Group Project (50%) - Student Grade Management Program

Overview

A senior developer who had led projects in your team for a long time recently left your company, the COMP0010 Software. This developer was the only one who had worked on the recent project requested by the customer, Awesome University (a fictional university). The customer wants a web based student grade management program, specifically built using Spring Boot and React. However, the customer did not specify features they wanted, but they said that they will leave it up to your company. The CEO of your company promised the customer that your team will demonstrate the first version of the program with wonderful features in December 2025.

Now, you need to collaborate with two other junior developers to develop the system. Unfortunately, your manager has informed you that you cannot expect the senior developer to answer your questions...

Fortunately (or unfortunately), the senior developer left behind some design documents and code snippets for you to use. You now need to figure out how to continue the project and demonstrate a minimum viable product (MVP) to the customer. The legacy left by the senior developer is outlined below:

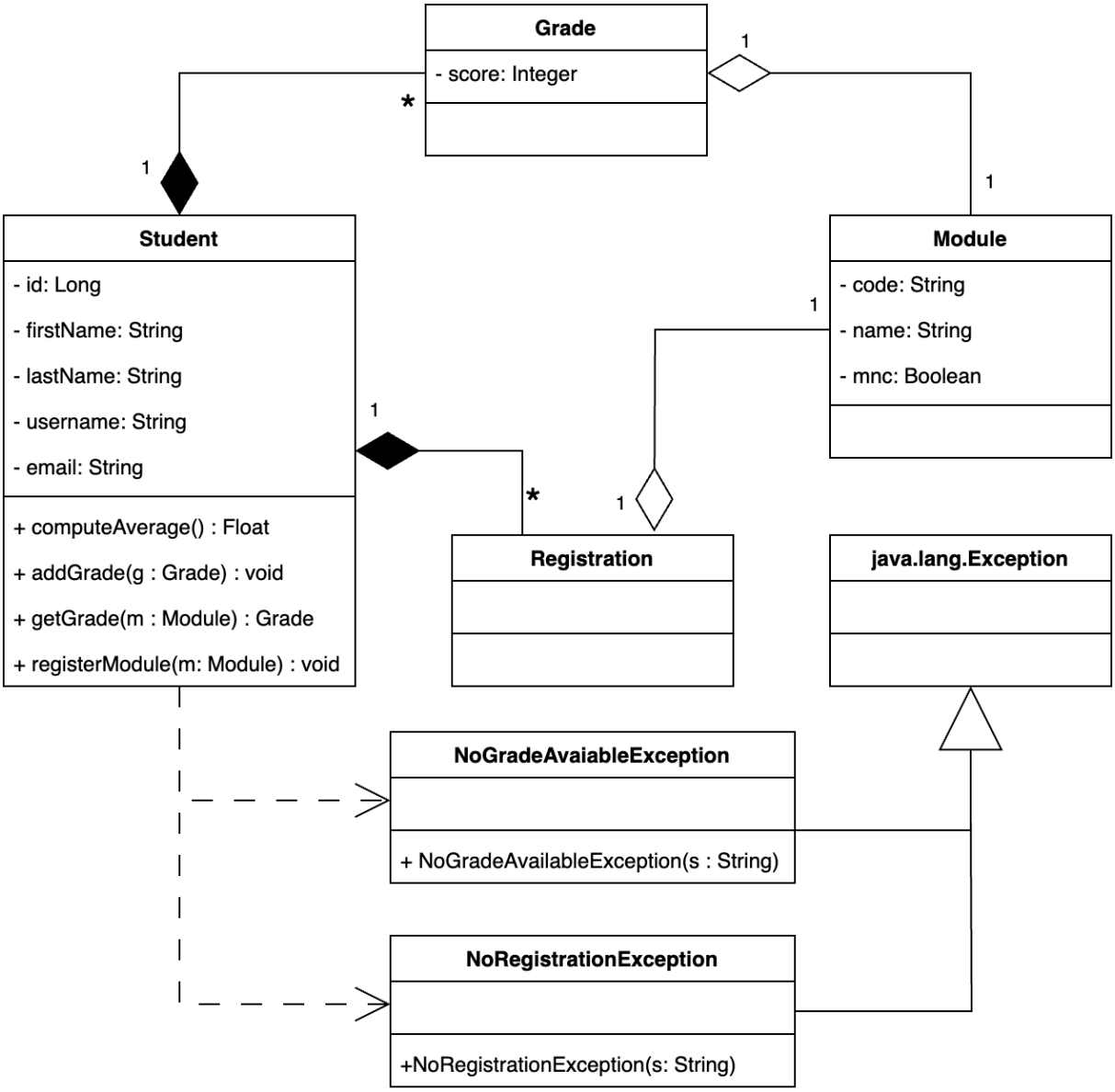
- A class diagram illustrating the basic design of model classes and their relationships
- Working frontend implementations written in TypeScript and JavaScript (your team can choose either one)
- Spring Boot project configurations

Requirements

- Your code **MUST** follow COMP0010 Coding Standards.
- Your group should create a team in GitHub Classroom and all works should be done within the repository.
  - If you use your own repository, we are not able to grade your project.
- You should use TDD process to implement your code.
  - In other words, all main code changes should come with tests for the changes.
- All your tasks should be specified in GitHub issues.

    - You need to update the issues (e.g., adding comments and update status) along your work.
- All commits should be reviewed by your team via pull request.
    - **No direct commits to the main branch**.
- Your group should brainstorm and implement the features your customer may want. For example, but not limited to:
    - Compute average grade for each module
    - Compute average grade for each student
    - Record academic year in the grade

Basic Design



- The classes should be located in the package, `uk.ac.ucl.comp0010.model`.
- You can locate the exception classes in a separate package.
- Getters & Setters are omitted in the diagram
- The fields are omitted if the diagram shows the relationship between two classes.

- The `mnc` field in `Module` stands for mandatory non-condonable.
- You should give meaningful names to the method parameters, although it appeared as a single character in the diagram due to the limited space.
- `NoGradeAvailableException` can be thrown if there is no grade available at all or not available for a specific module.
- `NoRegistrationException` can be thrown if a user try to access grades for unregistered modules.

Use of Spring

- Initialise a new Maven project by using this link
  - You may want to change the `articfactId` , `name` , and `description` to represent your group.
  - However, the package name should be `uk.ac.ucl.comp0010`
- Set up the Maven build plugins and reporting plugins based on COMP0010 coding standards
- Your repository classes should extend `org.springframework.data.repository.CrudRepository` .
- You should use `spring-boot-starter-data-rest` to expose your repository classes over REST.
- You need to implement `GradeController` to handle POST requests to the endpoint `/grades/addGrade` .
  - The method signature for the `addGrade` method should be as below:

```
@PostMapping(value = "/grades/addGrade")
public ResponseEntity<Grade> addGrade(@RequestBody Map<String, Strin
  // Find the student by using student_id
  // Find the module by using the module_code
  // Create a Grade object and set all values
  // Save the Grade object.
  // Return the saved Grade object.
}
```

- For the database, please use H2 in-memory database. We do not need to persist the data for now.
- Please use the `schema.sql` file which is written in the PostgreSQL syntax as below.
  - This file covers the classes described in the above class diagram.
  - If you extend your project, you may need to add more tables and columns in the schema file.

```
DROP TABLE IF EXISTS grade CASCADE;
DROP TABLE IF EXISTS registration CASCADE;
DROP TABLE IF EXISTS student CASCADE;
DROP TABLE IF EXISTS module CASCADE;


CREATE TABLE student(
  id INT PRIMARY KEY,
  firstName VARCHAR(30),
  lastName VARCHAR(30),
  username VARCHAR(30),
  email VARCHAR(50)
);

CREATE TABLE module(
  code VARCHAR(10) PRIMARY KEY,
  name VARCHAR(100),
  mnc BOOLEAN
);

CREATE TABLE grade(
  id SERIAL PRIMARY KEY,
  score INT,
  student_id INT,
  module_code VARCHAR(10),
  FOREIGN KEY (student_id)
    REFERENCES student (id),
  FOREIGN KEY (module_code)
    REFERENCES module (code)
);


CREATE TABLE registration(
  id SERIAL PRIMARY KEY,
  student_id INT,
  module_code VARCHAR(10),
  FOREIGN KEY (student_id)
    REFERENCES student (id),
  FOREIGN KEY (module_code)
    REFERENCES module (code)
);
```

- You should use the `application.properties` file with the below configuration.

```
spring.application.name=CW2
server.port=2800
spring.datasource.url=jdbc:h2:mem:test;MODE=PostgreSQL;
```

```
spring.datasource.driver-class-name=org.h2.Driver
springdoc.swagger-ui.path=/swagger-ui.html
springdoc.swagger-ui.enabled=true
```

- By adding the below dependency in your pom.xml file, you can access http://localhost:2800/swagger-ui/index.html to see the available API endpoints while running your project with the `mvn spring-boot:run` command

```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
  <version>2.0.2</version>
</dependency>
```

Validation

Code Integrity

Your code will be validated via the below command in a shell:

```
$ mvn compile test checkstyle:check spotbugs:check verify site
```

- Markers will configure your project with the recommended `pom.xml` file specified in the coding standards and use the same command above to mark your coursework.
- If you use a different configuration, it is your responsibility to validate your changes with the model configuration before your submission at least.

Running Frontend

- Please check the COMP0010 coding standards and install Node.js in your machine.
- You can find frontend code here:
  - JavaScript version
  - TypeScript version
  - There is no behavioural difference between two projects, feel free to select one and copy the files under your team's repository
- In the frontend folder, please run `npm ci` command to build the frontend.
  - If `npm ci` doesn't work, please run `npm install`.

- Once you successfully build the frontend, you can run it with the `npm run dev` command.
  - Then, you can see your frontend working on your browser at http://localhost:5173

Running Backend

- Open another terminal window and locate your backend folder.
- You can run a web server in your backend folder with the command `mvn spring-boot:run`
- Then, your frontend should work with your backend.

Cross-origin resource sharing (CORS) Configuration

- You need to configure CORS to allow your frontend to access the backend.
- Please add the below `SecurityConfig` class in your project.
- FYI, the class contains some Checkstyle issues, you need to address them by yourself.

```
package uk.ac.ucl.comp0010.config;

import static org.springframework.security.config.Customizer.withDef

import java.util.Arrays;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.H
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

@Configuration
public class SecurityConfig {

  @Bean
  public SecurityFilterChain filterChain(HttpSecurity http) throws E
    http.csrf((csrf) -> csrf.disable()).cors(withDefaults());

    return http.build();
  }

  @Bean
  public CorsConfigurationSource corsConfigurationSource() {
    CorsConfiguration config = new CorsConfiguration();
    config.setAllowedOriginPatterns(Arrays.asList("*"));
    config.setAllowedHeaders(Arrays.asList("*"));
```

```
        config.setAllowedMethods(Arrays.asList("*"));
        config.setAllowCredentials(false);
        config.applyPermitDefaultValues();

        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigu
        source.registerCorsConfiguration("/**", config);

        return source;
    }
}
```

If you haven't added `Spring Security`, please add the below dependency in your `pom.xml` file

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
```

Reveal Entity ID

- By default, Spring does not reveal the identifier of each object over REST.
- You should add the below configuration class in your project to include the id of objects.

```
package uk.ac.ucl.comp0010.config;


import org.springframework.context.annotation.Configuration;
import org.springframework.data.rest.core.config.RepositoryRestConfi
import org.springframework.data.rest.webmvc.config.RepositoryRestCon
import org.springframework.web.servlet.config.annotation.CorsRegistr
import uk.ac.ucl.comp0010.model.Grade;
import uk.ac.ucl.comp0010.model.Module;
import uk.ac.ucl.comp0010.model.Student;

@Configuration
public class RestConfiguration implements RepositoryRestConfigurer {

    @Override
    public void configureRepositoryRestConfiguration(RepositoryRestCon
            CorsRegistry cors) {
        config.exposeIdsFor(Student.class);
        config.exposeIdsFor(Module.class);
        config.exposeIdsFor(Grade.class);
    }
```

```
}
```

Grade

- Your group project will get a grade from 0 to 100.
- Individual group members will get theirs marks based on their contributions to the project.
- The contributions will be considered in three different factors: general, discussion, and code contributions.
- If all three members contributed enough, all of them can get the 100% of their group grades.

General Contribution Form

- After the project submission, the group should submit a form includes general contribution score of each member.
- The form should be agreed by all three members and only one submission should be made.
- You can measure your group members' contribution as below:

Score	Label	Description
110–120	Essential	Only you could do some hard part of the project: robot design, a coding challenge, or excellent management skills
100–109	Good	Nice working with you. The results were much better because of you.
60–99	Adequate	You did some simple coding. The project was better for your contribution.
40–59	Poor	Could be poor communication, meeting attendance or coding. You added to the project.
0–40	Missing	The project would have been as good (or better) without you. Maybe you contributed a little.

- Based on the response of the general contribution form, the general contribution for each group member can be computed as below:
- General Contribution = $\text{Score}/\ 100 * 0.4$

Discussion Contribution

- The number of your comments on issues and pull requests will be counted as a part of your contribution.
- Discussion Contribution = $\min(\frac{\#\text{ of your comments}}{\#\text{ of total comments}}, 0.4)$

Code Contribution

- The number of your merged pull requests over the total number of merged pull requests
- Code Contribution = $\min(\frac{\#\text{ of your merged pull requests}}{\#\text{ of total merged pull requests}}, 0.4)$

How to compute your grade

- Finally, your grade can be computed as below:

  Grade =
  $\text{round}(\min(1.0, (\text{General} + \text{Discussion} + \text{Code})) * \text{Group Score})$

- For example, if a student's group got 80 for the project and her general contribution is 70 (i.e., adequate). Her discussion and code contributions recorded as 0.35 and 0.3 based on the equation above respectively. Then, the student's final grade for the project will be
  $\text{round}(\min(1.0, ((70/100 * 0.4) + 0.35 + 0.3)) * 80) = 74$

Marking Criteria

Maven Configuration (10 marks)

Most groups will get 10 marks if they initialised a project via the link given in this document and use the recommended `pom.xml` configuration in COMP0010 Coding Standards.

- You should set up correct `Spring Boot` dependencies (4 marks).
- You should set up `Checkstyle`, `Spotbugs`, and `JaCoCo` (6 marks).

Does the code work? (10 marks)

Your code will be configured with a model `pom.xml` file and be validated. With the model configuration, your code should not show any error.

Description	Marks
No Compilation Error	2 marks
No Test Failures	2 marks

Description	Marks
No Spotbugs error with a model configuration	2 marks
No Checkstyle error with a model configuration	4 marks

Test Coverage (10 marks)

Assuming your project is correctly configured, marks will be given based on the test coverage:

Test Coverage	Marks
Test Failure	0 marks
< 59%	4 marks
60%–89%	6 marks
90%–99%	8 marks
100%	10 marks

Use of GitHub (40 marks)

Description	Marks
Only few team interactions were observed	0–10 marks
Almost no team dynamics observed, all commits directly merged into the main branch	11–20 marks
Not enough interactions on GitHub, but the group managed the project anyway	21–30 marks
Group members have interacted actively on GitHub issues and pull requests	31–40 marks

Javadoc (10 marks)

Most students have reasonable javadoc comments so get most of the available marks here.

- There should be no Checkstyle error regarding javadoc.
- The javadoc comments should provide enough information about classes and public methods.

Implementation (20 marks)

Description	Marks
Nothing implemented	0 marks
Your group implemented the basic design only, but doesn't work alright	0–5 marks
Your group implemented the basic design only and the code looks good	6–10 marks
Your group impelmented the basic design and some additional features	11–15 marks
The project contains impressive extension beyond the basic design	16–20 marks

Reference

- Full Stack Development with Spring Boot 3 and React - You can access the book online by using UCL account.
    - Click `SIGN IN` at the top right of the screen.
    - Type your UCL email address in the text field and click the `Continue` button
    - It shows you `I'm with: University College London`, then click the `Sign in with SSO` button
    - It may ask you to login by using your UCL account.
    - Now, you can read the book

Acknowledgment

This project is adapted from the original code by DongGyun Han (COMP0010 2024). We acknowledge his contribution and thank him for sharing the codebase.