



SCHOOL OF COMPUTER SCIENCES

ACADEMIC SESSION: 2021/2022

CAT201 Integrated Software Development Workshop

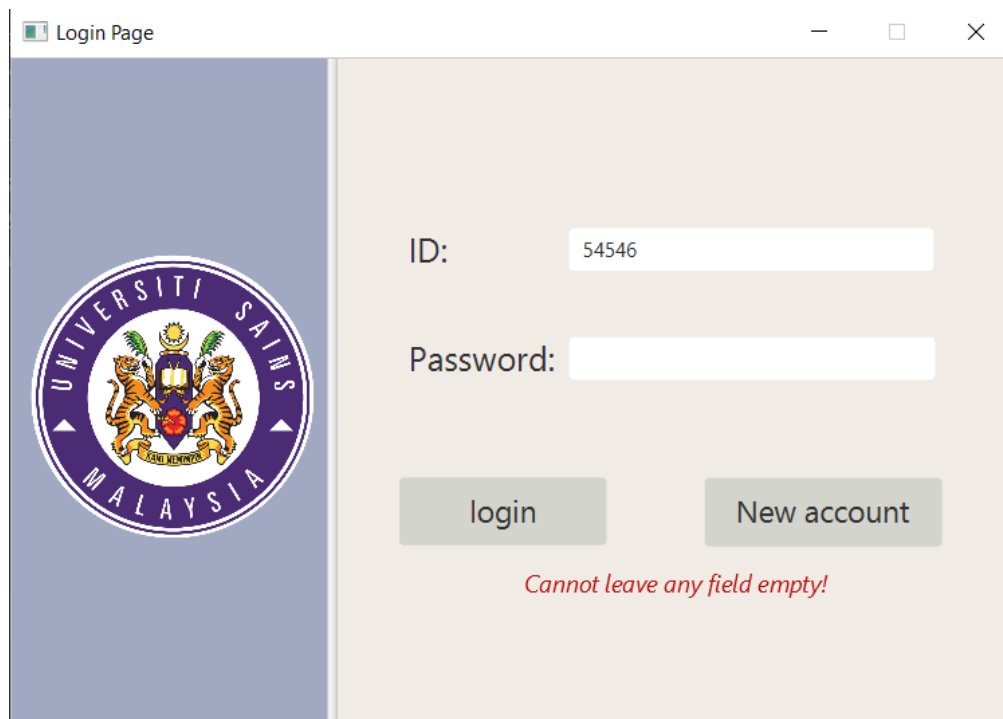
Project

Matrix No.	Name	Email
154648	Hazem Ahmed	hazemamn@student.usm.my
154385	CUI SAIFEI	cuisaifei@student.usm.my
154561	WANG YAN	yanwang@student.usm.my
154467	GUO SHIHAN	guoshihan@student.usm.my

1: Program description

With the development of technology and the advancement of paperless office, an app called "MyUni Companion" came into being. Its arrival greatly facilitates online teaching for teachers and students. Lecturers and students take it accompanies them along their journey, it allows them to register at the university, register for courses, check their assignments and solving them and many other features that make this app one of the must-haves for all teaching and teaching staff .


2: Screen shots



The screenshot shows a web browser window titled "Login Page". The page is split into two main sections. On the left, there is a vertical blue bar containing the circular logo of Universiti Sains Malaysia, which features two tigers and a central emblem. On the right, the background is a light beige color. This section contains the login form with the following elements:

- An "ID:" label followed by a white text input field containing the number "54546".
- A "Password:" label followed by an empty white text input field.
- Two grey buttons: "login" and "New account".
- A red italicized error message at the bottom: "Cannot leave any field empty!"

Login Page



ID:

Password:

Wrong Password!

Student Home Page

Student Info

Name: Mason Daniell

ID: 1021 Current Year: 1

Age: 19 Country: U.S

Email: MasonDaniell@student.my

Registered Courses

Course Code	Course Name	Course Type
AKW104	ECONMIC	T

Inspect Course

Course Info

Course Name: ECONMIC
Course code: AKW104
Course Type: T
Course Credits: 4
Lecturer Name: Emma Johnson

Course Assignments

Assignemnt Name	Assignment Type
Resources Scarcity	H.W.
Utility Maximization	H.W.

Return to home

Solve

3: Member's contributions

First iteration

Classes with CSV tables	
Hazem	Assignment, Assignment_Metadata
Cui SaiFei	Lecturer and Assignment
Guo Shrihan	Course and Course_Metadata
Wang Yan	Member

Second iteration

Scene design and data fill to CSV file	
Hazem	Design scene 7-14
Cui SaiFei	Design scene 1-7
Guo Shrihan	Data fill
Wang Yan	Data fill

Third iteration

Code and CSS	
Hazem	Controllers Code
Cui SaiFei	DatabaseFiller Code
Guo Shrihan	CSS Code
Wang Yan	CSS Code

4: Source codes and reference

4.1 Source codes

1) Assignment

```
4 public class Assignment {
5
6     private int assignment_ID;
7     private String assignment_name;
8     private String course_ID;
9     private String assignment_type;
10    private String assignment_question;
11
12
13    public void setAssignment_ID(int assignment_ID) {
14        this.assignment_ID = assignment_ID;
15    }
16
17    public int getAssignment_ID() {
18        return assignment_ID;
19    }
20
21    public void setAssignment_name(String assignment_name) {
22        this.assignment_name = assignment_name;
23    }
24
25    public String getAssignment_name() {
26        return assignment_name;
27    }
28
29    public void setCourse_ID(String course_ID) {
30        this.course_ID = course_ID;
31    }
32
33    public String getCourse_ID() {
34        return course_ID;
35    }
36
37    public void setAssignment_type(String assignment_type) {
38        this.assignment_type = assignment_type;
39    }
40
41    public String getAssignment_type() {
42        return assignment_type;
43    }
44
45
46    public void set_assignment_question(String file_path) {
47        assignment_question = new String(file_path);
48    }
49
50    public String get;
51
52    String assignment_question() {
53        return assignment_question;
54    }
55
56 }
```

2)Assignment Metadata

60 lines (46 sloc) 1.53 KB

```
1 package org.companion.myunicompanion.classes;
2
3
4 public class Assignment_Metadata {
5     private int assignment_id;
6     private String course_id;
7     private String answer;
8     String assignment_grade;
9
10    public Assignment_Metadata() {
11        assignment_id = 0;
12        course_id = "";
13        answer = null;
14        assignment_grade = "";
15    }
16
17    public Assignment_Metadata(int asgn_id, String course_id, String answer, String asgn_grade) {
18        this.assignment_id = asgn_id;
19        this.course_id = course_id;
20        this.answer = new String(answer);
21        this.assignment_grade = asgn_grade;
22    }
23
24    public int getAssignment_id() {
25        return this.assignment_id;
26    }
27
28    public String getCourse_id() {
29        return this.course_id;
30    }
31
32    public String getAnswer() {
33        return this.answer;
34    }
35
36    public String getAssignment_grade() {
37        return this.assignment_grade;
38    }
39
40    public void setAssignment_id(int assignment_id) {
41        this.assignment_id = assignment_id;
42    }
43
44    public void setCourse_id(String course_id) {
45        this.course_id = course_id;
46    }
47
48    public void setAnswer(String answer) {
49        this.answer = answer;
50    }
51
52    public void setAssignment_grade(String assignment_grade) {
53        this.assignment_grade = assignment_grade;
54    }
55
56
57    public String toString() {
58        return "Assignment_Metadata(assignment_id=" + this.getAssignment_id() + ", course_id=" + this.getCourse
59    }
60 }
```

3) Course

© 2022 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#)

62 lines (50 sloc) | 1.99 KB

```
1 package org.companion.myun[companion.classes];
2
3 public class Course_Metadata {
4     private String course_ID;
5     private String course_grade;
6
7     public Course_Metadata() {
8         course_ID = "";
9         course_grade = "";
10    }
11
12    public void setCourse_ID(String course_ID) {
13        this.course_ID = course_ID;
14    }
15
16    public String getCourse_ID() {
17        return course_ID;
18    }
19
20    public void setCourse_grade(String course_grade) {
21        this.course_grade = course_grade;
22    }
23
24
25    public String getCourse_grade() {
26        return this.course_grade;
27    }
28
29
30    public String toString() {
31        return "CourseMetadata(course_ID=" + this.getCourse_ID() + ", course_grade=" + this.getCourse
32    }
33
34    public boolean equals(final Object o) {
35        if (o == this) return true;
36        if (!(o instanceof Course_Metadata)) return false;
37        final Course_Metadata other = (Course_Metadata) o;
38        if (!other.canEqual((Object) this)) return false;
39        final Object this$course_ID = this.getCourse_ID();
40        final Object other$course_ID = other.getCourse_ID();
41        if (this$course_ID == null ? other$course_ID != null : !this$course_ID.equals(other$course_ID)
42        final Object this$course_grade = this.getCourse_grade();
43        final Object other$course_grade = other.getCourse_grade();
44        if (this$course_grade == null ? other$course_grade != null : !this$course_grade.equals(other$
45            return false;
46        return true;
47    }
48
49    protected boolean canEqual(final Object other) {
50        return other instanceof Course_Metadata;
51    }
52
53    public int hashCode() {
54        final int PRIME = 59;
55        int result = 1;
56        final Object $course_ID = this.getCourse_ID();
57        result = result * PRIME + ($course_ID == null ? 43 : $course_ID.hashCode());
58        final Object $course_grade = this.getCourse_grade();
59        result = result * PRIME + ($course_grade == null ? 43 : $course_grade.hashCode());
60        return result;
61    }
62 }
```

5) Member

[Go back to the top of the page](#) [Home](#) [About](#) [Privacy](#) [Contact](#) [Sitemap](#) [Terms](#) [Feedback](#) [Help](#)

```

124 lines (88 Sloc)   3.6 KB
1  package org.companion.mymusiccompanion.classes;
2
3  import java.time.LocalDate;
4  import java.time.Period;
5  import java.util.ArrayList;
6  import java.util.HashMap;
7
8
9  public class Student extends Person {
10     private LocalDate date_enrolled;
11     private HashMap<Integer, Assignment_Metadata> exams_metadata;
12     private HashMap<String, Course_Metadata> courses_metadata;
13
14
15     public Student() {
16         super();
17         date_enrolled = LocalDate.now();
18         courses_metadata = new HashMap<>();
19         exams_metadata = new HashMap<>();
20     }
21
22     public void set_date_enrolled(int year, int month num, int day of month) {
23         date_enrolled = LocalDate.of(year, month_num, day_of_month);
24     }
25
26     public int get_enrolled_year() {
27         return Period.between(date_enrolled, LocalDate.now()).getYears();
28     }
29
30
31     public void add_courses_metadata(String course_id, Course_Metadata course) {
32         courses_metadata.put(course_id, course);
33     }
34
35     public Course_Metadata get_course_metadata(String course_id) {
36         //verify id exist
37         if (courses_metadata.containsKey(course_id))
38             return courses_metadata.get(course_id);
39         else
40             return null;
41     }
42
43     public boolean set_course_grade(String course_id, String grade) {
44         //verify if id exist
45         if (courses_metadata.containsKey(course_id)) {
46             Course_Metadata course = courses_metadata.get(course_id);
47             return course.set_course_grade(grade);
48         } else
49             return false;
50     }
51
52     public String get_course_grade(String course_id) {
53         if (courses_metadata.containsKey(course_id))
54             return courses_metadata.get(course_id).get_course_grade();
55         else
56             return null;
57     }
58
59     public void add_exam_metadata(int exam_id, Assignment_Metadata exam) {
60         exams_metadata.put(exam_id, exam);
61     }
62
63     public Assignment_Metadata get_exam_metadata(int exam_id) {
64         //verify id exist
65         if (exams_metadata.containsKey(exam_id))
66             return exams_metadata.get(exam_id);
67         else
68             return null;
69     }
70
71     public boolean set_exam_grade(int exam_id, String grade) {
72         //verify if id exist
73         if (exams_metadata.containsKey(exam_id)) {
74             Assignment_Metadata exam = exams_metadata.get(exam_id);
75             return exam.set_exam_grade(grade);
76         } else
77             return false;
78     }
79
80     public String get_exam_grade(int exam_id) {
81         if (exams_metadata.containsKey(exam_id))
82             return exams_metadata.get(exam_id).get_exam_grade();
83         else
84             return null;
85     }
86
87     public void remove_exam(int exam_id) {
88         if (exams_metadata.containsKey(exam_id))
89             exams_metadata.remove(exam_id);
90     }
91
92     }
93
94     public boolean unenrolled_course(String course_id) {
95
96         if (courses_metadata.containsKey(course_id)) {
97             ArrayList<Integer> assign_keys = new ArrayList<>();
98             Course_Metadata course = courses_metadata.get(course_id);
99             //getting the assignment keys to delete them
100             exams_metadata.forEach((key, exam) -> {
101                 if (exam.get_course_id() == course_id)
102                     assign_keys.add(exam.get_assignment_id());
103             });
104             //deleting the assignments metadata
105             for (int i = 0; i < assign_keys.size(); i++)
106                 exams_metadata.remove(assign_keys.get(i));
107             return true;
108         }
109         //returns false if there is no
110         return false;
111     }
112     }
113
114     public ArrayList<Integer> Assignment_Metadatas get_all_assignments() {
115         return exams_metadata.keySet();
116     }
117
118     public HashMap<String, Course_Metadata> get_all_courses() {
119         return courses_metadata;
120     }
121
122
123
124 }

```

7)Lecturer

40 lines (28 sloc) | 934 Bytes

```
1 package org.companion.myunicompanion.classes;
2
3 import java.time.LocalDate;
4 import java.time.Period;
5 import java.util.ArrayList;
6
7 public class lecturer extends Member {
8
9     private ArrayList<String> courses_taught;
10    private LocalDate date_hired;
11
12    public lecturer() {
13        //super();
14        courses_taught = new ArrayList<>();
15        date_hired = LocalDate.now();
16    }
17
18    public void add_course_taught(String course_id) {
19
20        courses_taught.add(course_id);
21
22    }
23
24    public ArrayList<String> get_course_taught() {
25        return courses_taught;
26    }
27
28    public void deregister_course(String course_id) {
29        courses_taught.remove(course_id);
30
31    }
32
33    public int get_lecturing_period() {
34        return Period.between(date_hired, LocalDate.now()).getMonths();
35    }
36
37    public void set_hired_date(int year, int month, int dayofmonth) {
38        date_hired = LocalDate.of(year, month, dayofmonth);
39    }
40 }
```

8) LoginSceneController

```

1 package org.companion.myunicompanion;
2
3 import javafx.event.ActionEvent;
4 import javafx.fxml.FXML;
5 import javafx.fxml.FXMLLoader;
6 import javafx.scene.Parent;
7 import javafx.scene.*;
8 import javafx.scene.control.*;
9 import javafx.scene.text.Text;
10 import javafx.stage.Stage;
11
12 import java.io.IOException;
13
14 import static org.companion.myunicompanion.DatabaseFiller.db;
15
16 public class LoginSceneController {
17     @FXML
18     private Button loginBtn;
19     @FXML
20     private TextField idField;
21     @FXML
22     private PasswordField password;
23     @FXML
24     private Text errMsg;
25     @FXML
26     private Button newAccountBtn;
27     @FXML public void checkCredentials(ActionEvent e)
28     {
29
30         //not empty check
31         if(!idField.getText().isBlank() && !(password.getText().isBlank())){
32             //check if id is a number
33             try {
34                 int id = Integer.parseInt(idField.getText());
35                 // see if ID is in the students table
36                 if(db.students.containsKey(id)){
37                     //check password of student
38                     if(db.students.get(id).getPassword().equals(password.getText())){
39                         db.dataTransporter.put("student id",idField.getText());
40                         //go to student homepage scene
41                         goToStudentHomePage(e);
42                     }
43                     else { //password doesn't match
44                         errMsg.setText("Wrong Password!");
45                     }
46                     //the id is not a student so let's see if it is a lecturer
47                 }else if(db.lecturers.containsKey(id)){
48                     //check password for lecturer
49                     if(db.lecturers.get(id).getPassword().equals(password.getText())){
50                         db.dataTransporter.put("lecturer id",idField.getText());
51                         //go to lecturer homepage scene
52                     } //password doesn't match
53                     else {
54                         errMsg.setText("Wrong Password!");
55                     }
56                 } // the id doesn't belong to any student or lecturer
57                 else {
58                     errMsg.setText("This ID doesn't belong to any student or lecturer.\n" +
59                         "Please create a new account or contact the university admin.");
60                 }
61             }
62             //The wasn't an Integer
63             catch (NumberFormatException nfe){
64                 errMsg.setText("The ID must be an Integer");
65             }
66         }
67         // either one of the fields was empty
68         else{
69             errMsg.setText("Cannot leave any field empty!");
70         }
71     }
72
73     private void goToStudentHomePage(ActionEvent e) {
74         Parent stuPage = null;
75         try {
76             stuPage = FXMLLoader.load(LoginSceneController.class.getResource("Student HomePage(Scene3).fxml"));
77             Scene stuScene = new Scene(stuPage);
78             Stage window = (Stage) ((Node) e.getSource()).getScene().getWindow();
79             window.setScene(stuScene);
80             window.setTitle("Student Home Page");
81             window.show();
82         } catch (IOException ex) {
83             ex.printStackTrace();
84         }
85     }
86     private void goToLecturerHomePage(ActionEvent e){
87
88     }
89
90     @FXML
91     public void switchToCreateAccount(ActionEvent e) {
92         Parent accountPage = null;
93         try {
94             accountPage = FXMLLoader.load(LoginSceneController.class.getResource("New Account(Scene2).fxml"));
95             Scene accountScene = new Scene(accountPage);
96             Stage window = (Stage) ((Node) e.getSource()).getScene().getWindow();
97             window.setScene(accountScene);
98             window.setTitle("Create Account Page");
99             window.show();
100         } catch (IOException ex) {
101             ex.printStackTrace();
102         }
103     }
104 }

```

9) StudentHomeController

```
6 import javafx.event.ActionEvent;
7 import javafx.fxml.FXML;
8 import javafx.fxml.FXMLLoader;
9 import javafx.fxml.Initializable;
10 import javafx.scene.Node;
11 import javafx.scene.Parent;
12 import javafx.scene.Scene;
13 import javafx.scene.control.*;
14 import javafx.scene.control.cell.PropertyValueFactory;
15 import javafx.scene.text.Text;
16 import javafx.stage.Stage;
17 import org.companion.myunicompanion.classes.*;
18
19 import java.io.IOException;
20 import java.net.URL;
21 import java.util.ArrayList;
22 import java.util.HashMap;
23 import java.util.ResourceBundle;
24
25 import static org.companion.myunicompanion.DatabaseFiller.db;
26
27 public class StudentHomeController implements Initializable {
28
29     @FXML
30     private TableView<Course> table;
31     @FXML
32     private TableColumn<Course, String> courseCode;
33
34     @FXML
35     private TableColumn<Course, String> courseName;
36     @FXML
37     private TableColumn<Course, String> courseType;
38
39
40     @FXML
41     private Text stuAge;
42
43     @FXML
44     private Text stuCountry;
45
46     @FXML
47     private Text stuEmail;
48
49     @FXML
50     private Text stuID;
51
52     @FXML
53     private Text stuName;
54
55     @FXML
56     private Text stuYear;
57     @FXML
58     private Button inspectBtn;
59     @FXML
60     private Button unregisterBtn;
61
62
63     @Override
64     public void initialize(URL url, ResourceBundle resourceBundle) {
65         //getting the current student id from transporter
66         int stu_id = Integer.parseInt(db.dataTransporter.get("student id"));
67         Student stu = db.students.get(stu_id);
68
69         stuAge.setText("Age: " + Integer.toString(stu.get_student_year()));
70         stuCountry.setText("Country: "+stu.getCountry());
71         stuEmail.setText("Email: "+stu.getEmail());
72         stuID.setText("ID: "+Integer.toString(stu.getUni_ID()));
73         stuName.setText("Name: "+ stu.getName() + " " + stu.getName());
74         stuYear.setText("Current Year: " + Integer.toString(stu.get_student_year()));
75         //setting the columns
76         courseName.setCellValueFactory(new PropertyValueFactory<Course, String>("course_name"));
77         courseCode.setCellValueFactory(new PropertyValueFactory<Course, String>("course_id"));
78         courseType.setCellValueFactory(new PropertyValueFactory<Course, String>("course_type"));
79         ObservableList<Course> list = getCourses(stu_id);
80         if(!list.isEmpty())
81             table.setItems(list);
82
83         table.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
84     }
85
86     private ObservableList<Course> getCourses(int stu_id){
87         ObservableList<Course> list = FXCollections.observableArrayList();
88         //getting the course ids to get their information later
89         ArrayList<String> student_courses_ids = new ArrayList<>();
90         HashMap<String,Course_Metadata> stu_courses = db.students.get(stu_id).get_all_courses();
91         //checking if the student has enrolled in any course
92         if( stu_courses != null && !(stu_courses.isEmpty()) ) {
93             db.students.get(stu_id).get_all_courses().forEach((key, value) -> {
94                 student_courses_ids.add(key);
95             });
96             //extracting these courses from the database and filling our list with it
97             student_courses_ids.forEach((course_id) -> {
98                 list.add(db.courses.get(course_id));
99             });
100         }
101         return list;
102     }
103
104     //inspect course method
105     @FXML
```

```

110         //transferring the data
111         db.dataTransporter.put("course id",selected_course.getCourse_ID());
112         switchToInspectCourseScene("Student Inspect Course(Scene4).fxml", e);
113     }
114 }
115 @FXML
116 private void unregisterCourse(ActionEvent e){
117     Course selected_course = table.getSelectionModel().getSelectedItem();
118     if(selected_course !=null){
119         //Alert popup
120         Alert alert = new Alert(Alert.AlertType.CONFIRMATION, "If you unregister this course, all you your assignment solutions will be lost, Are you sure?", ButtonType.YES);
121         alert.showAndWait();
122         if (alert.getResult() == ButtonType.YES) {
123             table.getItems().remove(selected_course);
124             //getting the current student id from transporter
125             int stu_id = Integer.parseInt(db.dataTransporter.get("student id"));
126             //removing the course
127             db.students.get(stu_id).unregister_course(selected_course.getCourse_ID());
128         }
129     }
130 }
131 }
132
133 private void switchToInspectCourseScene(String fxmlPath, ActionEvent e){
134     Parent coursePage = null;
135     try {
136         coursePage = FXMLLoader.load(LoginScene1Controller.class.getResource(fxmlPath));
137         Scene stuScene = new Scene(coursePage);
138         Stage window = (Stage) ((Node) e.getSource()).getScene().getWindow();
139         window.setScene(stuScene);
140         window.setTitle("Inspect Course");
141         window.show();
142     } catch (IOException ex) {
143         ex.printStackTrace();
144     }
145 }
146
147
148 }

```

10) StudentInspectCourseController

```

1 package org.companion.myunicompanion;
2
3 import javafx.collections.FXCollections;
4 import javafx.collections.ObservableList;
5 import javafx.event.ActionEvent;
6 import javafx.fxml.FXML;
7 import javafx.fxml.FXMLLoader;
8 import javafx.fxml.Initializable;
9 import javafx.scene.Node;
10 import javafx.scene.Parent;
11 import javafx.scene.Scene;
12 import javafx.scene.control.Button;
13 import javafx.scene.control.SelectionMode;
14 import javafx.scene.control.TableColumn;
15 import javafx.scene.control.TableView;
16 import javafx.scene.control.cell.PropertyValueFactory;
17 import javafx.scene.text.Text;
18 import javafx.stage.Stage;
19 import org.companion.myunicompanion.classes.Assignment;
20 import org.companion.myunicompanion.classes.Assignment_Metadata;
21 import org.companion.myunicompanion.classes.Course;
22 import org.companion.myunicompanion.classes.Student;
23
24 import java.io.IOException;
25 import java.net.URL;
26 import java.util.ArrayList;
27 import java.util.HashMap;
28 import java.util.ResourceBundle;
29
30 import static org.companion.myunicompanion.DatabaseFiller.db;
31

```

```

32 public class StudentInspectCourseController implements Initializable {
33     @FXML
34     private TableView<Assignment> table;
35
36     @FXML
37     private TableColumn<Assignment,String> assignQuestionCol;
38
39     @FXML
40     private TableColumn<Assignment,String> assignTypeCol;
41
42     @FXML
43     private Text courseID;
44
45     @FXML
46     private Text courseName;
47
48     @FXML
49     private Text courseType;
50
51     @FXML
52     private Text lecturerName;
53
54     @FXML
55     private Button returnBtn;
56
57     @FXML
58     private Button solveBtn;
59
60
61     @Override
62     public void initialize(URL url, ResourceBundle resourceBundle) {
63         //getting the current student and course ids from transporter
64         int stu_id = Integer.parseInt(db.dataTransporter.get("student id"));
65         Student stu = db.students.get(stu_id);
66         String current_course_id = db.dataTransporter.get("course id");
67         Course course = db.courses.get(current_course_id);
68         //getting the courses's lecturer info
69         int lecturer_id = course.getLecturer_id();
70         String lec_fisrt_name = db.lecturers.get(lecturer_id).getFname();
71         String lec_full_name = lec_fisrt_name + " " + db.lecturers.get(lecturer_id).getlname();
72         // setting the course info on the scene
73         courseID.setText(course.getCourse_ID());
74         courseName.setText(course.getCourse_name());
75         courseType.setText(course.getCourse_type());
76         lecturerName.setText(lec_full_name);
77         //setting the columns
78         assignQuestionCol.setCellValueFactory(new PropertyValueFactory<Assignment, String>("assignment_name"));
79         assignTypeCol.setCellValueFactory(new PropertyValueFactory<Assignment, String>("assignment_type"));
80
81         ObservableList<Assignment> list = getAssignments(current_course_id);
82
83         if(!list.isEmpty())
84             table.setItems(list);
85
86         table.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
87     }
88     private ObservableList<Assignment> getAssignments(String current_course_id){
89         ObservableList<Assignment> list = FXCollections.observableArrayList();
90         //getting all the assignments ids
91         if(!db.assignments.isEmpty()) {
92             db.assignments.forEach((asn_id, asgn) -> {
93                 if (current_course_id.equals(asgn.getCourse_ID()))
94                     list.add(asgn);
95             });
96         }
97         return list;
98     }
99
100     @FXML
101     public void returnToHome(ActionEvent e) {
102         Parent homePage = null;
103         try {
104             homePage = FXMLLoader.load(StudentInspectCourseController.class.getResource("Student HomePage(Scene3).fxml"));
105             Scene homeScene = new Scene(homePage);
106             Stage window = (Stage) ((Node) e.getSource()).getScene().getWindow();
107             window.setScene(homeScene);
108             window.setTitle("Student Home Page");
109             window.show();
110         } catch (IOException ex) {
111             ex.printStackTrace();
112         }
113     }
114     @FXML
115     public void switchToSolvingScene(ActionEvent e) {
116         Assignment selected_asgn = table.getSelectionModel().getSelectedItem();
117         db.dataTransporter.put("assignment id",Integer.toString(selected_asgn.getAssignment_ID()));
118         Parent solvingPage = null;
119         try {
120             solvingPage = FXMLLoader.load(StudentInspectCourseController.class.getResource("Solving Assignment(Scene5).fxml"));
121             Scene solvingScene = new Scene(solvingPage);
122             Stage window = (Stage) ((Node) e.getSource()).getScene().getWindow();
123             window.setScene(solvingScene);
124             window.setTitle("Solving Assignment Page");
125             window.show();
126         } catch (IOException ex) {
127             ex.printStackTrace();
128         }
129     }
130 }
131
132 /*
133 HashMap<Integer, Assignment_Metadata> stu_asgns = db.students.get(stu_id).get_all_assignments();
134 //check if there is registered courses for the
135 if( stu_asgns != null && !(stu_asgns.isEmpty())) {
136     db.students.get(stu_id).get_all_assignments().forEach((asn_id, value) -> {
137         assignments_ids.add(asn_id);
138     });
139     //extracting these assignemnts from the database and filling our list with it
140     assignments_ids.forEach((asn_id) -> {
141         list.add(db.assignments.get(asn_id));
142     });
143 }
144 */

```


11) SolvingAssignmentController

```
1 package org.companion.myunicompanion;
2
3 import javafx.event.ActionEvent;
4 import javafx.fxml.FXML;
5 import javafx.fxml.FXMLLoader;
6 import javafx.fxml.Initializable;
7 import javafx.scene.*;
8 import javafx.scene.control.*;
9 import javafx.scene.text.Text;
10 import javafx.stage.Stage;
11 import org.companion.myunicompanion.classes.Assignment_Metadata;
12
13 import java.io.IOException;
14 import java.net.URL;
15 import java.util.ResourceBundle;
16
17 import static org.companion.myunicompanion.Databasefiller.db;
18
19 public class SolvingAssignmentController implements Initializable {
20
21     int stu_id, current_assignment_id;
22     String course_id;
23     @FXML
24     private Text questionArea;
25     @FXML
26     private TextArea solutionArea;
27     @FXML
28     private Button cancelBtn;
29     @FXML
30     private Button returnBtn;
31     @FXML
32     private Button submitBtn;
33
34     @Override
35     public void initialize(URL url, ResourceBundle resourceBundle) {
36         //getting the the transfered data
37         stu_id = Integer.parseInt(db.dataTransporter.get("student id"));
38         current_assignment_id = Integer.parseInt(db.dataTransporter.get("assignment id"));
39         course_id = db.assignments.get(current_assignment_id).getCourse_ID();
40         //initializing the texts in the scene
41         questionArea.setText(db.assignments.get(current_assignment_id).getAssignment_question());
42         if(db.students.get(stu_id).get_all_assignments().containsKey(current_assignment_id)) {
43             String stu_previous_answer = db.students.get(stu_id).get_all_assignments().get(current_assignment_id).getAnswer();
44             solutionArea.setText(stu_previous_answer + " " + stu_id + " " + current_assignment_id);
45         }
46     }
47
48     @FXML
49     public void returnToHome(ActionEvent e) {
50         //checking if there is a solution
51         if(!(solutionArea.getText().isBlank())) {
52             Alert alert = new Alert(Alert.AlertType.WARNING, "Your solution will not be saved, Are you sure?", ButtonType.YES, ButtonType.NO);
53             alert.showAndWait();
54             if (alert.getResult() == ButtonType.YES) {
55                 switchToHome(e);
56             }
57         }
58         else
59             switchToHome(e);
60     }
61
62     private void switchToHome(ActionEvent e) {
63         Parent homePage = null;
64         try {
65             homePage = FXMLLoader.load(StudentInspectCourseController.class.getResource("Student HomePage(Scene3).fxml"));
66             Scene homeScene = new Scene(homePage);
67             Stage window = (Stage) ((Node) e.getSource()).getScene().getWindow();
68             window.setScene(homeScene);
69             window.setTitle("Student Home Page");
70             window.show();
71         } catch (IOException ex) {
72             ex.printStackTrace();
73         }
74     }
75
76     @FXML
77     public void submitSolution(ActionEvent e){
78         //checking if the user wants to save the solution when it is blank
79         if(solutionArea.getText().isBlank()){
80             Alert alert = new Alert(Alert.AlertType.WARNING, "Your solution is empty, Are you sure you want to submit?", ButtonType.YES, ButtonType.NO);
81             alert.showAndWait();
82             if (alert.getResult() == ButtonType.YES)
83             {
84                 saveSolution();
85                 returnToInspectCourse(e);
86             }
87         }
88         else
89         {
90             saveSolution();
91             returnToInspectCourse(e);
92         }
93     }
94
95     private void saveSolution(){
96         Assignment_Metadata asgn_metadata = new Assignment_Metadata();
97         asgn_metadata.setAssignment_id(current_assignment_id);
98         asgn_metadata.setCourse_id(course_id);
99         asgn_metadata.setAnswer(solutionArea.getText());
100         //put solution in assignment metadata of the student
101         db.students.get(stu_id).add_asgn_metadata(current_assignment_id, asgn_metadata);
102     }
103
104     @FXML
```

```

104     @FXML
105     public void cancelSolution(ActionEvent e) {
106         if (solutionArea.getText().isBlank()) {
107             returnToInspectCourse(e);
108         } else {
109             Alert alert = new Alert(Alert.AlertType.WARNING, "Your solution will not be saved, Are you sure?", ButtonType.YES, ButtonType.NO);
110             alert.showAndWait();
111             if (alert.getResult() == ButtonType.YES) {
112                 returnToInspectCourse(e);
113             }
114         }
115     }
116 }
117
118 @FXML
119 public void returnToInspectCourse(ActionEvent e) {
120     Parent coursePage = null;
121     try {
122         coursePage = FXMLLoader.load(StudentInspectCourseController.class.getResource("Student Inspect Course(Scene4).fxml"));
123         Scene courseScene = new Scene(coursePage);
124         Stage window = (Stage) ((Node) e.getSource()).getScene().getWindow();
125         window.setScene(courseScene);
126         window.setTitle("Inspect Course Page");
127         window.show();
128     } catch (IOException ex) {
129         ex.printStackTrace();
130     }
131 }
132 }

```

12) SolvingAssignmentController



13) LecturerInspectCourse

[illegible]

14) LecturerHompPage

[illegible]

15) CreateAccountController

[illegible]

16) StudentRegisterCourse

```
111 lines (97 sloc) | 4.04 KB
1 package org.companion.myapplication;
2
3 import javafx.collections.FXCollections;
4 import javafx.collections.ObservableList;
5 import javafx.event.ActionEvent;
6 import javafx.fxml.FXML;
7 import javafx.fxml.FXMLLoader;
8 import javafx.fxml.Initializable;
9 import javafx.scene.Node;
10 import javafx.scene.Parent;
11 import javafx.scene.Scene;
12 import javafx.scene.control.*;
13 import javafx.scene.control.cell.PropertyValueFactory;
14 import javafx.stage.Stage;
15 import org.companion.myapplication.classes.*;
16
17 import java.io.IOException;
18 import java.net.URL;
19 import java.util.ArrayList;
20 import java.util.HashMap;
21 import java.util.ResourceBundle;
22
23 import static org.companion.myapplication.DatabaseFiller.db;
24
25 public class StudentRegisterCourse implements Initializable {
26     private int stu_id;
27     @FXML
28     private TableView<Course> table;
29     @FXML
30     private TableColumn<Course,String> courseCode;
31
32     @FXML
33     private TableColumn<Course,Integer> courseCredits;
34
35     @FXML
36     private TableColumn<Course,String> courseName;
37
38     @FXML
39     private TableColumn<Course,String> courseType;
40     @FXML
41     private Button registerBtn;
42
43     @FXML
44     private Button returnBtn;
45
46
47     @Override
48     public void initialize(URL url, ResourceBundle resourceBundle) {
49         //getting the transferred data
50         stu_id = Integer.parseInt(db.dataTransporter.get("student id"));
51         //setting the columns
52         courseCode.setCellValueFactory(new PropertyValueFactory<Course, String>("course_name"));
53         courseCode.setCellValueFactory(new PropertyValueFactory<Course, String>("course_id"));
54         courseType.setCellValueFactory(new PropertyValueFactory<Course, String>("course_type"));
55         courseCredits.setCellValueFactory(new PropertyValueFactory<Course, Integer>("credits_num"));
56         ObservableList<Course> list = getCourses();
57         if(list.isEmpty()){
58             table.setItems(list);
59             table.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
60         }
61         private ObservableList<Course> getCourses(){
62             ObservableList<Course> list = FXCollections.observableArrayList();
63             int stu_year = db.students.get(stu_id).get_student_year();
64             String stu_school = db.students.get(stu_id).get_school_name();
65             HashMap<String, Course_Metadate> stu_courses = db.students.get(stu_id).get_all_courses();
66             db.courses.forEach((id,course)->{
67                 if(course.getyear_applicable() == stu_year && course.getschool_name().equals(stu_school) ){
68                     if(!(stu_courses.containsKey(course.getcourse_id()))){
69                         list.add(course);
70                     }
71                 }
72             });
73             return list;
74         }
75     @FXML
76     void registerCourse(ActionEvent e) {
77         Alert alert = null;
78         if(table.getSelectionModel().getSelectedItem() == null){
79             alert = new Alert(Alert.AlertType.ERROR,"Please select a course to register",ButtonType.OK);
80             alert.showAndWait();
81         }
82         else
83             {
84                 //register the course and remove from list
85                 Course selected_course = table.getSelectionModel().getSelectedItem();
86                 Course_Metadate course_metadate = new Course_Metadate();
87                 course_metadate.setcourse_id(selected_course.getcourse_id());
88                 db.students.get(stu_id).add_course_metadate(course_metadate.getcourse_id(), course_metadate);
89                 alert = new Alert(Alert.AlertType.INFORMATION,"Course has been successfully registered, Check it out in your home page.",ButtonType.OK);
90                 alert.showAndWait();
91                 table.getItems().remove(selected_course);
92             }
93     }
94
95     @FXML
96     void switchToHome(ActionEvent e){
97         Parent homePage = null;
98         try {
99             homePage = FXMLLoader.load(StudentRegisterCourse.class.getResource("Student_HomePage(Scene3).fxml"));
100             Scene homeScene = new Scene(homePage);
101             Stage window = (Stage) (home e.getSource()).getScene().getWindow();
102             window.setScene(homeScene);
103             window.setTitle("Student Home Page");
104             window.show();
105         } catch (IOException ex) {
106             ex.printStackTrace();
107         }
108     }
109
110
111 }
```

17) LecturerCourseAssignments

```
35 lines (24 loc) 482 bytes
1 package org.companion.mynicompanion;
2
3 import javafx.fxml.FXML;
4 import javafx.scene.control.Button;
5 import javafx.scene.control.TableColumn;
6 import javafx.scene.control.TableView;
7 import org.companion.mynicompanion.classes.Assignment;
8
9 public class LecturerCourseAssignments {
10
11     @FXML
12     private Button addAssignBtn;
13
14     @FXML
15     private TableColumn<Assignment, String> assignName;
16
17     @FXML
18     private TableColumn<Assignment, String> assignQuestion;
19
20     @FXML
21     private TableColumn<Assignment, String> assignType;
22
23     @FXML
24     private Button gradeAssignBtn;
25
26     @FXML
27     private Button removeAssignBtn;
28
29     @FXML
30     private Button returnAssign;
31
32     @FXML
33     private TableView<Table> table;
34
35 }
```

18) LecturerGradeStudent

```
87 lines (54 loc) 2,414 bytes
1 package org.companion.mynicompanion;
2
3 import javafx.fxml.FXML;
4 import javafx.scene.control.Button;
5 import javafx.scene.control.Label;
6 import javafx.scene.control.TableColumn;
7 import javafx.scene.control.TableView;
8 import javafx.scene.control.TextField;
9 import javafx.scene.layout.AnchorPane;
10 import javafx.scene.layout.BorderPane;
11 import javafx.scene.layout.HBox;
12 import javafx.scene.layout.VBox;
13 import javafx.scene.layout.GridPane;
14 import javafx.scene.layout.StackPane;
15 import javafx.scene.layout.Border;
16 import javafx.scene.layout.BorderStroke;
17 import javafx.scene.layout.BorderStrokeStyle;
18 import javafx.scene.layout.BorderWidth;
19 import javafx.scene.layout.CornerRadius;
20 import javafx.scene.layout.Region;
21 import javafx.scene.layout.Spacer;
22 import javafx.scene.layout.StackPane;
23 import javafx.scene.layout.Border;
24 import javafx.scene.layout.BorderStroke;
25 import javafx.scene.layout.BorderStrokeStyle;
26 import javafx.scene.layout.BorderWidth;
27 import javafx.scene.layout.CornerRadius;
28 import javafx.scene.layout.Region;
29 import javafx.scene.layout.Spacer;
30 import javafx.scene.layout.StackPane;
31 import javafx.scene.layout.Border;
32 import javafx.scene.layout.BorderStroke;
33 import javafx.scene.layout.BorderStrokeStyle;
34 import javafx.scene.layout.BorderWidth;
35 import javafx.scene.layout.CornerRadius;
36 import javafx.scene.layout.Region;
37 import javafx.scene.layout.Spacer;
38 import javafx.scene.layout.StackPane;
39 import javafx.scene.layout.Border;
40 import javafx.scene.layout.BorderStroke;
41 import javafx.scene.layout.BorderStrokeStyle;
42 import javafx.scene.layout.BorderWidth;
43 import javafx.scene.layout.CornerRadius;
44 import javafx.scene.layout.Region;
45 import javafx.scene.layout.Spacer;
46 import javafx.scene.layout.StackPane;
47 import javafx.scene.layout.Border;
48 import javafx.scene.layout.BorderStroke;
49 import javafx.scene.layout.BorderStrokeStyle;
50 import javafx.scene.layout.BorderWidth;
51 import javafx.scene.layout.CornerRadius;
52 import javafx.scene.layout.Region;
53 import javafx.scene.layout.Spacer;
54 import javafx.scene.layout.StackPane;
55 import javafx.scene.layout.Border;
56 import javafx.scene.layout.BorderStroke;
57 import javafx.scene.layout.BorderStrokeStyle;
58 import javafx.scene.layout.BorderWidth;
59 import javafx.scene.layout.CornerRadius;
60 import javafx.scene.layout.Region;
61 import javafx.scene.layout.Spacer;
62 import javafx.scene.layout.StackPane;
63 import javafx.scene.layout.Border;
64 import javafx.scene.layout.BorderStroke;
65 import javafx.scene.layout.BorderStrokeStyle;
66 import javafx.scene.layout.BorderWidth;
67 import javafx.scene.layout.CornerRadius;
68 import javafx.scene.layout.Region;
69 import javafx.scene.layout.Spacer;
70 import javafx.scene.layout.StackPane;
71 import javafx.scene.layout.Border;
72 import javafx.scene.layout.BorderStroke;
73 import javafx.scene.layout.BorderStrokeStyle;
74 import javafx.scene.layout.BorderWidth;
75 import javafx.scene.layout.CornerRadius;
76 import javafx.scene.layout.Region;
77 import javafx.scene.layout.Spacer;
78 import javafx.scene.layout.StackPane;
79 import javafx.scene.layout.Border;
80 import javafx.scene.layout.BorderStroke;
81 import javafx.scene.layout.BorderStrokeStyle;
82 import javafx.scene.layout.BorderWidth;
83 import javafx.scene.layout.CornerRadius;
84 import javafx.scene.layout.Region;
85 import javafx.scene.layout.Spacer;
86 import javafx.scene.layout.StackPane;
87 import javafx.scene.layout.Border;
```

19) CreateAsgnController

```
78 lines (65 sloc) 2.33 KB
Raw Blame

1 package org.companion.myunicompanion;
2
3 import javafx.event.ActionEvent;
4 import javafx.fxml.FXML;
5 import javafx.fxml.FXMLLoader;
6 import javafx.fxml.Initializable;
7 import javafx.scene.Node;
8 import javafx.scene.Parent;
9 import javafx.scene.Scene;
10 import javafx.scene.control.*;
11 import javafx.stage.Stage;
12 import org.companion.myunicompanion.classes.Assignment;
13
14 import java.io.IOException;
15 import java.net.URL;
16 import java.util.ResourceBundle;
17
18 import static org.companion.myunicompanion.DatabaseFiller.db;
19
20 public class CreateAsgnController implements Initializable {
21
22     @FXML
23     private Button cancelBtn;
24
25     @FXML
26     private Button createBtn;
27
28     @FXML
29     private TextField name;
30
31     @FXML
32     private TextArea questions;
33
34     @FXML
35     private ChoiceBox<String> typesbox;
36
37     @Override
38     public void initialize(URL url, ResourceBundle resourceBundle) {
39         typesbox.getItems().setAll("H.W.", "Quiz", "Test", "Exam");
40         typesbox.getSelectionModel().selectFirst();
41     }
42
43     @FXML
44     void createAsgn(ActionEvent event) {
45         Alert alert = null;
46         if(!name.getText().isBlank() || questions.getText().isBlank()){
47             Assignment newAsgn = new Assignment();
48             db.incrementAsgnSurKey();
49             newAsgn.setAssignment_ID(db.getAsgnSurKey());
50             newAsgn.setAssignment_name(name.getText());
51             newAsgn.setCourse_ID(db.dataTransporter.get("course id"));
52             newAsgn.setAssignment_type(typesbox.getValue());
53             newAsgn.set_assignment_question(questions.getText());
54             db.assignments.put(newAsgn.getAssignment_ID(), newAsgn);
55
56         }
57         else
58         {
59             alert = new Alert(Alert.AlertType.ERROR, "Please fill all the assignment fields!", ButtonType.OK);
60             alert.showAndWait();
61         }
62     }
63
64     @FXML
65     void returnToCoursesAsgns(ActionEvent e) {
66         Parent coursePage = null;
67         try {
68             coursePage = FXMLLoader.load(LecturerCourseAssignments.class.getResource("lecturer-see course assignments(Scene11).fxml"));
69             Scene homeScene = new Scene(coursePage);
70             Stage window = (Stage) ((Node) e.getSource()).getScene().getWindow();
71             window.setScene(homeScene);
72             window.setTitle("Courses Assignments");
73             window.show();
74         } catch (IOException ex) {
75             ex.printStackTrace();
76         }
77     }
78 }
```


20) Lecturer-create course

90 lines (88 sloc) | 5.01 KB

Raw Blame



```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import javafx.scene.control.Button>
4 <?import javafx.scene.control.ChoiceBox>
5 <?import javafx.scene.control.Label>
6 <?import javafx.scene.control.TextField>
7 <?import javafx.scene.layout.AnchorPane>
8 <?import javafx.scene.layout.ColumnConstraints>
9 <?import javafx.scene.layout.GridPane>
10 <?import javafx.scene.layout.RowConstraints>
11
12 <AnchorPane maxHeight="400.0" maxWidth="600.0" minHeight="400.0" minWidth="600.0" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #c5baa5;" xmlns="http://jav
13 <children>
14 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0" />
15 <GridPane prefHeight="375.0" prefWidth="600.0">
16 <columnConstraints>
17 <ColumnConstraints hgrow="SOMETIMES" maxWidth="294.3999637890625" minWidth="10.0" prefWidth="144.7999725341797" />
18 <ColumnConstraints hgrow="SOMETIMES" maxWidth="457.6000518798828" minWidth="10.0" prefWidth="455.2000274658203" />
19 </columnConstraints>
20 <rowConstraints>
21 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
22 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
23 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
24 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
25 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
26 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
27 </rowConstraints>
28 <children>
29 <AnchorPane prefHeight="200.0" prefWidth="200.0">
30 <children>
31 <Label layoutX="60.0" layoutY="35.0" prefHeight="30.0" prefWidth="94.0" text="Course ID: " />
32 </children>
33 </AnchorPane>
34 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.rowIndex="1">
35 <children>
36 <Label layoutX="60.0" layoutY="35.0" text="Course name: " />
37 </children>
38 </AnchorPane>
39 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.rowIndex="2">
40 <children>
41 <Label layoutX="60.0" layoutY="35.0" text="Course type: " />
42 </children>
43 </AnchorPane>
44 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.rowIndex="3">
45 <children>
46 <Label layoutX="49.0" layoutY="35.0" text="Number of credits: " />
47 </children>
48 </AnchorPane>
49 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.rowIndex="4">
50 <children>
51 <Label layoutX="60.0" layoutY="35.0" text="Year applicable: " />
52 </children>
53 </AnchorPane>
54 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.columnIndex="1">
55 <children>
56 <TextField layoutX="50.0" layoutY="35.0" prefWidth="250.0" />
57 <Label layoutX="106.0" layoutY="6.0" prefHeight="18.0" prefWidth="127.0" text="New Course" textFill="#272b8c" />
58 </children>
59 </AnchorPane>
60 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="1">
61 <children>
62 <TextField layoutX="50.0" layoutY="35.0" prefWidth="250.0" />
63 </children>
64 </AnchorPane>
65 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="2">
66 <children>
67 <ChoiceBox layoutX="50.0" layoutY="35.0" prefWidth="250.0" style="-fx-background-color: #dadada;" />
68 </children>
69 </AnchorPane>
70 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="3">
71 <children>
72 <ChoiceBox layoutX="50.0" layoutY="35.0" prefWidth="250.0" style="-fx-background-color: #dadada;" />
73 </children>
74 </AnchorPane>
75 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="4">
76 <children>
77 <ChoiceBox layoutX="50.0" layoutY="35.0" prefWidth="250.0" style="-fx-background-color: #dadada;" />
78 </children>
79 </AnchorPane>
80 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.rowIndex="5" />
81 <AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="5">
82 <children>
83 <Button layoutX="317.0" layoutY="31.0" mnemonicParsing="false" prefHeight="30.0" prefWidth="111.0" style="-fx-background-color: #e6dcdf;" text="Cancel" />
84 <Button layoutX="100.0" layoutY="31.0" mnemonicParsing="false" prefHeight="30.0" prefWidth="111.0" style="-fx-background-color: #e6dcdf;" text="Create" />
85 </children>
86 </AnchorPane>
87 </children>
88 </GridPane>
89 </children>
90 </AnchorPane>
```

4.2 Reference

1. [Project Lombok](#) for boilerplate codes
2. [JavaFX \(openjfx.io\)](#) For Gui

5. Github repository

[Hazem-Ahmed-Abdelraouf/MyUni-Companion \(github.com\)](https://github.com/Hazem-Ahmed-Abdelraouf/MyUni-Companion)