ITWS Midterm 2

Theme: Student – Professor – Course data system

Student: Stefan Cliff   
ID: 2019/230449  
Email: [stefan.cliff.19@singimail.com](mailto:stefan.cliff.19@singimail.com)

Contents

[Introduction 3](#_Toc98958857)

[Architecture and technology 4](#_Toc98958858)

[Database model 5](#_Toc98958859)

[Database Structure 5](#_Toc98958860)

[The Database 5](#_Toc98958861)

[Students 5](#_Toc98958862)

[Professors 6](#_Toc98958863)

[Courses 6](#_Toc98958864)

[The Relationship diagram of the three tables: 7](#_Toc98958865)

[Implementation 8](#_Toc98958866)

[Index.html 8](#_Toc98958867)

[Index.css 9](#_Toc98958868)

[Emp.java (1/4) 9](#_Toc98958869)

[Emp.java (2/4) 10](#_Toc98958870)

[Emp.java (3/4) 11](#_Toc98958871)

[Emp.java (4/4) 12](#_Toc98958872)

[EmpDao.java (2/3) 13](#_Toc98958873)

[EmpDao.java (3/3) 14](#_Toc98958874)

[Create – SaveStudentServlet.java 15](#_Toc98958875)

[Read – ViewStudentServlet.java 16](#_Toc98958876)

[Update (1/2) – EditStudentServlet.java 17](#_Toc98958877)

[Update (2/2) – SaveStudentServlet2.java 18](#_Toc98958878)

[Delete – DeleteStudentServlet.java 19](#_Toc98958879)

[Functionalities 20](#_Toc98958880)

[Conclusion 23](#_Toc98958881)

# Introduction

The system I made an application for is a simple *Student – Professor – Course* type of project. Where you can create/add new Students or Professors to a course, remove them from the database or edit any or all of their credentials in a simple user friendly manner.  
  
The application is a CRUD (*Create, Read, Update, and Delete*) style, meaning via the app you can directly control certain/all the tables to the database you wish to alter. In this example, the Database is called ITWS\_Midterm2\_SC with its tables being: - students; - professors; &  
- courses;

I wanted the idea of adding new students and professors to be to main part of the application, with editing/viewing them being of equal importance. As well as deleting.   
All of these functions where done via Servlets and SQL commands that were run on my localhost server.

# Architecture and technology

As I mentioned before, the project is one based on *Servlet – Server* communications and information parsing. The enviroment I used was Eclipse, the server was TomCat v9. Originally I tried to use MySQL Workbench for my database modelling and so on but encountered some weird over-lapping with XAMPP I already had installed on my device (*which I forgot I had in the first place*).

After that I switched over to XMAPP/PhPMyAdmin for any and all database manipulation/creation. I chose to create and model all database related things via the graphical interface over the writing of SQL code, as it just seemed easier and faster at the time.

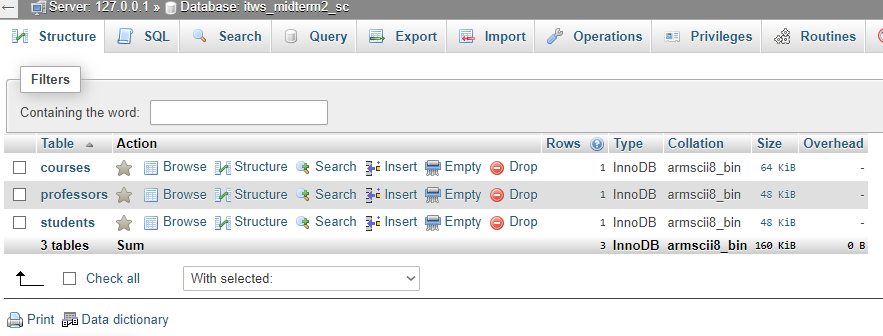
The code and guide I used for the project was the CRUDExample folder the professor shared with us to use as the ‘*baseline*’ of our project as well as the wonders of the internet for any errors or troubleshooting I had to do.

# Database model

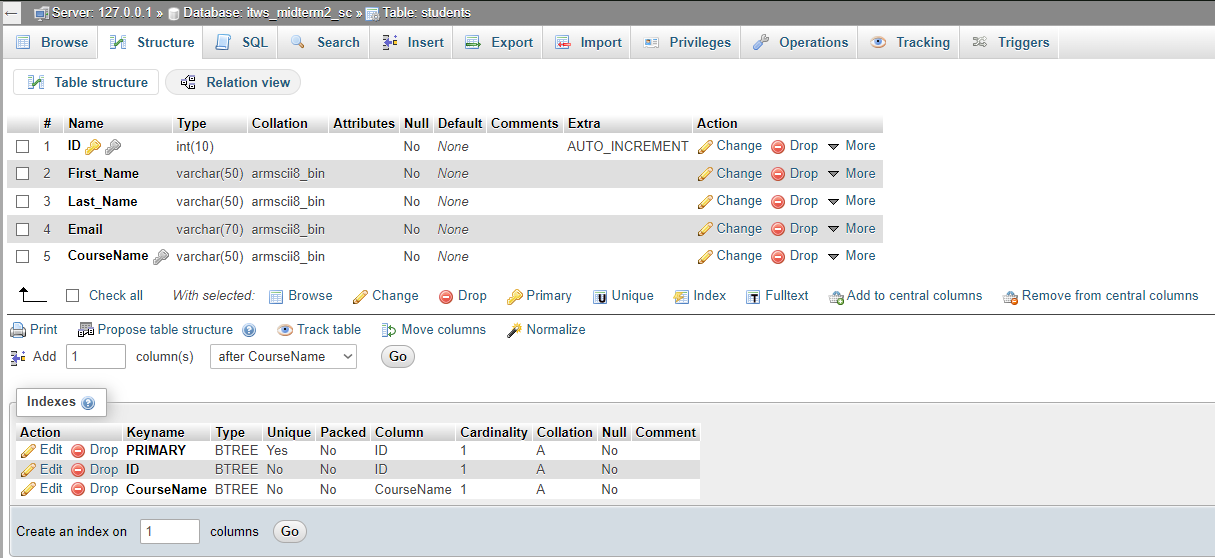
The database is called ITWS\_Midterm2\_SC, and its tables are professors, students and courses. All of this I mentioned before, but here I will show the tables themselves, their configurations and their relationships to one another.

## Database Structure

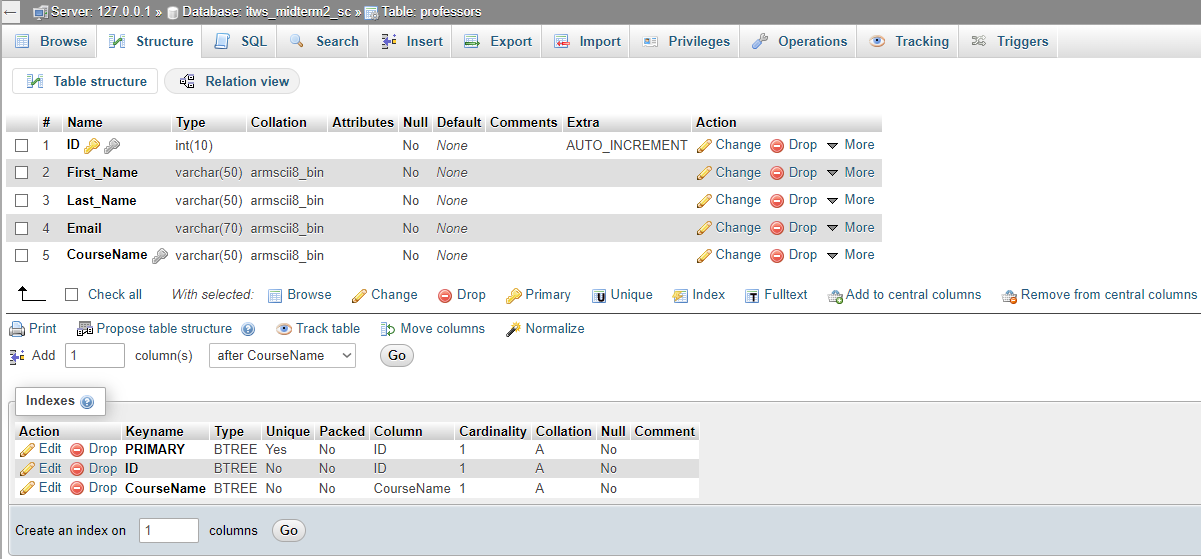
### The Database



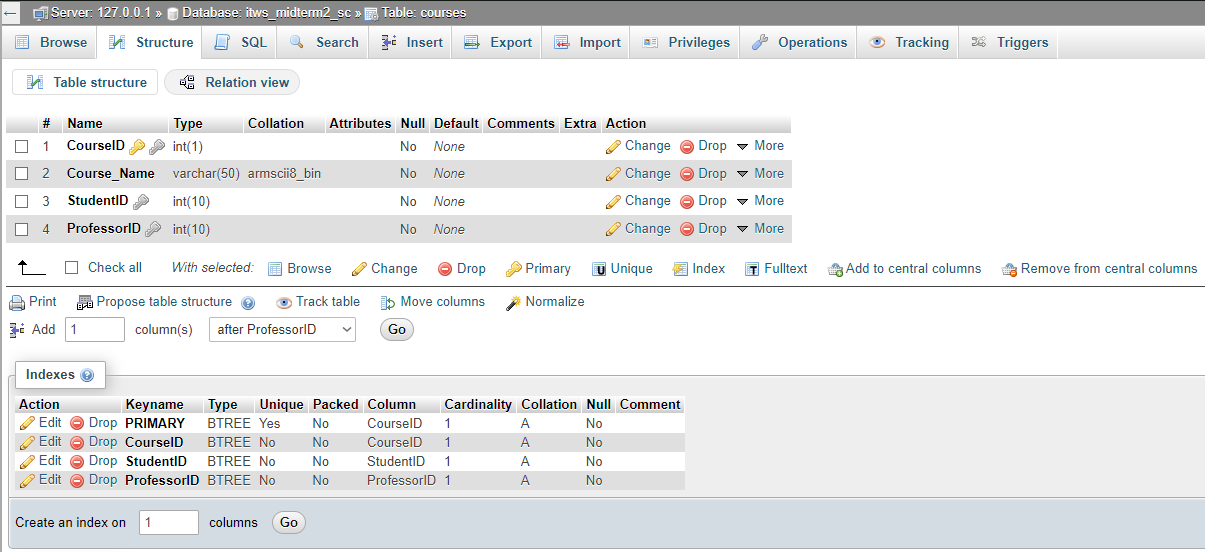
### Students



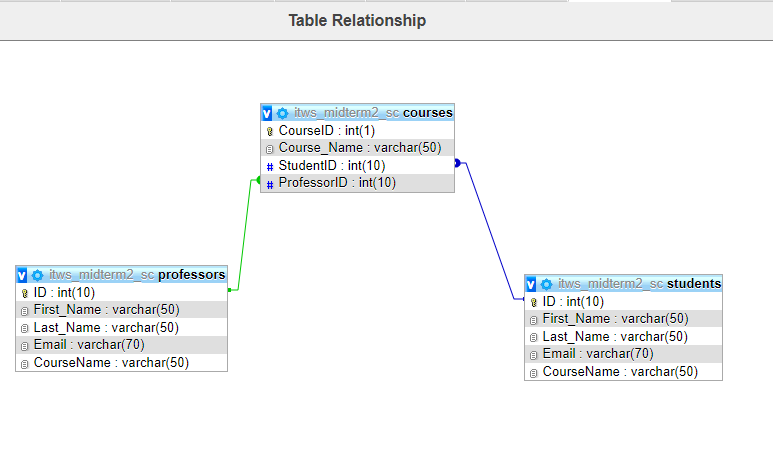
### Professors



### Courses



## The Relationship diagram of the three tables:



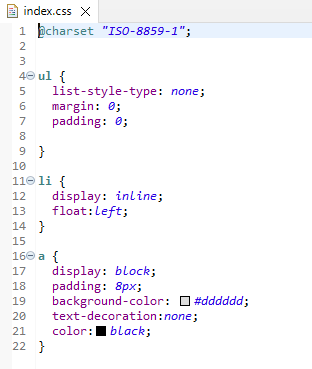
# Implementation

Bellow I will attach a good number of the code snippets that are important, and do my best to name all the snippets correctly.   
*I’ll only show how I did all the servlets for all student-based activities as to avoid copying very similar things over and over and over again.*

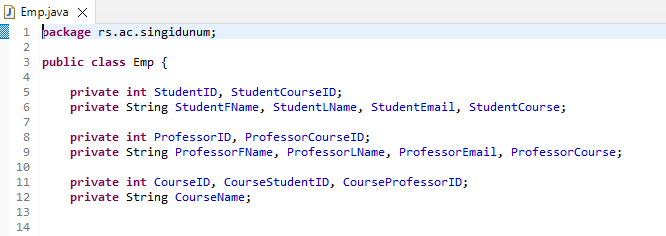
### Index.html



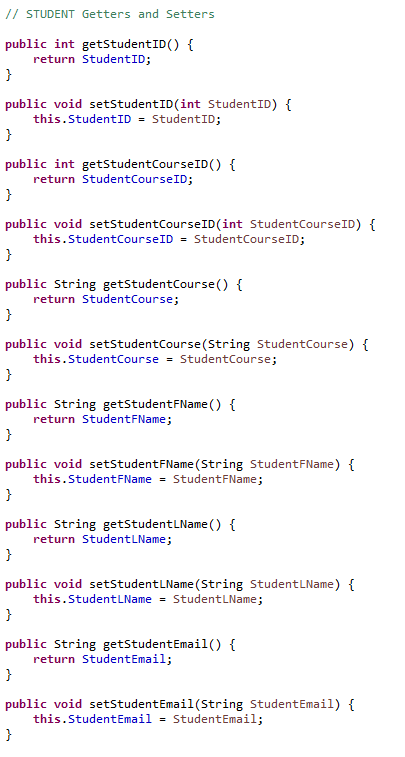
### Index.css



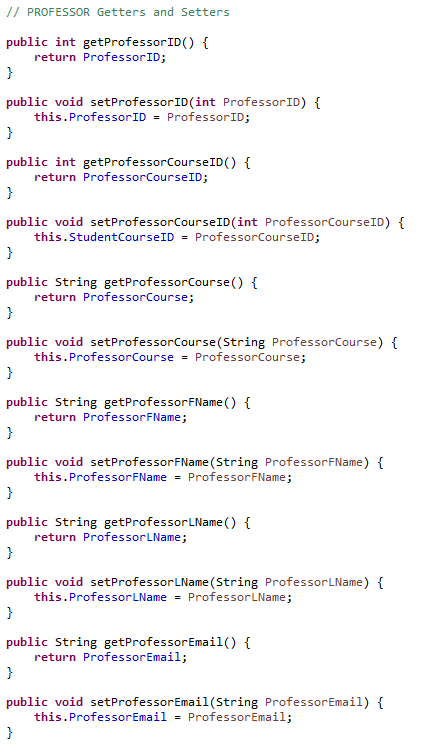
### Emp.java (1/4)



### Emp.java (2/4)



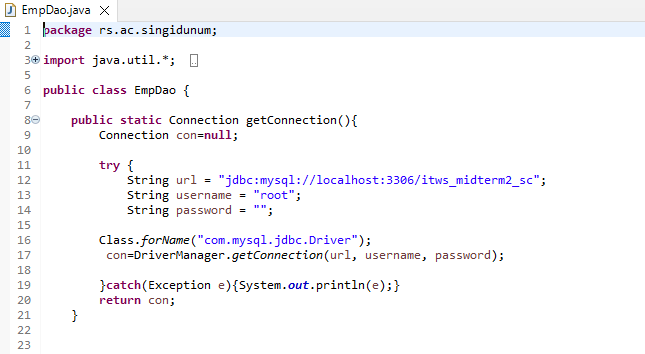
### Emp.java (3/4)



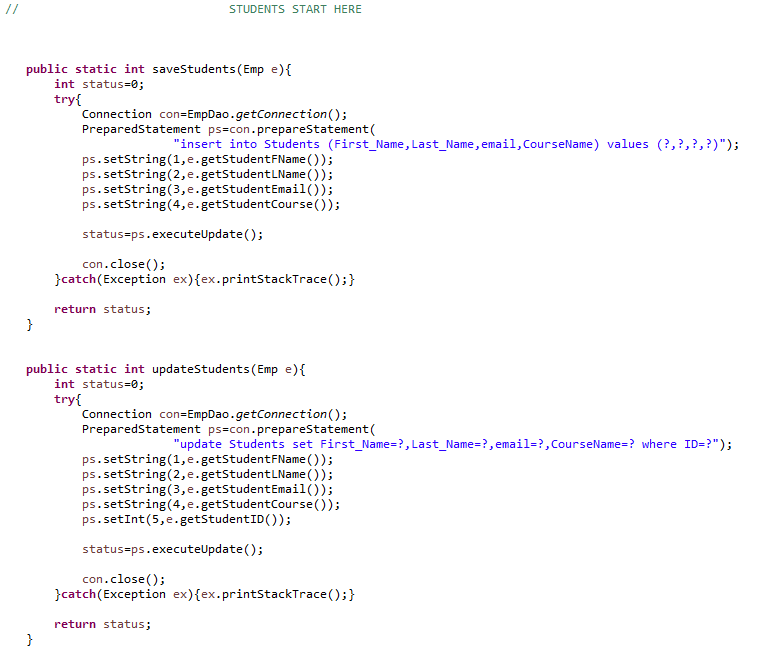
### Emp.java (4/4)



##### EmpDao.java (1/2)



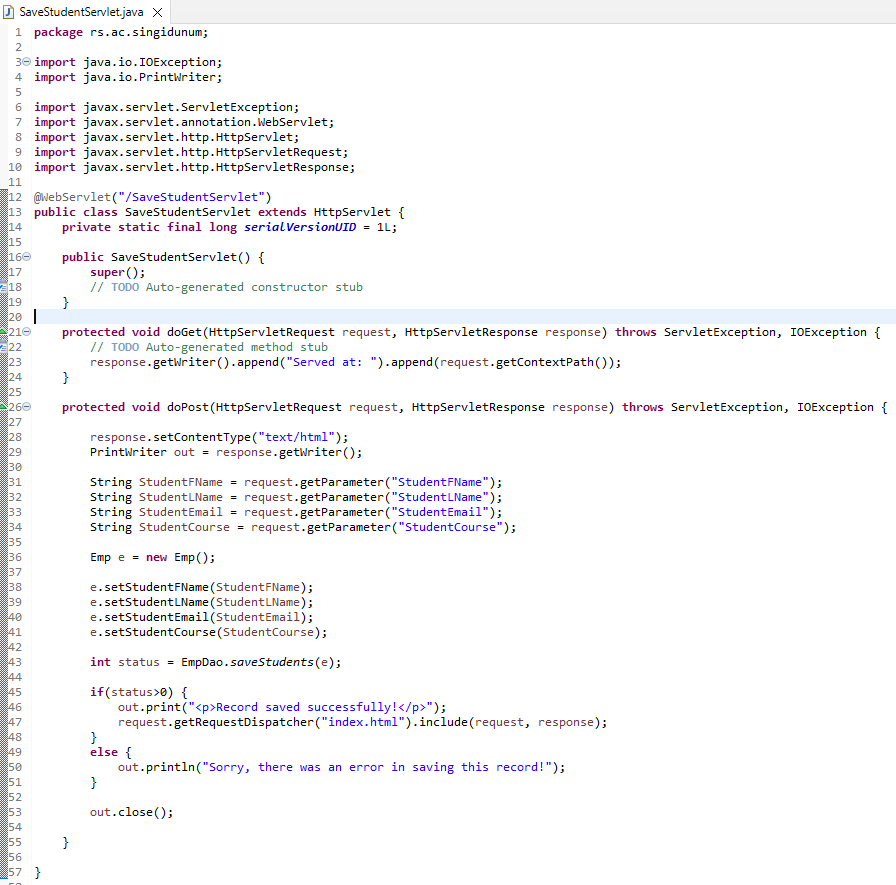
### EmpDao.java (2/3)



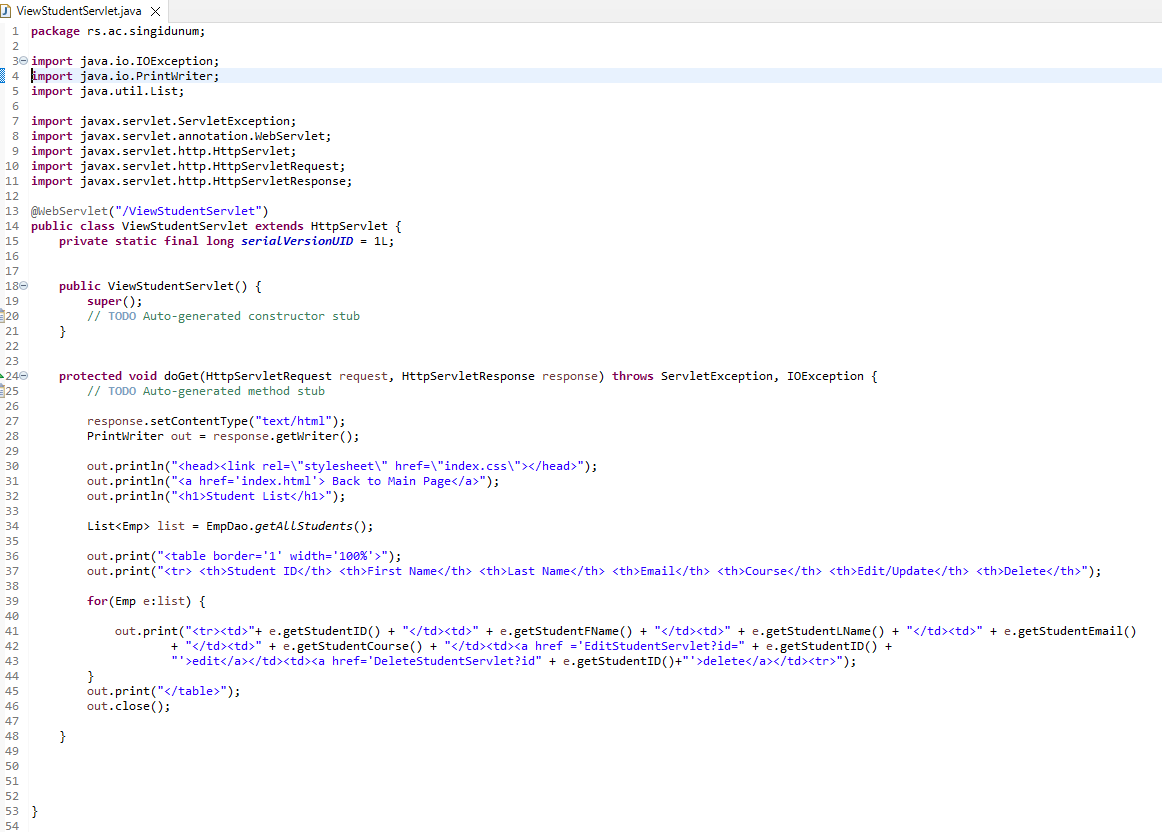
### EmpDao.java (3/3)



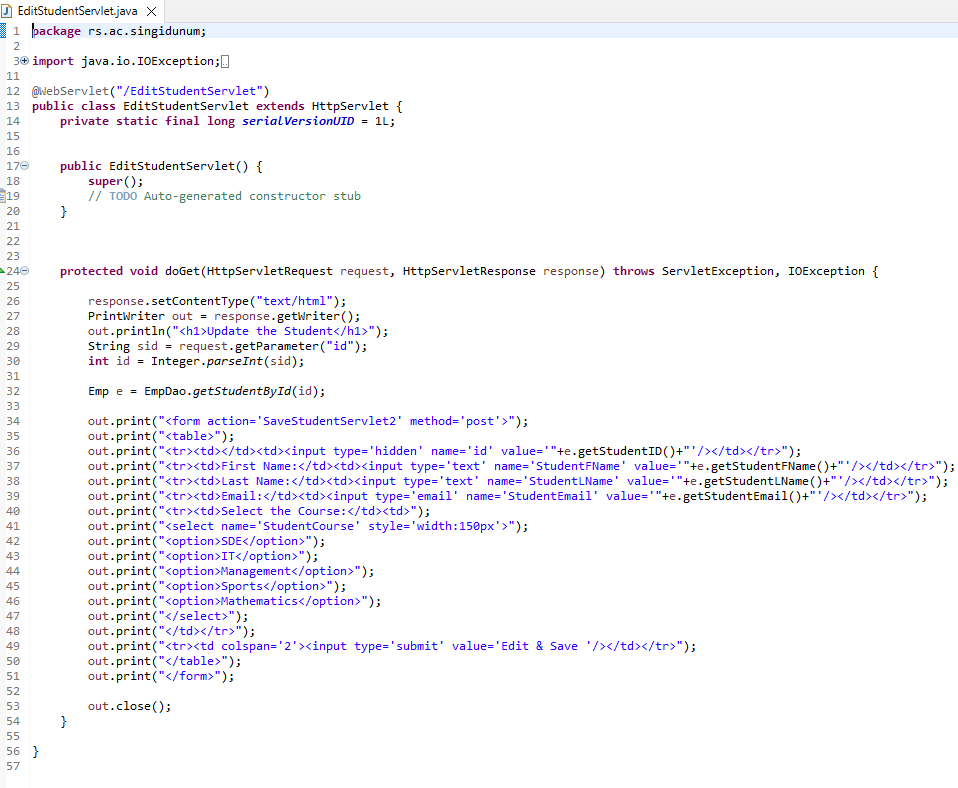
### Create – SaveStudentServlet.java



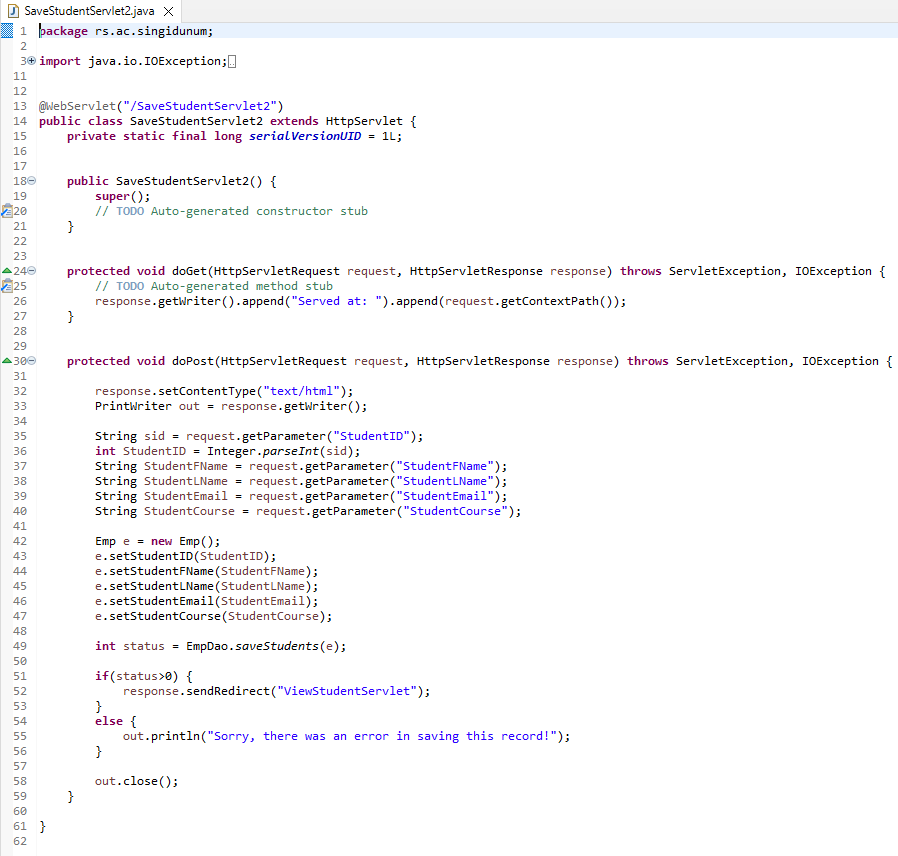
### Read – ViewStudentServlet.java



### Update (1/2) – EditStudentServlet.java



### Update (2/2) – SaveStudentServlet2.java



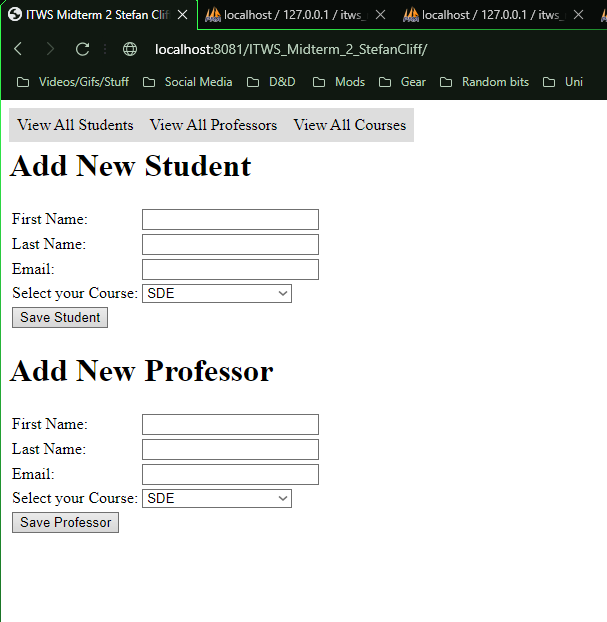
### Delete – DeleteStudentServlet.java



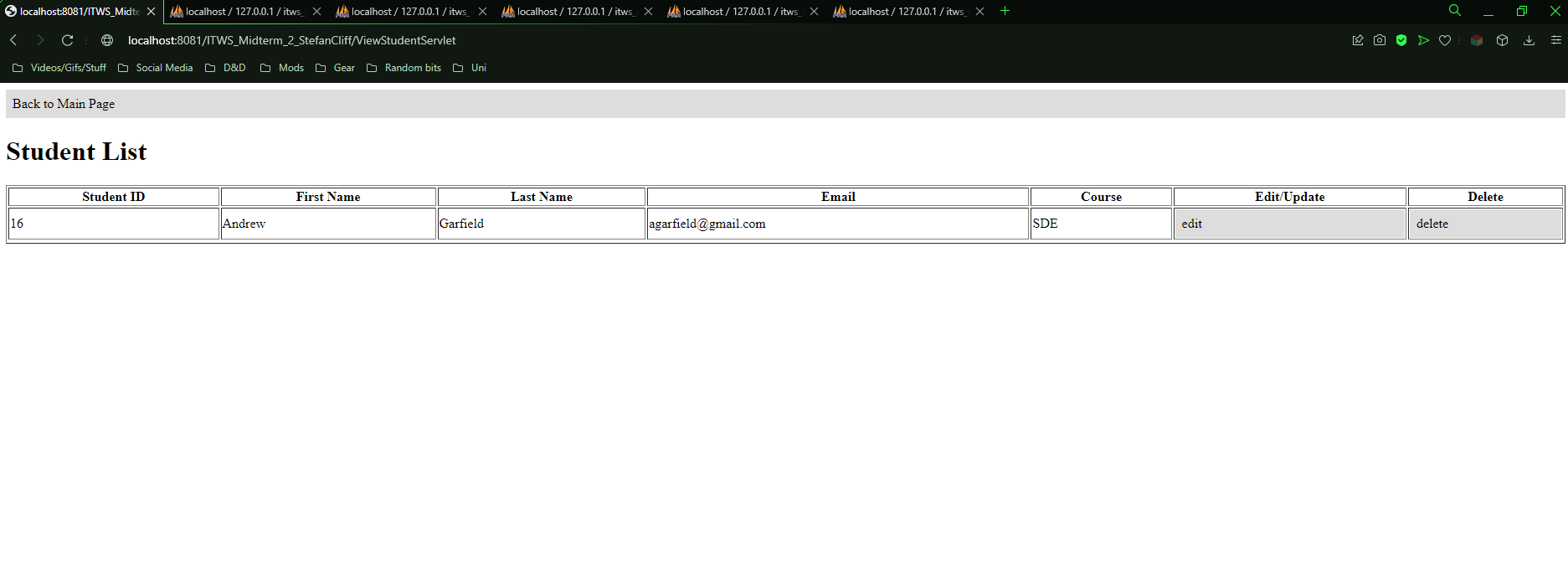
# Functionalities

In this part I shall the GUI (*Graphic User Interface*) of my actual project.

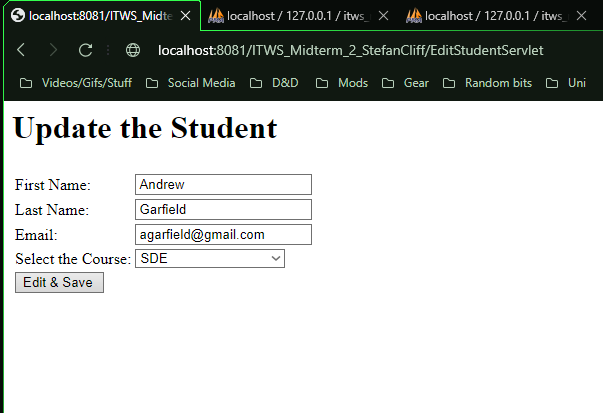
##### Home – index.html



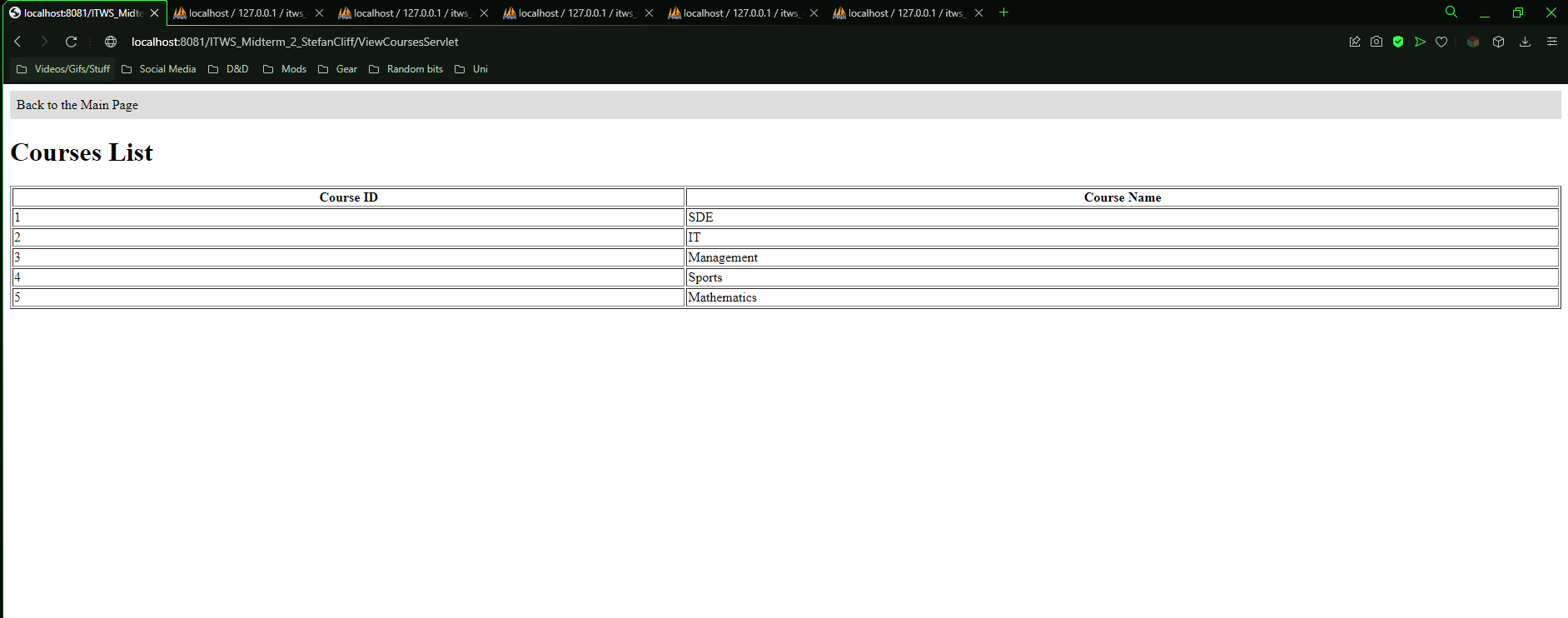
##### View all students



##### Edit/Update student



##### View all courses - Course list



# Conclusion

All and all, the task was simple in idea but it was a bit tricky in places still to implement. Some obvious improvements would be on the GUI side of this, the colouration and formatting could have been done better.

I didn’t write as many comments while writing the code as I usually do, which might prove to be a hindrance in future if I ever come back to this project to fix or add anything.   
  
But besides that, maybe adding a way to filter or sort through all the students based on GPA or something along those lines would be an interesting addition to be sure.   
As well as adding more students and professors as a whole, which would make the application feel a bit less barebones than it is in reality.

Besides that, I am unsure what I could change or add without redoing the whole project from the ground up and re-imagining its core idea to some extent.   
  
Maybe adding a way to see all the students that are in a course and its professor could have been a nice touch to add, but honestly I couldn’t really figure out how to would work to some extent. I have a pretty solid idea, but the more I think about it, the more it seems like a lot of extra work that I am certain I don’t have the time for.

I could add a few levels of user warnings for the deleting part of the application, because right now a simple miss click could in theory delete a whole person…. which isn’t the safest for practical use.

,, Always approach every situation as if the users are idiots. “

*Dr. Miodrag Živković 2019 Web Design*