# College of the North Atlantic
# CP3566 – Applied Java Programming
# Final Project

**Due Date: March 3, 2022**

## Final Project – Education System API

**Based on the Spring Framework**

**Context:**

Colleges and universities require efficient and feature-rich data management systems to provide a variety of necessary data to students, employees, and faculty. College of the North Atlantic (CNA) has awarded you a contract to provide a new student and employee system that will expose data to other related and existing systems that are already in place.

Ultimately, you will create a RESTful web service, using Spring, that will provide these capabilities to the College. The system will extend from work you completed in Assignment 4, and you should be able to continue with your development. If you do wish, or need, to start a new project, ensure that you select the following dependencies on project creation:

- Spring Web,
- Spring Data JPA
- MariaDB (or MySQL) Driver
- Remember, if you do miss one, or require others, you can certainly add them to your pom.xml file a little later.

**Requirements:**

Operational

- CNA will require an ability to test the provided API. CNA technical staff that will evaluate the system will complete all testing with Postman.
- Your database should be called **educationsystem**, and all tables required should be contained within.
- The structure and creation of your tables can be created and managed by Spring components, or manually created (or a mixture). Details of required attributes and/or relationships are outlined below.
- The application should be accessible at this URI (obviously your server name, port, and application/project name will come before this): **/api/cna/**
- Ensure that you use the naming conventions that are described below. As these are what other systems will use to interface with your API.

**Objectives (note that these objectives are not necessarily in the order that you may complete development):**

1. Create a class called Student.java (should exist from assignment 4) which will act as an entity class. The class should have the following attributes:
    - studentId (which will act as the table primary key, and should be auto-incremented)
    - firstName
    - lastName
    - email
    - address
    - city
    - postal
    - phone

    The class should have get/set methods for each attribute.

2. Create a class called Course.java (should exist from assignment 4) which will act as an entity class. The class should have the following attributes:
    - courseId (which will act as the table primary key, and should be auto-incremented)
    - courseName
    - courseNumber
    - capacity
    - year
    - semester (i.e. Fall, Winter, Spring)
    - pid (related to program records)

    The class should have get/set methods for each attribute.

3. Create a class called Programs.java which will act as an entity class. The class should have the following attributes:
    - pid (primary key, autoincrement)
    - programName
    - campus

    The class should have get/set methods for each attribute.

4. Create a class called Enrollment.java which will act as an entity class. The class should have the following attributes:
    - eid (primary key, autoincrement)
    - courseId (related to course records)
    - studentId (related to your course records)

    The class should have get/set methods for each attribute.

5. Create a class called Grades.java which will act as an entity class. The class should have the following attributes:
    - gid (primary key, autoincrement)

- studentId (related to student records)
- courseId (related to course records)
- grade

The class should have get/set methods for each attribute.

6. Create an interface for each of the above classes, that Spring will use to auto-implement them into a Bean.

7. Create a controller for each object, Student, Course, Program, Enrollment and Grades, that allows you to complete the following objectives through your API:

   StudentController:
   - Add – a particular student
   - List – all students
   - View – one student based on id
   - Modify a student record
   - Delete a student record
   - The path for student functionality, from a web access perspective, should be "**student**"

   CourseController:
   - Add – a particular course
   - List – all courses
   - View – one course based on id
   - Modify a course
   - Delete a course
   - The path for course functionality, from a web access perspective, should be "course"

   ProgramsController:
   - Add – a particular program
   - List – all programs
   - View – one program based on id
   - Modify a program
   - Delete a program
   - The path for program functionality, from a web access perspective, should be "program"

   EnrollmentController:
   - Add – a enrollment for a student in a course
   - List – all enrollment based on course
   - List – all enrollment based on student
   - Modify an enrollment
   - Delete an enrollment
   - The path for course functionality, from a web access perspective, should be "enrollment"

GradesController:

- Add – a particular grade
- List – all grades for a student
- List – all grades for a course
- Modify a grade
- Delete a grade
- The path for course functionality, from a web access perspective, should be “grades”

8. Even though there are some obvious relationships going on in the above data structures, you do not need to be too concerned about referential integrity as it relates to foreign keys and data that exists in other tables from a database perspective. Note that you should handle situations as best as you can programmatically.

9. Obviously, you will also require a main class to launch your Spring Boot application.

10. Make any adjustments or default overrides in your application.properties file.

11. Assume for this assignment that all testing will be done through Postman or IntelliJ’s internal generated HTTP requests.

**When complete, submit to the Project Dropbox.**