**CST-361 - Design Report Template**

|  |  |  |
| --- | --- | --- |
| **Topic:** | Activity 6: REST API | |
| **Date:** | January 25, 2019 | |
| **Revision:** | 6.0 | |
| **Team:** | 1. Fredrick Onduso Ondieki | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | Design MySQL database for the project | Fredrick Ondieki | 1.0 | 0.5 | | Integrated environment setup and project structure | Fredrick Ondieki | 0.5 |  | | Initial project set up and logic | Fredrick Ondieki | 1.0 |  | | Dummy data entry to the Database for the project test | Fredrick Ondieki | 1.0 | 0.5 | | Design REST API | Fredrick Ondieki | 2.5 | 1.0 | | Design End-to-End flow of the REST API | Fredrick Ondieki | 1,0 |  | | Created UML diagram -Sequence diagram, activity diagram | Fredrick Ondieki | 2.0 |  | | Program deployment and code validation | Fredric Ondieki | 1.0 |  | |  |  |  |  | | |
| **GIT URL:** | https://github.com/FREDDYSMALLZ/jdbc-student-tracker-final | |
| **Peer Review:** | *Y/N* | Yes. I acknowledge that I have reviewed this report and agree to the approach I will be taking. |

**Planning Documentation**

|  |
| --- |
| **What Went Well** |
| I managed to make a transition from the application that I made previously to a new application “Student\_tracker” App that counted the CRUD functions to a school system. |
| I was able to successfully set up my environment to successfully carry out the task at hand by installing the tools needed as well as all the dependencies and jar files needed for the project. |
| I successfully created JSF files, beans and classes that would be used for the project. |
| Managed to execute the program successful to confirm its functionality |
| Successfully Connected the project to the relational database by updating the MySQL jar file compared to the earlier versions that I used in the project |

|  |  |  |
| --- | --- | --- |
| **What Did Not Go Well** | **Action Plan** | **Due Date** |
| The database program failed successfully to connect to my relational database to display the users that I entered manually for the purpose of project setup. | I am thinking that there may be a connection error from the application server towards the MySQL connector jar file. I will start looking specifically on the connection of these two sections | January 21, 2019 |
|  |  |  |
|  |  |  |

**Design Documentation**

**General Technical Approach:**

Decided to create a JSF Application that connects to the database. This is because JSF (Java Server Faces) are one of the frameworks used for building web applications. In addition, they are the standard of Java Enterprise Edition (Project activity needs us to build an enterprise application) based on the Model View Controller.

On the other hand, I used java beans to hold the form data from user inputs as well as to hold the application business logics. Lastly, the project over view includes;

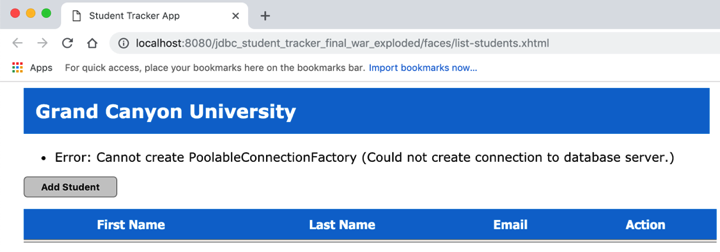
* Setting up the relational database- MySQL Workbench
* List the students from the database and viewed to the browser by JSF
* Add a new student to the database
* Update the student records from the database in case of error entry or change of data
* Delete the student record form the database.

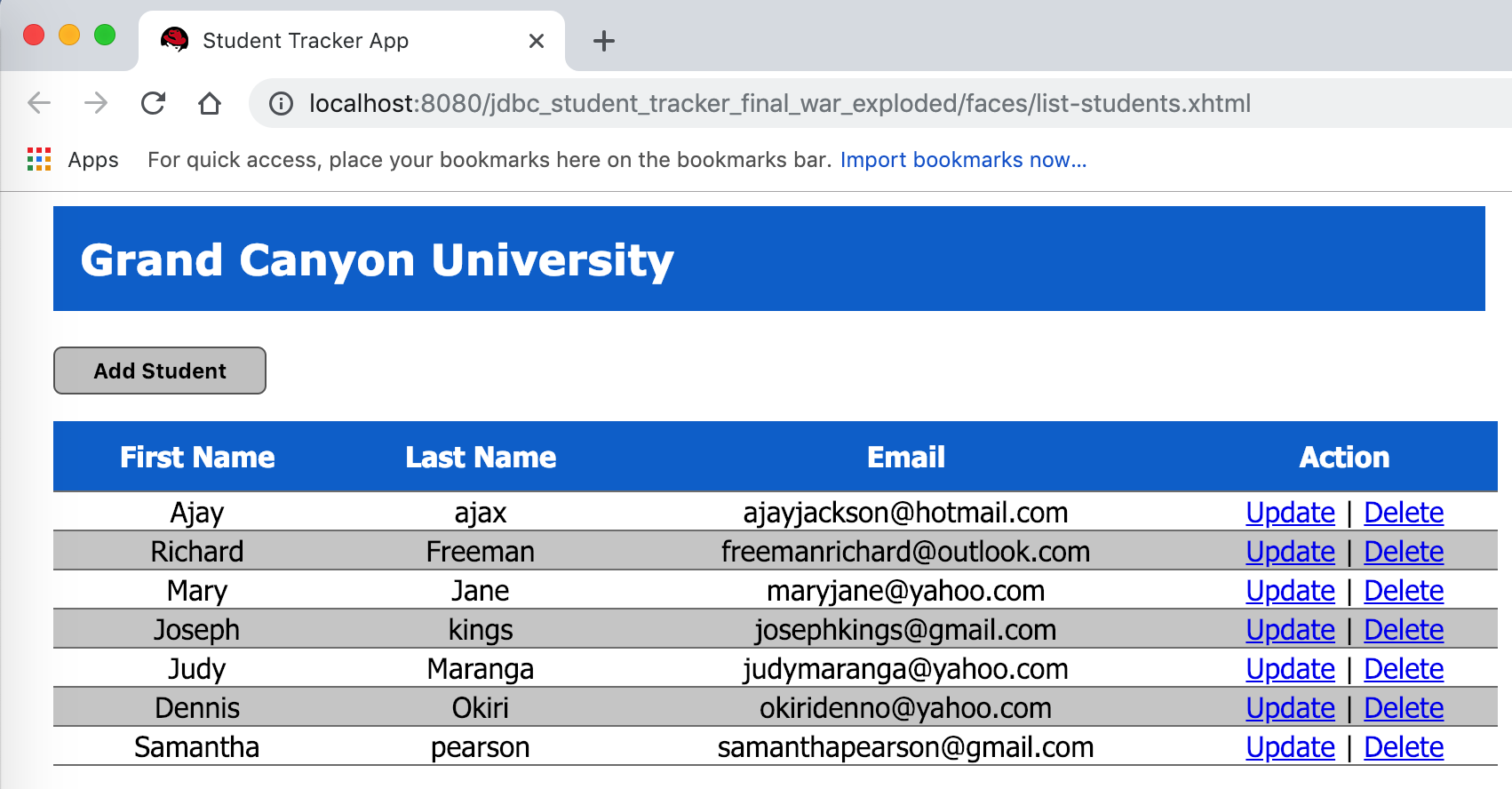
**Key Technical Design Decisions:**

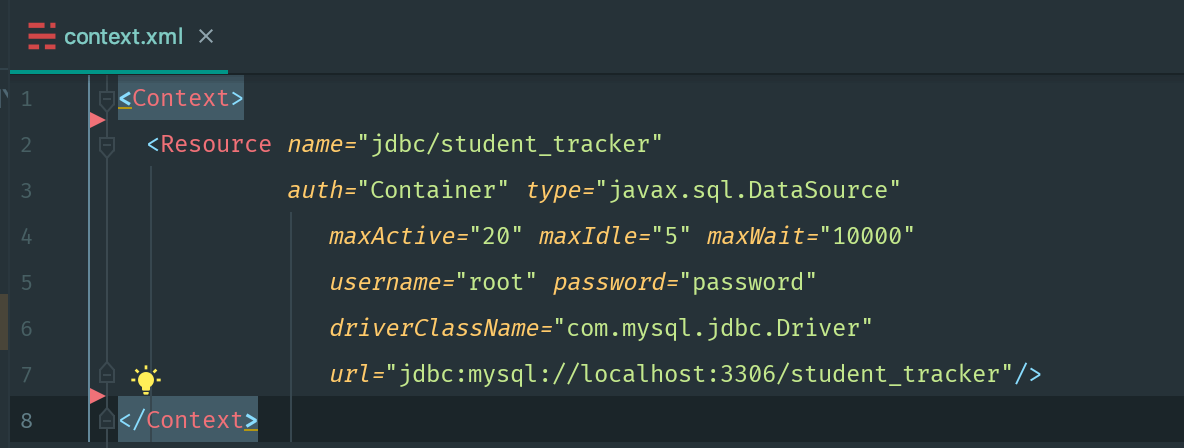
**JSF CST-361 Student Tracker Application -- Setup Instructions**

Student tracker application  
============================  
To run the application, one will need the following software as well as the steps below:  
1. MySQL database as a relational database- used to store the student records for the Application.  
2. MySQL script used for testing the application  
3. Open the source code for the right version  
4. Tom cat Server version 9.0.14  
5. Development environment- IntelliJ IDEA: However, it can be run on any Java Development environment.  
6. To run the application:  
 a. Select the file WebContent/index.html  
 b. Right-Click, select Run As > Run On Server

**Known Issues:**

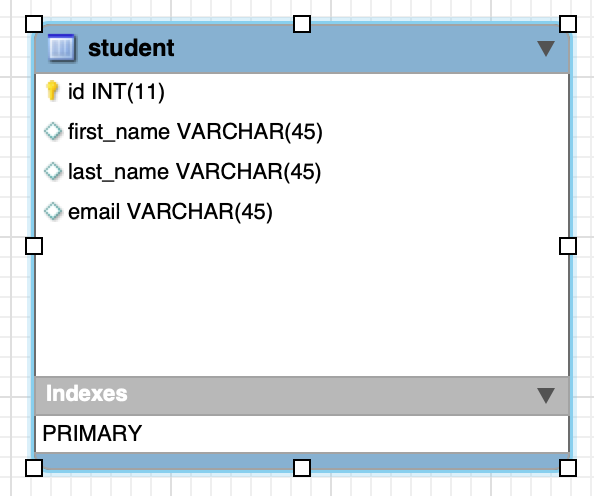
Failed successfully to create a connection pool to the relational database used in the project. See the screenshot bellow.

However, after updating the project resource files for example the MySQL JDBC connector jar file as well as the relational data pool file, I successful deployed the application as shown on the screen shot bellow as it lists the students available from the records.

The root file used for the connection to the MySQL database is shown below. I updated this file to match up and use the latest version of the MySQL jar file. The latest version of the context.xml file is shown below.

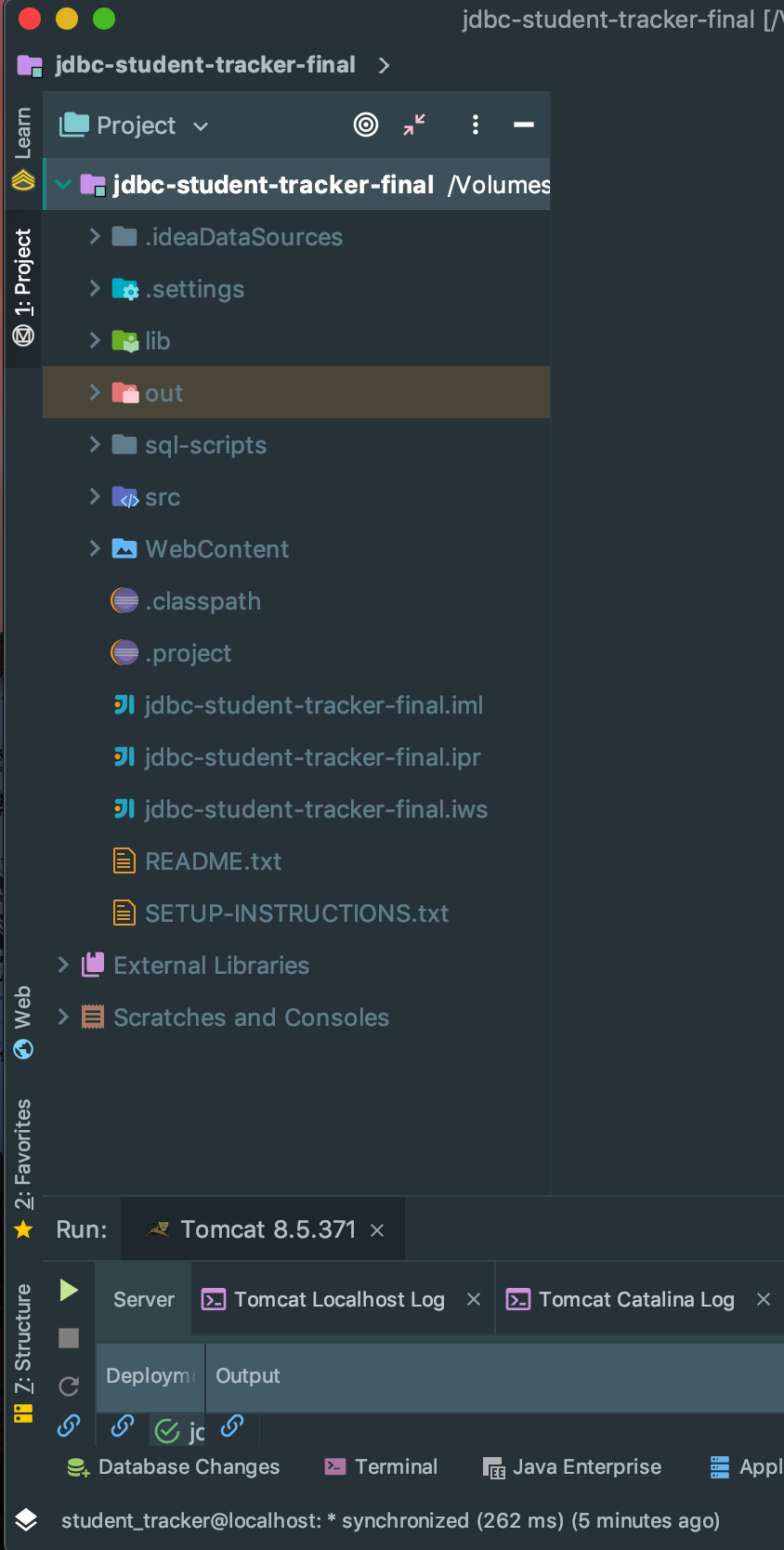
**Risks:**

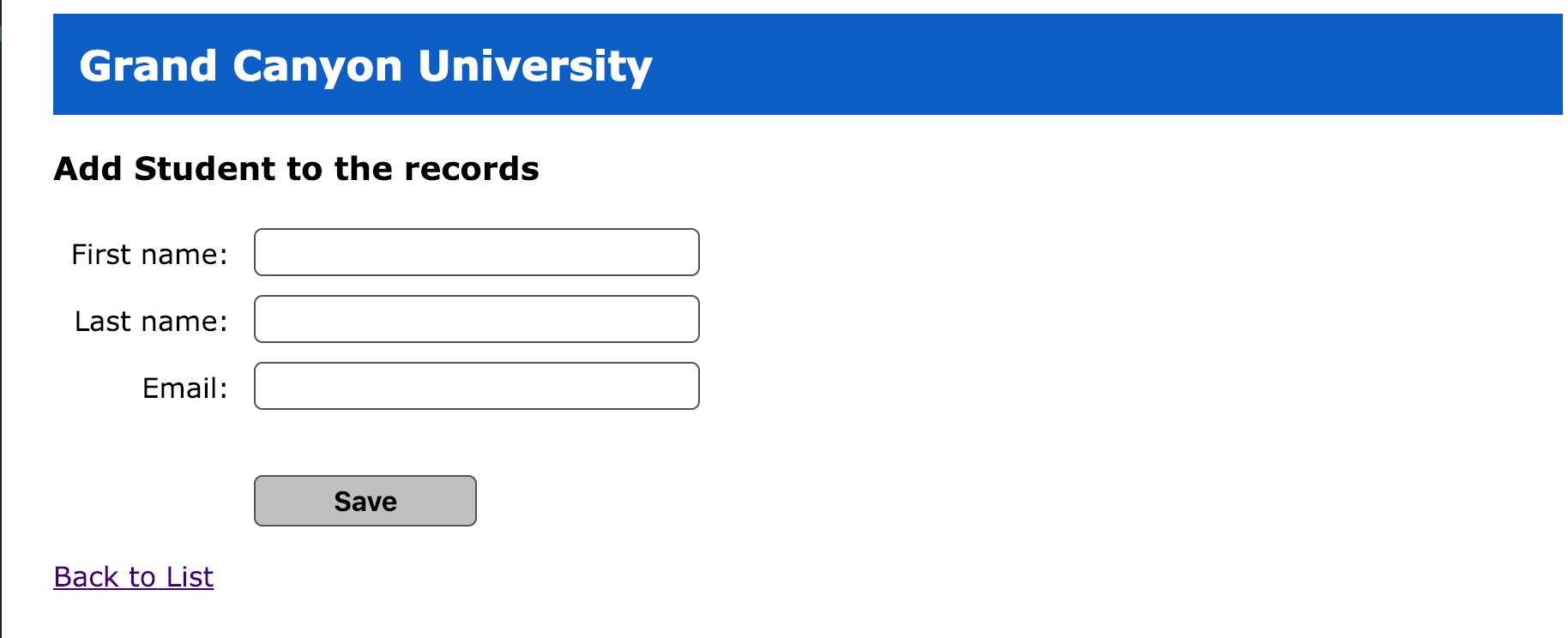
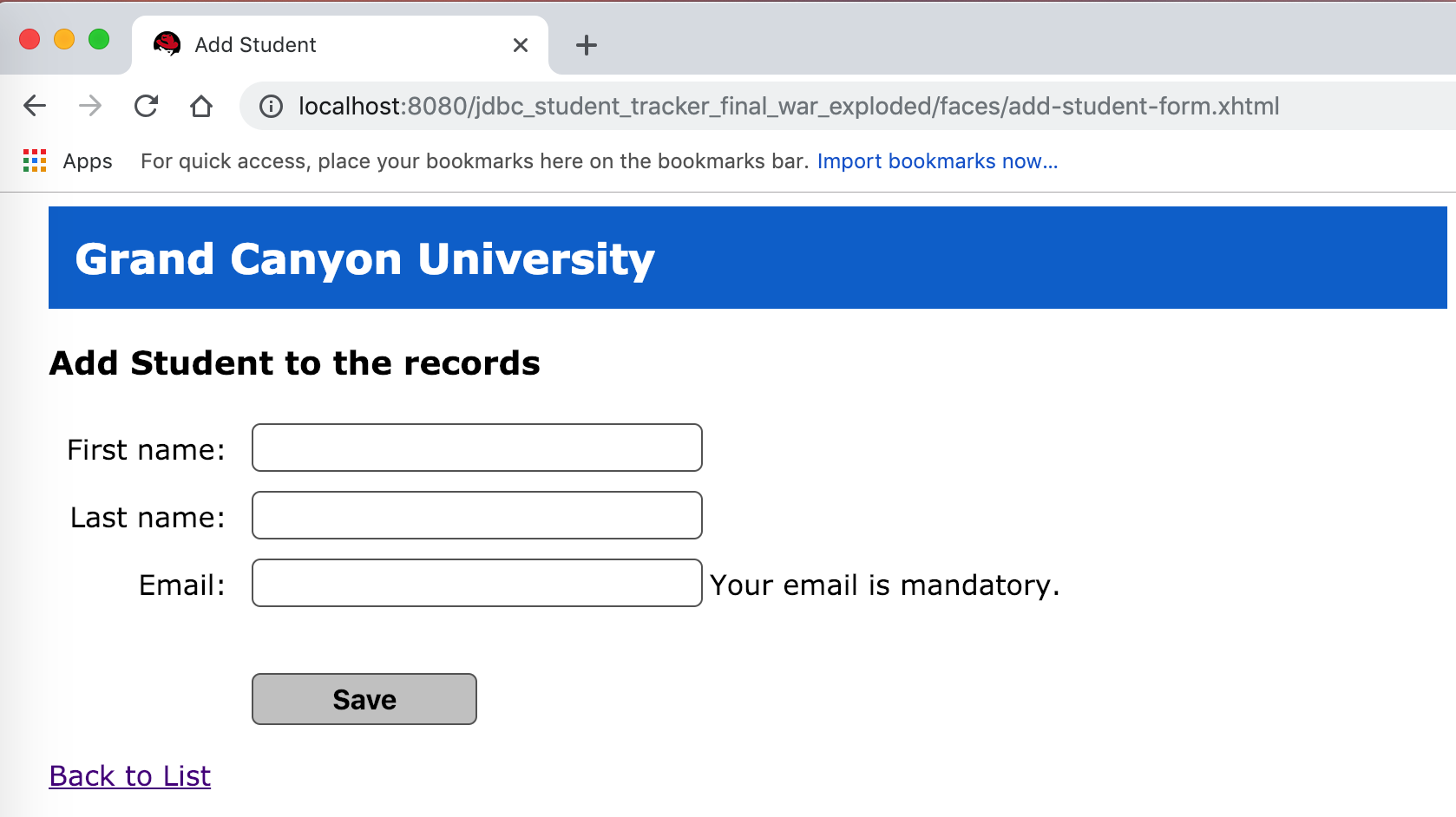
Loading or importing the project from the source code into your machine will create some errors. Therefore, you can remove these errors by checking on the project properties, JavaBuld path and configure the application to use your environment resources.

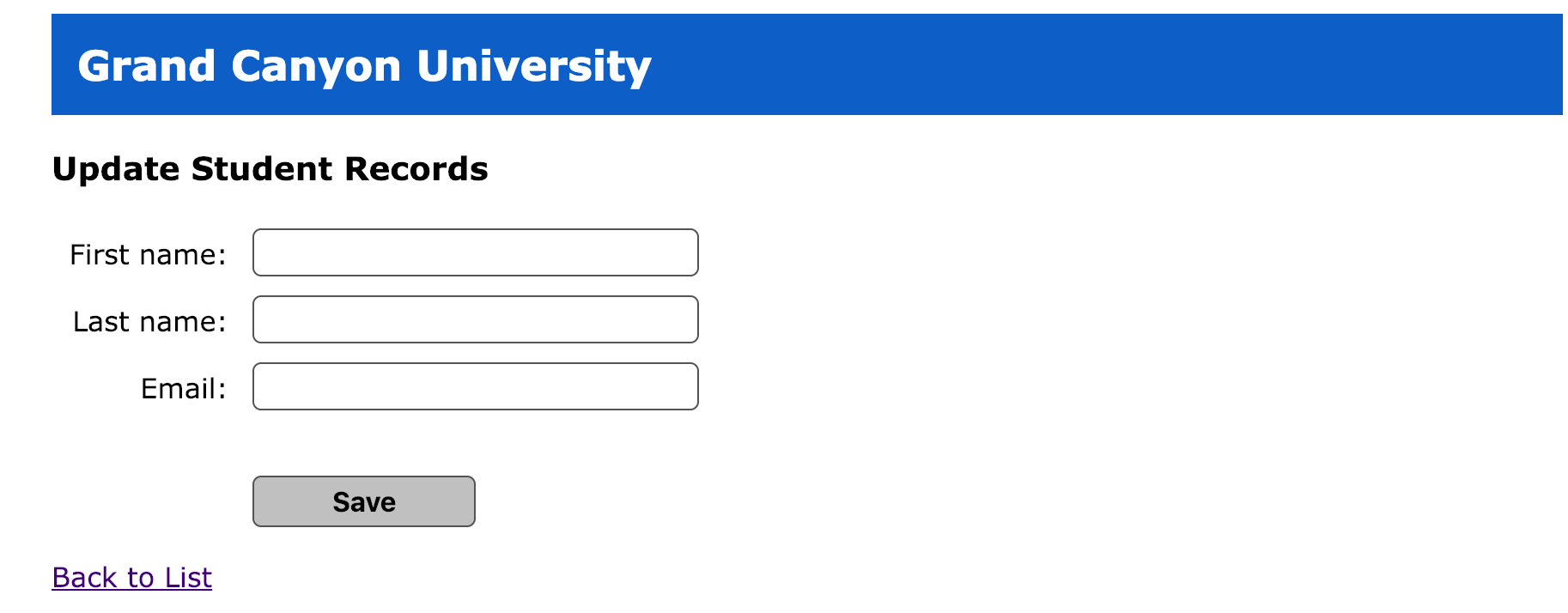
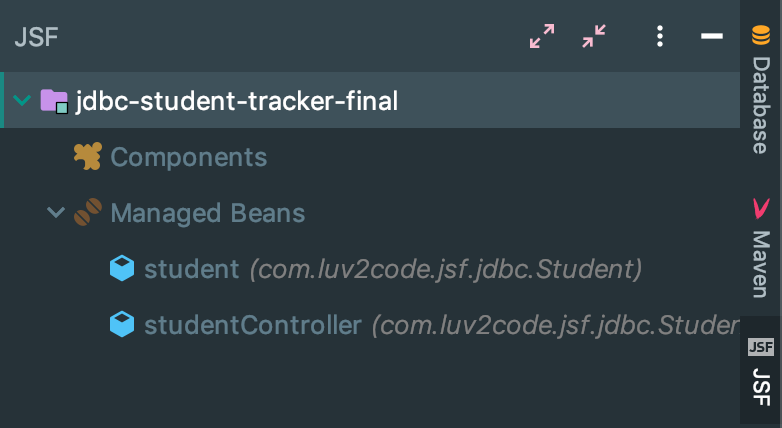
**ER Diagram:**

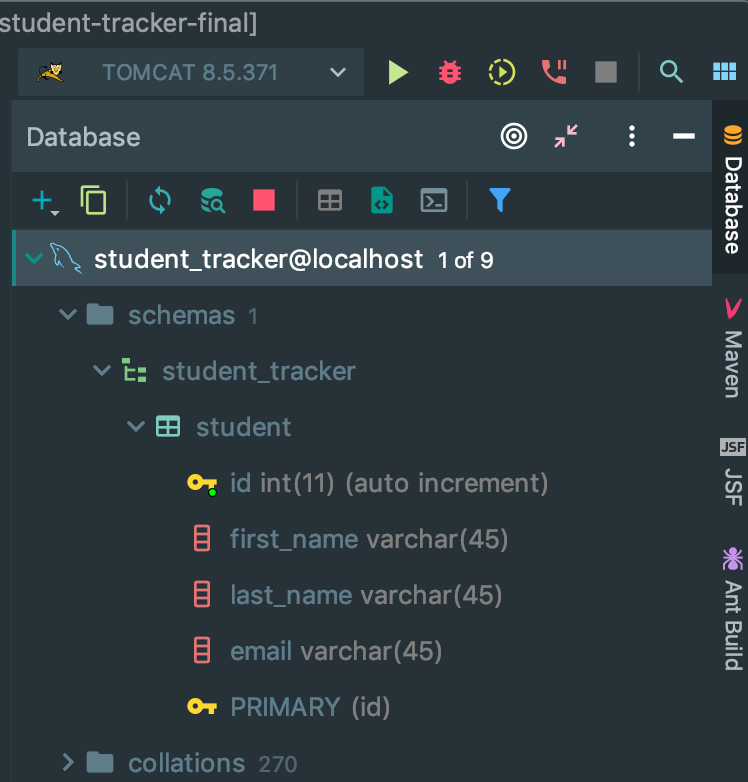
**Program Structure and Code Screenshots**

The following section outlines the program structure as well as the program files screen shots obtained both before and after the program execution. I have also included the class structure for the managed bean class for the student tracker application.

Project structure



Project Beans

Database connection with MySQL

**REST API and Transfer object Pattern**

The model object used in designing the REST API is the Student. I have also employed the use of the transfer object pattern for

this project which can be summarized by the UML class diagrams bellow.

First, the transfer Object pattern is used when we want to pass data with multiple attributes in one shot from client to server. In addition, the transfer object is the simple POJO class having getter/setter methods and is serializable so that it can be transferred over the network. It does not have any behavior. Server-Side business class normally fetches data from the database and fills the POJO and send it to the client or pass it by value. For client, transfer object is read-only. Client can create its own transfer object and pass it to server to update values in database in one shot (TutorialsPoint, n.d.). This design pattern has the following entities.

* **Business Object** - Business Service fills the Transfer Object with data.
* **Transfer Object** - Simple POJO having methods to set/get attributes only.
* **Client** - Client either requests or sends the Transfer Object to Business Object.

|  |
| --- |
| StudentController |
| +StudentController()  +listAll():ArrayList<Students>  +onDelete(Student):id  +onEdit(student):id  +onSubmit():String |

|  |
| --- |
| Student |
| +id:int  +firstname:String  +lastname:String  +email:String |
| +Student()  +getId()  +setId()  +getFirstName()  +setFirstName()  +getLastName()  +stLastName()  +getEmail()  +setEmail() |

|  |
| --- |
| DatabaseService |
| +dbURL:String  +user:String  +password:String  +c:Connection  +sqlstatement:PreparedStatement  +stmt:Statement  +rs:ResultSet  +rowsAffected:int  +rowsAffected:int |
| +delete(id):int  +insert(studeny):int  +listAll():ArrayList<Student>  +update(Student, id):int |

**Pseudo Code:**

I have attached a source code for the project on this project which can also be accessed via the following Git hub repository link

https://github.com/FREDDYSMALLZ/jdbc-student-tracker-final

**References**

The Java EE 6 Tutorial. What Are RESTful Web Services? (n.d.). Retrieved January 25, 2019, from https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

Design Pattern - Transfer Object Pattern. (n.d.). Retrieved January 25, 2019, from https://www.tutorialspoint.com/design\_pattern/transfer\_object\_pattern.htm