Campus Connect

SE352 – Object Oriented Enterprise Application Development

Campus Connect

Contents

[Overview 2](#_Toc524527007)

[Requirements 2](#_Toc524527008)

[Use Case 2](#_Toc524527009)

[Description of problem 2](#_Toc524527010)

[Design 2](#_Toc524527011)

[Sequence of major functionality 2](#_Toc524527012)

[Web UI (Common case) 3](#_Toc524527013)

[Table layout 5](#_Toc524527014)

[Deployment 6](#_Toc524527015)

[Discussion of how your design met the requirements 6](#_Toc524527016)

[Discussion of lessons learned 6](#_Toc524527017)

Milestone Screenshots 7

Milestone #1 7

Milestone #2 7

Milestone #3 8

Milestone #4 10

Finals Submission 11

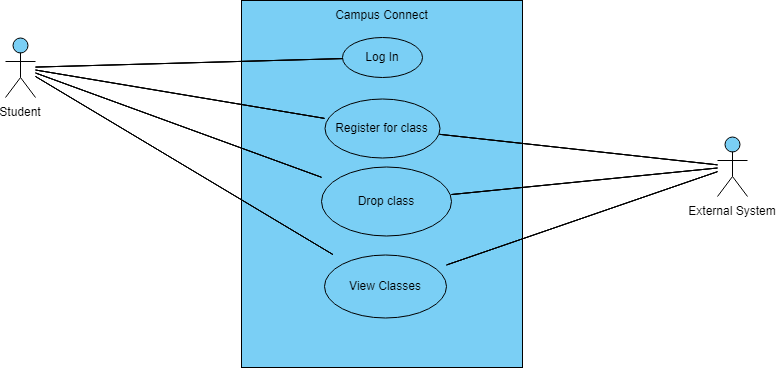
[Decision Log 13](#_Toc524527018)

# Overview

An application that allows students to register, drop, and view their enrolled classes for a given quarter.

# Requirements

## Use Case



## Description of problem

Students log in to campus connect to manage their classes by registering, dropping, and viewing their classes for each of DePaul’s academic quarters (Winter, Spring, Summer, and Fall).

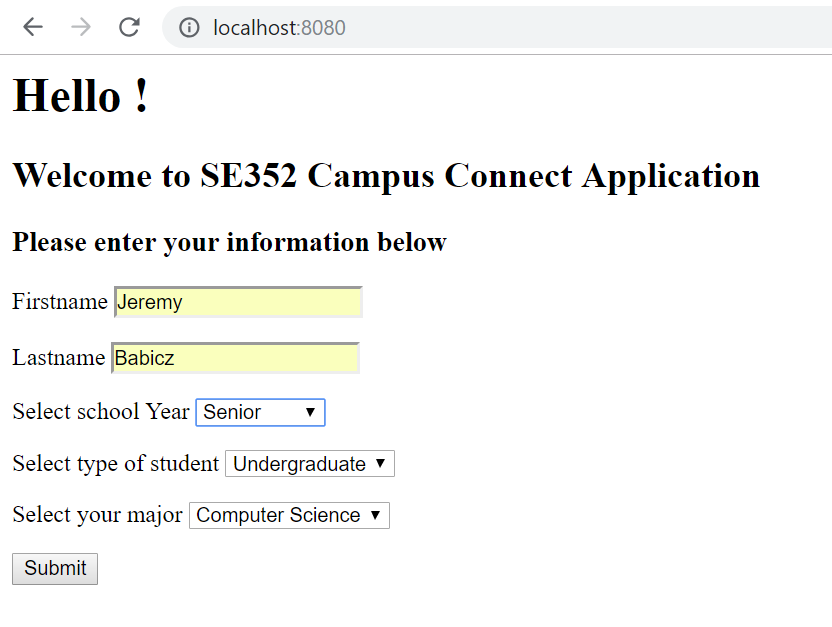
# Design

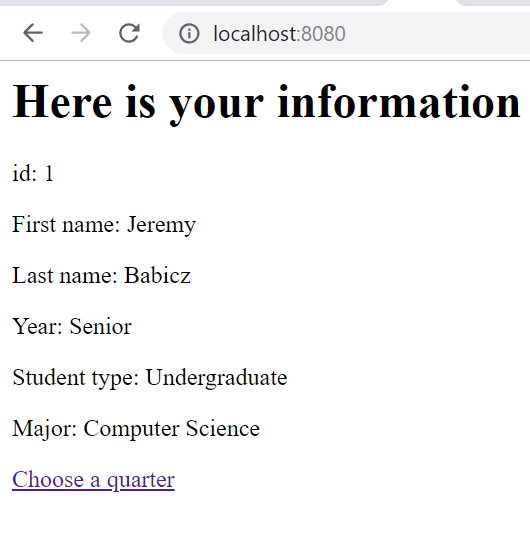
## Sequence of major functionality (Pictures of sequence shown in Web UI with corresponding numbers)

1. Student first inputs their information and then is shown their entered information.
2. They then are redirected to the quarter page. They pick their desired quarter then press submit.
3. They are now on the choose classes page. Here, the student picks their desired classes for that quarter.
4. They are redirected to page showing their current enrolled classes. They then have 3 options upon entering classes:
5. They can choose to drop their enrolled classes (go to #5)
6. They can choose to go back to the enrollment screen and add more classes (go to #3)
7. They can finish enrolling and be officially enrolled for their chosen quarter (go to #6)
8. The student now on the DropClass page. They select the classes they wish to drop and then press submit. Upon submit they are then shown their classes that they choose to drop. They then go back to their current enrolled classes page (#4).
9. The student is officially enrolled for their selected classes for their chosen quarter. Here, their finalized class list is shown for that quarter.

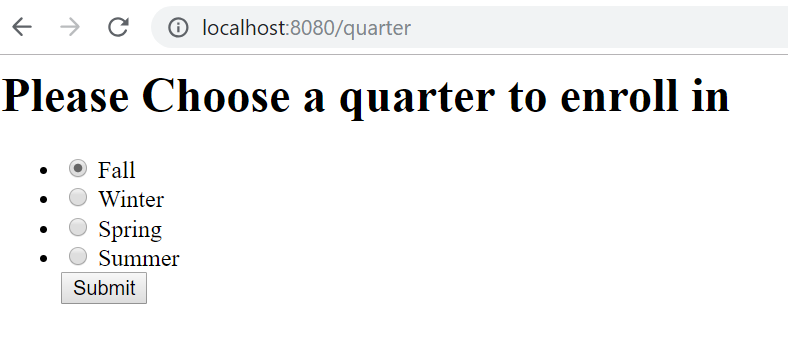
### Web UI (Common case)

1. Home page & Student Result pages

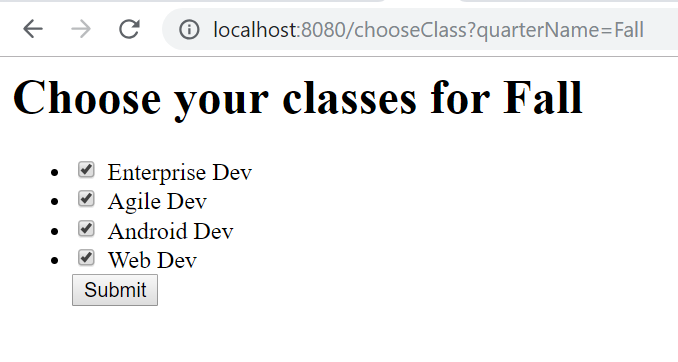




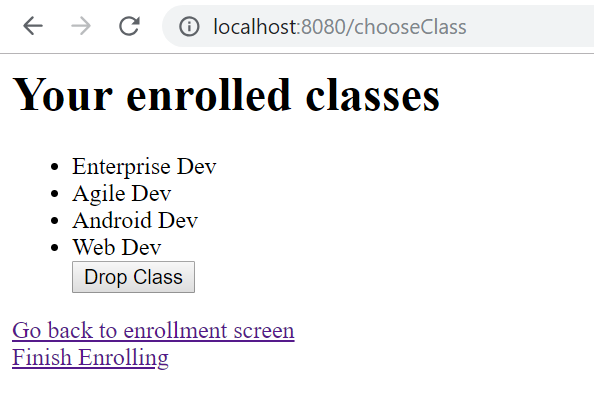
2. Quarter page



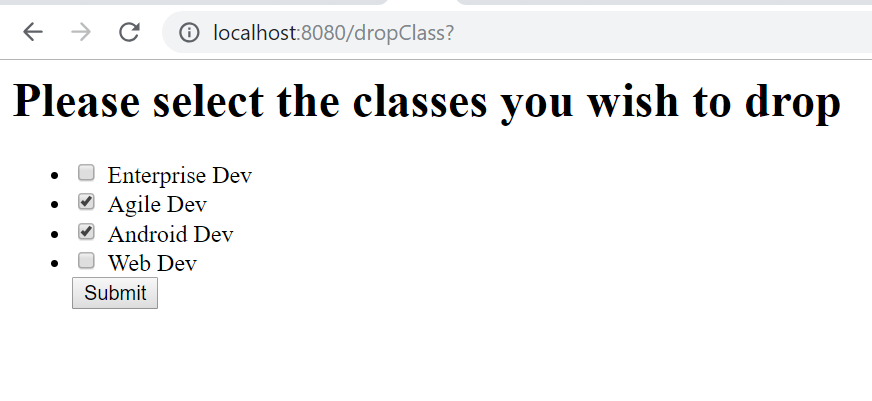
3. Choose Class page

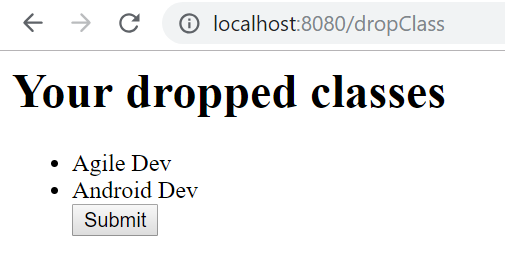


4. Choose Class Result page

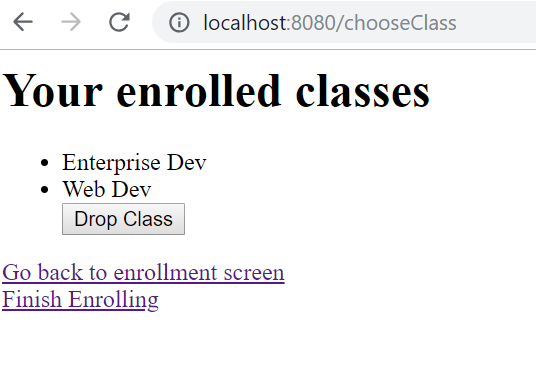


5. Drop Class and Drop Class result pages

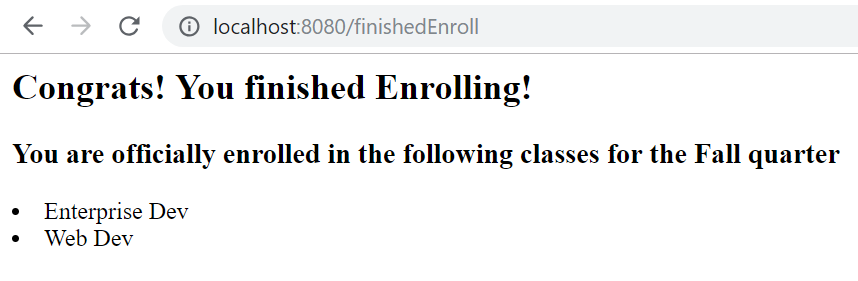




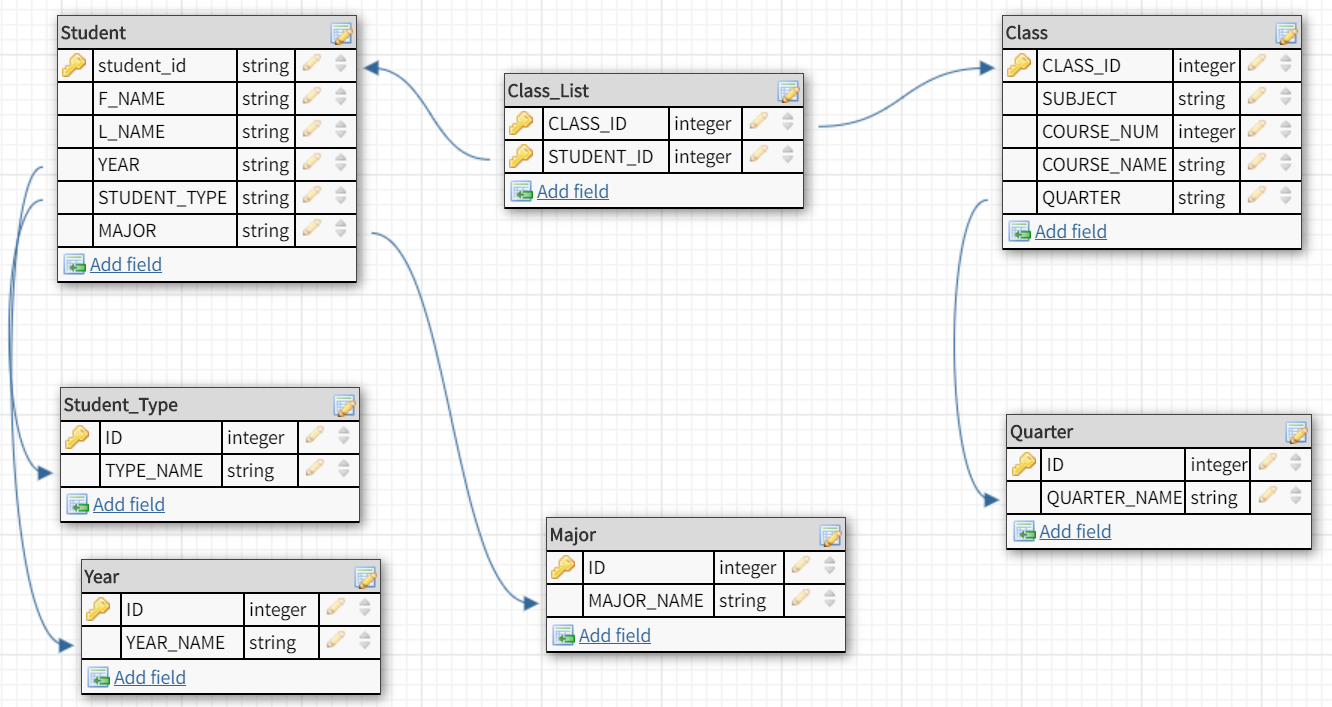
4. Choose class result **(2nd time!)**



6. Finished Enrolling Page



## Table layout



## Deployment

Under the edu.depaul.cdm.se.campusconnect package, run CampusConnectProjectApplication.java. In the browser, go to http://localhost:8080/

# Discussion of how your design met the requirements

1. Students are stored in the “Student table”. Classes are stored in the “Class table”.
2. A student enrolls for a specific quarter referenced from the “Quarter table”.
3. A query is then made to get all the classes who’s “quarter name” is equivalent to the “quarter name” of the quarter the student chose to enroll in.
4. The student then selects classes from the queried list and the chosen class ids along with the student’s id are stored in the “Class\_List table”. Thus, adding the class to the student’s list of classes.
5. When a student decides to drop classes, the matched record with the dropped classes’ ids and the student ids are removed from the “Class\_List table.” Thus, removing the class from the student’s list of classes.

# Discussion of lessons learned

For using the MongoDB, in order to utilize the MongoRepository ID functions, the attribute name must be “id” in the java class which represents that table. Or else, the functions will not work. I had to change Class’s attribute name “class\_id” to “id”, as well as changing Student’s attribute name “student\_id” to just “id”.

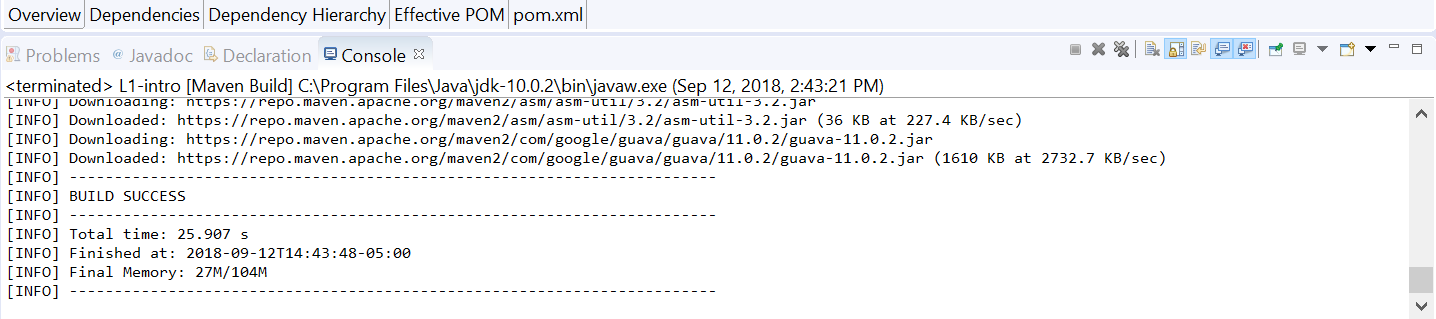
When I was trying to list out all the records on my quarter table in html through ThymeLeaf. I had to change my own id’s names from id to \_id. The auto generated id’s mongo provides was giving me an error as the spring framework couldn’t convert org.bson.types.ObjectId to java.lang.long.

The Spring Framework also needs the variable names to match the getters and setters. Thus, I had to use the column attribute and rename my variables to be compliant of what spring needed to access my entities’ attributes.

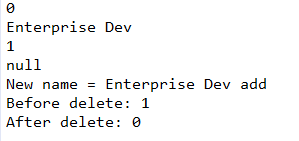
I was having issues trying to insert test values into my h2 database. I originally thought that the script “mydb.script” was being executed for my spring classes. However, this was not the case. I looked deeper into the Spring Boot documentation and found that h2 will automatically create a schema based on my entities; thus, my script wasn’t being executed. So, I had to create a file called “data.sql” in my resources folder because the documentation stated that Spring will pick up the file and insert my values in the database.

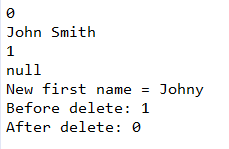
# Milestone Screenshot

## Milestone #1



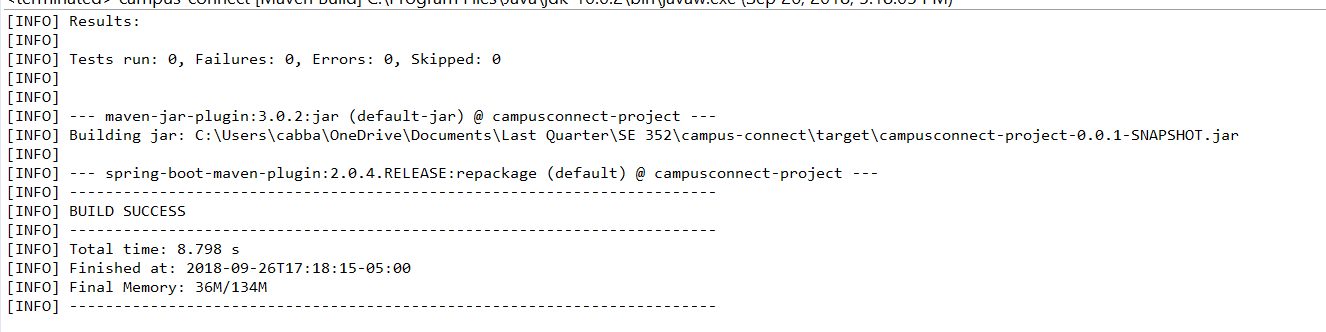
## Milestone #2



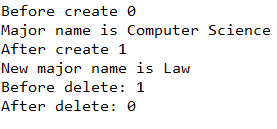


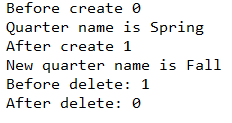
I was able to set up my hsqldb, create tables in it, and was able to use crud operations to populate it. These tests illustrate that I first get count of items in the table, then populate the table with the class name/ student name, get the number of items in each table (should be 1) then I update the name (shown by “new name” and “new first name”), and then I delete the records.

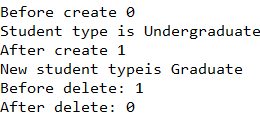
Below is my maven build success for this milestone.

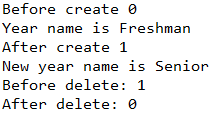


## Milestone #3



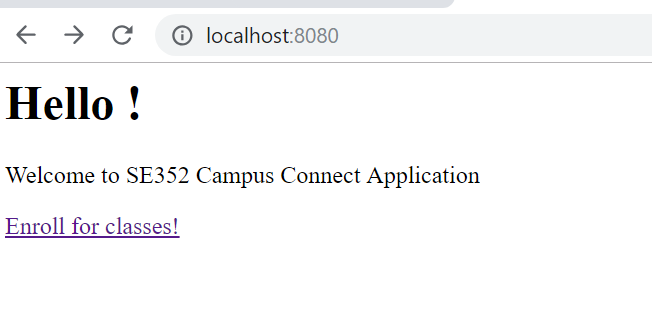


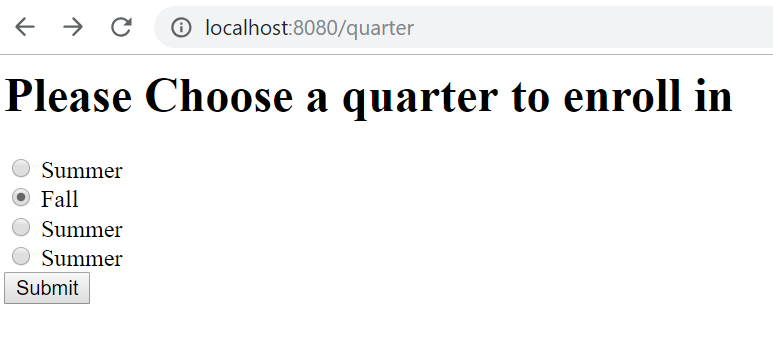


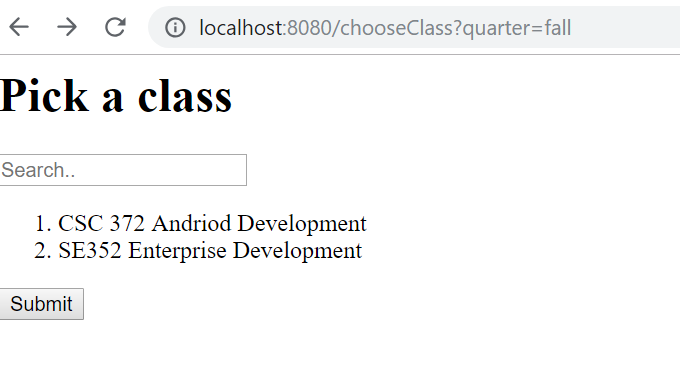


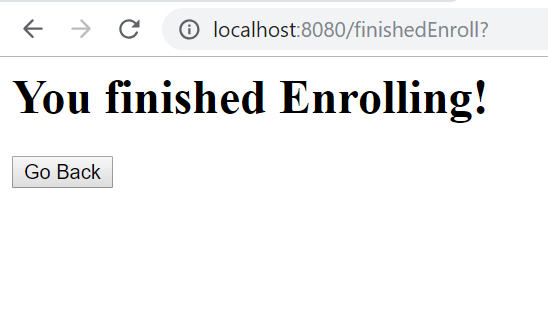
Like milestone #2 I was able to set up mongodb and ensure that all the crud operations were in working order for it. I plan on using mongodb for the tables: major, quarter, student\_type, and year.

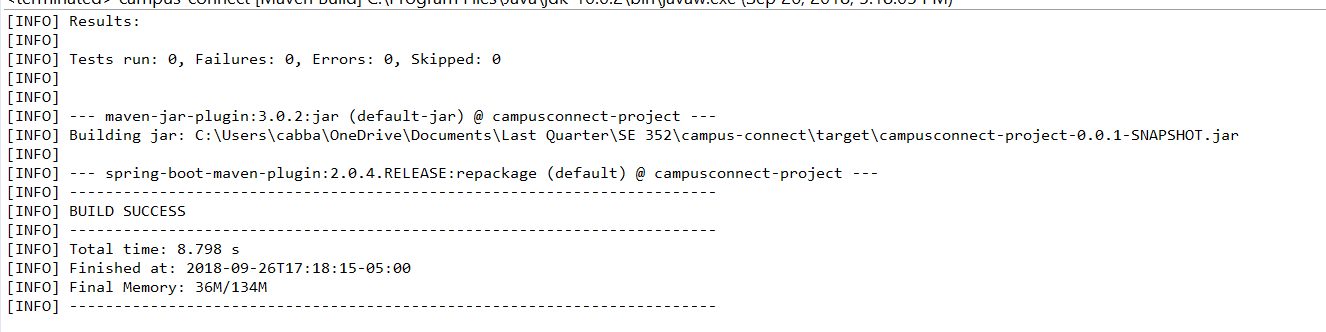
I was also able to implement my html in relation to my mockup. Flow shown below.









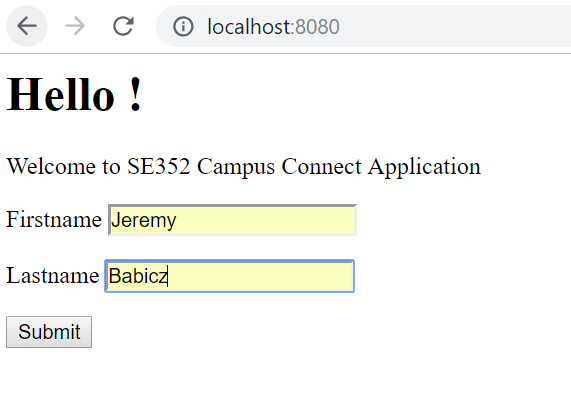


## Milestone #4

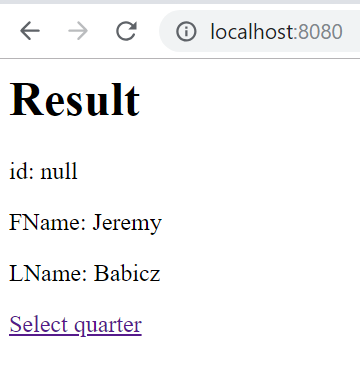
For this milestone, I was able to use the spring mvc and Thymeleaf to use to save and display data from my database. I was only able to do this in my index page where the user signs in and in my quarters page, to display all the documents in my quarters collection. I also removed hardcoded links from my UI in the previous milestone

Shown below are screenshots of the “Home” and “Quarters” pages that interact with the database.

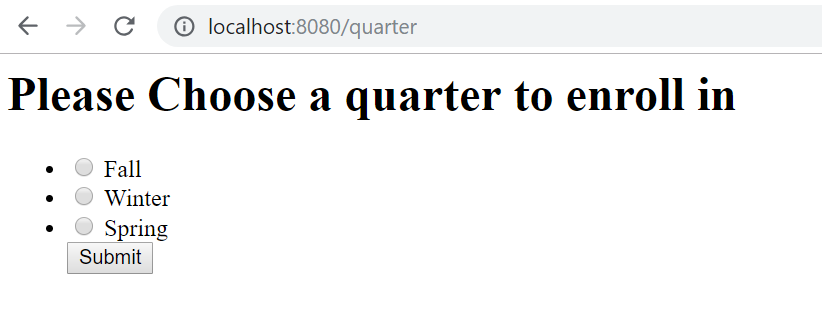
Home(Get)

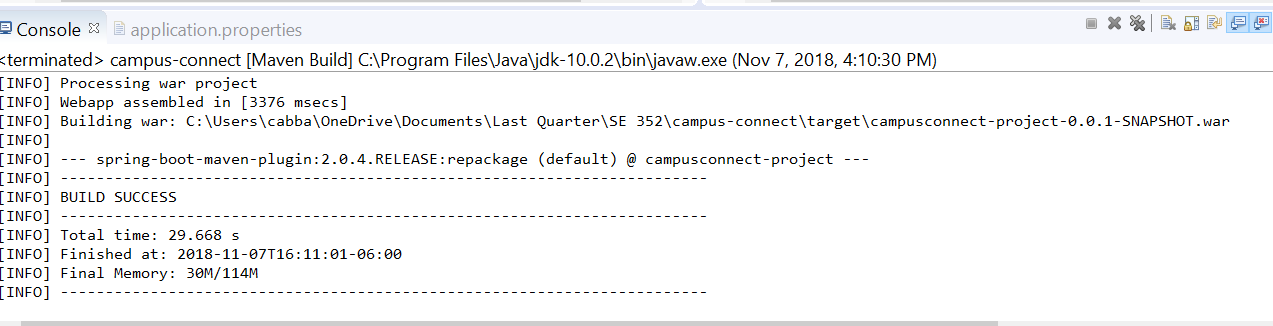


Home(Post)



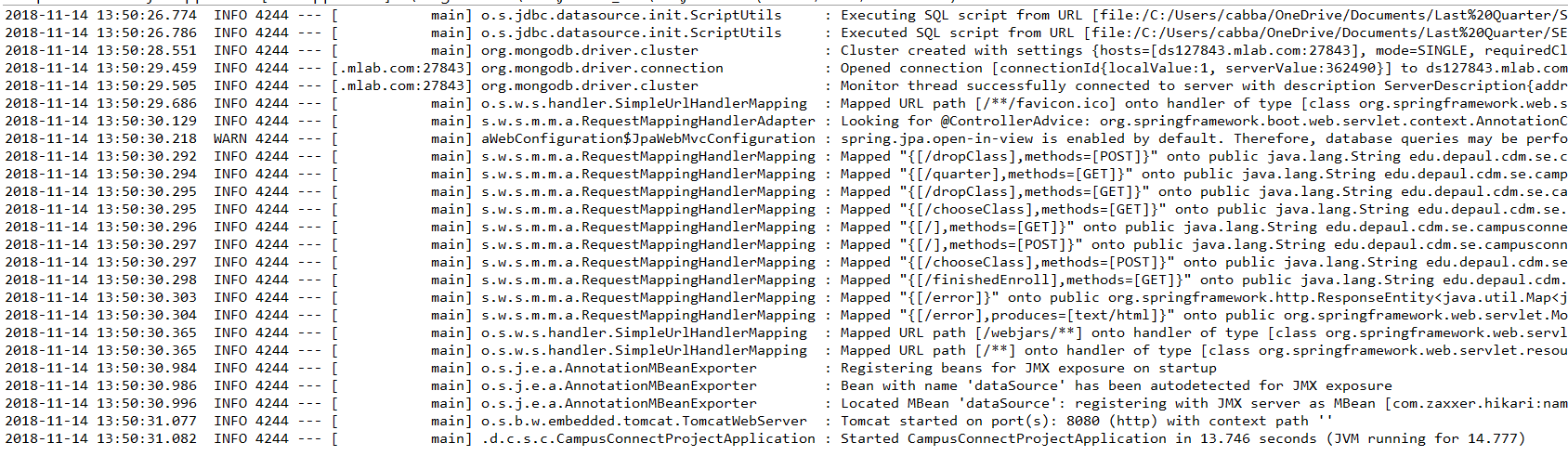
Quarters



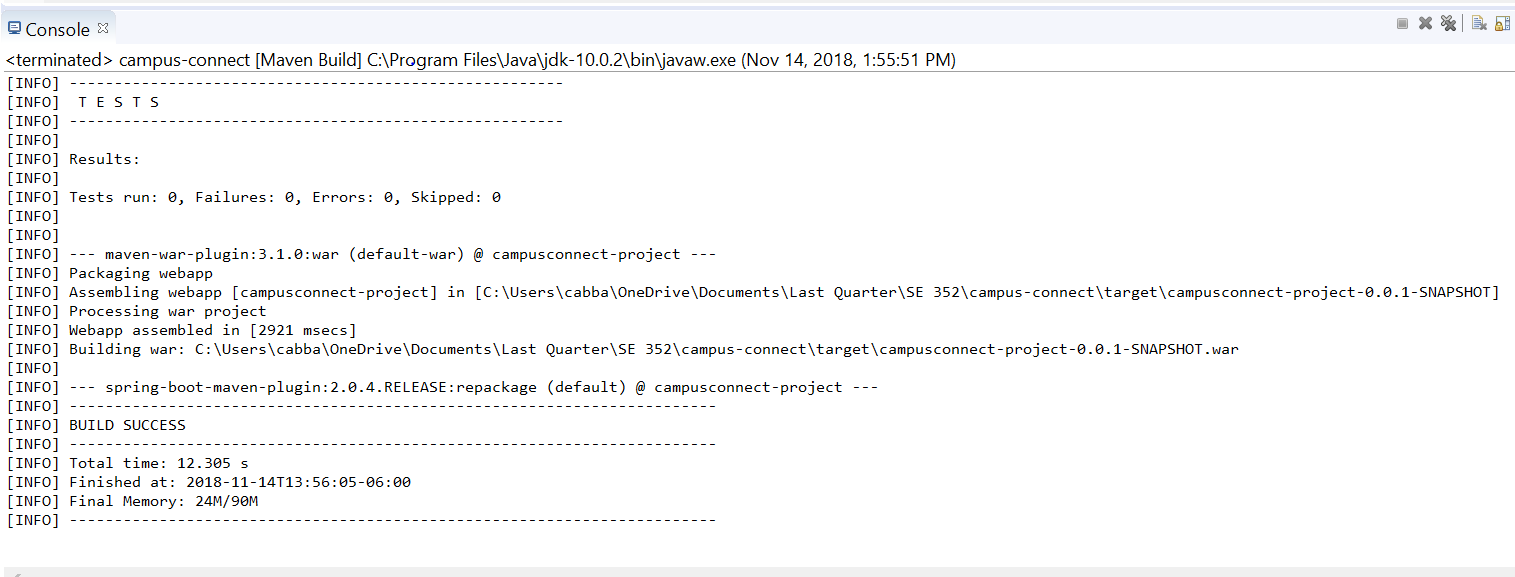


## Final Submission

Project successfully runs:



Successful build:



# Decision Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem** | **What was decided** | **Alternatives considered** | **Rationale** |
| Which IDE to use | Eclipse | Netbeans | Eclipse had a maven plugin and I am most familiar with the IDE |
| Which NoSql to use | MongoDB | Spring-Data | MongoDB was simple, was intuitive and would work well with app. |
| Which template engine to use for Spring MVC | Thymeleaf | FreeMarker | Thymeleaf had good documentation, great examples, and was intuitive. |