Web Server Project: Instructions

Step 1: Install dependencies

You will need Gradle on your computer to run the server project. Gradle is a build tool, which means it will help you download libraries and compile your Java code.

Step 2: Run the starter code

Check out the repo and go to the project directory in your terminal.

Next, run this command to download libraries and compile your code:

```
gradle build
```

This might take a while the first time.

Next, run this command to start up the server:

```
gradle run
```

You will notice a progress bar pause at 75%: "Run". That means the server is now running! The server is different from other homework we've done in the past, because it is a *long-running process*, meaning it needs to keep running in order to do its work rather than running quickly and finishing.

Remember that you can stop the server by pressing **CTRL+C**. You will need to restart your server each time you want to test out a code change, just like how you need to run `javac` and `java` on your terminal for smaller programs.

Now, open your web browser and navigate to http://localhost:3000. If you see the grading system front end, this means all of the starter code is working and you can start on your part of the project!

Step 3: Finish implementing the server

Remember that the project has two parts: the *server* and the *client*. The client code is already finished, but you need to finish up the server code. You will be able to perform some actions

without changing anything, but you will need to implement two final API endpoints to make the whole system work. These are the endpoints in the system:

- `GET /api/students` (finished)
- `GET /api/students/{id}` (finished)
- `POST /api/students` (finished)
- `PUT /api/students/{id}` (finished)
- `PUT /api/students/{id}/grade` (TODO!)
- `DELETE /api/students/{id}` (TODO!)

The **API Specification Document** has the full details on what these endpoints should look like. Your endpoints need to return JSON data like the example responses in the document.

In order to implement the endpoints, you will need to change code in the `Main`, `School`, and `Student` classes.

Step 3, Part 1: Implement the endpoint to change a student's grade

You'll notice that the logic for this endpoint has already been started in the `Main` class.

Implement a `setGrade` method in the `School` class. The method should take two parameters: a student's ID and a new grade. It should look through the students until it finds the right one, then change the grade to the new value. You will also need to create a setter method in the `Student` class.

Step 3, Part 2: Implement the endpoint to delete a student

Implement a `deleteStudent` method in the `School` class. The method should take one parameter: the ID of the student to delete. It should find the student that has that ID and remove it out of the list of students.

There is not a handler method for this endpoint, so you will need to create one yourself. You will need to follow the same pattern that the other endpoints follow. This part of the Spark documentation can help you know which method to call to create the endpoint.

Submitting the project

Put your work on Githbu and submit a link to your repo on Canvas, like previous projects.