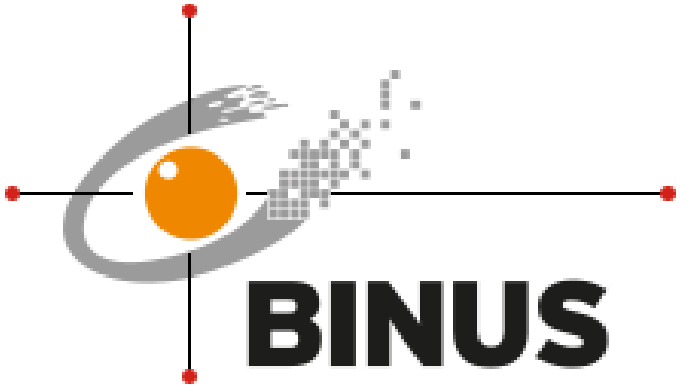
Programming Languages

PROJECT REPORT

Binus University/School registration form simulation



BINUS
UNIVERSITY
INTERNATIONAL

Prepared by:
Stephanus Jovan Novarian 2201832856

Even semester 2018/2019
Computer Science Program
Binus University International

Table of Content

Introduction

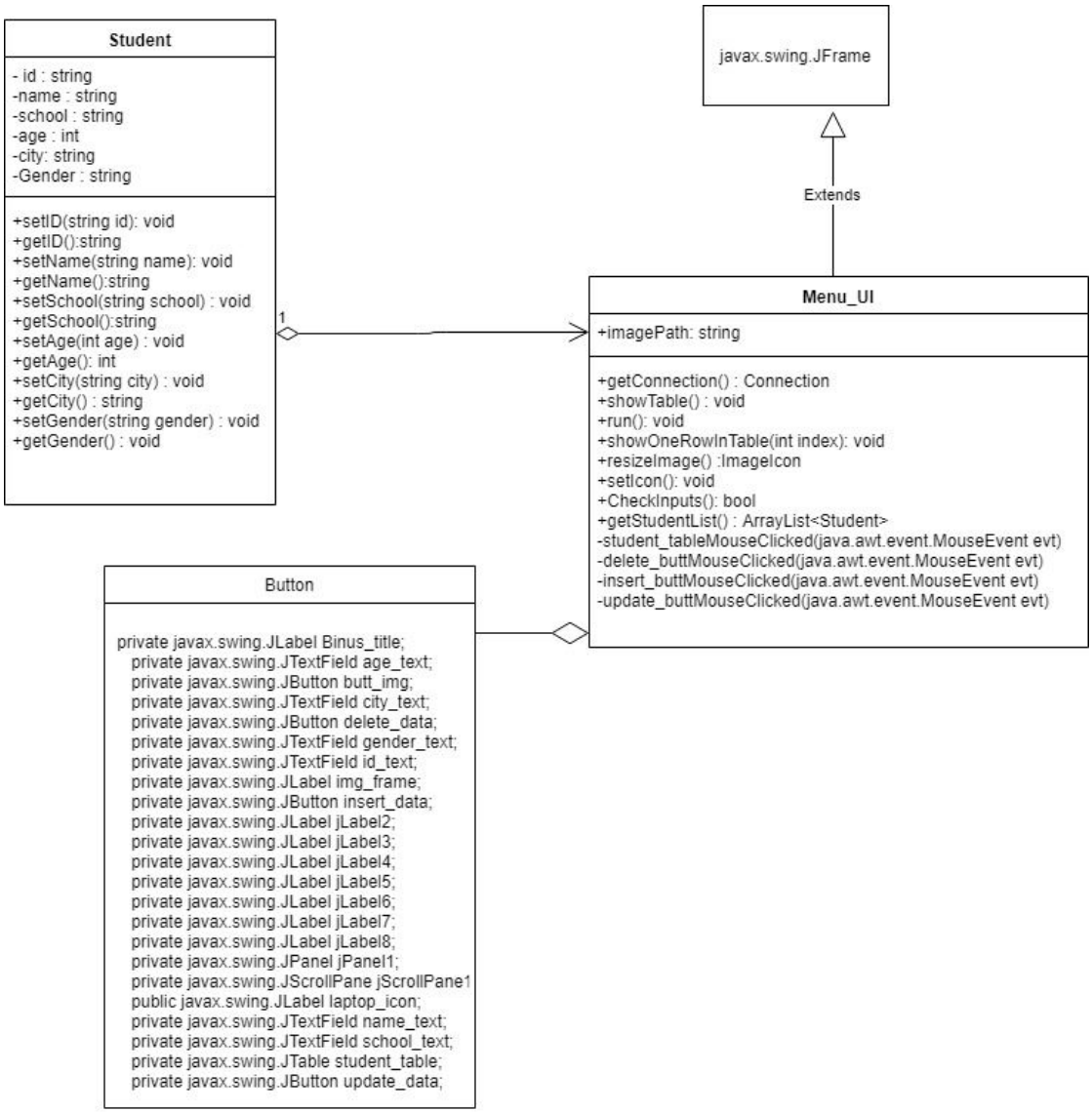
Java is one of the programming languages that has been commonly used by most of the programmers for making several programs related to basically anything, mostly for making different kinds of applets with different kinds of usages. In this semester we knew a lot of java imports in order for us to create lots of function to let the program to do other than process in terminals. In this case Javax.swing is one of the imports that is always used for creating an applet , a simple interface and so on. Based on the Internet the swing is toolkit for making a Graphical UI, and JFrame is one of the components inside the import of javax.swing.

In this final project I connected my JFrame to a SQL (Query) database and my aim is to create a simple registration form simulation specifically for lots of student who is aiming to study in our major which is computer science. Help them to understand how a simple registration form works or maybe how to create a form on their own.

Class Diagram

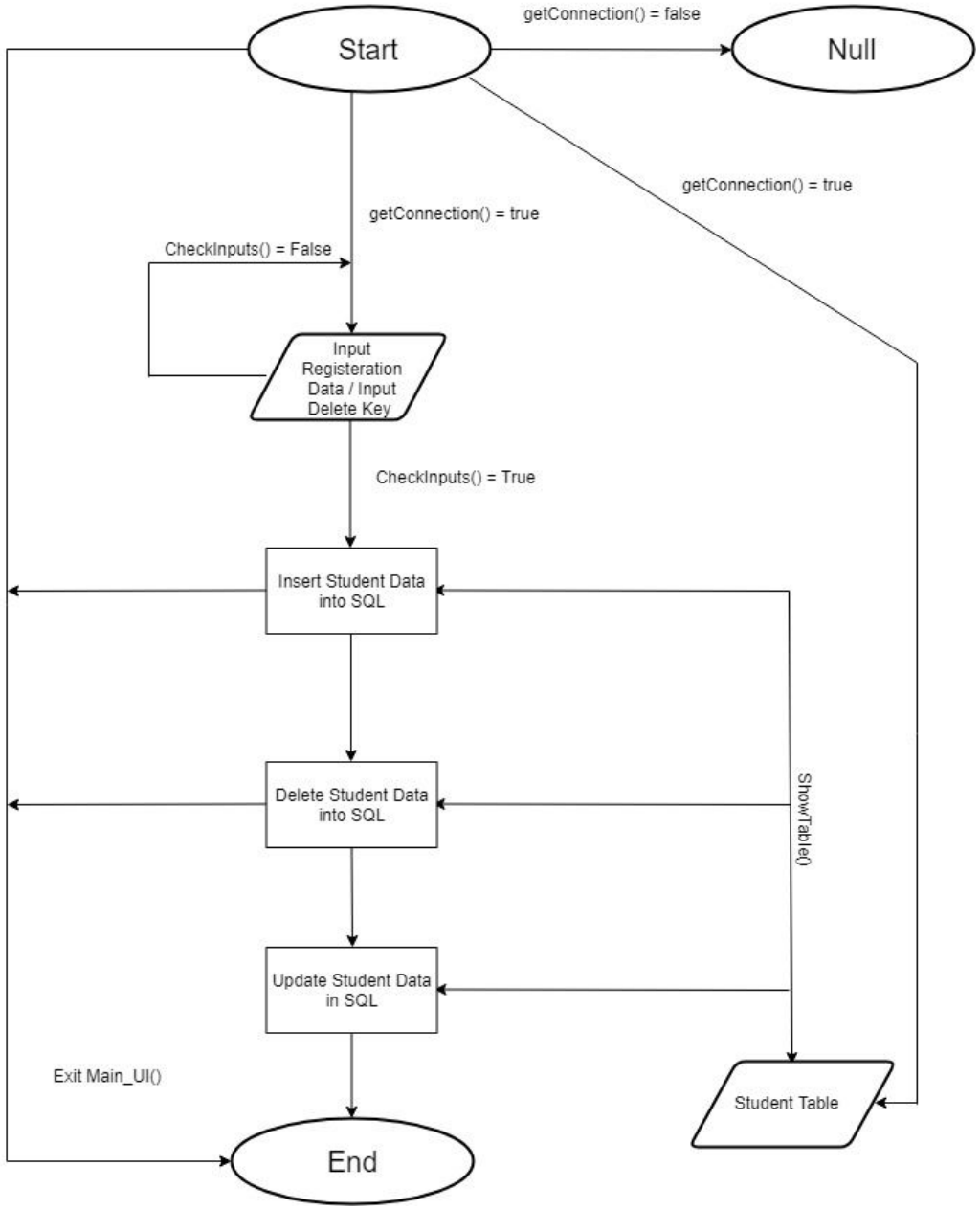
In this final project i created only 2 classes, one is the encapsulation of student data and the other class implements java swing JFrame to create the user interface for my simple registration form simulation. Technique has been used for creating the project is Encapsulation, Inheritance from Java Swingx module, Dynamic polymorphism for override function and etc.

Here is my class diagram:



Flowchart

Attached below is the generalized flowchart for my final project.



Class Explanation

Student Class

```
public class Student {
    private String ID;
    private String name;
    private String gender;
    private int age;
    private String city;
    private String School;

    public Student(String ID, String name, String gender, int age, String city, String School) {
        this.ID = ID;
        this.name = name;
        this.gender = gender;
        this.age = age;
        this.city = city;
        this.School = School;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getID() {
        return ID;
    }
```

```
public String getID() {
    return ID;
}

public void setID(String ID) {
    this.ID = ID;
}

public String getGender() {
    return gender;
}

public void setGender(String gender) {
    this.gender = gender;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public String getCity() {
    return city;
}
```

```
    public void setCity(String city) {
        this.city = city;
    }

    public String getSchool() {
        return School;
    }

    public void setSchool(String School) {
        this.School = School;
    }

}
```

This is the class that I used for getting a readable and writable data to be stored inside the student arraylist which will be then showed into a table (showTable()). This technique is called the encapsulation where data is wrapped together inside one container, and to be accessed inside the Menu_UI Class JFrame.

Import List

```
import java.awt.Image;
import java.awt.Toolkit;
import java.io.File;
import java.io.FileInputStream;
//import java.io.FileNotFoundException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileNameExtensionFilter;
```

```
import javax.swing.table.DefaultTableModel;
```

Button List

```
 private javax.swing.JLabel Binus_title;
    private javax.swing.JTextField age_text;
    private javax.swing.JButton butt_img;
    private javax.swing.JTextField city_text;
    private javax.swing.JButton delete_data;
    private javax.swing.JTextField gender_text;
    private javax.swing.JTextField id_text;
    private javax.swing.JLabel img_frame;
    private javax.swing.JButton insert_data;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    public javax.swing.JLabel laptop_icon;
    private javax.swing.JTextField name_text;
    private javax.swing.JTextField school_text;
    private javax.swing.JTable student_table;
    private javax.swing.JButton update_data;
```

Main_UI Class

```
private void setIcon()
{
    setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().getResource("binus.png")));
}

public Menu_UI() {
    initComponents();
    setIcon();
    showTable();
    //getConnection();
}

public Connection getConnection(){
    Connection con = null;
    try {
        con = DriverManager.getConnection("jdbc:mysql://localhost/db_students","root","");
        return con;
    } catch (SQLException ex) {
        Logger.getLogger(Menu_UI.class.getName()).log(Level.SEVERE, null, ex);
        return null;
    }

}
```

setIcon( ) → setIcon function is for setting the JFrame browser icon from a default JFrame icon to an image that the user uploads in the java file code package.

Menu_UI( ) → Main user interface for the project, contains functions in order user to view inside the JFrame user interface. (code will be explained below)

getConnection() → In this code , Connection import from java is for me to connect my sql to the JFrame , data inserted towards the table later on will be stored inside mySQL database, in this project the database name is call db_students with the table name student_list. Catch the error if the SQL is not active and return nothing inside the JFrame.

```
String imagePath = null;
// resizing images
public ImageIcon resizeImage(String imagePath,byte[] pict){
    ImageIcon myImg = null;
    if (imagePath != null)
    {
        myImg = new ImageIcon(imagePath);
    }
    else
    {
        myImg = new ImageIcon(pict);
    }
    Image img = myImg.getImage();
    Image img2 = img.getScaledInstance(img_frame.getWidth(), img_frame.getHeight(), Image.SCALE_SMOOTH);
    ImageIcon Image =  new ImageIcon(img2);
    return Image;
}
```

ResizeImage() : ImageIcon → this function was coming from the import called ImageIcon in order to let programmer to do their code for adding , uploading images etc. Image that has been uploaded by the user set into the size of the JFrame Label, in other words, image is resized in order to match the length or the Label.

```
public boolean checkInputs()
{
    if (id_text.getText()== null
            || name_text.getText() == null
            || age_text.getText() == null
            || city_text.getText()== null
            || school_text.getText()== null
            || gender_text.getText()== null)
    {
        return false;
    }
    else
    {
        try{
            Float.parseFloat(id_text.getText());
            return true;
        }catch(Exception ex){
            return false;
        }
    }
}
```

checkInputs() : bool → this function is basically for letting the java check the data inputs that the user has been inputted inside the JFrame, return false if one of the data is not entered correctly or the data still empty. Return true after the ID has been converted return false to catch exception error message.

```
// to fill the student data inside an Array
public ArrayList<Student> getStudentList()
{
    ArrayList<Student> sl = new ArrayList<Student>();
    Connection con = getConnection();
    String query_db = "SELECT * FROM student_list";
    Statement st;
    ResultSet rs;

    try
    {
        st = con.createStatement();
        rs = st.executeQuery(query_db);
        Student stud;
        while (rs.next())
        {
            stud = new Student(rs.getString("ID"),rs.getString("name"),rs.getString("gender"),
                    Integer.parseInt(rs.getString("age")),rs.getString("city"),rs.getString("school"));
            sl.add(stud);

        }

    } catch (SQLException ex)
    {
        Logger.getLogger(Menu_UI.class.getName()).log(Level.SEVERE, null, ex);
    }
    return sl; // sl = student list
}
```

ArrayList<Student> getStudentList() : → this function is to set and get the Student constructor inside the Student Class into an ArrayList towards the SQL by using Statement and Result import from Java Package. Return student_list after the process finishes.

```
// To show the Student inside the Table
public void showTable() // show any contents from the sql inside the table
{
    // arraylist for student
    ArrayList<Student> studentlist = getStudentList();
    // for table model
    DefaultTableModel tablemodel = (DefaultTableModel)student_table.getModel();
    tablemodel.setRowCount(0); // set the row count into 0 so it will not loop again
    Object[] row = new Object[6];
    for (int i = 0;i<studentlist.size();i++)
    {
        row[0] = studentlist.get(i).getID();
        row[1] = studentlist.get(i).getName();
        row[2] = studentlist.get(i).getGender();
        row[3] = studentlist.get(i).getAge();
        row[4] = studentlist.get(i).getCity();
        row[5] = studentlist.get(i).getSchool();
        tablemodel.addRow(row); // too add rows inside the java table
    }
}
```

showTable() : void → this function is to get the student data form into an Array list which then copied into the table inside the Main_User interface.

```
public void showOneRowItemInTable(int index) // show(highlight) item based on one row from the table
{
    id_text.setText(getStudentList().get(index).getID());
    gender_text.setText(getStudentList().get(index).getGender());
    age_text.setText(Integer.toString(getStudentList().get(index).getAge()));
    city_text.setText(getStudentList().get(index).getCity());
    school_text.setText(getStudentList().get(index).getSchool());
    name_text.setText(getStudentList().get(index).getName());
}
```

showOneRowItemInTable(int index) : void → To show one row of table by clicking into the row that has been selected by the user. (see student_tableMouseClicked(event)), text is set based on the order of the table.

```
private void butt_imgActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser file = new JFileChooser();
    file.setCurrentDirectory(new File(System.getProperty("user.home")));

    FileNameExtensionFilter filter = new FileNameExtensionFilter("*.images","jpg","png");

    file.addChoosableFileFilter(filter);
    int result = file.showSaveDialog(null);

    if(result == JFileChooser.APPROVE_OPTION)
    {
        File selectedFile = file.getSelectedFile();
        String path = selectedFile.getAbsolutePath();
        img_frame.setIcon(resizeImage(path,null));
        imagePath = path;

    }
    else
    {
        System.out.println("No Image File Selected");
    }
}
```

butt_img.ActionPeformed(event) :void → function of action on click, butt_img is a button that is user for uploading image inside the JFrame. User will be directed to the his own File explorer and select an image with the path jpg or png, image will then resized based on the function (resizeImage()), if no image selected print out “No image file selected”.

```
private void insert_dataActionPerformed(java.awt.event.ActionEvent evt) {

    if (checkInputs()&& imagePath != null)
    {
        Connection con = getConnection();
        try {
            PreparedStatement ps = con.prepareStatement("INSERT INTO student_list(ID,Name,Gender,Age,City,School)"
                    + "values(?,?,?,?,?,?);");
            ps.setString(1, id_text.getText());
            ps.setString(2, name_text.getText());
            ps.setString(3, gender_text.getText());
            ps.setString(4, age_text.getText());
            ps.setString(5, city_text.getText());
            ps.setString(6, school_text.getText());
            ps.executeUpdate();
            showTable();
            JOptionPane.showMessageDialog(null, "Student Data has been inserted");
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
        }
    }
    else
    {
        JOptionPane.showMessageDialog(null, "Check your fields please, Thank you");
    }
}
```

insert_data.ActionPerformed(event) : void → function for inserting the data into mySQL by clicking the button confirm inside the main user interface, PreparedStatement java import has been used as a script to input all of the data one by one, then the data will be shown on the table after update execution. Catch an error message with pane if checkInputs() : bool is false.

```
private void delete_dataActionPerformed(java.awt.event.ActionEvent evt) {

    if (!id_text.getText().equals("")){

        try {
            Connection con = getConnection();
            PreparedStatement ps = con.prepareStatement("DELETE from student_list WHERE ID = ?");
            int id = Integer.parseInt(id_text.getText());
            ps.setInt(1, id);
            ps.executeUpdate();
            JOptionPane.showMessageDialog(null, "Student Deleted");
        } catch (SQLException ex) {
            Logger.getLogger(Menu_UI.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(null, "Delete Fail");
        }
    }
    else
    {
        JOptionPane.showMessageDialog(null, "Delete Fail , No Primary Key selected");
    }

}
```

delete_data.ActionPerformed(event) :void → Delete button for delete one pair of student data inside the SQL by entering the student's ID since ID is the Primary key inside the SQL and it will be easier to look for numbers rather than other data which is string.

```
private void update_dataActionPerformed(java.awt.event.ActionEvent evt) {
    if(checkInputs() && id_text.getText()!= null)
    {
        String UpdateQuery = null;
        PreparedStatement ps = null;
        Connection con = getConnection();
        // update without the images
        if (imagePath == null)
        {
            try {
                UpdateQuery = "UPDATE student_list SET Name = ?,Gender = ?"
                    + ",Age = ?,City = ?,School = ? WHERE ID = ?";
                ps = con.prepareStatement(UpdateQuery);

                ps.setString(1, name_text.getText());
                ps.setString(2, gender_text.getText());
                ps.setString(3, age_text.getText());
                ps.setString(4, city_text.getText());
                ps.setString(5, school_text.getText());
                ps.setString(6, id_text.getText());

                ps.executeUpdate();
                showTable();
                JOptionPane.showMessageDialog(null, "Data successfuly Updated");

            } catch (SQLException ex) {
                Logger.getLogger(Menu_UI.class.getName()).log(Level.SEVERE, null, ex)
```

```
    // this is the update with the images
    else
    {
        try {
            InputStream img = new FileInputStream(new File(imagePath));
            UpdateQuery = "UPDATE student_list SET Name = ?,Gender = ?"
                + ",Age = ?,City = ?,School = ? WHERE ID = ?";
            ps = con.prepareStatement(UpdateQuery);
            ps.setString(1, name_text.getText());
            ps.setString(2, gender_text.getText());
            ps.setString(3, age_text.getText());
            ps.setString(4, city_text.getText());
            ps.setString(5, school_text.getText());
            ps.setString(6, id_text.getText());

            ps.executeUpdate();
            JOptionPane.showMessageDialog(null, "Successfully Updated");

        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
        }
    }
}
else
{
    JOptionPane.showMessageDialog(null, "Please check your data input");
}
```

update_data.ActionPerformed(event): void → The update button is for updating one student data that exists inside my SQL, two loops for updating one's data with Image and without image, text for query update has been

```
    private void student_tableMouseClicked(java.awt.event.MouseEvent evt) {
        int rowOnClick = student_table.getSelectedRow();
        showOneRowItemInTable(rowOnClick);
    }
```

student_tableMouseClicked (event) : void → To show one group of data inside the row by clicking the selected row.

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
     Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Menu_UI().setVisible(true);
        }
    });
}
```

run() :void → To set the menu_ui interface viewable for users to do tasks for the JFrame, it will not be visible by the user if the parameter of setVisible is false.

Difficulties

Although this project looks simple after user runs it but to code this project was not an easy task for me as a Computer Science student. There are several difficulties that I have encountered during the making of this project and one of them was to separate different functions into a different file, some of the functions I need to make it in the same file in order for it to run in order for example setIcon() (see page 9). I tried to read some tutorials about the way to set but it did not work when i want to separate that function from the main. So to encounter this problem obviously to make those function in the Main in order to run the UI correctly.

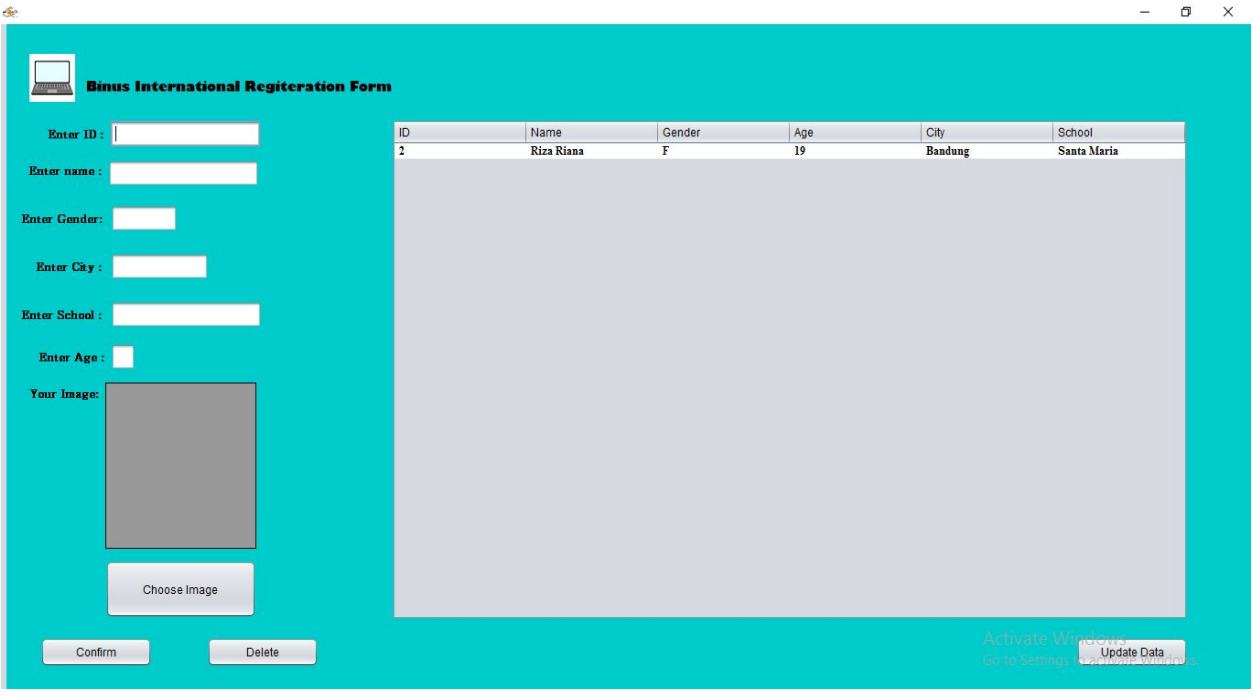
Second problem that I have encountered was the debugging of the code since one error could lead to so many errors, I tried to debug several times and things that popped up inside my IDE is error messages which i do not understand very well, the only thing I did was trying to restart and

run every time I changed one function until everything works in order. The rest of the situation was not a big deal for me since Java had lots of resources for explanation that i could check to help my code.

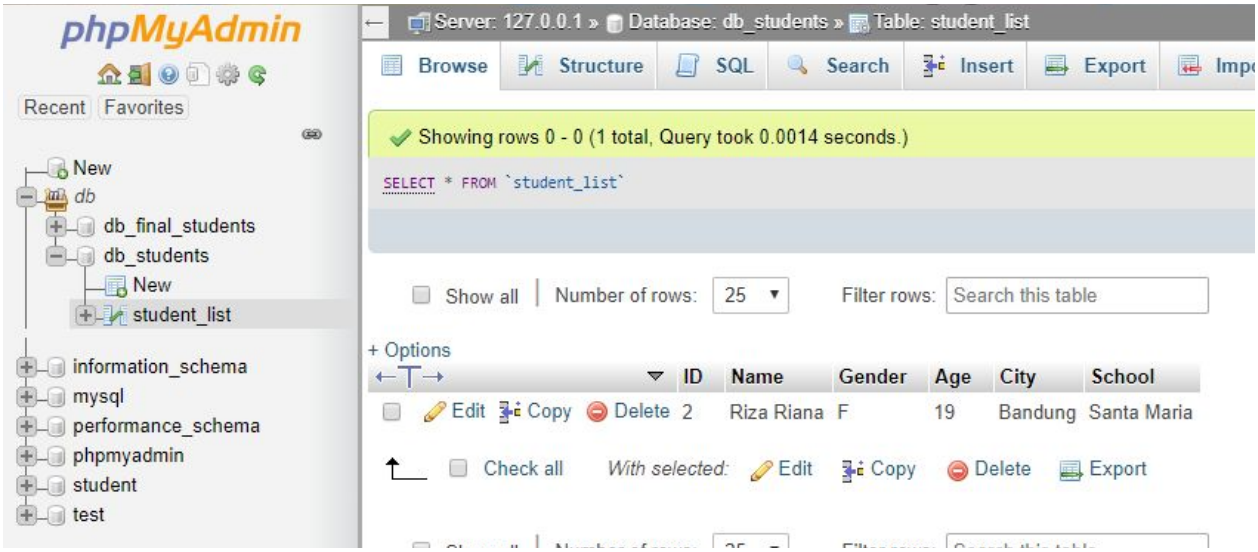
Conclusion

Overall, the code runs well based on what I want, there are lots of things that I can improve but unfortunately according to my time management, I did not have enough time to implement new methods for the project, the design of the interface was very simple but it is very visible for users. For the future development of this project I might be able to put more features inside such as user sign up for new students and also by using hyperlinks (button on click) to connect different panels at one run time to look it more professional and tidy.

Result in Picture



Pic 1.3 (Main_UI JFrame)



Pic 1.4 (MySQL LocalHost)

References

I would love to thank some of the resources that helped me to finish my final project successfully.

java2s.com. *JFrame.setIcon()*. May 2019. Friday June 2019. <http://www.java2s.com/Code/JavaAPI/javax.swing/JFramesetIconImageImageimage.htm>.

JavaTutorial. *Java Swing*. May 2019. June 2019. <http://www.java2s.com/Tutorials/Java/Java_Swing/0400__Java_Swing_JFrame.htm>.