



Assignment Cover Letter

(Individual Work)

Student Information:

1.

Surname

Lucianto

Given Names

William

Student ID Number

2301890390

Course Code : COMP6510

Course Name : Introduction to Programming

Class : L2AC

Name of Lecturer(s) :Jude Joseph Lamug
Martinez

Major : CS

Title of Assignment :StudentDatabaseManagement
(if any)

Type of Assignment : Final Project

Submission Pattern

Due Date : 20-06-20

Submission Date : 20-06-20

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

(William Lucianto Santoso)

1. William Lucianto Santoso

A handwritten signature in black ink, appearing to read 'William Lucianto Santoso', written in a cursive style.

Table of content

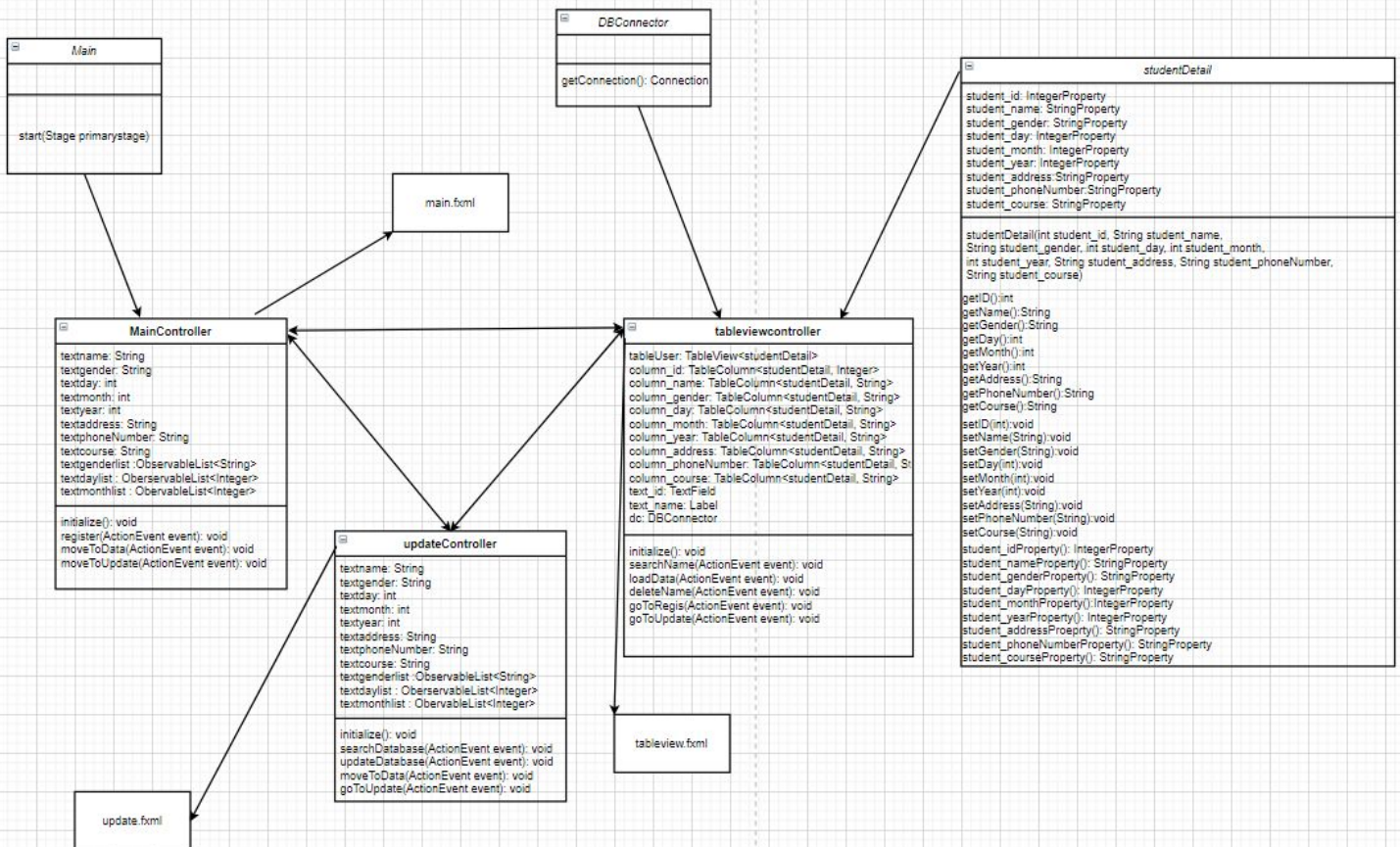
I.	Project Specifications	4
II.	Solution Design	5-15
	A. Class Diagram	5
	B. How it works	6-15
III.	Evidence of Working Programs	16
IV.	Resources	19

Project Specifications

This project is created using eclipse where I install the javafx, scene builder and download the mysql connector jar file before starting this project. I already tried using Jframe for the first time but i think using javafx is more flexible and simple so I started over again using javafx. I had to add the javafx library and add an external jar file in the build source so I can connect the database to my program and to create the database I use the mysql workbench so I can easily create and check my data before continuing to create the program.

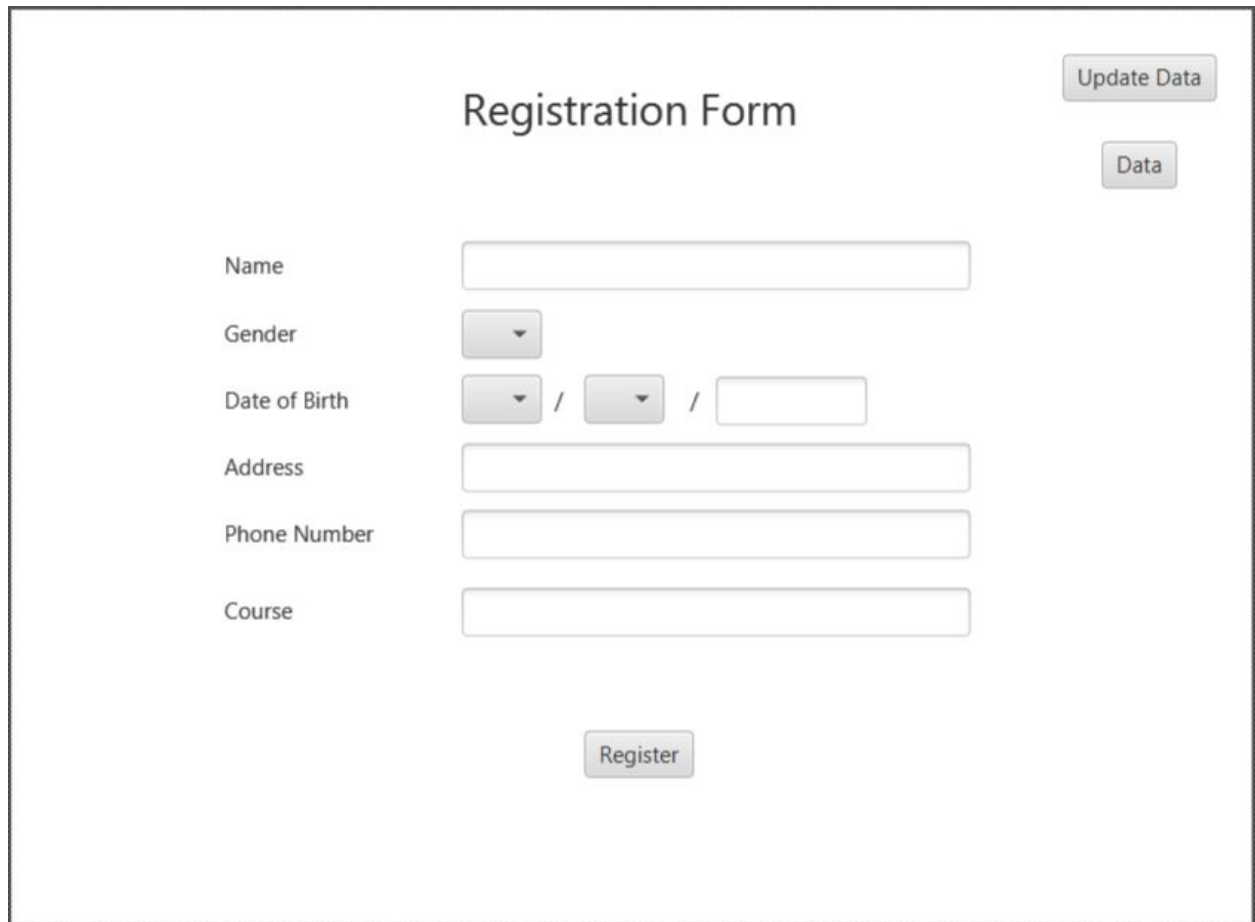
For this final project, I created a database system that stores student information using mysql and for better user experience I use javafx to make a GUI(Graphic User Interface). This program can register a student along with his information and the program will automatically assign the student with an id so we can use the id to search the student from the database and then we can change their information if needed or maybe delete the student data from the database. I decide to make this program because last semester I also create a database program for item in python with an extra function for cashier, so because I have little experience about it i choose to make a database program and try to make the app to able to show the data because my python program can't show the data and need to use the database application to see the data.

Solution design



In this project there are 3 main fxml files since my program has a total of 3 windows with each have different functions, the 3 functions are registration form, search & update, and table that shows all the data and the use of the id to delete the data inside the database. To create this fxml I use SceneBuilder because it is more convenient and it is really simple.

Main.fxml



The image shows a user interface for a "Registration Form". The title "Registration Form" is centered at the top. In the top right corner, there are two buttons: "Update Data" and "Data". The form fields are arranged vertically on the left side, each with a corresponding input field on the right:

- Name:** A single-line text input field.
- Gender:** A dropdown menu.
- Date of Birth:** Three dropdown menus separated by slashes, followed by a single-line text input field for the day.
- Address:** A single-line text input field.
- Phone Number:** A single-line text input field.
- Course:** A single-line text input field.

At the bottom center of the form, there is a button labeled "Register".

This is the first one where you can fill in the data and then press register to add this student into the student database. The two buttons in the upper right is to change window to the another program.

update.fxml

Update Database

Add Data
Data

ID

Search

Name

Gender

▼

Date of Birth

▼

 /

▼

 /

Address

Phone Number

Course

Update

This is the second program where you can fill in the text field in ID and then when you press the search button it will automatically fill in all the data if the ID is found on the database. Once the data is filled you can change the name, gender, date of birth, address, phone number, and course with exception of ID to ensure us that there will be no same id will be created. Just like the first program the two buttons in upper right is to go to the other program window.

tableView.fxml

ID	Name	Gender	day	Month	Year	Address	PhoneNumber	Course
No content in table								

Add Data

Update Data

Load Data

Delete Data by ID

ID

Search ID

Name

Delete

This is the third program where there is a huge table where it will show all the data you have in the database by pressing the load data where it will connect to our database. The left lower button it to go to other window program which is the previous two and the bottom right is a new function i add to delete the data in the database by fill in the id and then use the search button that will change the “name” label into the name inside of the data where his/her id is the one we searched, and then you can press the delete button to remove the data from the database.

Each of this fxml have its own controller where inside the controller will be the function for the button.

1. MainController.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
```

This controller import is going to import all of this where the upper part is for the mysql database and the bottom part is for the javafx.

```
public class MainController {
    //the choice box selection
    ObservableList<String> textgenderlist = FXCollections.observableArrayList("M","F");
    ObservableList<Integer> textdaylist = FXCollections.observableArrayList(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31);
    ObservableList<Integer> textmonthlist = FXCollections.observableArrayList(1,2,3,4,5,6,7,8,9,10,11,12);

    @FXML
    public TextField textname;
    @FXML
    public ChoiceBox<String> textgender;
    @FXML
    public ChoiceBox<Integer> textday;
    @FXML
    public ChoiceBox<Integer> textmonth;
    @FXML
    public TextField textyear;
    @FXML
    public TextField textaddress;
    @FXML
    public TextField textphoneNumber;
    @FXML
    public TextField textcourse;
```

In this main controller I call the textfield, choice box and then the observablelist which inside will be all the string or integer we are going to input into the choice box so the user will have to pick one of them for their data. This textname, textgender, and so on is created from SceneBuilder where you can set the fx:id for each of the components.

```
//at the start of running this program it will run this code that will add the choice box selection to the choice box
@FXML
private void initialize() {
    textgender.setItems(textgenderlist);
    textday.setItems(textdaylist);
    textmonth.setItems(textmonthlist);
}
```

This part is going to be initialized for the choice box in fxml with the list where i put all the answers you can select from the previous part of the code.

```
//add the filled in information of student into the the database
public void register(ActionEvent actionEvent) {
    try {
        //connecting to the database
        Class.forName("com.mysql.jdbc.Driver");
        Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb","root","password");
        //the "?" is the value that i inserted in below this statement which mean the student_name is textname.getText(), and so on.
        PreparedStatement ps = connection.prepareStatement("insert into student(student_name,student_gender,student_day,"
            + "student_month,student_year,student_address,student_phonenumber,student_course) values(?,?,?,?,?,?,?,?)");
        ps.setString(1, textname.getText());
        ps.setObject(2, textgender.getValue());
        ps.setObject(3, textday.getValue());
        ps.setInt(4, textmonth.getValue());
        ps.setInt(5, Integer.parseInt(textyear.getText()));
        ps.setString(6, textaddress.getText());
        ps.setString(7, textphoneNumber.getText());
        ps.setString(8, textcourse.getText());
        int x = ps.executeUpdate();
        if(x>0) {
            System.out.println("Registration success");
        }else {
            System.out.println("failed");
        }
    } catch (Exception e1) {
        System.out.println(e1);
    }
}
```

In this function I connect to my database using the “root” as the username and “password” as password, and then in the PreparedStatement i connect it with a sql statement that tell the program to insert the data into the student database with the values that we going to insert with the string / integer in the textfield / choicebox. For the textfield you can;t get integer but instead you can convert the string into an integer which is why there is Integer.parseInt in the year data. use the executeUpdate() to execute the change into the database.

```

}
//change window to the TableView.fxml
public void moveToData(ActionEvent event) {
    try {
        Parent root1 = FXMLLoader.load(getClass().getResource("tableView.fxml"));
        Scene scene = new Scene(root1);
        Stage stage = new Stage();
        stage.setScene(scene);
        stage.show();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
//change window to the update.fxml
public void moveToUpdate(ActionEvent event) {
    try {
        Parent root2 = FXMLLoader.load(getClass().getResource("update.fxml"));
        Scene scene = new Scene(root2);
        Stage stage = new Stage();
        stage.setScene(scene);
        stage.show();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

This two functions are used to change the window, from the Main.fxml to tableView.fxml or update.fxml.

2. updateController.java

```

}
public void searchDataBase(ActionEvent event) {
    try {
        text_name.clear();
        text_year.clear();
        text_address.clear();
        text_phoneNumber.clear();
        text_course.clear();
        Class.forName("com.mysql.jdbc.Driver");
        Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb","root","password");
        PreparedStatement ps = connection.prepareStatement("select * from student where student_id = ?");
        ps.setInt(1, Integer.parseInt(text_id.getText()));
        ResultSet rs = ps.executeQuery();
        while(rs.next()) {
            String setName = rs.getString("student_name");
            text_name.setText(setName);
            String setGender = rs.getString("student_gender");
            text_gender.setValue(setGender);
            int setDay = rs.getInt("student_day");
            text_day.setValue(setDay);
            int setMonth = rs.getInt("student_month");
            text_month.setValue(setMonth);
            String setYear = rs.getString("student_year");
            text_year.setText(setYear);
            String setAddress = rs.getString("student_address");
            text_address.setText(setAddress);
            String setPhoneNumber = rs.getString("student_phoneNumber");
            text_phoneNumber.setText(setPhoneNumber);
            String setCourse = rs.getString("student_course");
            text_course.setText(setCourse);
        }
    } catch (Exception e1) {}
    System.out.println(e1);
}
}
}

```

For this updateController the inside is mostly the same as the main.java the only main difference is in the function because this function we are going to use ResultSet so we need to import java.sql.ResultSet.

Just in case I clear the textField first and then once again connect to the database. This time the statement we going to use is “select *” which means it will get all the information(name,gender,day,month,year,address,phone number and course) and “where the student_id=?” which means we are going to get the data from the one with the id = ? and then we are going to get the id from the textField and put it into the statement and execute the query. After getting all the data we need from the database we are going to loop the data and get the data from the ResultSet and then put it into the textField.

```
//if you want to change the data / update we can use this function to update the data and replace the original data
public void updateDatabase(ActionEvent event) {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb","root","password");
        PreparedStatement ps = connection.prepareStatement("update student set student_name=?, student_gender=?, student_day=?, "
            + "student_month=?, student_year=?, student_address=?, student_phoneNumber = ?, student_course=? where student_id = ?");
        ps.setString(1, text_name.getText());
        ps.setString(2, text_gender.getText());
        ps.setInt(3, text_day.getValue());
        ps.setInt(4, text_month.getValue());
        ps.setString(5, text_year.getText());
        ps.setString(6, text_address.getText());
        ps.setString(7, text_phoneNumber.getText());
        ps.setString(8, text_course.getText());
        ps.setInt(9, Integer.parseInt(text_id.getText()));
        int x = ps.executeUpdate();
        if(x>0) {
            System.out.println("Update success");
        }else {
            System.out.println("Failed");
        }
    }
    catch(Exception e2){System.out.println(e2);}
}
```

This function is really close with the add data into the database from the MainController.java, the main difference is in the sql statement because we are going to update and not insert. everything else is the same but in this one i tried to use another method which didn't make any different (from “(student_id) values=(?)” into “(student_id=?)”).

3. DBConnector.java, studentDetail.java & tableviewcontroller.java

The last fxml file which is tableView.fxml is the most confusing one in my program because to run this I use 3 different java files to run it. To complete this last function I had to watch several 30-40 minutes videos on youtube just to understand how it works. I really wanted to finish this last function because this is the function I regret I didn't make in last semester python.

DBConnector.java

```
import java.sql.Connection;

public class DBConnector {
    public Connection getConnection() throws SQLException {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb","root","password");

            return connection;
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return null;
    }
}
```


For the first class is the DBConnector, I need this function for the later part to use the resourceBundle.

studentDetail.java

```
import javafx.beans.property.*;

public class studentDetail {

    private final IntegerProperty student_id;
    private final StringProperty student_name;
    private final StringProperty student_gender;
    private final IntegerProperty student_day;
    private final IntegerProperty student_month;
    private final IntegerProperty student_year;
    private final StringProperty student_address;
    private final StringProperty student_phoneNumber;
    private final StringProperty student_course;

    public studentDetail(int student_id, String student_name, String student_gender, int student_day,
        int student_month, int student_year, String student_address, String student_phoneNumber, String student_course) {
        this.student_id = new SimpleIntegerProperty(student_id);
        this.student_name = new SimpleStringProperty(student_name);
        this.student_gender = new SimpleStringProperty(student_gender);
        this.student_day = new SimpleIntegerProperty(student_day);
        this.student_month = new SimpleIntegerProperty(student_month);
        this.student_year = new SimpleIntegerProperty(student_year);
        this.student_address = new SimpleStringProperty(student_address);
        this.student_phoneNumber = new SimpleStringProperty(student_phoneNumber);
        this.student_course = new SimpleStringProperty(student_course);
    }
}
```

For the second class it's the studentDetail where the data here is for the tableview data. I put all the data (id,name,gender,day,month,year,address,phone number, course) with the javafx.beans.property and create the SimpleStringProperty / SimpleIntegerProperty.

```
public int getID() {return student_id.get();}
public String getName() {return student_name.get();}
public String getGender() {return student_gender.get();}
public int getDay() {return student_day.get();}
public int getMonth() {return student_month.get();}
public int getYear() {return student_year.get();}
public String getAddress() {return student_address.get();}
public String getPhoneNumber() {return student_phoneNumber.get();}
public String getCourse() {return student_course.get();}

public void setID(int x) {student_id.set(x);}
public void setName(String x) {student_name.set(x);}
public void setGender(String x) {student_gender.set(x);}
public void setDay(int x) {student_day.set(x);}
public void setMonth(int x) {student_month.set(x);}
public void setYear(int x) {student_year.set(x);}
public void setAddress(String x) {student_address.set(x);}
public void setPhoneNumber(String x) {student_phoneNumber.set(x);}
public void setCourse(String x) {student_course.set(x);}

public IntegerProperty student_idProperty() {return student_id;}
public StringProperty student_nameProperty() {return student_name;}
public StringProperty student_genderProperty() {return student_gender;}
public IntegerProperty student_dayProperty() {return student_day;}
public IntegerProperty student_monthProperty() {return student_month;}
public IntegerProperty student_yearProperty() {return student_year;}
public StringProperty student_addressProperty() {return student_address;}
public StringProperty student_phoneNumberProperty() {return student_phoneNumber;}
public StringProperty student_courseProperty() {return student_course;}
```

Create more functions for to set and name and the Integer / StringProperty.

tableviewController.java

```
public class tableviewController implements Initializable {
    @FXML
    private TableView<studentDetail> tableUser;
    @FXML
    private TableColumn<studentDetail, Integer> column_id;
    @FXML
    private TableColumn<studentDetail, String> column_name;
    @FXML
    private TableColumn<studentDetail, String> column_gender;
    @FXML
    private TableColumn<studentDetail, Integer> column_day;
    @FXML
    private TableColumn<studentDetail, Integer> column_month;
    @FXML
    private TableColumn<studentDetail, Integer> column_year;
    @FXML
    private TableColumn<studentDetail, String> column_address;
    @FXML
    private TableColumn<studentDetail, String> column_phoneNumber;
    @FXML
    private TableColumn<studentDetail, String> column_course;
    @FXML
    private TextField text_id;
    @FXML
    private Label text_name;

    ObservableList<studentDetail> data = FXCollections.observableArrayList();

    private DBConnector dc;
```

Create the table view and the table column using SceneBuilder and add the fx:id for each component. For the table view function the TextField and Label is not going to be used but instead it going to use the ObservableList where we put the studentDetail class as the parameter and use the FXCollection to make it an array list.

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    dc = new DBConnector();
}
```

This dc is only going to be used in the tableview which is why we use the ResourceBundle as the parameter.

```

@FXML
private void loadData(ActionEvent event) {
    try {
        Connection conn = dc.getConnection();
        data = FXCollections.observableArrayList();
        ResultSet rs = conn.createStatement().executeQuery("select * from student");
        while(rs.next()) {
            data.add(new studentDetail(rs.getInt(1), rs.getString(2), rs.getString(3),
                rs.getInt(4), rs.getInt(5), rs.getInt(6), rs.getString(7), rs.getString(8), rs.getString(9)));
        }
    } catch (SQLException ex) {
        System.err.println("Error" + ex);
    }

    column_id.setCellValueFactory(new PropertyValueFactory<>("student_id"));
    column_name.setCellValueFactory(new PropertyValueFactory<>("student_name"));
    column_gender.setCellValueFactory(new PropertyValueFactory<>("student_gender"));
    column_day.setCellValueFactory(new PropertyValueFactory<>("student_day"));
    column_month.setCellValueFactory(new PropertyValueFactory<>("student_month"));
    column_year.setCellValueFactory(new PropertyValueFactory<>("student_year"));
    column_address.setCellValueFactory(new PropertyValueFactory<>("student_address"));
    column_phoneNumber.setCellValueFactory(new PropertyValueFactory<>("student_phoneNumber"));
    column_course.setCellValueFactory(new PropertyValueFactory<>("student_course"));

    tableUser.setItems(null);
    tableUser.setItems(data);
}

```

For this function I use the dc from the DBConnector and I going to use sqlStatement that get all the data in the database so I use the “select * from student” and then add them to the data which is an observableArrayList with using resultSet. After we got all the data I set the data into a table column by creating a new PropertyValueFactory and then set the data to the tableUser which is table view.

Program Result Register

— □ ×

Registration Form

Update Data

Data

Name

brian

Gender

M ▾

Date of Birth

13 ▾ / 6 ▾ / 2000

Address

jalan kemanggisan

Phone Number

08123011231

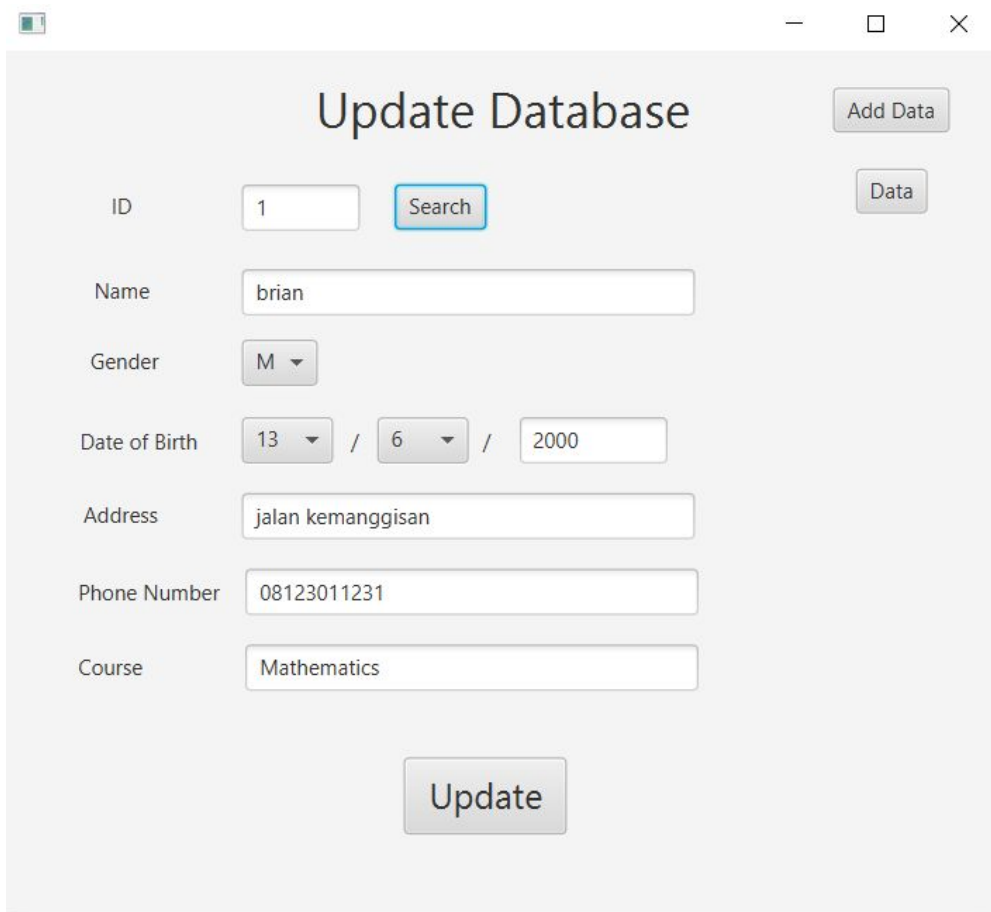
Course

Mathematics

Register

[illegible]

Search



The screenshot shows a web application window titled "Update Database". In the top right corner, there are two buttons: "Add Data" and "Data". The form contains several input fields: "ID" with the value "1", "Name" with "brian", "Gender" with a dropdown menu showing "M", "Date of Birth" with dropdowns for "13", "6", and a text field for "2000", "Address" with "jalan kemanggisan", "Phone Number" with "08123011231", and "Course" with "Mathematics". A blue "Search" button is highlighted next to the ID field. A large "Update" button is at the bottom center.

Update Database

Add Data

Data

ID 1 Search

Name brian

Gender M

Date of Birth 13 / 6 / 2000

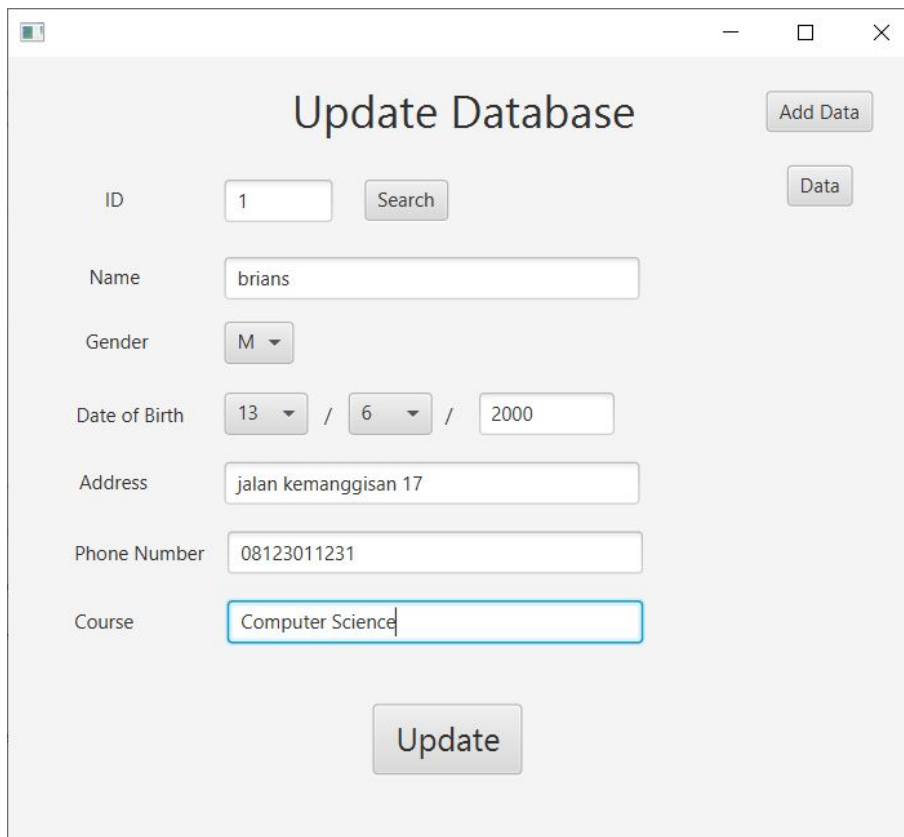
Address jalan kemanggisan

Phone Number 08123011231

Course Mathematics

Update

Update



The screenshot shows the same "Update Database" form, but with updated values. The "Name" field now contains "brians", the "Address" field contains "jalan kemanggisan 17", and the "Course" field contains "Computer Science". The "Search" button is no longer highlighted. The "Update" button remains at the bottom center.

Update Database

Add Data

Data

ID 1 Search

Name brians

Gender M

Date of Birth 13 / 6 / 2000

Address jalan kemanggisan 17

Phone Number 08123011231

Course Computer Science

Update

ID	Name	Gender	day	Month	Year	Address	PhoneNumber	Course
1	brians	M	13	6	2000	jalan kemanggisan 17	08123011231	Computer Science

Deletion

Delete Data by ID

ID

brians

ID	Name	Gender	day	Month	Year	Address	PhoneNumber	Course
No content in table								

Add Data

Update Data

Delete Data by ID

ID

name

Resources

- https://www.youtube.com/watch?v=oVn6_2KuYbM
Setting Up Java FX 13 with Eclipse
- <https://www.youtube.com/watch?v=2PxU7q9xl38>
JavaFX with Eclipse and Scene Builder in 4 min
- <https://www.youtube.com/watch?v=ThTgN90PA44>
How to connect Java Project with MySQL Workbench
- <https://www.youtube.com/watch?v=Phog21DF4xk>
JavaFX Tutorial 1: Installation and Hello World with Scene Builder
- https://www.youtube.com/watch?v=uh5R7D_vFto
Adding and populating a TableView object using SceneBuilder