# ReadMe file

**FaceLite Social network application Description**

FaceLite is a social networking application that was developed as part of the ICS-108: Object-Oriented Programming course project. The main goal of the project was to create a user-friendly application that facilitates the management of a basic social network. With FaceLite, users have the ability to create personalized profiles, connect with friends, share status updates, and engage with their network through various interactive features.

**Table of contents:**

1. Features

2. To run the FaceLite application, follow these steps

3. Technologies Used

4. Design Overview

* FaceLiteApplication
* FaceLiteController
* User

5. Usage

* Add button
* Change Status
* Change picture
* Add friend button
* Lookup button
* Delete button

Features

1. **Data Persistence:** We have implemented a robust data persistence feature by incorporating a file-based storage system using CSV format. This enables the application to save and load user data consistently, ensuring that user profiles and associated information remain intact even after the application is restarted.

2. **File Explorer for Profile Picture:** Our application now includes a user-friendly file explorer functionality within the "Change Picture" feature. This enhancement empowers users to conveniently select a desired photo from their local storage as their profile picture. The file explorer interface streamlines the process of browsing and selecting image files, augmenting the overall user experience.

3. **Scene Switching User Interface:** We have meticulously designed an intuitive user interface that seamlessly facilitates scene switching. This dynamic interface enables the application to present distinct screens or views, adapting to user interactions and specific functionalities. By providing clear and logical navigation between different sections of the application, the scene switching feature enhances usability and ensures a polished user experience.

4. **Account Deletion Confirmation**: We have added a confirmation prompt to the "Delete" button functionality. This ensures that users are prompted to confirm their intention to delete their account before proceeding. By incorporating this confirmation step, we prioritize user safety and prevent accidental deletion of user profiles, promoting a user-centric approach to account management.

To run the FaceLite application, follow these steps:

1.Ensure that you have intelliJ IDEA and SceneBuilder on your system.

2.Open the project in your preferred Java IDE.

3.run the application.

4.The application will launch with a user interface displaying options to add, delete, and lookup profiles, along with options to update status and profile pictures.

Technologies Used:

1. intelliJ IDEA

2. SceneBuilder

Design Overview:

1. **FaceLiteApplication**

**Purpose:** The FaceLiteApplication class serves as the entry point for the FaceLite application. It extends the JavaFX Application class and controls the loading and saving of user data, as well as the initialization and display of the application's UI.

**Variables:**

None explicitly defined in the code snippet.

**Methods:**

**loadUserData():** This method loads user data from the "users.txt" file. It reads each line from the file, splits it into individual data elements, creates User objects using the data, and adds them to the FaceLiteController's users list.

**saveUserData():** This static method saves user data from the FaceLiteController's users list to the "users.txt" file. It iterates over the list of users, converts each User object to its string representation using the toString() method, and writes it to the file.

**start(Stage stage):** This method is an override of the start() method from the JavaFX Application class. It is called when the application is launched. Inside this method, user data is loaded using loadUserData(). It then loads the FXML file that defines the application's UI using the FXMLLoader, creates a Scene with the loaded UI, sets it on the Stage, and displays the Stage.

**main(String[] args):** This is the entry point for the application. It launches the JavaFX application by calling the launch() method inherited from the Application class and then calls saveUserData() to save the user data before exiting

2. **FaceLiteController**

**Purpose:** The FaceLiteController class serves as the controller for the FaceLite application's UI. It handles various UI elements, such as labels, text fields, buttons, and image views, and performs actions based on user interactions.

**Variables:**

**logo:** Represents an ImageView for displaying the FaceLite logo.

**userName:** Represents a Label for displaying the user's name.

**friendsLabel**: Represents a Label for displaying the user's friends.

**update**: Represents a Label for displaying update messages.

**statusLabel**: Represents a Label for displaying the user's status.

**nameTextField**: Represents a TextField for entering the user's name.

**addFriendTextField**: Represents a TextField for entering a friend's name.

**friendList**: Represents a VBox container for displaying the user's friend list.

**comboBox**: Represents a ComboBox for selecting the user's status.

**circle**: Represents a Circle shape for displaying the user's profile picture.

**users**: An ArrayList of User objects to store the registered users.

**Methods:**

**showLogo():** Loads and displays the FaceLite logo image in the logo ImageView.

**clear():** Clears the UI elements related to the user's profile, including name, friends, status, profile picture, and friend list.

**add():** Handles the action when the "Add" button is clicked. It checks if the entered name already exists and creates a new user profile if it doesn't.

**addUser(String name):** Adds a new user with the specified name to the application.

**changeStatus():**It updates the user's status when a status is entered.

**lookup():** Handles the action when the "Lookup" button is clicked. It retrieves and displays the profile information of the entered name.

**changePicture():** Handles the action when the "Change Picture" button is clicked. It allows the user to select and update their profile picture.

**addFriend():** Handles the action when the "Add Friend" button is clicked. It adds the entered friend's name to the user's friend list.

**delete():** Handles the action when the "Delete" button is clicked. It prompts the user for confirmation before deleting their profile.

3. **User**

**Purpose:** The User class is responsible for creating and managing user profiles in the FaceLite application. It encapsulates the user's data and provides methods to access and modify the profile information.

**Variables:**

**name**: Represents the user's name. It is a String variable.

**imagePath**: Represents the file path of the user's profile picture. It is a String variable.

**status**: Represents the user's current status. It is a String variable.

**friends**: Represents the user's friend list. It is a LinkedHashSet of Strings, ensuring uniqueness and preserving insertion order.

**Methods:**

**User(String name):** Constructor method that initializes a User object with the specified name. It sets the default values for the profile picture, status, and friend list.

**getName():** Returns the user's name as a String.

**getPicturePath():** Returns the file path of the user's profile picture as a String.

**setPicturePath(String imagePath):** Sets the file path of the user's profile picture using the specified imagePath.

**getStatus():** Returns the user's current status as a String.

**setStatus(String status):** Sets the user's current status using the specified status.

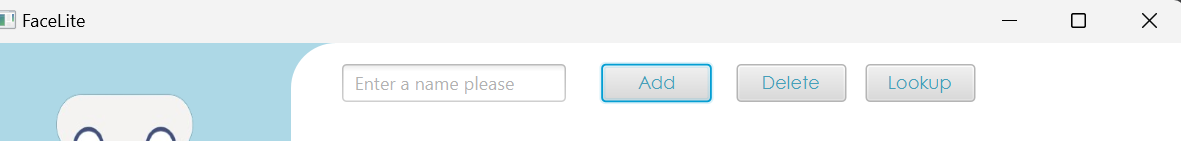
**userAddFriend(String friend):** Adds a friend to the user's friend list. It takes a String parameter representing the friend's name.

**getFriend():** Returns the user's friend list as a LinkedHashSet of Strings.

**toString():** Overrides the toString() method to return a String representation of the User object. It includes the user's name, status, profile picture path, and friend list.

Usage:

* Add button



**To use the "Add" button:**

1.Enter the desired name for the new user in the nameTextField.

2.Click the "Add" button

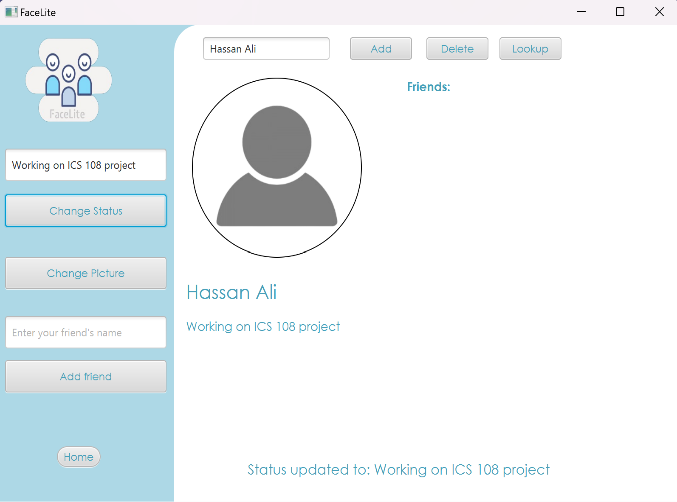
**The code of the add button**:

public void add() {  
 showLogo();  
 try {  
 String name = nameTextField.getText();  
 boolean userExists = false;  
 friendList.getChildren().clear();  
  
  
 if (!name.isEmpty()) {  
 for (User user : *users*) {  
 if (user.getName().equals(name)) {  
 userExists = true;  
 break;  
 }  
 }  
 if (userExists) {  
 clear();  
 update.setText("A profile with the name " + name + " already exists");  
 } else {  
 addUser(name);  
 }  
 }  
 } catch (Exception e) {  
 userName.setText("Error");  
 }  
  
  
}

A screenshot of a social media account

Description automatically generated

* Change Status



**To use the status selection feature:**

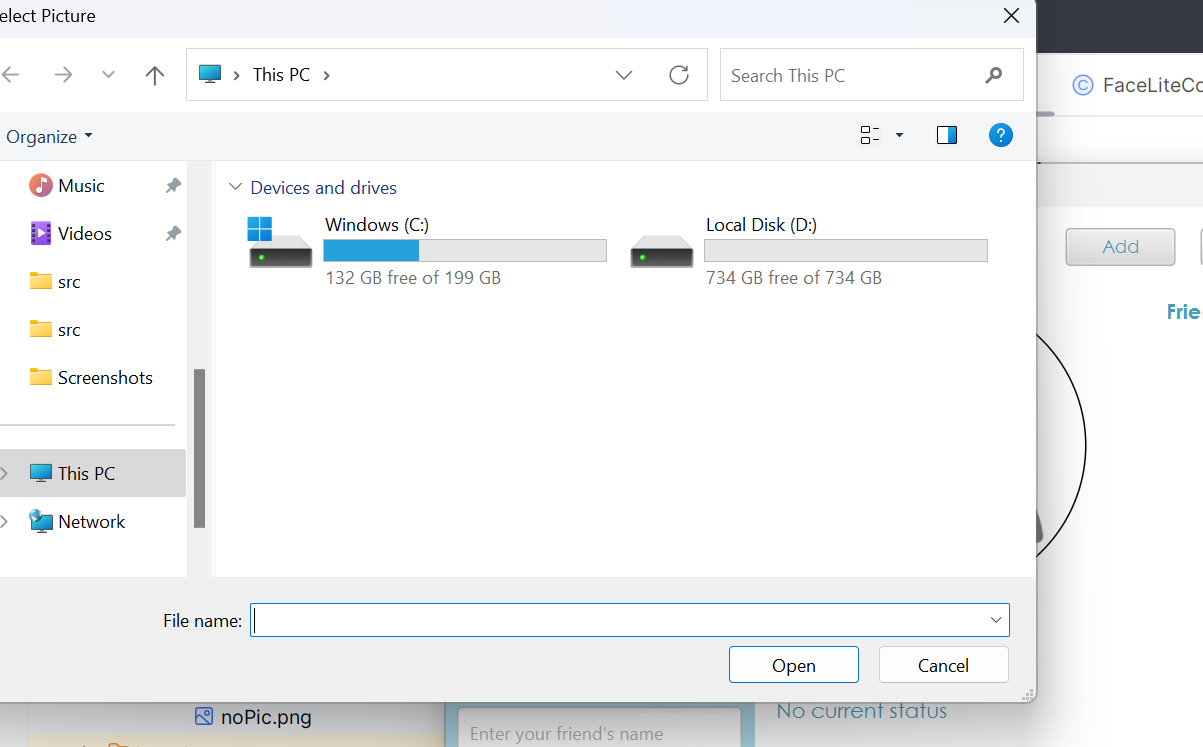
1.Enter the desired name for the new user in the changeStatusTextField.

2.Click the "Change Status" button

**The code of the change status:**

public void changeStatus() {  
 showLogo();  
 String name = nameTextField.getText();  
 if (!*users*.isEmpty()) {  
 String status = changeStatusTextField.getText();  
 if (!status.isEmpty()) {  
 *// Update the status and display a message* statusLabel.setText(status);  
 update.setText("Status updated to: " + status);  
 } else {  
 update.setText("Status updated to: " + status);  
 }  
 *// Find the user and update their status* for (User user : *users*) {  
 if (user.getName().equals(name)) {  
 user.setStatus(status);  
 break;  
 }  
 }  
 }  
}

* Change picture

A screenshot of a computer

Description automatically generated

**To use the "Change Picture" feature:**

1.Click on the "Change Picture" button

2.A file selection dialog titled "Select Picture" will appear.

3.Choose an image file from your local device.

4.Click "Open" or an equivalent button in the file selection dialog.

**The code of the change picture button:**

public void changePicture() {  
 showLogo();  
 try {  
 if (!*users*.isEmpty()) {  
 String name = nameTextField.getText();  
 FileChooser fileChooser = new FileChooser();  
 fileChooser.setTitle("Select Picture");  
 File selectedFile = fileChooser.showOpenDialog(null);  
  
 if (selectedFile != null) {  
 String imagePath = selectedFile.toURI().toString();  
 Image newImage = new Image(imagePath);  
 circle.setFill(new ImagePattern(newImage));  
 circle.setStroke(*BLACK*);  
 update.setText("Picture updated");  
  
 for (User user : *users*) {  
 if (user.getName().equals(name)) {  
 user.setPicturePath(imagePath);  
 break;  
 }  
 }  
 }  
 }  
 } catch (Exception ex) {  
 System.*out*.println(ex.getMessage());

* Add friend button

A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generated

**To use the "Add Friend" button:**

1.Enter the name of the friend you want to add in the addFriendTextField.

2.Click the "Add Friend" button.

**The code of the add friend button:**

public void addFriend() {  
 showLogo();  
 String friendName = addFriendTextField.getText();  
 if (!*users*.isEmpty()) {  
 if (!friendName.isEmpty()) {  
 if (friendName.equals(userName.getText())) {  
 update.setText("You cannot add your profile as a friend");  
 } else {  
 boolean profileExists = false;  
 User friendProfile = null;  
  
  
 *// Check if the friend's profile exists```java* for (User user : *users*) {  
 if (friendName.equals(user.getName())) {  
 profileExists = true;  
 friendProfile = user;  
 break;  
 }  
 }  
 if (profileExists) {  
 *// Check if the friend is already in the current user's friend list* boolean alreadyFriends = false;  
 for (String friend : friendProfile.getFriend()) {  
 if (friend.equals(userName.getText())) {  
 alreadyFriends = true;  
 break;  
 }  
 }  
 if (alreadyFriends) {  
 update.setText("You are already friends with " + friendName);  
 } else {  
 update.setText(friendName + " added as a friend");  
 friendProfile.userAddFriend(userName.getText());  
 User currentUser = null;  
  
  
 for (User user : *users*) {  
 if (user.getName().equals(userName.getText())) {  
 currentUser = user;  
 break;  
 }  
 }  
 if (currentUser != null) {  
 currentUser.userAddFriend(friendName);  
 }  
 }  
 } else {  
 update.setText("A profile with the name " + friendName + " does not exist");  
 }  
 }  
 friendList.getChildren().clear();  
 User currentUser = null;  
 for (User user : *users*) {  
 if (user.getName().equals(userName.getText())) {  
 currentUser = user;  
 break;  
 }  
 }  
 if (currentUser != null) {  
 for (String friend : currentUser.getFriend()) {  
 Label friendLabel = new Label();  
 friendLabel.setText(friend);  
 friendList.getChildren().add(friendLabel);  
 }  
 }  
  
 }  
  
 }  
  
}

* A screenshot of a computer

  Description automatically generatedLookup button

**To use the "Lookup" button:**

1.Enter the name of the user you want to lookup in the nameTextField.

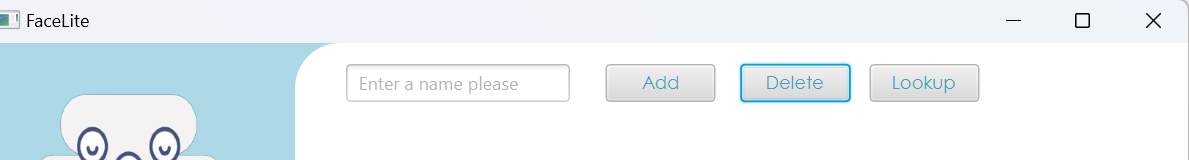
2.Click the "Lookup" button.

**The code of the lookup button:**

public void lookup() {  
 showLogo();  
 try {  
 if (!*users*.isEmpty()) {  
 friendList.getChildren().clear();  
 String name = nameTextField.getText();  
 if (!*users*.isEmpty()) {  
 for (User user : *users*) {  
 if (user.getName().equals(name)) {  
 userName.setText(name);  
 friendsLabel.setText("Friends: ");  
 statusLabel.setText(user.getStatus());  
 update.setText("Displaying " + name);  
 Image newImage = new Image(user.getPicturePath());  
 circle.setFill(new ImagePattern(newImage));  
 circle.setStroke(*BLACK*);  
 for (String friend : user.getFriend()) {  
 Label friendLabel = new Label();  
 friendLabel.setText(friend);  
 friendList.getChildren().add(friendLabel);  
 }  
 break;  
 } else {  
 clear();  
 update.setText("A profile with the name " + name + " does not exist");  
 }  
 }  
  
 } else {  
 clear();  
 update.setText("A profile with the name " + name + " does not exist");  
 }  
 }  
  
 } catch (Exception e) {  
 userName.setText("Error");  
 }

A screenshot of a computer

Description automatically generated

* Delete button

**To use the "Delete" button:**

1.Enter your name in the nameTextField.

2.Click the "Delete" button.

**The code of the Delete button:**

public void delete() {  
 showLogo();  
 String name = nameTextField.getText();  
 if (!*users*.isEmpty()) {  
  
 Alert alert = new Alert(Alert.AlertType.*CONFIRMATION*);  
 alert.setTitle("Confirmation Dialog");  
 alert.setHeaderText("Delete Confirmation");  
 alert.setContentText("Are you sure you want to delete your profile?");  
 ButtonType confirmButton = new ButtonType("Confirm");  
 ButtonType cancelButton = new ButtonType("Cancel");  
 alert.getButtonTypes().setAll(confirmButton, cancelButton);  
  
 alert.showAndWait().ifPresent(response -> {  
 if (response == confirmButton) {  
 Iterator<User> iterator = *users*.iterator();  
 while (iterator.hasNext()) {  
 User user = iterator.next();  
 LinkedHashSet<String> friends = user.getFriend();  
 friends.remove(name);  
 if (user.getName().equals(name)) {  
 iterator.remove();  
 }  
 }  
  
 clear();  
 update.setText("Profile of " + name + " deleted");  
 }  
 });  
 }  
}