Aaron Willis

Billy Stockton

Ariana Konhilas

Jeremy Hunton

12/6/2021

CPT275 - Team Sezmi

**Sezmi Product Documentation - Developer**

**Project Description**

The Sezmi Course Registration project is a web based program designed to allow students to register for classes and track their progress towards graduation. The project has JavaScript, CSS and HTML on the frontend that communicates to a MySQL server via Spring Boot APIs. All relevant information like courses available at Trident, majors available at Trident, and student information is all stored within the MySQL server. To use the application, students must create a user profile and log in each time. Student’s major selection and classes are tracked within the program and will guide students to sign up for relevant classes that they need to graduate, as well as classes that they have prerequisite classes for.

**GitHub Location:**

The project is currently located at:

<https://github.com/andersonjackson-ttc/SezmiCourseRegistration>

**Downloading Instructions:**

**To download the project from GitHub you will need to:**

1. Log into GitHub. If you do not have a GitHub account, you will need to create one.
2. In the Repository search bar, search for ***andersonjackson-ttc/SezmiCourseRegistration***.
3. When the repository is pulled up, select the branch in the upper left corner to **master**. Click the green button labeled **Code** and select **Open with GitHub Desktop** to open the project in GitHub Desktop or **Download ZIP** to download the project directly to your computer. **Note: if you download as a ZIP, you will not have the benefits of backups or previous iterations of the project like you would using GitHub Desktop**

**To download the project using GitHub Desktop:**

1. Download GitHub Desktop if you haven’t already.
2. Go through steps 1-5 in **To download the project from GitHub** above. When you get the step 5, select **Open with GitHub Desktop**
3. Once in GitHub Desktop, select **SezmiCourseRegistration** from the *Current repository* dropdown menu.
4. Make Sure that the *Current branch* is **master**. If you want to go to another branch, you can, but those are not up-to-date versions of the software.
5. Click **Pull** to create a new pull request. This will download the files to your local machine.

**Installation Instructions:**

This project uses a stack composed of HTML/CSS, JavaScript, Spring Boot (Java), and MySQL. To use the project, you will need to have Spring Boot and MySQL downloaded.

**To Download Spring Boot (Spring Tool Suite):**

1. Go to <https://spring.io/tools> to download Spring Tool Suite. Select your relevant operating system using either Eclipse or VS Code IDEs.
2. Open the downloaded file and follow the instructions to install Spring Tool Suite.
3. Once downloaded, open Spring Tool Suite. Set your workplace to the **SezmiCourseRegistration** folder you pulled using GitHub Desktop. Click **Launch** once the workspace location is selected.
4. When inside Spring Tool Suite, go to **File > Import.** Click the arrow next to **General** and select **Projects from Folder or Archive** from the dropdown. Click **Next**.
5. Click **Directory…** and go to the location of the **SezmiCourseRegistration** folder you pulled from GitHub. Click **Finish.**
6. The project files will now import to Spring Tool Suite. You will know this is finished because the package will appear within the **Package Explorer**.

**To Download MySQL Server:**

1. Go to <https://dev.mysql.com/downloads/workbench/> to download MySQL Workbench for your system. You will need to create an Oracle account to successfully download MySQL Workbench, so please do so if you do not have an account.
2. Once downloaded, run the installation program and follow the prompts.

**Running the Project**

1. To run the project, open up the **SezmiCourseRegistration** folder in Spring Tool Suite (see Download Spring Tool Suite if needed) and navigate in the **Project Explorer** (left pane) to **src/main/java > Sezmi.TridentTechCourseRegistration > TridentTechCourseRegistrationApplication.java**. You can either **right click** on the file and select **Run As > 1 Java Application** *or* left click the file and click the run icon at the top toolbar (green button with an arrow).
2. The project server is currently set up at the url ***//192.168.6.3:3306***. If you want to use your own MySQL Server, you would need to change the **application.properties** file in **src/main/resources > application.properties** (more info in Changing the Server below).

**Changing the Server**

1. To change the server from the default server for the project, open Spring Tool Suite and load the project.
2. Go to **src/main/resources > application.properties** to access the application properties.
3. To change the server URL, edit line 3 **spring.datasource.url** and set the address equal to the port you want to use for your server. If you aren’t using a MySQL server, you will also need to change the **jdbc:mysql** on that line to **jdbc:[server\_type]**.
4. **Note: if you change the server, you must also update the username and password in lines 4 and 5 of the application properties**.
5. MAKE SURE IF YOU CREATE A NEW SERVER THAT YOU KEEP THE SAME NAMING CONVENTION FOR TABLES OR YOU WILL HAVE LOAD ISSUES.

**Detailed Project Description**

**Current Database Layout**

1. course
   1. **PK**: course\_id
   2. course\_name
   3. course\_type
2. course\_prereq (a Join table of course and course)
   1. **PK**: course\_id & course\_prereq
3. course\_section (a Join table of course and section)
   1. **PK**: course\_id & section\_id
4. major
   1. **PK**: major\_id
   2. major\_name
5. major\_course (a Join table of major and course)
   1. **PK**: major\_id & course\_id
6. section
   1. **PK**: section\_id
   2. instructor\_id
   3. schedule
   4. term
   5. duration
   6. time
   7. course\_format
   8. remaining\_spaces
   9. **FK**: course\_id
7. student
   1. **PK**: id
   2. email
   3. last\_name
   4. first\_name
   5. salt
   6. password
   7. **FK**: major\_id
8. student\_course(a Join table of student and course)
   1. **PK:** id & course\_id
9. student\_section(a Join table of student and section)
   1. **PK:** id & section\_id
10. user
    1. **PK**: id
    2. User

**Main Model Classes**

1. TridentTechCourseRegistrationApplication
   1. **Description**: responsible for starting the program.
2. Course
   1. **Description**: responsible for housing information from the course table in the database. Also responsible for holding course\_section and course\_prereq information within Sets.
3. CourseController
   1. **Description:** responsible for handling the API creation for the Course class.
4. CourseRepository
   1. **Description:** responsible for handling Hibernate/SQL communication between Spring Boot and the MySQL server for the Course class.
5. CourseService
   1. **Description:** responsible for handling the business logic for the Course class from the repository.
6. Major
   1. **Description**: responsible for housing information from the major table in the database. Also responsible for holding major\_course information within a Set.
7. MajorController
   1. **Description:** responsible for handling the API creation for the Major class.
8. MajorIdName
   1. **Description:** interface that handles limited information from the server for specific API calls (only contains the major\_id and major\_name)
9. MajorRepository
   1. **Description:** responsible for handling Hibernate/SQL communication between Spring Boot and the MySQL server for the Major class.
10. MajorService
    1. **Description:** responsible for handling the business logic for the Major class from the repository.
11. Section
    1. **Description**: responsible for housing information from the section table in the database.
12. SectionController
    1. **Description:** responsible for handling the API creation for the Section class.
13. SectionRepository
    1. **Description:** responsible for handling Hibernate/SQL communication between Spring Boot and the MySQL server for the Section class.
14. SectionService
    1. **Description:** responsible for handling the business logic for the Section class from the repository.
15. Student
    1. **Description**: responsible for housing information from the student table in the database. Also responsible for holding student\_section and student\_course information within Sets.
16. StudentController
    1. **Description:** responsible for handling the API creation for the Course class.
17. StudentEmailMajor
    1. **Description:** interface that handles limited information from the server for specific API calls (only contains the email and major\_id)
18. StudentRepository
    1. **Description:** responsible for handling Hibernate/SQL communication between Spring Boot and the MySQL server for the Student class.
19. StudentService
    1. **Description:** responsible for handling the business logic for the Student class from the repository.

**API URLs**

**CourseController class**

1. **/courses**
   1. Get mapping responsible for getting all of the courses offered at Trident. Also has post mapping to allow an admin to add a course to the list.
2. **/courses/{courseID}**
   1. Get mapping responsible for getting a single course by the course ID. Also has put mapping to allow an admin to update the information in the course, as well as a delete mapping to remove a course by an admin.
3. **/courses/{course\_id}/sections**
   1. Get mapping responsible for getting all the sections available for a single course.
4. **/courses/{course\_id}/pre\_reqs**
   1. Get mapping responsible for getting the set of prerequisite courses needed to take a course.

**MajorController class**

1. **/majors**
   1. Get mapping that returns a list of all majors offered at Trident. Also has post mapping to allow an admin to add a major to the major table if needed.
2. **/majors/{major\_id}**
   1. Get mapping that returns a single major given the major id. Also has put mapping to allow an admin to update the info in the major if needed, as well as a Delete mapping if an admin wants to delete a major.
3. **/majors/{email}/courses**
   1. Get mapping that returns a list of courses that a student hasn’t taken based upon their email address.

**SectionController class**

1. **/section**
   1. Get mapping that lists all of the sections available at Trident.Also has post mapping to allow an admin
2. **/section/{section\_id}**
   1. Get mapping that displays the section based on the section ID. Also has delete mapping if an admin wants to delete a section.
3. **/section/{courseID}**
   1. Put mapping that allows an admin to edit the section based upon the section ID.

**StudentController class**

1. **/student/**
   1. Mapping responsible for getting all of the students on the server. Post mapping also exists to add a new student.
2. **/student/{email}**
   1. Get mapping responsible for getting a student based upon their email address.
3. **/student/{email}/course\_prereq\_true**
   1. Get mapping responsible for returning a list of courses that the student has prerequisite classes for.
4. **/student/{email}/course\_prereq\_false**
   1. Get mapping responsible for returning a list of courses that the student does not have prerequisite classes for.
5. **/student/{email}/courses**
   1. Get mapping responsible for returning the list of courses that the student has taken.
6. **/student/{email}/sections**
   1. Get mapping responsible for returning the list of sections that the student has chosen.
7. **/student/{email}/major**
   1. Get mapping responsible for returning only the student’s email and major.
8. **/student/{id}**
   1. Put mapping that allows the student to edit their information based upon their id number. Also has Delete mapping for potential Admin use.
9. **/student/{id}/{course\_id}**
   1. Patch mapping for adding a course the student has taken to the set of courses taken.
10. **/student/{id}/{section\_id}**
    1. Patch mapping for adding a section to the student set of sections enrolled in.

**Future Direction**

It would be beneficial to add admin functions to the program. While a lot of admin data handling has been created in Spring Boot, an admin user would need to be created within the database and given super privileges to safely access the admin methods like deleting and editing contents in the database. Along with admin privileges, it would also be beneficial to allow for users to be able to change their password via an email confirmation. It would also help users to have email updates when they sign up for a class, or it is nearing time for them to sign up for more classes.