

**FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY**

**PROJECT SSK 3101**

**SEMESTER II 2017/2018**

**LECTURER:**

**DR. NG KENG YAP**

**TITLE :**

**MEMBERSHIP MANAGEMENT**

**GROUP MEMERS :**

|  |  |  |
| --- | --- | --- |
| **NO.** | **NAME** | **MATRIC NUMBER** |
| 1. | Muhammad Afif Mirza bin Mohd Farid | 192128 |
| 2. | Muhammad Harith bin Zainudin | 192171 |
| 3. | Nadirah binti Kamaruddin | 194096 |
| 4. | Tean Yew Cheean | 194698 |
| 5. | Muhammad Haris bin Harun | 194701 |

**INTRODUCTION**

Membership Management is a registration program that used to register new member for an organization, Social Welfare Club for Faculty of Computer Science and Information Technology. This program required the user to fill out their personal information and job information. It also enables the user to view registered members in the organization and update their personal details such as status, add the family members and so on. This program is really helpful for the organization to keep track of their members so that they could organize any events smoothly in the future.

**PROJECT DESCRIPTION**

* Membership Management program has been build for the user to register into the organization easily. In addition, the system enables each member of the organization view other members of the organization, upload profile photo, update their personal details and job information.
* Problem Statements :
* How to register into the organization without having to submit the registration form personally to the president of the organization?
* Does the program enable us to update our own personal details?
* Are we able to view the members of the organization?
* Objectives :
* To view members of the organization.
* To help the user to register into the organization easily.
* To get personal information and job information of each member of the organization.

**FUNCTIONALITIES**

Membership Management required the user to enter the password to enter the system. If the user is a new member, they have to use default password to enter the system. After they enter the system, they have to register as a new member first before proceed to other actions (change password,view all members). For the registration, the user have to set new password before the system prompt the user to enter their honorary title, full name, identity card number, gender, date of birth, place of birth, religion, race, home address, district, postcode, state, country, marital status and citizenship for personal information (Part A) while the system prompt user to enter staff number, division, email address, mobile phone number, office phone number, office address, district, postcode, state and country for job information (Part B). In order to apply a good interface, we use application GUI to collect the data. Then, we apply object-oriented and exception in processing the information needed. After that, the user has to upload profile picture into the system to complete the registration process. In addition, all registered members are able to change their password and view members of the organization. But, if they want to view a specific person in the organization, they have to type the name of the specific person on the search engine while the system will searching after the search button is clicked.

Inputs or information needed from the users:

* Part B (Job Information)
* Staff Number
* Division
* Email Address
* Mobile Phone Number
* Office Phone Number
* Office Address ( District, Postcode

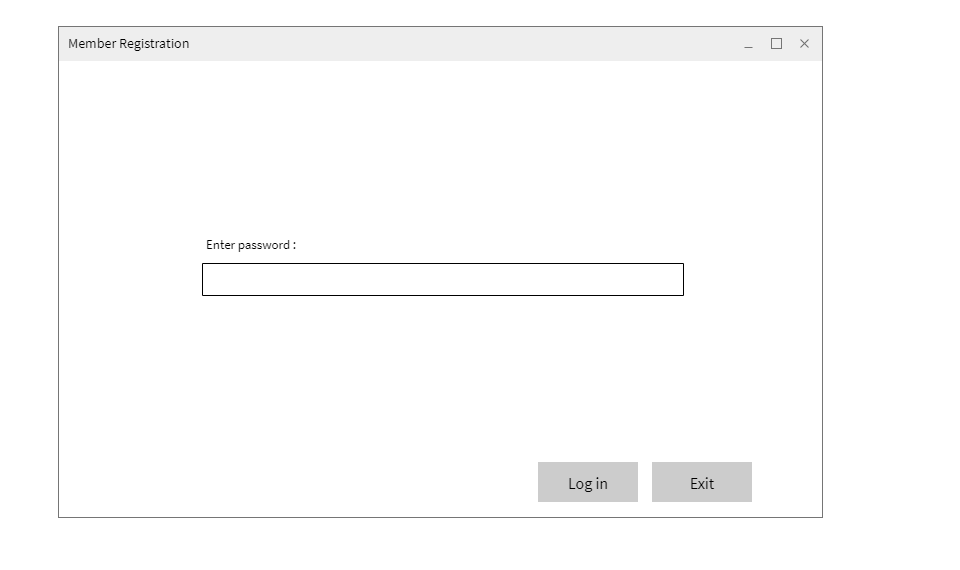
State, Country )

* Part A (Personal Information)
* Honorary Title
* Full Name
* Identity Card Number
* Gender
* Date of Birth
* Place of Birth
* Religion
* Race
* Home Address ( District, Postcode,

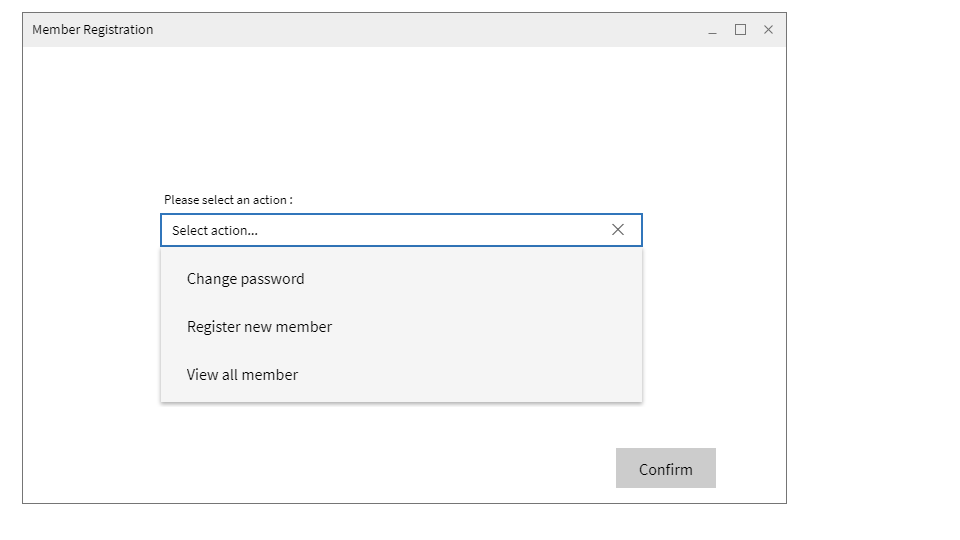
State, Country )

* Marital Status
* Citizenship

**USER INTERFACE**



*Figure 1 : Login*

*Figure 2 : Select an action (change password, register new member, view all member)*

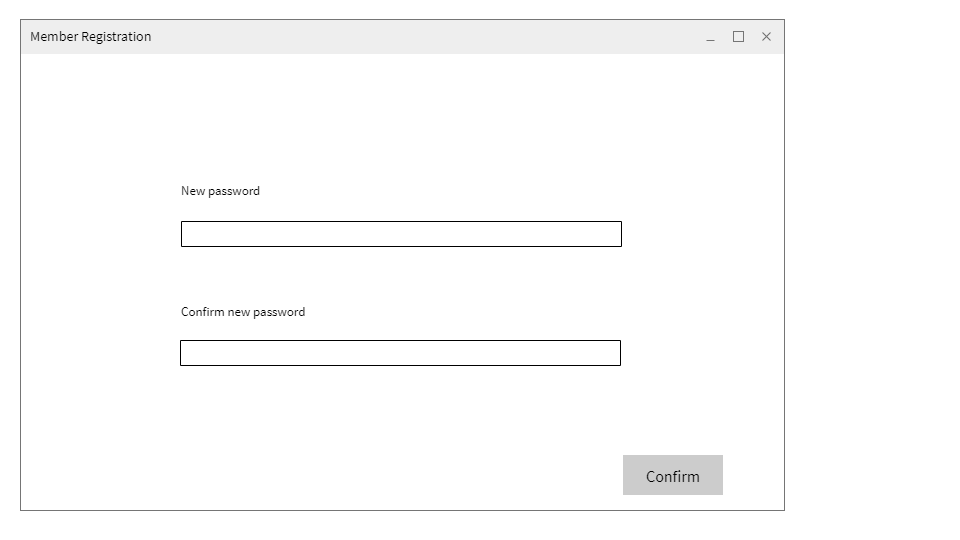
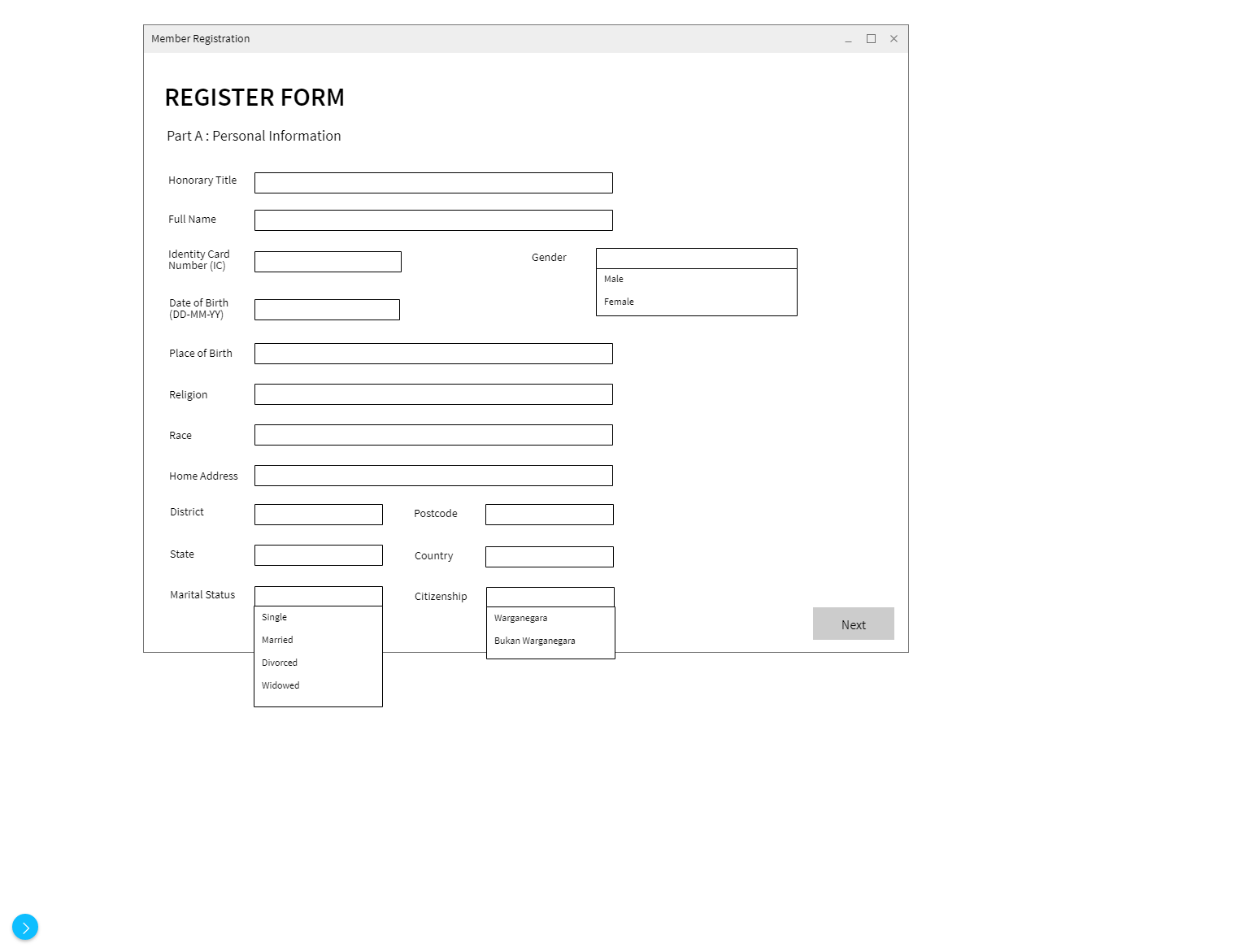
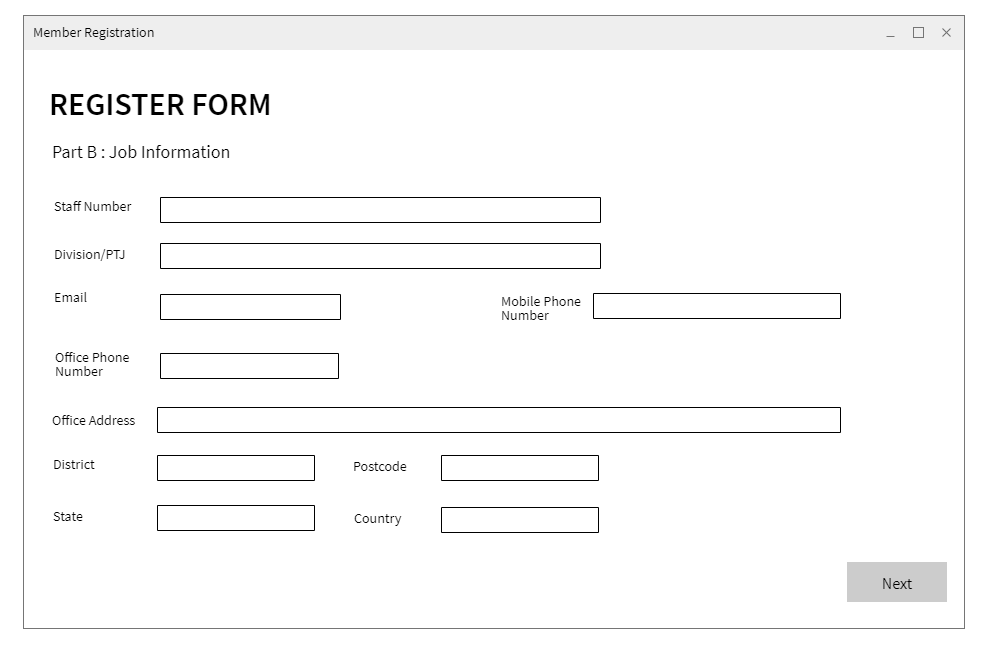
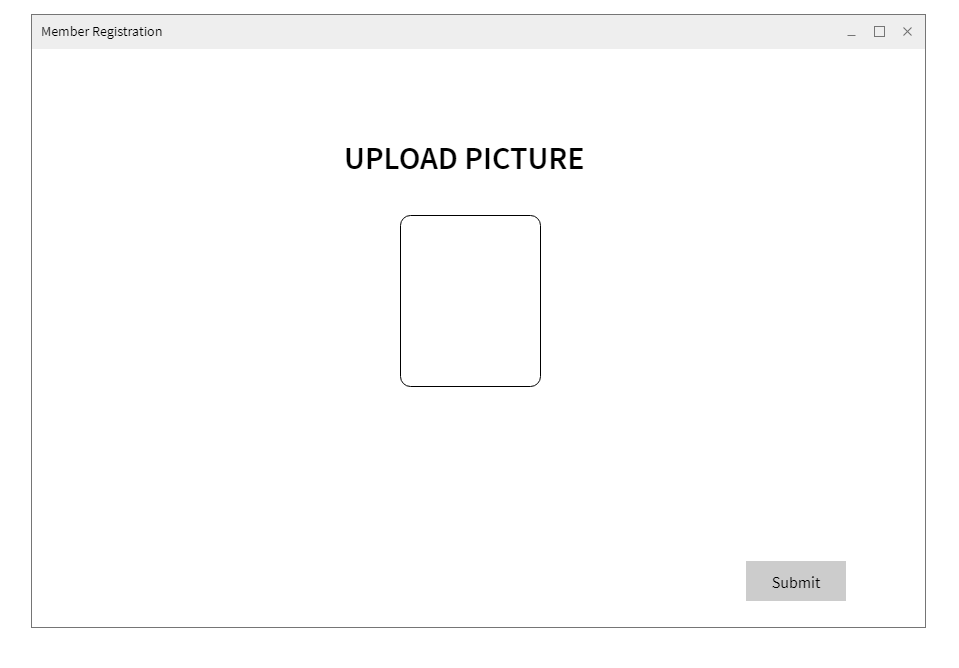


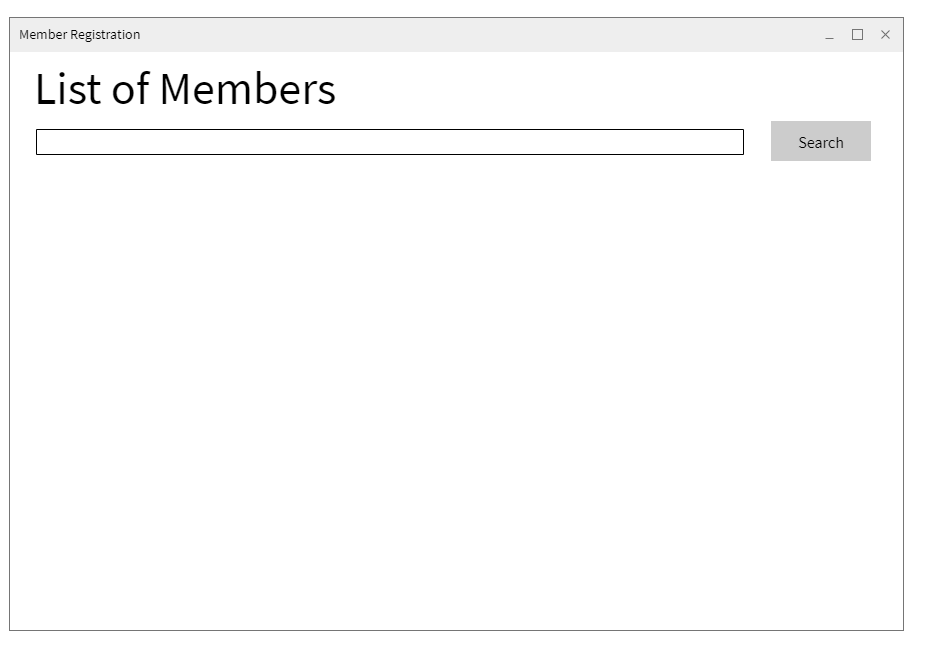
Figure 3 : Change password



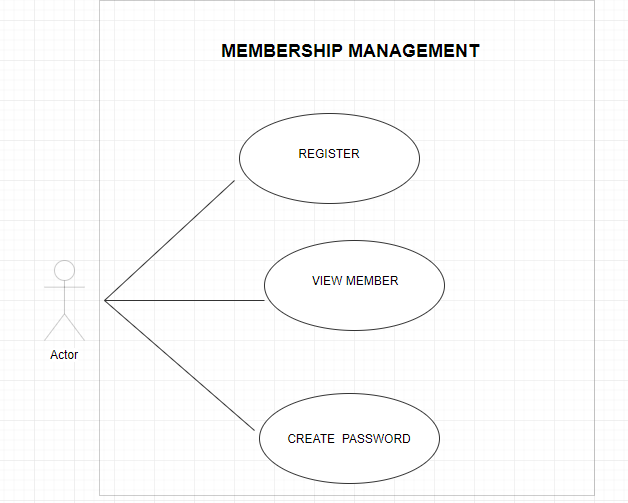
*Figure 4 : Register Form ( Part A : Personal Information )*

*Figure 5 : Register Form ( Job Information )*

*Figure 6 : Upload Profile Picture*

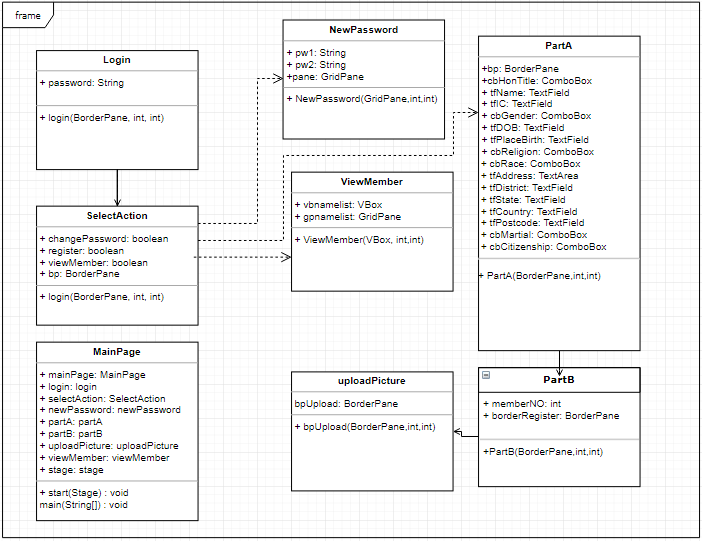
*Figure 7 : List of members / View member*

**USE CASE**

****

*Figure 8 : Use Case for Membership Management*

**CLASS DIAGRAM**

****

**SOURCE CODE**

LOGIN

/\*\*

\* This is a classthe login page for the system.

\* If the user enter using the default password, it will bring the user to the select action scene.

\* If wrong password is enter, the user cannot login.

\*/

package project;

import javax.swing.JOptionPane;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

public class Login extends Scene {

static BorderPane borderP1 = new BorderPane();

static String password = new String();

public Login(BorderPane borderP1, int a, int b) {

super(borderP1, a, b);

this.borderP1 = borderP1;

VBox vbox = new VBox(15); // create vertical box for top pane of border pane and 15 is the gap between

HBox horBox1 = new HBox(15); // create horizontal box and also the text field

// BorderPane borderP1 = new BorderPane();

borderP1.setTop(vbox);

vbox.getChildren().add(new javafx.scene.control.Label("Enter Password"));

horBox1.setAlignment(Pos.CENTER);// Setting the allignment of the Billing Account at the centre

vbox.setAlignment(Pos.CENTER);

// create new label

Label Message = new Label();

// set the initial password

password = "1234";

// Created a grid pane to enter the textfield and all the label

GridPane gridP1 = new GridPane();

gridP1.setAlignment(Pos.CENTER); // allow textfield and labels to be at the center

gridP1.setVgap(15);// Allow the textfield to have gap vertically

gridP1.setHgap(30); // Allow the textfield to have gap with the label horizontally

gridP1.add(Message, 1, 1, 3, 1);

// Inserting the textfield to allow user to enter the password (pw1 stands for

// password)

PasswordField tfPassword = new PasswordField();

tfPassword.setPrefColumnCount(10);

// Adding the password in the vbox - vertical!

vbox.getChildren().addAll(tfPassword);// adding all the text and textfield in horbox

vbox.setPadding(new Insets(15, 15, 15, 15)); // padding to make sure gap between the textfield and the box

// Set the grid at the centre of the border

borderP1.setCenter(gridP1);

HBox hbox = new HBox(15); // Making space horizontally between process and exits buttons

Button btExit = new Button("Exit"); // Create the button exit

Button btLogin = new Button("Login"); // Create the button login

horBox1.getChildren().addAll(btLogin, btExit);// Adding the button into the hbox

borderP1.setBottom(horBox1); // Setting both of the exits and process buttons at the bottom of the pane

horBox1.setAlignment(Pos.CENTER); // Setting the buttons so that it will be at the centre of the pane

horBox1.setPadding(new javafx.geometry.Insets(0, 0, 15, 0)); // setting the buttons 15 distance from the bottom

btExit.setOnAction((ActionEvent event) -> {

// once click, the method for button exit

System.exit(0);

});

btLogin.setOnAction(new EventHandler<ActionEvent>() {

public void handle(ActionEvent event) {

Message.setStyle("-fx-text-fill: red;");

String password1 = tfPassword.getText();

// set the rules fo user to enter password so that it will not empty

if (password1.equals("")) {

Message.setText("Please enter the password");

} else if (password1.equals(password)) {

Message.setStyle("-fx-text-fill: green;");

password = NewPassword.pw1;

JOptionPane.showMessageDialog(null, "Successfully login.", null, JOptionPane.INFORMATION\_MESSAGE);

MainPage.stage.setScene(MainPage.selectAction);

} else {

Message.setStyle("-fx-text-fill: red;");

Message.setText("Incorrect password");

}

}

});

}

} // end of class login

MAIN PAGE

/\*\*

\* This is the class where the main stage is.

\*/

package project;

import java.io.IOException;

import javafx.application.Application;

import javafx.stage.Stage;

public class MainPage extends Application {

static MainPage mainPage = new MainPage();

static Login login = new Login(Login.borderP1, 300, 200);

static SelectAction selectAction = new SelectAction(SelectAction.bp, 800, 600);

static NewPassword newPassword = new NewPassword(NewPassword.pane, 800, 600);

static PartA partA = new PartA(PartA.bp, 850, 900);

static PartB partB = new PartB(PartB.borderRegister, 1000, 600);

static UploadPicture uploadPicture = new UploadPicture(UploadPicture.bpUpload, 1000, 600);

static ViewMember viewMember = new ViewMember(ViewMember.vbnamelist, 1000, 600);

static Stage stage = new Stage();

public void start(Stage primaryStage) {

// set the title of the stage, scene and show the stage

stage.setTitle("Membership Management");

stage.setScene(login);

stage.show();

}

// main method

public static void main(String[] args) throws IOException {

launch(args);

}

} // end of class MainPage

NEW PASSWORD

/\*\*

\* This is a class for Scene 3 as known as the scene for the user to enter the new password

\* The password will be updated if the user enter the new password

\* After the user have enter the new password, it will bring back the user to the select action scene.

\*/

package project;

import static javafx.application.Application.launch;

import javax.swing.JOptionPane;

import javafx.geometry.HPos;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.stage.Stage;

public class NewPassword extends Scene {

static GridPane pane = new GridPane();

static String pw1 = new String();

public NewPassword(GridPane pane, int a, int b) {

super(pane, a, b);

this.pane = pane;

// Create a pane and set its properties

pane.setAlignment(Pos.CENTER);

pane.setPadding(new Insets(20, 20, 20, 20));

pane.setHgap(10);

pane.setVgap(10);

// create message label

Label Message = new Label();

// Place nodes in the pane

pane.add(new Label("New password:"), 0, 0);

PasswordField pf = new PasswordField();

pane.add(pf, 0, 1);

pane.add(new Label("Confirm new password:"), 0, 3);

PasswordField pf1 = new PasswordField();

pane.add(pf1, 0, 4);

pane.add(Message, 1, 1, 3, 1);

Button btAdd = new Button("Confirm");

pane.add(btAdd, 1, 6);

GridPane.setHalignment(btAdd, HPos.RIGHT);

// set action for button

btAdd.setOnAction(e -> {

String pw1 = pf.getText();

String pw2 = pf1.getText();

if (pw1.equals(pw2)) {

pw1 = Login.password;

JOptionPane.showMessageDialog(null, "Password has been changed.", null,

JOptionPane.INFORMATION\_MESSAGE);

MainPage.stage.setScene(MainPage.selectAction);

} else

Message.setText("Password not match");

});

}

} // end of class new password

FORM PART A : PERSONAL INFORMATION

/\*\*

\* This is a class for scene 4 as known as the class for the user to enter the new member info for part A

\* part A is the personal information form that the user have to enter

\* If all the part have been enter, the user have to click next to proceed to the next from

\*/

package project;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.control.ScrollBar;

import javafx.scene.control.ScrollPane;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import javafx.scene.control.TextFormatter;

import javafx.scene.layout.\*;

import javafx.scene.text.\*;

class PartA extends Scene {

static BorderPane bp = new BorderPane();

static ComboBox cbHonTitle = new ComboBox();

static TextField tfName = new TextField();

static ComboBox cbGender = new ComboBox();

static TextField tfIC = new TextField();

static TextField tfDOB = new TextField();

static TextField tfPlaceBirth = new TextField();

static ComboBox cbReligion = new ComboBox();

static ComboBox cbRace = new ComboBox();

static TextField tfCountry = new TextField();

static ComboBox cbMarital = new ComboBox();

static ComboBox cbCitizenship = new ComboBox();

static TextArea tfAddress = new TextArea();

static TextField tfDistrict = new TextField();

static TextField tfPostcode = new TextField();

static TextField tfState = new TextField();

public PartA(BorderPane bp, int a, int b) {

super(bp, a, b);

// change the font and all the characteristics of the title form

Label labelTitle = new Label("REGISTER FORM");

labelTitle.setFont(Font.font("Verdana", FontWeight.BOLD, 30));

Label labelPart = new Label("Part A : Personal Information");

// put labelTitle and labelPart in horizontal box and set all the

// characteristics

VBox vb1 = new VBox(10);

vb1.getChildren().addAll(labelTitle, labelPart);

// set vb1 to the top of the border pane

bp.setTop(vb1);

// create gridpane to hold all the label,textfield and combo box into place, and

// set the characteristics

GridPane gp = new GridPane();

gp.setPadding(new Insets(20, 0, 0, 0));

gp.setHgap(10);

gp.setVgap(15);

// create all label,textfield and combo box, then place it in the gridpane and

// also set the characteristics

Label labelHonTitle = new Label("Honorary Title");

cbHonTitle.getItems().addAll("MR.", "MRS.");

gp.add(labelHonTitle, 0, 0);

gp.add(cbHonTitle, 1, 0);

cbHonTitle.setMinWidth(150);

// create the label for Full name and set it all to uppercase

Label labelFName = new Label("Full Name");

gp.add(labelFName, 0, 1);

gp.add(tfName, 1, 1);

tfName.setMinWidth(400);

tfName.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

// create label for IC and limit it to only have 12 inputs, and cannot input

// characters

Label labelIC = new Label("Identity Card Number (IC)");

gp.add(labelIC, 0, 3);

gp.add(tfIC, 1, 3);

tfIC.setMaxWidth(150);

tfIC.textProperty().addListener(new ChangeListener<String>() {

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if (!newValue.matches("\\d{0,12}?")) {

tfIC.setText(oldValue);

}

}

});

// create label for gender and set the width of the combo box

Label labelGender = new Label("Gender");

cbGender.getItems().addAll("MALE", "FEMALE", "OTHERS");

gp.add(labelGender, 0, 2);

gp.add(cbGender, 1, 2);

cbGender.setMinWidth(150);

// create label for DOB and limit it to input only 6 numbers at maximum and

// cannot enter characters

Label labelDOB = new Label("Date of Birth (DD-MM-YY)");

gp.add(labelDOB, 0, 4);

gp.add(tfDOB, 1, 4);

tfDOB.setMaxWidth(150);

tfDOB.textProperty().addListener(new ChangeListener<String>() {

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if (!newValue.matches("\\d{0,6}?")) {

tfDOB.setText(oldValue);

}

}

});

// create label for place of birth and set it all to uppercase

Label labelPlaceBirth = new Label("Place of Birth");

gp.add(labelPlaceBirth, 0, 5);

gp.add(tfPlaceBirth, 1, 5);

tfPlaceBirth.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

// create label for religion

Label labelReligion = new Label("Religion");

cbReligion.getItems().addAll("ISLAM", "CHRISTIAN", "BUDDHA", "HINDHU", "OTHERS");

gp.add(labelReligion, 0, 6);

gp.add(cbReligion, 1, 6);

cbReligion.setMinWidth(150);

// create label for race

Label labelRace = new Label("Race");

cbRace.getItems().addAll("MALAY", "CHINESE", "INDIAN", "SIKH", "OTHERS");

gp.add(labelRace, 0, 7);

gp.add(cbRace, 1, 7);

cbRace.setMinWidth(150);

// create label for home address and set it input to all uppercase

Label labelAddress = new Label("Home Address");

gp.add(labelAddress, 0, 8);

gp.add(tfAddress, 1, 8);

tfAddress.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

// create label for district and also the textfield. Set also the

// characteristics that is Uppercase for all lettler

Label labelDistrict = new Label("District");

gp.add(labelDistrict, 0, 9);

gp.add(tfDistrict, 1, 9);

tfDistrict.setMaxWidth(150);

tfDistrict.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

// create label and textfield for postcode. Limit the user for entering the

// input that is only 5

Label labelPostcode = new Label("Postcode");

gp.add(labelPostcode, 0, 10);

gp.add(tfPostcode, 1, 10);

tfPostcode.setMaxWidth(150);

tfPostcode.textProperty().addListener(new ChangeListener<String>() {

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if (!newValue.matches("\\d{0,5}?")) {

tfPostcode.setText(oldValue);

}

}

});

// create label and textfield for state. Change all the letters to uppercase

// when user enter the input

Label labelState = new Label("State");

gp.add(labelState, 0, 11);

gp.add(tfState, 1, 11);

tfState.setMaxWidth(150);

tfState.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

// create label and textfield for country. change the input that user enter to

// uppercase

Label labelCountry = new Label("Country");

gp.add(labelCountry, 0, 12);

gp.add(tfCountry, 1, 12);

tfCountry.setMaxWidth(150);

tfCountry.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

// create label and combo box for marital status. set the minimum width of the

// combo box

Label labelMarital = new Label("Marital Status");

cbMarital.getItems().addAll("SINGLE", "MARRIED", "DIVORCED", "WIDOWED", "OTHERS");

gp.add(labelMarital, 0, 13);

gp.add(cbMarital, 1, 13);

cbMarital.setMinWidth(150);

// create label and combobox for citizenship and set the minimum width

Label labelCitizenship = new Label("Citizenship");

cbCitizenship.getItems().addAll("CITIZEN", "NON-CITIZEN");

gp.add(labelCitizenship, 0, 14);

gp.add(cbCitizenship, 1, 14);

cbCitizenship.setMinWidth(150);

// create a next button and set it go to the next scene if click

Button btnNext = new Button("Next");

btnNext.setOnAction(e -> {

MainPage.stage.setScene(MainPage.partB);

});

// align the gridpane to it position

bp.setCenter(gp);

bp.setBottom(btnNext);

bp.setPadding(new Insets(20));

bp.setAlignment(btnNext, Pos.BOTTOM\_RIGHT);

}

} // end of class PartA

FORM PART B : JOB INFORMATION

/\*\*

\* This is a class for the form part B

\* This scene is from part A form

\* Part B is the form where the user have to enter the information regarding the work

\* The user also can go to previous part A form to edit back.

\* If the user click next, all the information will be saved and this will bring the user to the upload picture scene

\*

\*/

package project;

import java.io.File;

import java.io.\*;

import java.io.PrintWriter;

import javax.swing.JOptionPane;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import javafx.scene.control.TextFormatter;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.text.Text;

public class PartB extends Scene {

static int memberNO = 0;

static BorderPane borderRegister = new BorderPane();

public PartB(BorderPane borderRegister, int a, int b) {

super(borderRegister, a, b);

this.borderRegister = borderRegister;

VBox vbox = new VBox(); // create vertical box for top pane of border pane

HBox horBox1 = new HBox(); // create horizontal box for the patient name and also the text field

vbox.getChildren().addAll(new Label("Register Form"));

borderRegister.setTop(vbox);

horBox1.setAlignment(Pos.CENTER);// Setting the allignment of the grading activity at the centre

vbox.setAlignment(Pos.CENTER);

// Creating a grid pane to insert (LABEL,TEXTFIELD)

GridPane gridP1 = new GridPane();

gridP1.setAlignment(Pos.CENTER); // allow textfield and labels to be at the center

gridP1.setVgap(15);// Allow the textfield to have gap vertically

gridP1.setHgap(30); // Allow the textfield to have gap with the label horizontally

Text PtB = new Text(" Part B : Job Information : ");

horBox1.getChildren().addAll(PtB);// adding all the text and textfield in horbox

vbox.getChildren().add(horBox1);

// Creating the Label for the grammar, spelling, Correct Length and content

// attributes

Label L1 = new Label("Staff Number : ");

TextField tf1 = new TextField();

gridP1.add(L1, 0, 0);

gridP1.add(tf1, 1, 0);

// set all to uppercase when user enter input

tf1.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

Label L2 = new Label("Division/PTJ :");

TextField tf2 = new TextField();

gridP1.add(L2, 0, 1);

gridP1.add(tf2, 1, 1);

// set to all uppercase when user enter input

tf2.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

Label L3 = new Label("Email :");

TextField tf3 = new TextField();

gridP1.add(L3, 0, 2);

gridP1.add(tf3, 1, 2);

// set to all uppercase when user enter input

tf3.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

Label L4 = new Label("Office Phone Number : ");

TextField tf4 = new TextField();

gridP1.add(L4, 0, 3);

gridP1.add(tf4, 1, 3);

// set the limit that user can enter for office phone number

tf4.textProperty().addListener(new ChangeListener<String>() {

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if (!newValue.matches("\\d{0,10}?")) {

tf4.setText(oldValue);

}

}

});

Label L5 = new Label("Office Address : ");

TextField tf5 = new TextField();

TextField tf6 = new TextField();

gridP1.add(L5, 0, 4);

gridP1.add(tf5, 1, 4);

gridP1.add(tf6, 1, 5);

// set tf5 to use all uppercase when user enter input

tf5.setPrefColumnCount(20); // Setting the length for the length of the textfield

tf5.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

// set tf6 to uppercase

tf6.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

Label L7 = new Label("State : ");

TextField tf7 = new TextField();

gridP1.add(L7, 0, 6);

gridP1.add(tf7, 1, 6);

// set tf7 to uppercase

tf7.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

Label L8 = new Label("Mobile Phone Number :");

TextField tf8 = new TextField();

gridP1.add(L8, 2, 2);

gridP1.add(tf8, 3, 2);

// set the limit for the user to enter for mobile phone number

tf8.textProperty().addListener(new ChangeListener<String>() {

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if (!newValue.matches("\\d{0,11}?")) {

tf8.setText(oldValue);

}

}

});

Label L9 = new Label("Postcode : ");

TextField tf9 = new TextField();

gridP1.add(L9, 2, 6);

gridP1.add(tf9, 3, 6);

// set the limit for the postcode that is 5

tf9.textProperty().addListener(new ChangeListener<String>() {

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if (!newValue.matches("\\d{0,5}?")) {

tf9.setText(oldValue);

}

}

});

Label L10 = new Label("Country ;");

TextField tf10 = new TextField();

gridP1.add(L10, 2, 7);

gridP1.add(tf10, 3, 7);

// set tf10 to uppercase

tf10.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

Label L11 = new Label("District :");

TextField tf11 = new TextField();

gridP1.add(L11, 0, 7);

gridP1.add(tf11, 1, 7);

borderRegister.setCenter(gridP1);

// set tf11 to uppercase

tf11.setTextFormatter(new TextFormatter<>((change) -> {

change.setText(change.getText().toUpperCase());

return change;

}));

// Creating the process and exit button inside the HBox

HBox hbox = new HBox(15); // Making space horizontally between process and exits buttons

Button btNext = new Button("Next");

Button btPrevious = new Button("Previous");

// When user clicks at the button Process , it will then proceed to the next

// scene

btNext.setOnAction(e -> {

File memberFile = new File("member" + memberNO++ + ".txt");

try {

memberFile.createNewFile();

PrintWriter writeMember = new PrintWriter(memberFile);

writeMember.println(PartA.cbHonTitle.getValue());

writeMember.println(PartA.tfName.getText());

writeMember.println(PartA.tfIC.getText());

writeMember.println(PartA.cbGender.getValue());

writeMember.println(PartA.tfDOB.getText());

writeMember.print(PartA.tfPlaceBirth.getText());

writeMember.println(PartA.cbReligion.getValue());

writeMember.println(PartA.cbRace.getValue());

writeMember.println(PartA.tfAddress.getText());

writeMember.println(PartA.tfDistrict.getText());

writeMember.println(PartA.tfPostcode.getText());

writeMember.println(PartA.tfState.getText());

writeMember.println(PartA.tfCountry.getText());

writeMember.println(PartA.cbMarital.getValue());

writeMember.println(PartA.cbCitizenship.getValue());

// information in Part B

writeMember.println(tf1.getText());

writeMember.println(tf2.getText());

writeMember.println(tf3.getText());

writeMember.println(tf8.getText());

writeMember.println(tf4.getText());

writeMember.println(tf5.getText());

writeMember.println(tf6.getText());

writeMember.println(tf11.getText());

writeMember.println(tf9.getText());

writeMember.println(tf7.getText());

writeMember.println(tf10.getText());

writeMember.close();

JOptionPane.showMessageDialog(null, "Member Information Saved Successfully" + memberFile.getName(),

null, JOptionPane.INFORMATION\_MESSAGE);

MainPage.stage.setScene(MainPage.uploadPicture);

} catch (Exception e1) {

e1.printStackTrace();

// ignore error otherwise, if we can safely do so.

}

});

// When user clicks at the button Process , it will then go back to the previous

// scene

btPrevious.setOnAction((ActionEvent event) -> {

MainPage.stage.setScene(MainPage.partA);

});

hbox.getChildren().addAll(btNext, btPrevious);// Adding the button into the hbox

borderRegister.setBottom(hbox); // Setting both of the exits and process buttons at the bottom of the pane

hbox.setAlignment(Pos.CENTER); // Setting the buttons so that it will be at the centre of the pane

hbox.setPadding(new Insets(0, 0, 15, 0)); // Making space for the buttons from the bottom

}

}

SELECT ACTION

/\*\*

\* This is class for the scene select action.

\* The user have to choose what action they want to make.

\* Depending on what the user choose, it will bring the user to the next scene.

\* There are 3 choices that is change password, register new member, and see all the list of the members.

\*/

package project;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

//import javafx.scene.Group;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.SceneAntialiasing;

import javafx.scene.control.Button;

import javafx.scene.control.RadioButton;

import javafx.scene.control.ToggleGroup;

import javafx.scene.control.Label;

import javafx.scene.layout.\*;

public class SelectAction extends Scene {

static BorderPane bp = new BorderPane();

public SelectAction(BorderPane bp, int a, int b) {

super(bp, a, b);

this.bp = bp;

// create a label

Label labelSelectAction = new Label("Please select an action: ");

// create radio button for user to select what action

RadioButton rbRegistration = new RadioButton("Register New Member");

RadioButton rbViewMember = new RadioButton("View All Member");

RadioButton rbChangePW = new RadioButton("Change Password");

ToggleGroup actionGroup = new ToggleGroup();

rbRegistration.setToggleGroup(actionGroup);

rbViewMember.setToggleGroup(actionGroup);

rbChangePW.setToggleGroup(actionGroup);

// create a button

Button btnConfirm = new Button("Confirm");

Button btnLogout = new Button("Logout");

// create a vertical box

VBox vb = new VBox(15);

vb.getChildren().addAll(labelSelectAction, rbRegistration, rbViewMember, rbChangePW);

// set the action for the button

btnConfirm.setOnAction(e -> {

if (rbRegistration.isSelected())

MainPage.stage.setScene(MainPage.partA);

else if (rbViewMember.isSelected())

MainPage.stage.setScene(MainPage.viewMember);

else

MainPage.stage.setScene(MainPage.newPassword);

});

// set the action for button logout, and will go back to login page if click

btnLogout.setOnAction(e -> {

MainPage.stage.setScene(MainPage.login);

});

// create HBox for the button at bottom

HBox hb = new HBox();

hb.getChildren().addAll(btnConfirm, btnLogout);

btnConfirm.setAlignment(Pos.BOTTOM\_RIGHT);

// assign all the elements to the border pane, and set it to the right place

bp.setCenter(vb);

bp.setBottom(hb);

bp.setAlignment(vb, Pos.TOP\_CENTER);

bp.setAlignment(hb, Pos.BOTTOM\_RIGHT);

bp.setPadding(new Insets(250, 50, 25, 50));

}

} // end of class Scene2

UPLOAD PICTURE

/\*\*

\* This is a class for scene uploading picture. It is from the member registration form.

\*/

package project;

import javafx.scene.control.Label;

import javafx.scene.control.TabPane;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.stage.Stage;

import javafx.stage.FileChooser;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.geometry.\*;

import javafx.event.\*;

import java.io.\*;

import javax.imageio.ImageIO;

import javafx.embed.swing.SwingFXUtils;

import javafx.scene.shape.Rectangle;

import java.awt.image.BufferedImage;

import java.util.logging.Logger;

import java.util.logging.Level;

public class UploadPicture extends Scene {

// main create Layout

static BorderPane bpUpload = new BorderPane();

// constructor

UploadPicture(BorderPane bpUpload, int a, int b) {

super(bpUpload, a, b);

this.bpUpload = bpUpload;

// Create VBox,insert Label and ImageView, Button

VBox vbUpload = new VBox();

vbUpload.setAlignment(Pos.CENTER);

Label lUpload = new Label("UPLOAD PICTURE");

ImageView profilepic = new ImageView();

StackPane paneForPic = new StackPane();

paneForPic.getChildren().addAll(new Rectangle(100, 100, Color.BLACK), profilepic);

Button choosepic = new Button("Choose Picture");

choosepic.setOnAction(e -> {

// set extension filter

FileChooser browsePic = new FileChooser();

FileChooser.ExtensionFilter weWantPNG = new FileChooser.ExtensionFilter("PNG files (\*.png)", "\*.PNG");

FileChooser.ExtensionFilter weWantJPG = new FileChooser.ExtensionFilter("JPG files 9\*.jpg)", "\*.JPG");

browsePic.getExtensionFilters().addAll(weWantPNG, weWantJPG);

// Open file dialog

File weWantPic = browsePic.showOpenDialog(null);

try {

BufferedImage buffMyPic = ImageIO.read(weWantPic);

Image setMyPic = SwingFXUtils.toFXImage(buffMyPic, null);

profilepic.setImage(setMyPic);

} catch (IOException ex) {

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

}

});

// UploadPic

vbUpload.getChildren().addAll(lUpload, paneForPic, choosepic);

// Insert element into main layout

Button submit = new Button("Submit");

submit.setOnAction(e -> {

MainPage.stage.setScene(MainPage.selectAction);

});

bpUpload.setCenter(vbUpload);

bpUpload.setBottom(submit);

bpUpload.setAlignment(submit, Pos.BOTTOM\_RIGHT);

}

}

VIEW MEMBER

/\*\*

\* This is a class for viewing the members. All the members that have been register will be shown here.

\*/

package project;

import javafx.scene.control.TextField;

import javafx.scene.control.Button;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.layout.VBox;

import javafx.scene.control.Label;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import java.io.File;

import java.util.Scanner;

public class ViewMember extends Scene {

// main layout

static VBox vbnamelist = new VBox();

static GridPane gpnamelist = new GridPane();

// constructor

ViewMember(VBox namelist, int a, int b) {

super(namelist, a, b);

this.vbnamelist = namelist;

// Create HBox,insert TextField andButton

HBox hbnamelist = new HBox(10);

Button bSearch = new Button("Search");

bSearch.setPrefWidth(125);

TextField tfSearch = new TextField();

tfSearch.setPrefWidth(300);

hbnamelist.getChildren().addAll(tfSearch, bSearch);

// set attributes for gridpane

gpnamelist.setHgap(200);

gpnamelist.setAlignment(Pos.TOP\_LEFT);

gpnamelist.add(new Label("Name"), 0, 0);

gpnamelist.add(new Label("Status"), 1, 0);

// inset elements for main layout, and set attributes

vbnamelist.setPadding(new Insets(20, 50, 50, 50));

vbnamelist.getChildren().addAll(hbnamelist, new Label("MEMBER LIST"), gpnamelist);

for (int d = 0; d <= PartB.memberNO; d++) {

try {

Scanner scanMember = new Scanner(new File("member" + d++ + (".txt")));

for (int bb = 1; bb <= ((PartB.memberNO) + 1); bb++) {

String memTitle = scanMember.next();

String memName = scanMember.next();

gpnamelist.add(new Label(memTitle), 0, b);

}

scanMember.close();

} catch (Exception e3) {

e3.printStackTrace();

}

}

// create button back and pin it to the

Button btnBack = new Button("Back to Select Action");

HBox toBack = new HBox();

toBack.setAlignment(Pos.BOTTOM\_RIGHT);

toBack.getChildren().add(btnBack);

vbnamelist.getChildren().add(toBack);

btnBack.setOnAction(e -> {

MainPage.stage.setScene(MainPage.selectAction);

});

}

}

**BENEFITS OF SYSTEM**

1. The users can register in the organization easily without having to submit their registration form personally to the president of the organizartion (save time).
2. The president of the organization can keep track of each member easily which it can help them to organize any events with the involvement of each member of the organization in the future.
3. The president of the organization can access member’s profile from the system.
4. The members of the organization can get know each other without having to meet face to face.