Full Name: Abdul Muhaymin Abdul Hafiz

Portico ID: 23091179

Email: zcabam3@ucl.ac.uk

Group: 15

Repository URL: https://github.com/ucl-comp0010-2024/G-15

Section 1: Technical Description

The coursework task was to develop a web-based “Student Grade Management Program”. The frontend of the application was already developed using React, and the backend of the application was to be developed using Spring Boot – a Java framework. We were given a UML diagram containing the basic design of the model in the backend, as well as a working implementation of the React frontend in TypeScript to begin with. Our group’s primary assignment was to complete the backend from scratch, as well as creating any additional extensions to the frontend.

The Spring Boot backend we created consists of 6 main packages – config, controller, exception, model, repository, and service. We ensured any necessary tests were made for each part of the backend. When a user makes a HTTP request from the frontend, the request is handled by one of the controllers in the backend. Spring Boot can also automatically handle some of the HTTP requests. In the controllers, it calls classes within the service package, which contain the main business logic of the application. These service classes then interact with interfaces within the repository package which communicate with the database to perform CRUD operations. The model classes we made – with the main ones being the Student, Module, Registration and Grade classes – define the structure of the data used in the application and is mapped (via an ORM) to the database schema. For the purposes of the coursework the database used was an in-memory H2 database.

The React frontend we developed upon consists of 4 main pages – the home page, modules page, students page, and grades page. The home page gives a brief overview of the application. The modules page allows you to view all existing modules (GET request), add new modules (POST request), or delete modules (DELETE request). The students page allows you to view all the students (GET request), add new students (POST request), or delete students (DELETE request). The grades page allows you to view all of the grades (GET request), add new grades (POST request), or delete grades (DELETE request). We also added many extensions on top of this basic design. This includes a navigation bar, a better designed and dynamic UI with hover animations, a dark/light mode toggle, export to PDF or CSV feature, and a sorting feature that lets you sort the modules, students, or grades by a particular field.

In terms of the backend, I was responsible for setting up the overall project, and creating the Module class, Grade class, and Registration classes, as well as the addGrade, getGrade, registerModule methods in the Student class. I then went on to also create the RegistrationRepository, GradeRepository, and Module Repository interfaces, as well as the GradeService and GradeController classes. Throughout the entire development process, I followed the test-driven development (TDD) paradigm and created tests for each of the pieces of code I was making where necessary. I also ensured that the code written by myself, or even other people, did not contain any checkstyle violations.

With regards to the frontend, I was responsible for improving the layout of the pages and ensuring consistency between them. I also added things such as padding, icons, and iterated upon the navbar design. Furthermore, I was also responsible for adding the sorting feature to the students, modules, and grades pages, which is a functionality that operates entirely within the frontend without requiring any interactions with the backend.

Section 2: Technical Challenges

One technical challenge we faced was during the implementation of the delete feature in our application. We noticed that students were not being removed from the students list despite the delete button being pressed, and instead an error was being shown. I took on the responsibility to identify the cause of this issue and ensure that the delete functionality worked as intended without errors. I began by analysing the StudentController class and trying to better my understanding of how Spring Boot handled user requests. I recalled learning before during the development of the GradeController class that Spring Boot (specifically Spring Data REST) can expose some repository operations as REST endpoints without needing you to write explicit controller methods. After playing around with the StudentController, I also learnt that Spring Boot won't automatically use these REST endpoints for any HTTP request if you already explicitly implemented a controller method for another HTTP request (e.g. GET, PUT etc.) at that endpoint already. This meant that DELETE requests were no longer being handled automatically as I would have expected since I was handling PUT requests explicitly. So, to fix this issue, I simply removed the other controller methods that were handling other HTTP requests at the same endpoint. As a result, DELETE requests were now being handled automatically, and the application worked as intended when the delete button was pressed on students.

Another technical challenge we faced during the implementation of the backend of our application was ensuring 100% test coverage. During the development of our program, I noticed that sometimes some of my team members would not get complete line or branch coverage using the tests that they wrote for the specific classes that they made. So, I took on the task of creating new issues for each class that had a lack of complete test coverage, so that our group could eventually resolve it and get 100% test coverage. I used the jacoco test report that is automatically generated to identify which classes were deficient in line or branch coverage, and where in that class specifically. Using this information, I created new issues that address these classes, and I then built upon the existing tests that we had to ensure complete test coverage. As a result, the tests our group wrote achieved 100% test coverage, suggesting our code was somewhat correct.

Section 3: Teamwork

Overall, our groups teamwork was pretty good. In our group, we were all already friends, therefore being comfortable enough to communicate our thoughts or describe any issues we were facing was not a problem. Throughout the duration of the coursework, we maintained contact through a WhatsApp group chat where we discussed what tasks we were doing or any bugs we encountered. Furthermore, we ensured that whenever we were developing important features in our program, or even fixing bugs, we created a GitHub issue. Each issue contained the required actions that we needed to do, and the expected deliverables. For each of these issues, we typically created a new branch where we would make commits. After we were done, we would then make a new pull request and merge the feature branch it into the main branch before closing the original issue. On most of these issues and pull requests, we commented on it and shared our opinions about the issue or pull request.

One of the main challenges we experienced whilst working in a team was some members working on an issue which was already assigned to someone else. Sometimes, even though an issue was already assigned to someone, another team member would take over and start working on the issue themselves. This may have been because they needed that issue to be completed so that they can complete their assigned issue, or because they felt that the person assigned was taking too slow to work on it for example. As a result, there were some

instances where multiple people were working on an issue at the same time, and people would attempt to make a pull request to deal with an issue that was already dealt with, sometimes resulting in merge conflicts. It was clear that we needed to have better communication about who was working on which issue at the current moment, and whether someone was planning on taking over. Some members took action and made sure to communicate with the other team members about working on an issue which was already assigned to someone else, asking for their permission to work on it. As a result, there wasn't as many problems faced where multiple people were working on the same issue.

Section 4: Retrospective

In retrospect, I believe our group worked quite well together. We were able to create a working backend which had all the expected features in the coursework requirements, and we managed to get complete test coverage. We were able to also create considerable extensions to the frontend. Furthermore, our group had good communication throughout the coursework, and we were comfortable working together since we were already friends. All the group members contributed a decent amount to the project and made sure to follow the coding standards. We also made good use of GitHub by consistently creating issues and pull requests, and having it checked out by other members. Overall, I can't say that anyone did not do enough to help meet the coursework requirements. If I had to say what went wrong in our team it's that sometimes our communication was not the best. Sometimes people worked on an issue that was already assigned to someone else, or two people would start developing the same feature without telling anyone else. This sometimes led to merge conflicts and some frustration amongst the team members.

If I were to work on a similar project in the future, I would ensure that our group members follow good programming practices on GitHub (creating issues and pull requests, not making direct commits to the main branch, reviewing and commenting on other people's code, ensuring commits are not too big, etc.), and maintain active communication on a messaging app like WhatsApp for example. Furthermore, I believe it would be good if our group was strict about the task assignments, not letting other people work on an issue or feature when someone else was already working on it. Another thing is that we should avoid merging code into the main branch that was not properly tested or contains bugs. This is because generally the main branch should be free from any bugs and work as intended. To help with this in the future I think we should be more careful when opening new pull requests, and other members should pay closer attention when performing a code review. Moreover, I would also try to ensure that our group members start working on the project as soon as possible and try to complete the basic requirements as quickly as we can without delaying it. This is so that we have more time at the end to develop extensions for the project and to discover any potential bugs. It would also be good if to work consistently hard throughout to make the project as good as possible.