid credentials. Please try again.", AlertType.ERROR);

        }

    }

    private void handleRegistration() {

        String username = usernameField.getText();

        String password = passwordField.getText();

*// Check if username or password is empty*

        if (username.isEmpty() || password.isEmpty()) {

            showAlert("Registration Failed", "Username or Password cannot be empty.", AlertType.WARNING);

            return;

        }

        String encryptedPassword = encryptionUtil.encrypt(password);

        if (fileHandler.saveUser(username, encryptedPassword)) {

            showAlert("Registration Successful", "You can now log in with your credentials.", AlertType.INFORMATION);

        } else {

            showAlert("Registration Failed", "Username may already exist. Please try another.", AlertType.ERROR);

        }

    }

    private void showAlert(String title, String message, AlertType alertType) {

        Alert alert = new Alert(alertType);

        alert.setTitle(title);

        alert.setHeaderText(null);

        alert.setContentText(message);

        alert.showAndWait();

    }

}

*package* com.noteapp;

*import* javax.crypto.Cipher;

*import* javax.crypto.SecretKey;

*import* javax.crypto.spec.SecretKeySpec;

*import* java.util.Base64;

public class EncryptionUtil {

    private static final String ALGORITHM = "AES";

    private static final byte[] KEY = "1234567890123456".getBytes();

    public String encrypt(String data) {

        try {

            SecretKey secretKey = new SecretKeySpec(KEY, ALGORITHM);

            Cipher cipher = Cipher.getInstance(ALGORITHM);

            cipher.init(Cipher.ENCRYPT\_MODE, secretKey);

            byte[] encrypted = cipher.doFinal(data.getBytes());

            return Base64.getEncoder().encodeToString(encrypted);

        } catch (Exception e) {

            e.printStackTrace();

            return null;

        }

    }

    public String decrypt(String data) {

        try {

            SecretKey secretKey = new SecretKeySpec(KEY, ALGORITHM);

            Cipher cipher = Cipher.getInstance(ALGORITHM);

            cipher.init(Cipher.DECRYPT\_MODE, secretKey);

            byte[] decoded = Base64.getDecoder().decode(data);

            return new String(cipher.doFinal(decoded));

        } catch (Exception e) {

            e.printStackTrace();

            return null;

        }

    }

}

*package* com.noteapp;

*import* java.io.*\**;

*import* java.util.HashMap;

*import* java.util.Map;

public class FileHandler {

    private static final String USER\_FILE = "data/users.dat";

    private static final String NOTES\_FILE = "data/notes.dat";

    private Map<String, String> users;

    private Map<String, String> notes;

    public FileHandler() {

        users = loadUsers();

        notes = loadNotes();

    }

*// Load users from file*

    private Map<String, String> loadUsers() {

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(USER\_FILE))) {

            return (Map<String, String>) ois.readObject();

        } catch (IOException | ClassNotFoundException e) {

            return new HashMap<>(); *// Return empty map if file does not exist*

        }

    }

*// Save users to file*

    private void saveUsers() {

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(USER\_FILE))) {

            oos.writeObject(users);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

*// Save a new user*

    public boolean saveUser(String username, String password) {

        if (users.containsKey(username)) {

            return false; *// User already exists*

        }

        users.put(username, password);

        saveUsers();

        return true;

    }

*// Verify a user's credentials*

    public boolean verifyUser(String username, String password) {

        return password.equals(users.get(username));

    }

*// Load notes from file*

    private Map<String, String> loadNotes() {

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(NOTES\_FILE))) {

            return (Map<String, String>) ois.readObject();

        } catch (IOException | ClassNotFoundException e) {

            return new HashMap<>(); *// Return empty map if file does not exist*

        }

    }

*// Save notes to file*

    private void saveNotes() {

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(NOTES\_FILE))) {

            oos.writeObject(notes);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

*// Save a note*

    public void saveNote(String title, String content) {

        notes.put(title, content);

        saveNotes();

    }

*// Retrieve a note*

    public String getNote(String title) {

        return notes.get(title);

    }

*// Delete a note*

    public void deleteNote(String title) {

        notes.remove(title);

        saveNotes();

    }

*// Get all notes*

    public Map<String, String> getAllNotes() {

        return notes;

    }

    public boolean doesUserExist(String username) {

        throw new UnsupportedOperationException("Unimplemented method 'doesUserExist'");

    }

}

*package* com.noteapp;

*import* java.util.ArrayList;

*import* java.util.List;

*import* java.util.Map;

public class NoteManager {

    private final FileHandler fileHandler;

    public NoteManager(FileHandler fileHandler) {

*this*.fileHandler = fileHandler;

    }

*/\*\**

*\* Create a new note.*

*\**

*\* @param title   The title of the note.*

*\* @param content The content of the note.*

*\* @return True if the note was created successfully, false if the title already exists.*

*\*/*

    public boolean createNote(String title, String content) {

        if (fileHandler.getAllNotes().containsKey(title)) {

            return false; *// Note with the same title already exists*

        }

        fileHandler.saveNote(title, content);

        return true;

    }

*/\*\**

*\* Edit an existing note.*

*\**

*\* @param title   The title of the note to edit.*

*\* @param content The new content for the note.*

*\*/*

    public void editNote(String title, String content) {

        fileHandler.saveNote(title, content);

    }

*/\*\**

*\* Delete a note by its title.*

*\**

*\* @param title The title of the note to delete.*

*\*/*

    public void deleteNote(String title) {

        fileHandler.deleteNote(title);

    }

*/\*\**

*\* Get the content of a note by its title.*

*\**

*\* @param title The title of the note.*

*\* @return The content of the note, or null if the note does not exist.*

*\*/*

    public String getNoteContent(String title) {

        return fileHandler.getNote(title);

    }

*/\*\**

*\* Search for notes containing a specific keyword in the title or content.*

*\**

*\* @param keyword The keyword to search for.*

*\* @return A list of matching note titles.*

*\*/*

    public List<String> searchNotes(String keyword) {

        Map<String, String> notes = fileHandler.getAllNotes();

        List<String> matchingNotes = new ArrayList<>();

        for (Map.Entry<String, String> entry : notes.entrySet()) {

            if (entry.getKey().toLowerCase().contains(keyword.toLowerCase()) ||

                    entry.getValue().toLowerCase().contains(keyword.toLowerCase())) {

                matchingNotes.add(entry.getKey());

            }

        }

        return matchingNotes;

    }

}

*package* com.noteapp;

*import* javafx.fxml.FXML;

*import* javafx.scene.Scene;

*import* javafx.scene.control.Alert;

*import* javafx.scene.control.PasswordField;

*import* javafx.scene.control.TextField;

public class SignupController {

    @FXML

    private TextField usernameField;

    @FXML

    private PasswordField passwordField;

    private FileHandler fileHandler;

*/\*\**

*\* Sets the FileHandler instance for file operations.*

*\**

*\* @param fileHandler the FileHandler instance*

*\*/*

    public void setFileHandler(FileHandler fileHandler) {

*this*.fileHandler = fileHandler;

    }

*/\*\**

*\* Handles the signup action when the signup button is clicked.*

*\*/*

    @FXML

    private void handleSignup() {

        String username = usernameField.getText().trim();

        String password = passwordField.getText().trim();

        final AppManager appManager = new AppManager();

*//  Validate input*

        if (username.isEmpty() || password.isEmpty()) {

            showAlert(Alert.AlertType.ERROR, "Invalid Input", "Please enter both a username and a password.");

            return;

        }

*// Save user credentials to file*

        try {

            if (fileHandler == null) {

                throw new IllegalStateException("FileHandler is not set. Please set it before using this controller.");

            }

*// Check if the username already exists*

            if (fileHandler.doesUserExist(username)) {

                showAlert(Alert.AlertType.ERROR, "Signup Failed", "The username already exists. Please choose a different one.");

                return;

            }

*// Save credentials*

            fileHandler.saveUser(username, password);

            showAlert(Alert.AlertType.INFORMATION, "Signup Successful", "You have successfully signed up!");

*// Clear fields after successful signup*

            usernameField.clear();

            passwordField.clear();

        } catch (Exception e) {

            e.printStackTrace();

            showAlert(Alert.AlertType.ERROR, "Error", "An error occurred while signing up. Please try again.");

        }

    }

*/\*\**

*\* Displays an alert dialog with the given type, title, and content.*

*\**

*\* @param alertType the type of alert*

*\* @param title     the title of the alert*

*\* @param content   the content of the alert*

*\*/*

    private void showAlert(Alert.AlertType alertType, String title, String content) {

        Alert alert = new Alert(alertType);

        alert.setTitle(title);

        alert.setHeaderText(null);

        alert.setContentText(content);

        alert.showAndWait();

    }

}

*package* com.noteapp;

*import* javafx.scene.control.Alert;

*import* javafx.scene.control.Alert.AlertType;

public class UIUtils {

*/\*\**

*\* Display an alert dialog to the user.*

*\**

*\* @param type    The type of the alert (INFO, WARNING, ERROR).*

*\* @param title   The title of the alert window.*

*\* @param message The message content of the alert.*

*\*/*

    public static void showAlert(AlertType type, String title, String message) {

        Alert alert = new Alert(type);

        alert.setTitle(title);

        alert.setHeaderText(null);

        alert.setContentText(message);

        alert.showAndWait();

    }

*/\*\**

*\* Apply a custom CSS stylesheet to the scene.*

*\**

*\* @param scene      The current scene.*

*\* @param stylesheet Path to the CSS file.*

*\*/*

    public static void applyStylesheet(javafx.scene.Scene scene, String stylesheet) {

        scene.getStylesheets().clear();

        scene.getStylesheets().add(stylesheet);

    }

}

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane xmlns:fx="http://javafx.com/fxml" fx:controller="com.noteapp.LoginController">

    <children>

        <VBox alignment="CENTER" spacing="10.0" layoutX="100" layoutY="50" prefWidth="300" prefHeight="250">

            <Label text="Login" styleClass="heading"/>

            <TextField fx:id="usernameField" promptText="Username"/>

            <PasswordField fx:id="passwordField" promptText="Password"/>

            <HBox spacing="10">

                <Button fx:id="loginButton" text="Login"/>

                <Button fx:id="registerButton" text="Register"/>

            </HBox>

        </VBox>

    </children>

</AnchorPane>

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<BorderPane xmlns:fx="http://javafx.com/fxml" fx:controller="com.noteapp.MainController">

    <top>

        <MenuBar>

            <Menu text="File">

                <MenuItem text="New Note" onAction="#createNote"/>

                <MenuItem text="Save All Notes" onAction="#saveNotes"/>

                <MenuItem text="Exit" onAction="#exitApp"/>

            </Menu>

            <Menu text="Edit">

                <MenuItem text="Search Notes" onAction="#searchNotes"/>

            </Menu>

            <Menu text="View">

                <MenuItem text="Themes" onAction="#changeTheme"/>

            </Menu>

        </MenuBar>

    </top>

    <center>

        <ListView fx:id="notesList" prefWidth="200"/>

    </center>

    <right>

        <TextArea fx:id="noteContent" promptText="Type your note here..." prefWidth="300"/>

    </right>

</BorderPane>

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.PasswordField?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.VBox?>

<VBox xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.noteapp.SignupController" spacing="10" alignment="CENTER" style="-fx-padding: 20;">

    <Label text="Signup" style="-fx-font-size: 24pt; -fx-text-fill: darkblue;"/>

    <TextField fx:id="usernameField" promptText="Username"/>

    <PasswordField fx:id="passwordField" promptText="Password"/>

    <Button text="Signup" onAction="#handleSignup" style="-fx-background-color: darkblue; -fx-text-fill: skyblue; -fx-padding: 10px 20px;"/>

</VBox>