



Object Oriented Programming
Lab Assignment 3

Submitted to: Sir Shahid Bhatti

Submitted by: Sheheryar Tariq - SP24-BSE-112 and Khadija Noor
SP24-BSE-052

Project:

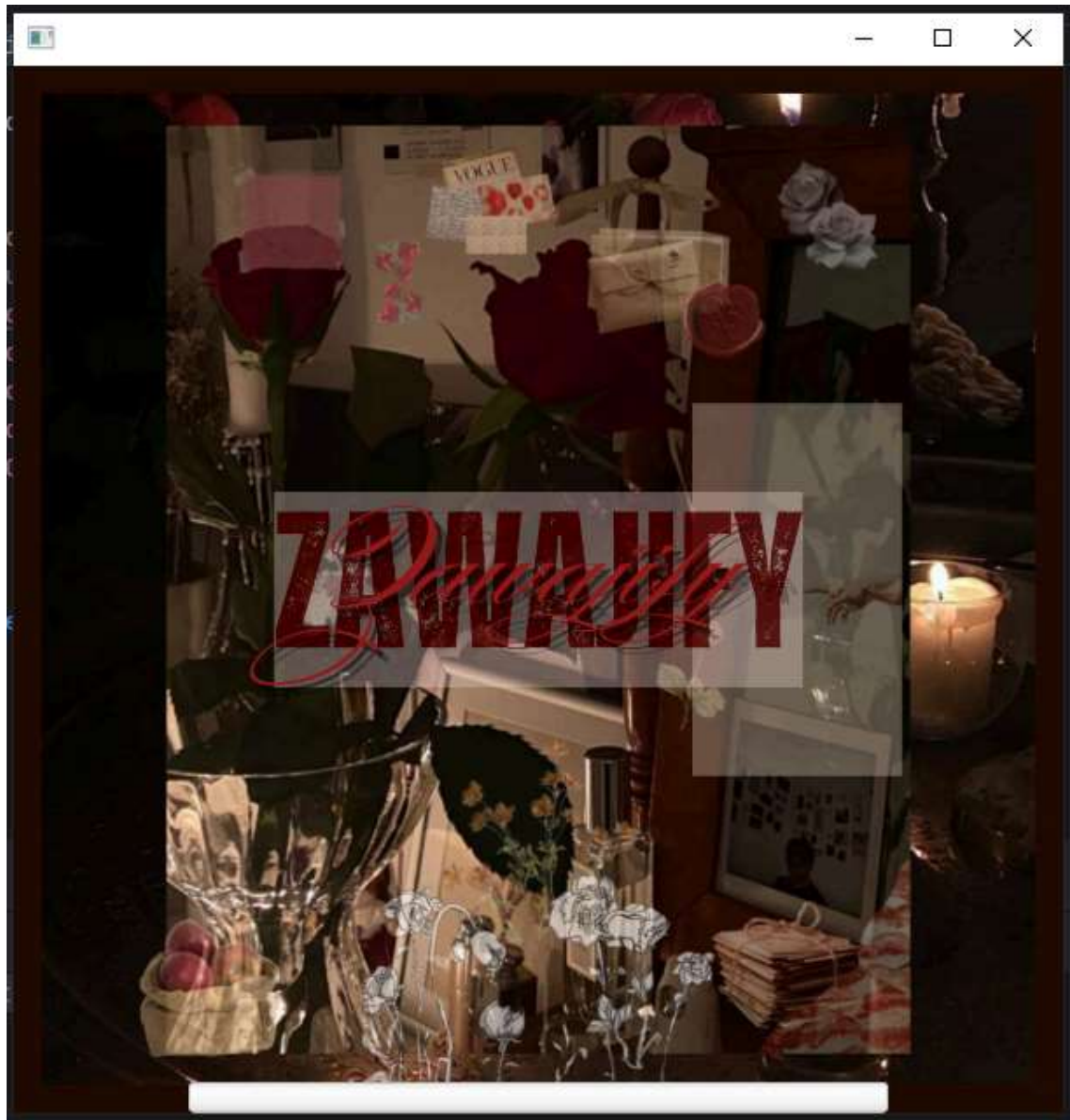
Marriage Application (Partially Completed)

Overview:

The application works in the following sequence. It has the following screens.

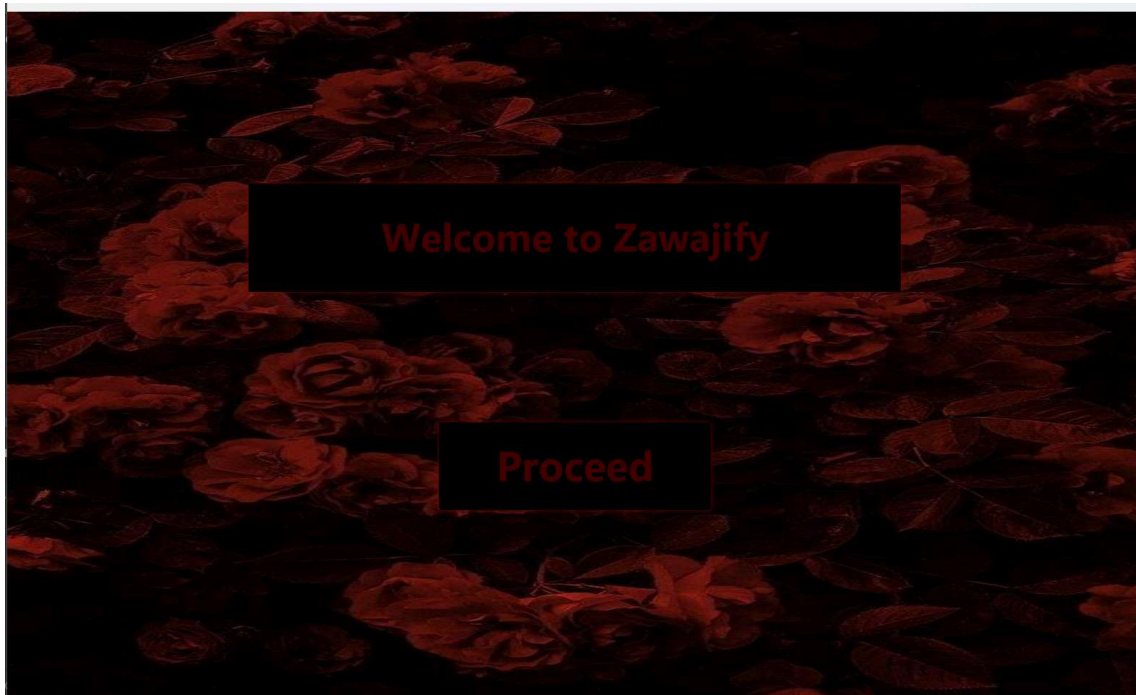
Screen no.1:

This screen stays for a few seconds for as long as the progress bar loads.



Screen no.2:

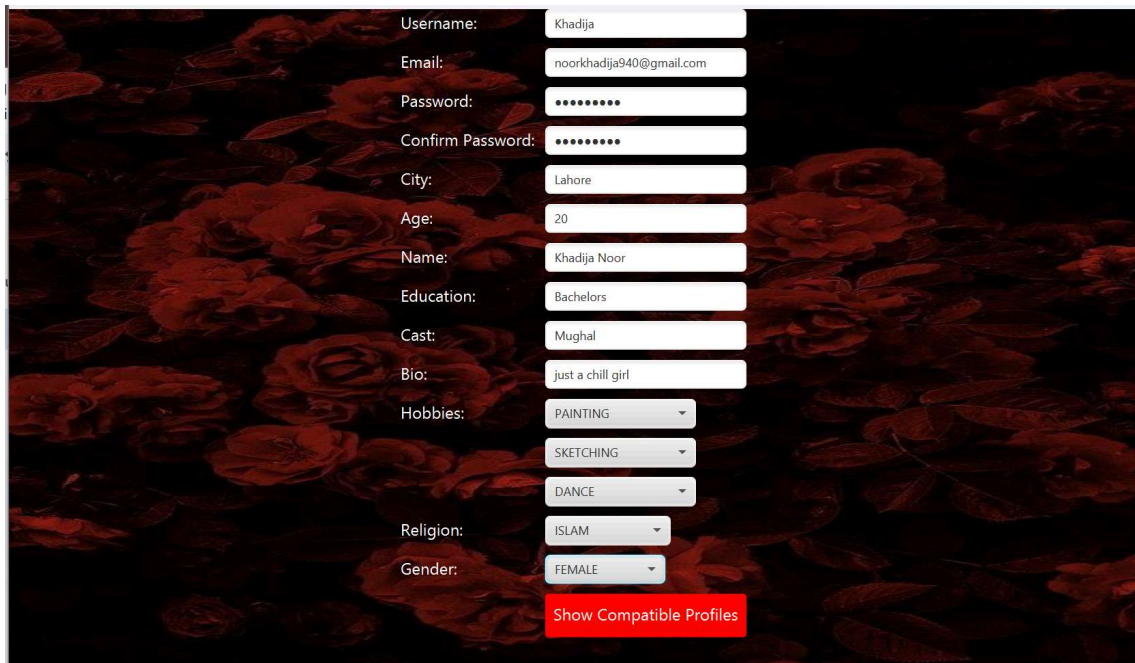
This screen has one button that takes you to the next screen when clicked upon.



Screen no.3:

This screen shows a menu with text input boxes. It allows the user to add their data in them with certain conditions e.g. using special characters in password, etc.

A screenshot of a mobile application interface showing a registration form. The background is a dark, textured pattern of red roses. The form consists of several input fields with labels to their left: "Username:" (a text box with a cursor), "Email:" (a text box with placeholder text "Enter your email"), "Password:" (a text box with placeholder text "Enter your password"), "Confirm Password:" (a text box with placeholder text "Re-enter your password"), "City:" (a text box with placeholder text "Enter your city"), "Age:" (a text box with placeholder text "Enter your age"), "Name:" (a text box with placeholder text "Enter your name"), "Education:" (a text box with placeholder text "Enter your education"), "Cast:" (a text box with placeholder text "Enter your cast"), "Bio:" (a text box with placeholder text "Tell us about yourself"), "Hobbies:" (three dropdown menus with placeholder text "Hobby 1", "Hobby 2", and "Hobby 3"), "Religion:" (a dropdown menu with placeholder text "Select religion"), and "Gender:" (a dropdown menu with placeholder text "Select gender"). At the bottom right of the form, there is a red button with the text "Show Compatible Profiles" in white.



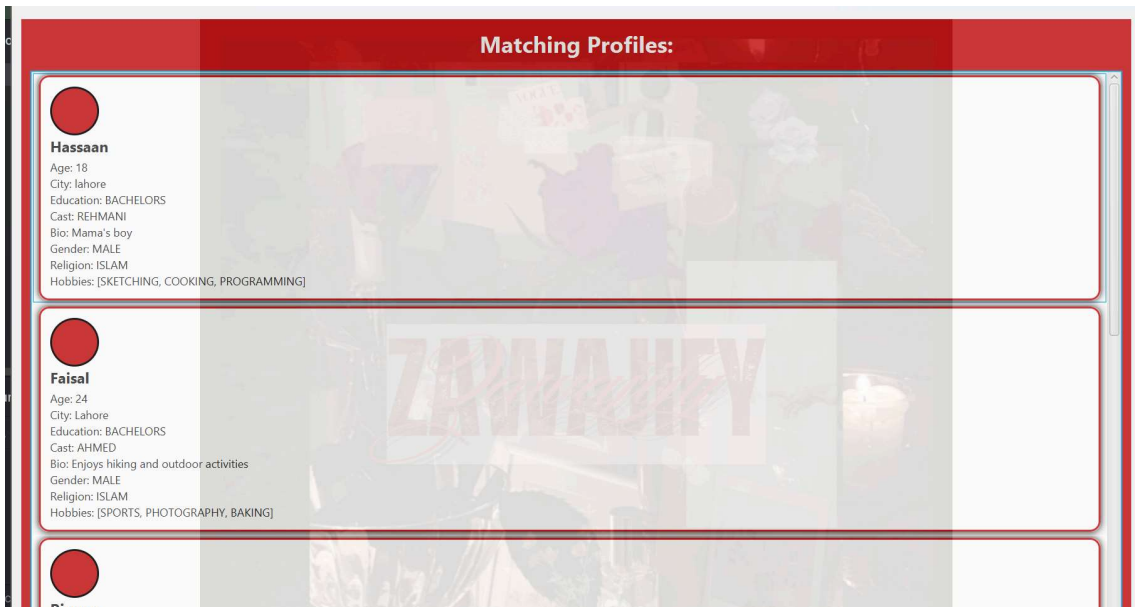
Registration form fields:

- Username: Khadija
- Email: noorkhadija940@gmail.com
- Password: [Redacted]
- Confirm Password: [Redacted]
- City: Lahore
- Age: 20
- Name: Khadija Noor
- Education: Bachelors
- Cast: Mughal
- Bio: just a chill girl
- Hobbies: PAINTING, SKETCHING, DANCE
- Religion: ISLAM
- Gender: FEMALE

[Show Compatible Profiles](#)

Screen no.4:

When clicked on the button ‘Show Compatible Profiles’, it shows a list of profiles that match with the details added.

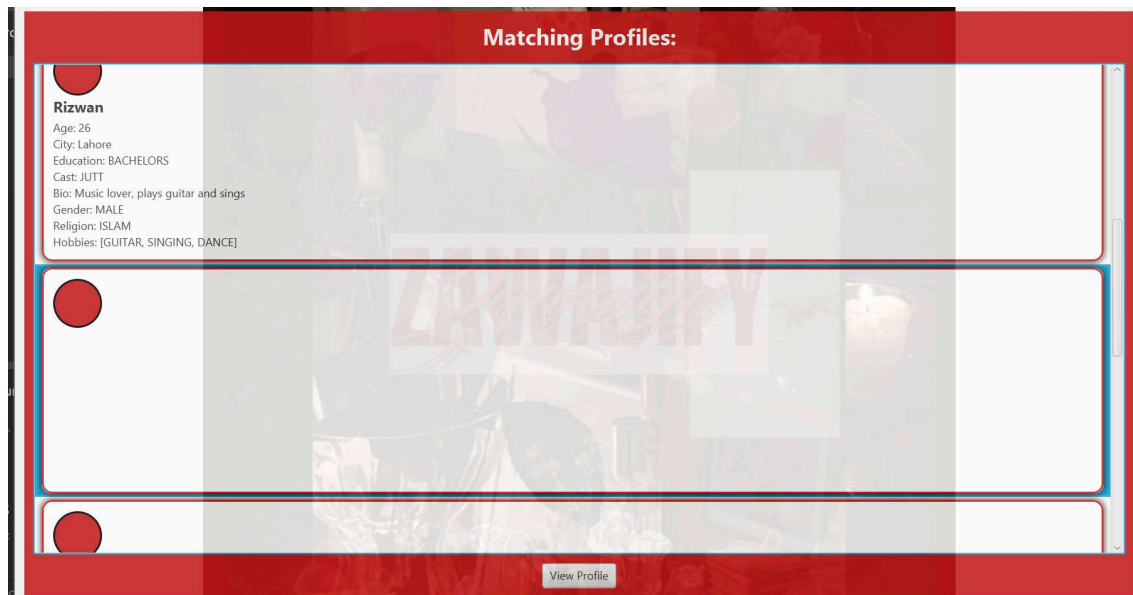


Matching Profiles:

- Hassaan**
Age: 18
City: Lahore
Education: BACHELORS
Cast: REHMANI
Bio: Mama's boy
Gender: MALE
Religion: ISLAM
Hobbies: [SKETCHING, COOKING, PROGRAMMING]
- Faisal**
Age: 24
City: Lahore
Education: BACHELORS
Cast: AHMED
Bio: Enjoys hiking and outdoor activities
Gender: MALE
Religion: ISLAM
Hobbies: [SPORTS, PHOTOGRAPHY, BAKING]
- Dimash**

Screen no.5:

When clicked on any profile, you can view it by then clicking on the ‘View Profile’ button.



Screen no. 6:

The profile details are shown. The back button takes you back to the previous screen of the list of profiles.

Name: Yasir

Age: 29

City: Lahore

Education: BACHELORS

Email: yasirraja13@gmail.com

Bio: Love for DIY projects and home decor

Hobbies: [DIYS, PAINTING, PHOTOGRAPHY]

Religion: ISLAM

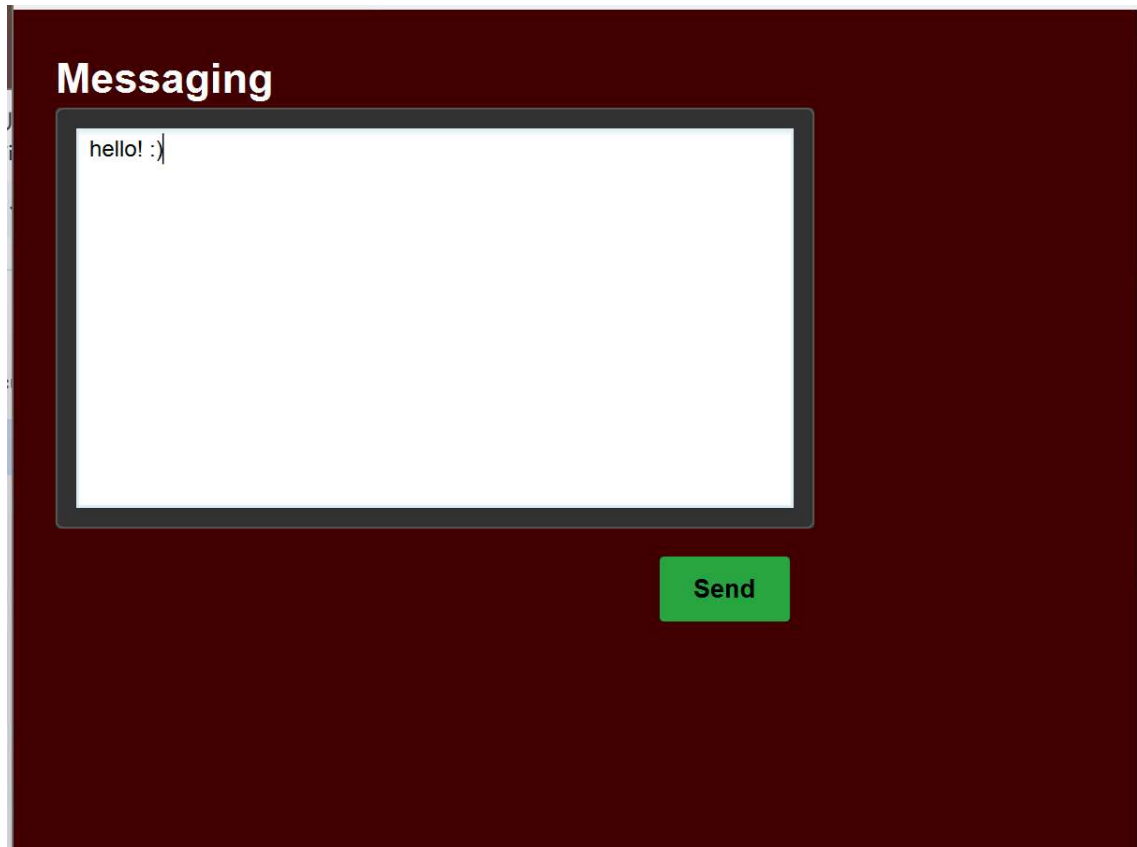
Gender: MALE

[Back](#)

[Message](#)

Messaging:

You can message the person as shown:



Code and functioning of the Screens:

Following are the code snippets and functioning of the screens.

HelloApplication Class:

This class basically starts the program and sets the stage of the application.


```

package com.example.connect;

import ...

public class HelloApplication extends Application {
    private Stage primaryStage; no usages
    private Scene startupScene, intermediateScene, signupScene, matchingProfilesScene, detailedProfileScene; no usages
    private ArrayList<Profile> profiles; no usages

    @Override
    public void start(Stage stage) throws IOException {
        try {
            FXMLLoader loader = new FXMLLoader(HelloApplication.class.getResource( name: "startupScene.fxml"));
            Scene scene = new Scene(loader.load());
            stage.setScene(scene);
            stage.show();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) { launch(args); }
}

```

IntermediateController Class:

This class creates a link between the GUI and the functioning code.

It loads the FXML files from the code and also inputs the font used in the application.

```

package com.example.connect;

import ...

public class IntermediateController {

    @FXML
    private Label welcomeLabel; // Add this declaration to link to the FXML's fx:id

    @FXML
    void handleProceed(ActionEvent event) {
        try {
            FXMLLoader loader = new FXMLLoader(getClass().getResource( name: "signupScene.fxml"));
            Parent root = loader.load();
            Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
            stage.setScene(new Scene(root));
            stage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void initialize() {
        // Load the Riffon font
        Font riffonFont = Font.loadFont(getClass().getResourceAsStream( name: "/com/example/connect/riffon-regular.otf"), 24);
        if (riffonFont != null) {
            welcomeLabel.setFont(riffonFont); // Apply the font
        } else {
            System.out.println("Font could not be loaded!");
        }
    }
}

```

SignupUser Class:

Takes input from the user.

```
4
5 public class Signup_user {
6     private static String city; 2 usages
7     private static int age; 2 usages
8     private static String name; 2 usages
9     private static String education; 2 usages
10    private static String cast; 2 usages
11    private static String bio; 2 usages
12    private static ArrayList<Hobbies> hobby = new ArrayList<>(); 2 usages
13    private static Religion religion; 2 usages
14    private static Gender gender; 2 usages
15    private static int userId; 2 usages
16    private static String username; 2 usages
17    private static String email; 2 usages
18    private static String password; 2 usages
19 }
```

SignController Class:

Stores data.

```
public class SignController {
    void handleSignup(ActionEvent event) {

        // Create a profile for the student
        p1 = new Profile(
            Signup_user.getUsername(), Signup_user.getEmail(), Signup_user.getPassword(),
            Signup_user.getCity(), Signup_user.getAge(), Signup_user.getName(),
            Signup_user.getEducation(), Signup_user.getCast(), Signup_user.getBio(),
            Signup_user.getHobby(), Signup_user.getReligion(), Signup_user.getGender()
        );
    }
}
```

Match Class:

Checks for matches.

```

import java.util.*;

public class Match {

    private int[] compatibility; // religion=50, cast=20, city=5, age difference=5, hobbies=15 8usages

    public Match(ArrayList<Profile> profiles, Profile p1) { 1usage
        compatibility = new int[profiles.size()];
        matchHobbies(profiles, p1);
        matchCast(profiles, p1);
        matchCity(profiles, p1);
        matchAgeDifference(profiles, p1);
        matchReligion(profiles, p1);
    }

    public void matchHobbies(ArrayList<Profile> profiles, Profile p1) { 1usage
        for (int i = 0; i < profiles.size(); i++) {
            Profile p = profiles.get(i);
            for (Hobbies hobby : p1.getHobby()) {
                if (p.getHobby().contains(hobby)) {
                    compatibility[i] += 5;
                    break; // Break the loop once a match is found to avoid duplicate increments
                }
            }
        }
    }
}

```

FXML Files:

They run the gui.

Messaging Class:

```
2
3 import javafx.fxml.FXML;
4 import javafx.scene.control.TextArea;
5
6 public class MessagingController {
7
8     @FXML
9     private TextArea messageTextArea;
10
11     private Profile recipientProfile; 2 usages
12
13     public void setRecipientProfile(Profile profile) { 1 usage
14         this.recipientProfile = profile;
15     }
16
17     @FXML 1 usage
18     public void handleSendMessage() {
19         String message = messageTextArea.getText();
20         if (!message.isEmpty()) {
21             System.out.println("Message sent to " + recipientProfile.getName() + ": " + message);
22             messageTextArea.clear();
23         } else {
24             System.out.println("Message is empty!");
25         }
26 }
```