**PROJECT GENERAL DESCRIPTION**

**Course Name:**

**CPT-275 COMPUTER TECH SENIOR PROJECT (W02)**

**Instructor:**

Anderson H. Jackson

**Date:**

January 3, 2023 - April 27, 2023

**Project Name:**

**2023 Spring Data General Registration**

**Developers:**

**Stephen A. Prine, Curtis Wilson, John H. Percival, Joshua T. Mcmorrow**

**Project Description:**

The 2023 Spring Data General Registration, the project has been designed with JavaScript, CSS, and HTML in the interface, it also integrates the Spring Boot and Spring Thymeleaf libraries and APIs with the MVC architecture, to give a fresh and modern appearance to the html templates.

The frontend communication is carried out through the MySql server, where all the data entered in the application is stored.

The program has two types of secure access: as a student and as an Administrator.

In the student section, the student can create a profile. After meeting the registration requirements, they will have access to a variety of functionalities and menus, among which are: a description of the core (required prerequisites, necessary electives), search and registration of courses, status, and description of courses, see the obtained grades, filter, and choose courses by terms, see registered courses and schedule.

In the administrator section, the administrator can search by student, see their information, and modify it, in the same way, they can search and modify courses, core, and add grades and modify the status of the courses.

**Project Location:**

The project is currently located at:

https://github.com/andersonjackson-ttc/2023-Spring-Data-General-Registration

**Downloading Instructions:**

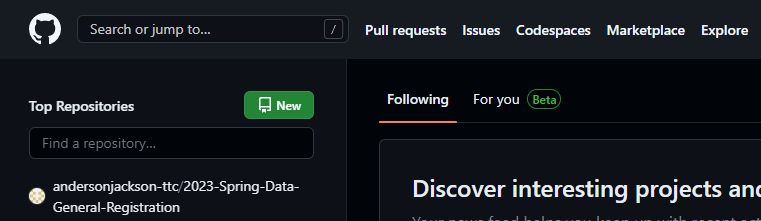
**To download the project from GitHub you will need to:**

1. Log into GitHub. If you do not have a GitHub account, you will need to create one.

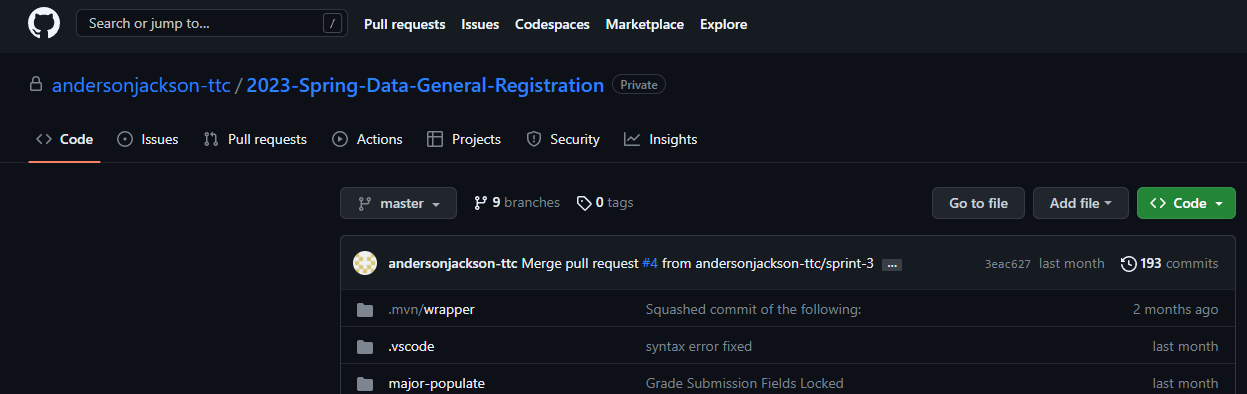
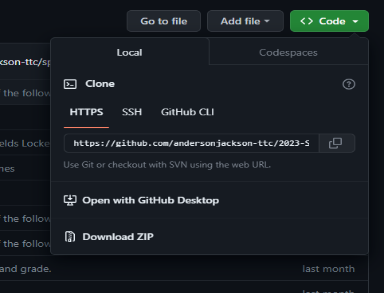
A screenshot of a computer

Description automatically generated

1. In the Repository search bar, search for **andersonjackson-ttc/2023-Spring-Data-General-Registration**

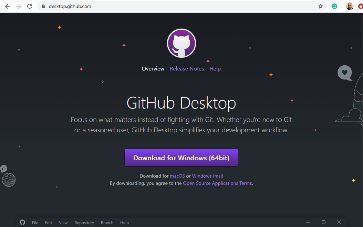


1. When the repository is pulled up, select the branch in the upper left corner to **master**. Click the green button labeled **Code** and select **Open with GitHub Desktop** to open the project in GitHub Desktop or **Download ZIP** to download the project directly to your computer.

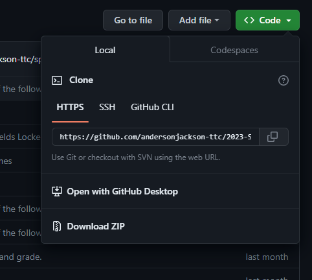


**To download the project using GitHub Desktop:**

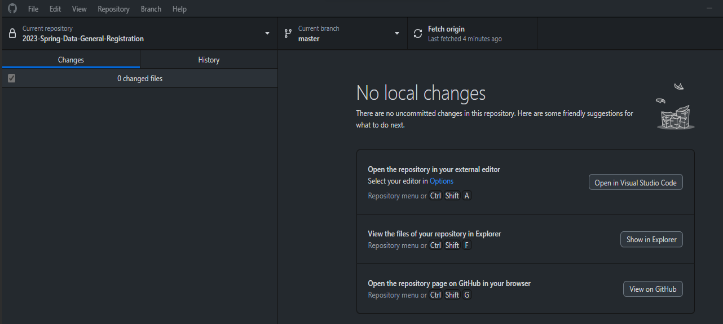
1. Download GitHub Desktop if you haven’t already.



1. Go through steps 1-5 in **To download the project from GitHub** above. When you get the step 5, select **Open with GitHub Desktop**



1. Once in GitHub Desktop, select **2023-Spring-Data-General-Registration** from the *Current repository* dropdown menu.
2. Make Sure that the *Current branch* is **master**.



1. Click **Pull** to create a new pull request. This will download the files to your local machine.

**Installation Required:**

This project includes HTML/CSS, JavaScript, Spring boot (Java), Thymeleaf, maven-Wrapper and MySQL. In order to use this project, you need to download Spring Boot and MySQL Workbench and Add Thymeleaf Snippets extension to you text editor(vscode recommended).

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

1. A white background with black text

   Description automatically generated with low confidenceGo 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 **2023-Spring-Data-General-Registration** 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 **2023-Spring-Data-General-Registration** 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**.

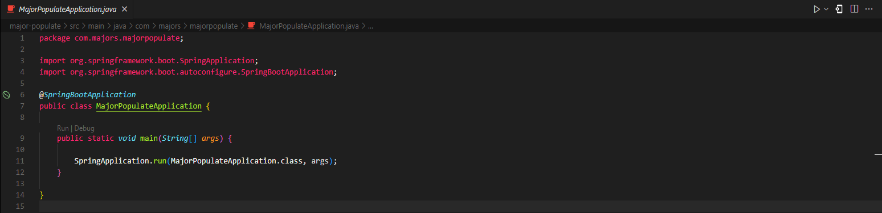
**To Download MySQL Server:**

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.

**Running the Project**

1. To run the project, open up the **2023-Spring-Data-General-Registration** folder in Code Editor (Visual Studio Code) and navigate in the **Project Explorer** (left pane) to **major-populate/src/main/java/com/majors/majorpopulate/MajorPopulateApplication.java A screenshot of a computer program

   Description automatically generated with medium confidence**
2. 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.



1. The project server is currently set up at the URL ***//127.0.0.1:3306***.
2. **A screen shot of a computer

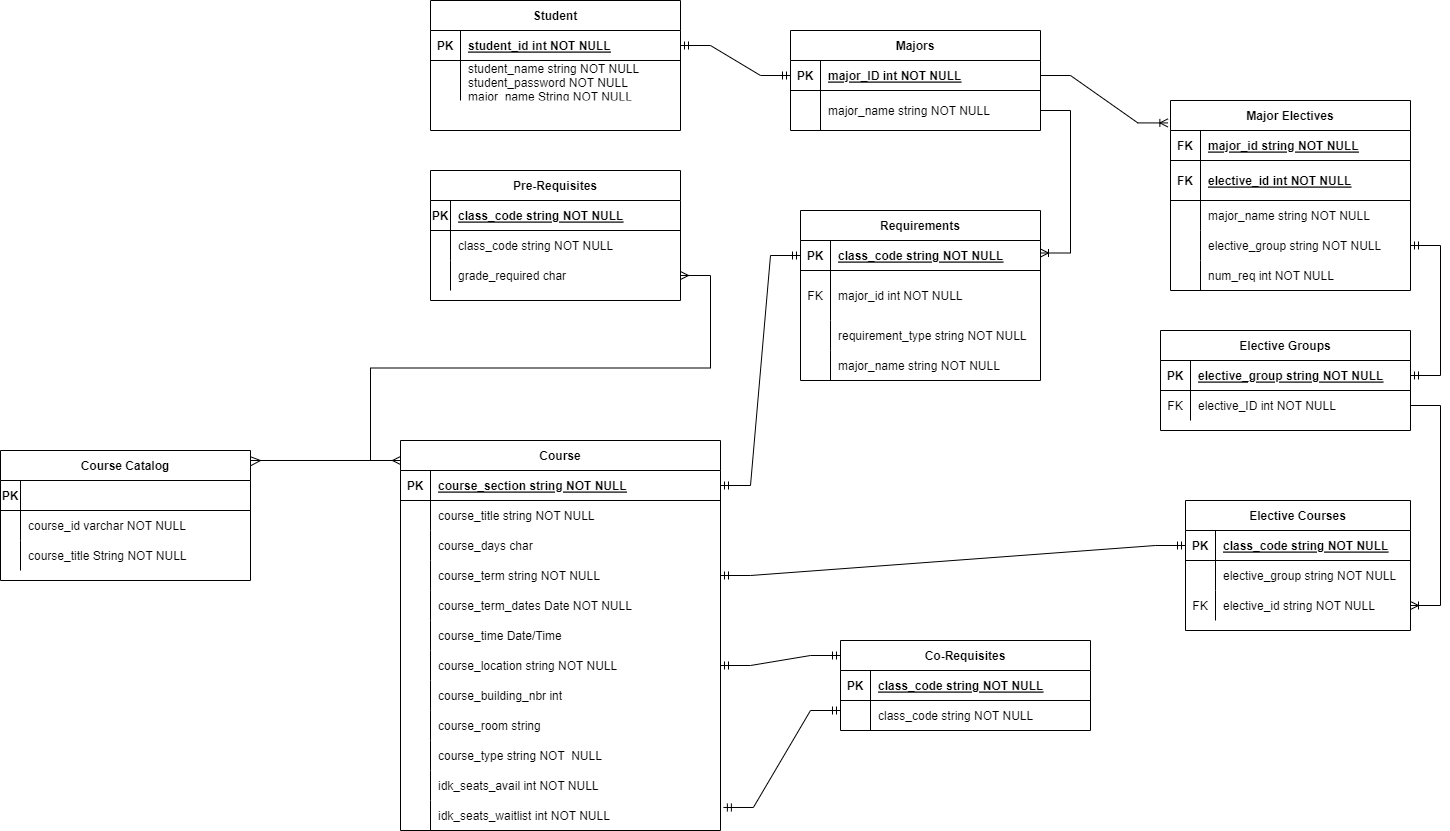
   Description automatically generated with medium confidence**If you want to use your own MySQL Server, you would need to change the the **application.properties** file in **major-populate/src/main/resources/application.pro perties>**

**Install Local Server To Get Access To Database**

1. Choose a local server like Wamp or Xampp according to your work environment and operating system.
2. <https://www.wampserver.com/en/> or
3. <https://www.apachefriends.org/>
4. Once downloaded, run the installation program and follow the prompts.
5. Once your local server is running, make sure the workbench database is setup and connected:
   1. Host: 127.0.0.1
   2. Port: 3306
   3. Username: root
   4. Password: root
   5. Database: cpt275\_db

**DATABASE TABLES:**

1. **tbl\_co\_req**
   1. course\_id\_c1
   2. course\_id\_c2
2. **tbl\_course\_catalog**
   1. course\_id
   2. course\_title
3. **tbl\_courses\_offered**
   1. course\_title
   2. course\_section
   3. course\_days
   4. course\_term
   5. course\_term\_dates
   6. course\_time
   7. course\_location
   8. course\_building\_nbr
   9. course\_room
   10. course\_type
   11. total\_seats
   12. seats\_taken
   13. section\_id
   14. course\_id
4. **tbl\_elective\_courses**
   1. elective\_id
   2. elective\_name
   3. course\_id
5. **tbl\_elective\_groups**
   1. elective\_name
   2. elective\_id
6. **tbl\_grad\_requirement**
   1. major\_id
   2. major\_name
   3. req\_type
   4. course\_id
7. **tbl\_major\_electives**
   1. major\_id
   2. major\_name
   3. elective\_group
   4. nbr\_required
8. **tbl\_majors**
   1. major\_id
   2. major\_name
9. **tbl\_pre\_reqs**
   1. course\_id
   2. prereq
   3. min\_grade
10. **tbl\_registration**
    1. student\_id
    2. major\_id
    3. course\_id
    4. section\_id
    5. term
    6. reg\_dts
11. **tbl\_student**
    1. **PK.** id
    2. name
    3. password
    4. major\_name
12. **tbl\_student\_login**
    1. student\_id
    2. student\_password
    3. password\_expiration
    4. password\_hint\_q1
    5. password\_hint\_a1
    6. password\_hint\_q2
    7. password\_hint\_q1
13. **tbl\_student\_profile**
    1. **PK.** student\_id
    2. student\_first\_name
    3. student\_last\_name
    4. major\_id



**Main Model Classes**

**API URLs**

1. AdminController
   1. **Description:** responsible for handling the API creation for the MajorService class.
2. CourseController
   1. **Description:** responsible for handling the API creation for the CourseService class.
3. GradeController
   1. **Description:** responsible for handling the API creation for the GradeService class.
4. MajorPopulateController
   1. **Description:** responsible for handling the API creation for the Major class, Section class, AdminService, MajorService class, Login class, Student Class.
5. POJO
   1. Co.Req

**a. Description:** this class represents the corequisites table in the database. This is used to perform database operations including retrieving a course’s corequisite data using a course id.

* 1. CourseDTO

**a. Description:** this class represents the course catalog table in the database. This is used to perform database operations involving the course catalog including getting course names from the course id and saving courses to the catalog.

* 1. CourseOffers

**a. Description:** used to store data pertaining to individual courses. Used when retrieving courses offerings from the database.

* 1. ElectiveCourses

**a. Description:** this class represents the elective courses table in the database. This is used to perform database operations including retrieving elective courses by elective group.

* 1. Grade

**a. Description:** this class represents the student transcript (grades) table in the database. This is used to perform database operations including saving student grades.

* 1. GradRequirements
  2. **Description:** this class represents the graduation requirements table in the database. This is used to perform database operations including retrieving graduation requirements based on major.
  3. MajorDTO

**a. Description:** this class represents the majors table in the database. This is used to perform database operations including retrieving all majors.

* 1. MajorElectives

**a. Description:** this class represents the majors electives table in the database. This is used to perform database operations including retrieving individual major electives based on their ids in the database.

* 1. Prereq

**a. Description:** this class represents the prerequisites table in the database. This is used to perform database operations including retrieving course prerequisites based on a course id.

* 1. RegisteredSection

**a. Description:** this class is used to store registered section data using a student id and represents the student schedule.

* 1. SearchTerm

**a. Description:** this class stores a search term description and supports the ability to search for a course.

* 1. SectionDTO

**a. Description:** this class represents the courses offered table in the database. This is used to perform database operations including retrieving offered courses according to a course id.

1. repository
   1. AdminRepository

**a. Description:** this class supports the ability to perform database operations for application administrative functions.

* 1. CoReqRepository

**a. Description:** this class supports the ability to perform database operations against the corequisites table.

* 1. CourseDTORepository

**a. Description:** this class supports the ability to perform database operations against the course catalog table.

* 1. ElectiveCourseRepository

**a. Description:** this class supports the ability to perform database operations against the elective courses table.

* 1. GradeRepository

**a. Description:** this class supports the ability to perform database operations against the student transcript (grades) table.

* 1. GradReqRepository

**a. Description:** this class supports the ability to perform database operations against the graduation requirements table.

* 1. MajorDTORepository

**a. Description:** this class supports the ability to perform database operations against the major’s table.

* 1. MajorElectRepository

**a. Description:** this class supports the ability to perform database operations against the major electives table.

* 1. PreReqRepository

**a. Description:** this class supports the ability to perform database operations against the PreReq table.

* 1. RegistrationRepository

**a. Description:** this class supports the ability to perform database operations against the registration table.

* 1. SectionRepository

**a. Description:** this class supports the ability to perform database operations against the courses offered table.

* 1. SqlCaller

**a. Description:** this class represents the data access layer for the service as, much like the other repositories in this list, it allows the service to execute custom database operations.

1. service
2. AdminService

**a. Description:** this service class contain operations that allows the service to manage the student within the overall application. These operations include the ability to add students, update grades, change majors, and set courses to completed.

1. CourseService

**a. Description:** this service interface contains operations that allow the service to retrieve data with respect to a course or elective id. These operations include the ability to retrieve course prerequisites, get course requirements, get course name, and get courses offered.

1. CourseServiceImpl

**a. Description:** this class is an implementation of the CourseService interface. This contains logic that executes the aforementioned operations mentioned in CourseService.

1. GradeService

**a. Description:** this service interface contains an operation that allow the service to save student grades.

1. GradeServiceImpl

**a. Description:** this class is an implementation of the GradeService interface. This contains logic that executes the aforementioned operations mentioned in GradeService.

1. MajorService

**a. Description:** this service class contain operations that allows the service to retrieve/modify major, term, and course data and also manages user authentication (login and registration).

1. MajorService2

**a. Description:** this service interface contains an operation that allows the service to retrieve all stored majors.

1. MajorServiceImpl

**a. Description:** this class is an implementation of the MajorService2 interface. This contains logic that executes the aforementioned operation mentioned in MajorService2.

1. RegistrationService

**a. Description:** this service interface contains an operation that allows the service to retrieve all stored registration.

1. RegistrationServiceImpl

**a. Description:** this class is an implementation of the RegistrationService. This contains logic that executes the operation mentioned in registrationService.

1. StudentService

**a. Description:** this service interface contains an operation that allows the service to retrieve all stored student.

1. RegistrationServiceImpl

**a. Description:** this class is an implementation of the StudentService. This contains logic that executes the operation mentioned above in StudentService.

1. student
2. Login
   1. **Description:** this class contains all of the data required by students to login to the application. This class also associates major data to the student.
3. Student

**a. Description:** this class contains student information including name, further authentication details, and major.

1. Course

**a. Description:** this class contains information pertaining to a course including data on available classes, course name, id, prerequisites, corequisites, status, and grade.

1. Major

**a. Description:** this class contains information pertaining to a major including data on required courses, name, id, and major elective group.

1. MajorPopulateApplication

**a. Description:** this class contains a static method `main()`, and a @SpringBootApplication annotation, which configure and execute the Spring Boot application.

1. Section

**a. Description:** this class contains information pertaining to available courses including data on course name, date/times, availability, and location.

1. resources
   1. static (javascript and styles.css)

**a. Description:** these files contain logic and styling specifications that determine the behavior of the web application.

* 1. templates (html files)
  2. **Description:** these files contain templating information that will determine the layout and content - from the service - to be present within the web application.
  3. application.properties

**a. Description:** this file contains configuration properties that will be utilized within the service. It primarily contains properties that define the data source location and its authentication details for access during database operations.