CS 102 *Spring 2020/21*

Project Group **G2C**

Instructor: **Aynur Dayanık**
Assistant: **Haya Shamim Khan Khattak**

Criteria	TA/Grader	Instructor
Presentation		
Overall		

~ LabConnect ~

Borga Haktan Bilen 22002733
Vedat Eren Arıcan 22002643
Berkan Şahin 22003211
Berk Çakar 22003021
Alp Ertan 22003912

Detailed Design Report

(version 2.0)

May 4, 2021

1 Introduction

LabConnect facilitates communication between students, TA's, tutors, and instructors. In the background, it is a web application that aims to assist CS introductory courses in organization and communication. Proposed ideas for features include priority queuing for TA zoom rooms among many other enhancements to TA/instructor productivity. For example, those who have completed their labs can be tested using pre-defined (by TA or instructor) unit tests, and then placed into a queue to optimize the TA-student meeting arrangement process in general. Much of the repetitive work that course staff need to do can be reduced substantially by automated actions, allowing TA's and tutors to allocate more time for more hands-on help towards students. In summary, LabConnect is a developing project that aims to make education more productive for students, and more efficient for teaching staff, above all.

2 System Overview

2.1 Organisation & Architecture

Shown below is the diagram of the organization of LabConnect's architecture. Users of varying roles interact with the interface displayed using the ReactJS library, which also makes HTTP requests to the REST API powered by the Spring framework, over the internet. The Spring framework acts mostly as the controller segment of the project, delivering data that is obtained through model classes and their communication with the databases.

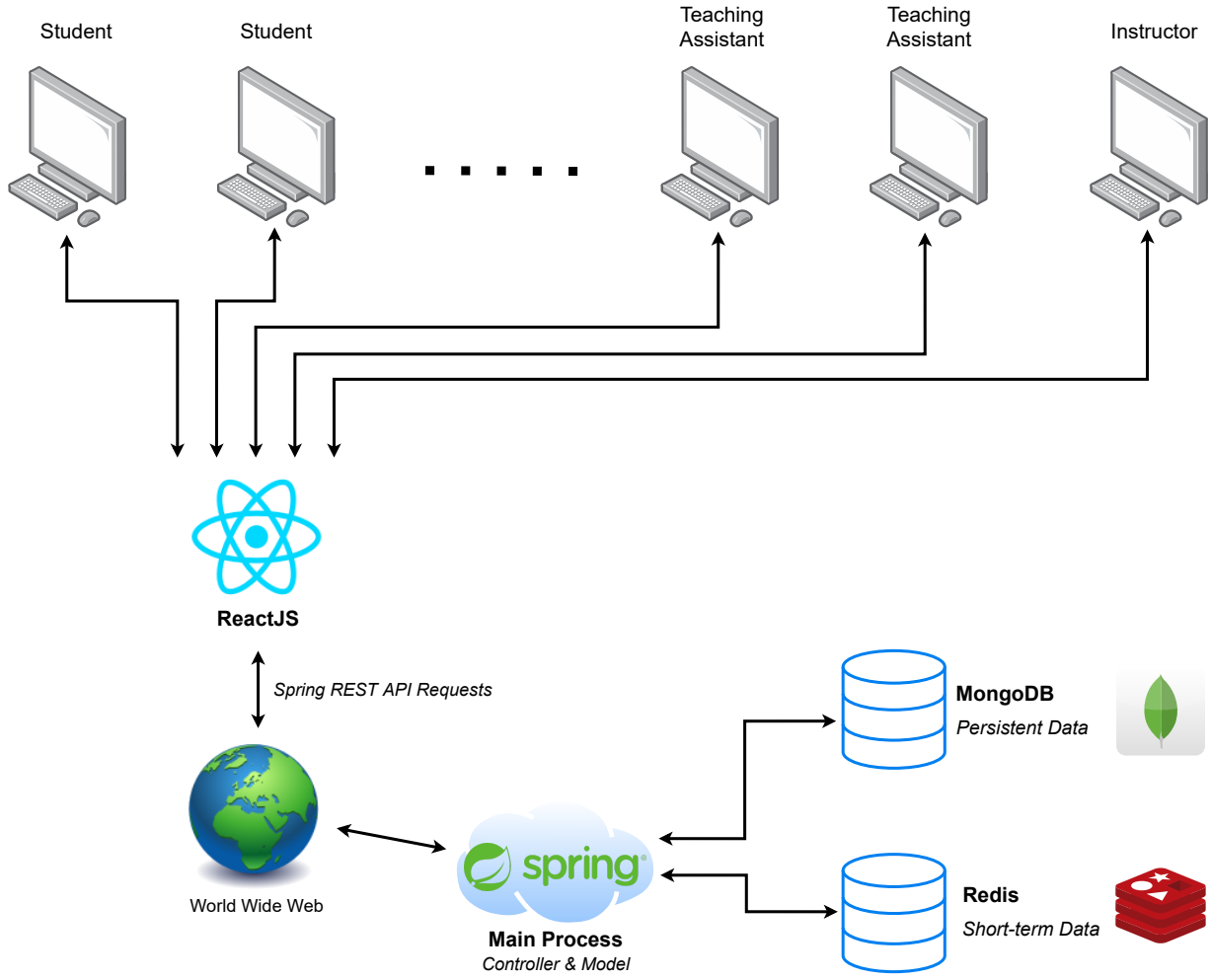

Figure 1: Overview of LabConnect's Organisation

2.2 Technologies

2.2.1 Back-end

- **Spring** - Framework to be used to power the REST API at the /api/ data endpoint. All necessary data will be exposed at the API endpoint, but only with proper authentication. Requests are only authorized accordingly with the user's account permission level. The *Spring Security* and *Spring MVC* frameworks may also be taken advantage of.
- **MongoDB** - To be used as persistent storage; account data, assignment data, etc.
- **Redis** - To be used as short-term storage; user session, authentication, etc.

2.2.2 Front-end

- **SASS** - Useful preprocessor to write CSS more productively.
- **ReactJS** - Will be used to construct a single-page-app user interface, which will serve components according to the API call responses.

2.2.3 Build & Utility Tools

- **Spring Boot** - May be used to simplify the development of Spring components.
- **Maven** - Build automation tool, good for any medium to large scale project.
- **Docker** - Facilitates the deployment of the project, and it may also be viable to use *docker-compose* to deploy separate containers for databases and other components simultaneously.

2.2.4 Domain & Host

- **Domain** - *labconnect.me* is the proposed domain for the website.
- **Hosting** - The project will most likely be run on either a container deployment service, or a VPS service.

3 Core Design Details

Most of the data, being of persistent nature, is stored in a database. But the model classes perform the necessary queries and subsequent actions to the data as necessary, essentially grouping database queries logically. Additionally, model classes are also responsible for serving functionallity (service layer) to the controller layer which then controller layer handles HTTP requests (GET, POST, DELETE, PUT) in order to create a REST API. In the first class diagram [2] below, all model Java classes can be seen with their methods and fields with inheritance relations (excluding any other hierarchical relation). Below the class diagram, hierarchical grouping of model classes can be seen (figures from 3 to 10). At the very end, compresed UML diagram can be seen [11] with all the Java classes, interfaces, enums and their relations. However, because of the magnitude of the project the diagrams below might not be clear enough to examine. Thus, we uploaded the UML diagrams (compresed and uncompresed versions) (with magnification script) and Javadocs to the LabConnect website under a new distinct subdomain for convience. Links for these are respectively: http://docs.labconnect.me/uml/, http://docs.labconnect.me/umlwc/ and http://docs.labconnect.me/.

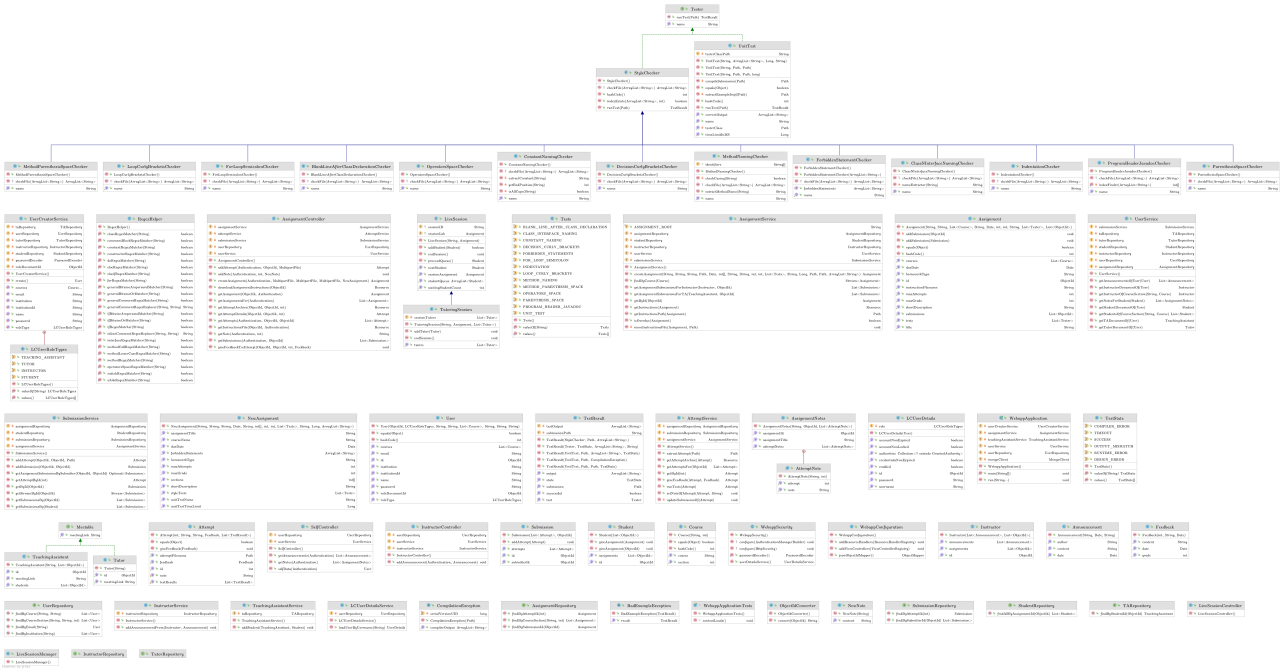

Figure 2: UML Class Diagram without Relations for LabConnect

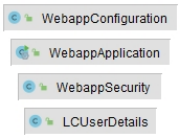

Figure 3: Configurator Classes

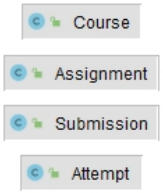

Figure 4: Course Classes

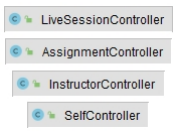

Figure 5: Controller Layer Classes

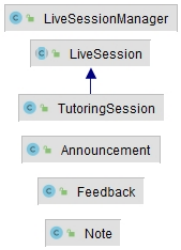

Figure 6: Assignment Function Classes

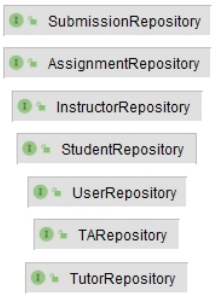

Figure 7: Repository Classes

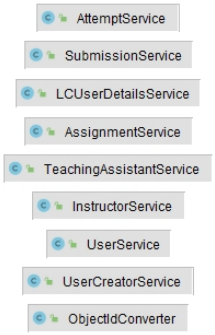

Figure 8: Service Layer Classes

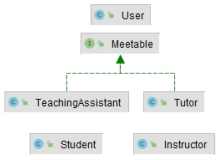

Figure 9: User Classes

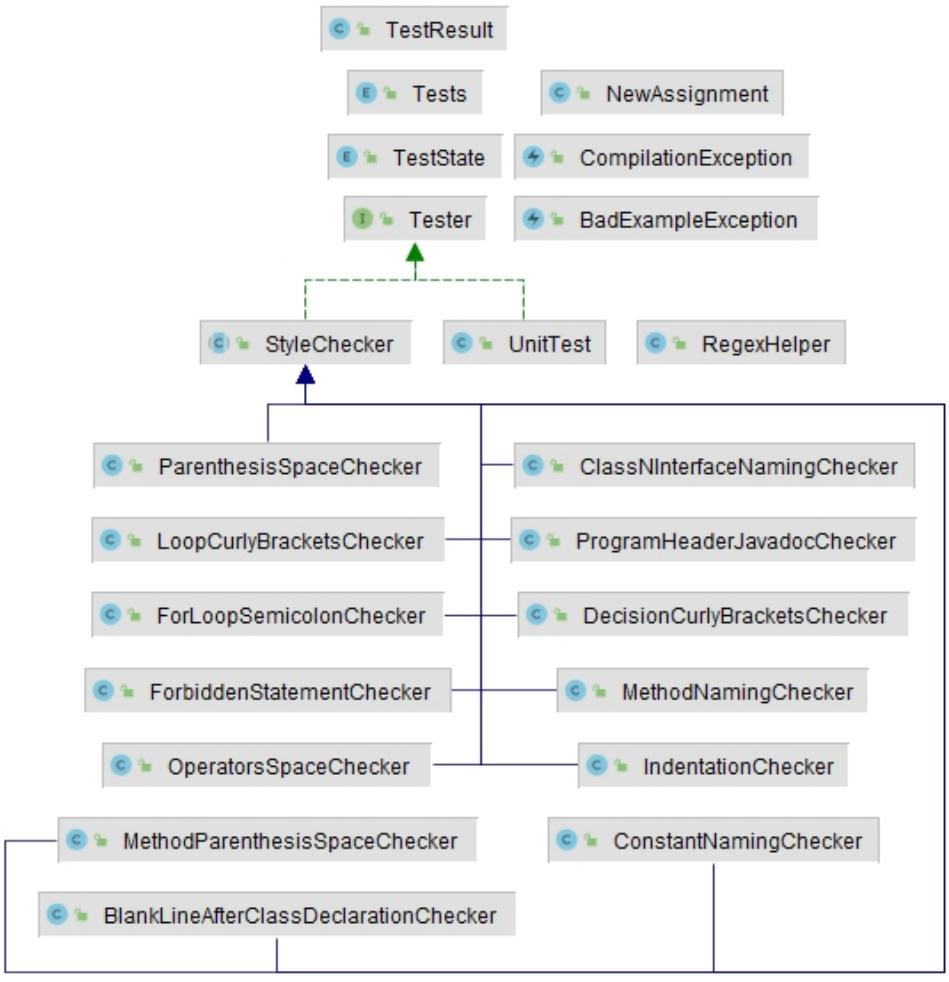

Figure 10: Testing Classes

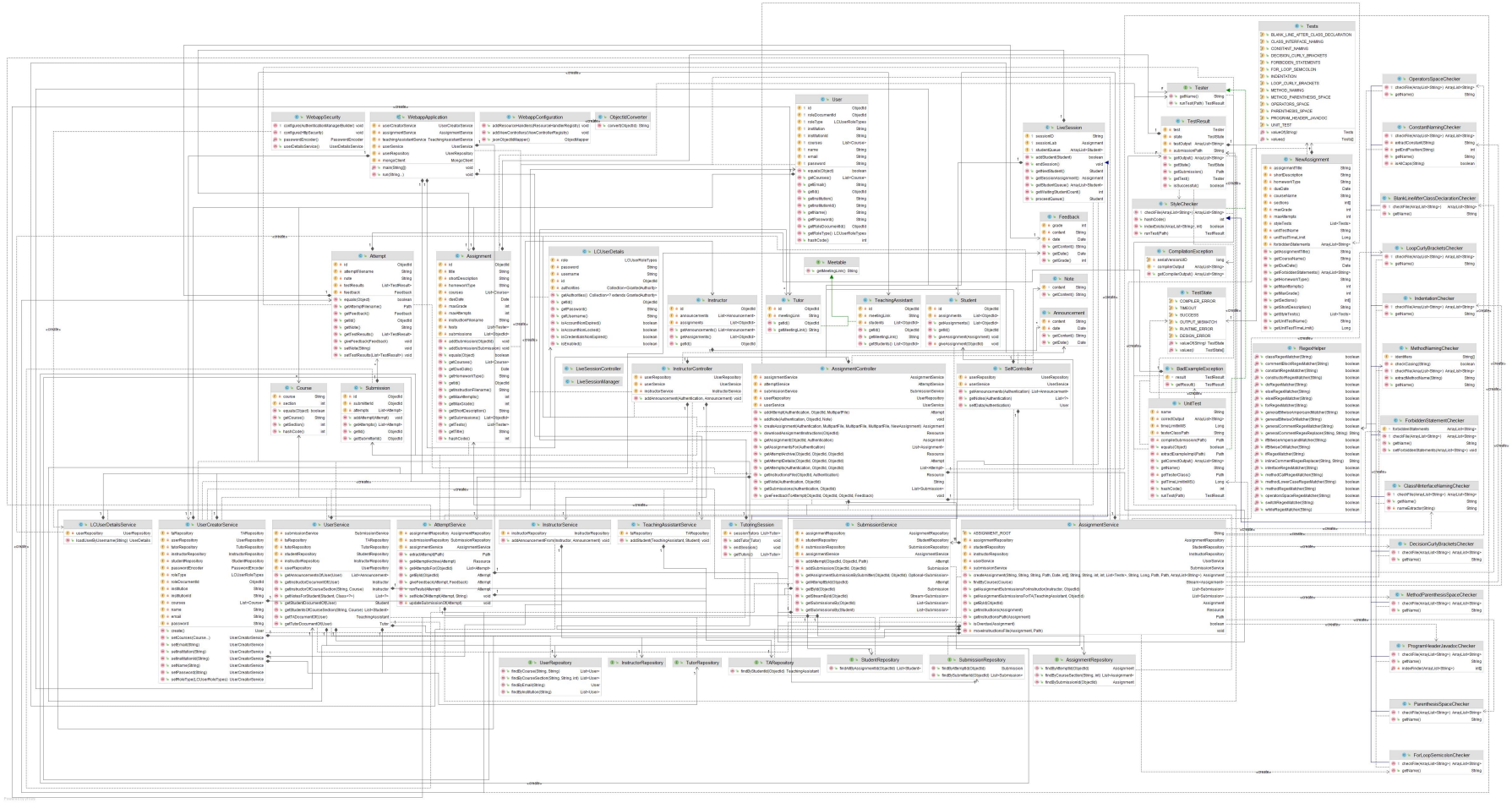

Figure 11: UML Class Diagram with Relations for LabConnect

4 Task Assignment

The division of work for the model classes is as follows;

Borga Haktan Bilen	Vedat Eren Arıcan	Berkan Şahin	Berk Çakar	Alp Ertan
AssignmentController	NewAssignment	NewAssignment	NewAssignment	LoopCurlyBracketsChecker
SelfController	Note	AssignmentController	Note	ClassNInterfaceNamingChecker
AttemptService	AssignmentController	InstructorController	AssignmentController	RegexHelper
Assignment	InstructorController	SelfController	SelfController	CompilationException
Feedback	SelfController	AssignmentService	Announcement	BadExampleException
Meetable	AssignmentService	AttemptService	Course	AssignmentNotes
LineAfterClassChecker	AttemptService	SubmissionService	Instructor	AssignmentController
ClassNInterfaceNamingChecker	Announcement	Assignment	OperatorsSpaceChecker	SelfController
ForbiddenStatementChecker	Attempt	Attempt	ParenthesisSpaceChecker	Tester
MethodNamingChecker	Course	Submission	ForbiddenStatementChecker	DecisionCurlyBracketsChecker
OperatorsSpaceChecker	Feedback	LiveSession	ConstantNamingChecker	ProgramHeaderJavadocChecker
ProgramHeaderJavadocChecker	ForbiddenStatementChecker	LiveSessionManager	ForLoopSemicolonChecker	ForLoopSemicolonChecker
RegexHelper	MethodNamingChecker	TutoringSession	MethodNamingChecker	StyleChecker
Tester	RegexHelper	RegexHelper	IndentationChecker	MethodNamingChecker
UserService	TestResult	StyleChecker	DecisionCurlyBracketsChecker	TestResult
Student	UserService	BadExampleException	MethodParenthesisSpaceChecker	WebappConfiguration
TeachingAssistant	Tutor	TestResult	LineAfterClassChecker	InstructorController
Tutor	User	TestState	RegexHelper	
User	InstructorRepository	UnitTest	StyleChecker	
InstructorRepository	StudentRepository	Instructor	UserService	
StudentRepository	SubmissionRepository	Student	LCUserDetails	
SubmissionRepository	TARepository	TeachingAssistant		
TARepository	TutorRepository	User		
TutorRepository	UserRepository	AssignmentRepository		
UserRepository	ObjectIdConverter	WebappSecurity		
LCUserDetails	WebappApplication	WebappConfiguration		
	InstructorService	WebappApplication		
	React.js Frontend Dev.	LCUserDetailsService		
		TeachingAssistantService		
		UserCreatorService		

The division above is only for the implemented Java classes. As for the remaining work, we have decided to not limit anyone to work on a particular technology involved in the project. This project is, above all, intended for us to learn new technologies and gain experience for both teamwork and medium/large scale development. In which case, it works against this goal to have clean-cut distinctions in task assignment. Theoretically, having all group members to strive to experience a variety of technologies should also ensure an even partition of work. Lastly, note that the project proposed thus far is of scale large enough to accommodate members working on a specific component without clashing.