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.

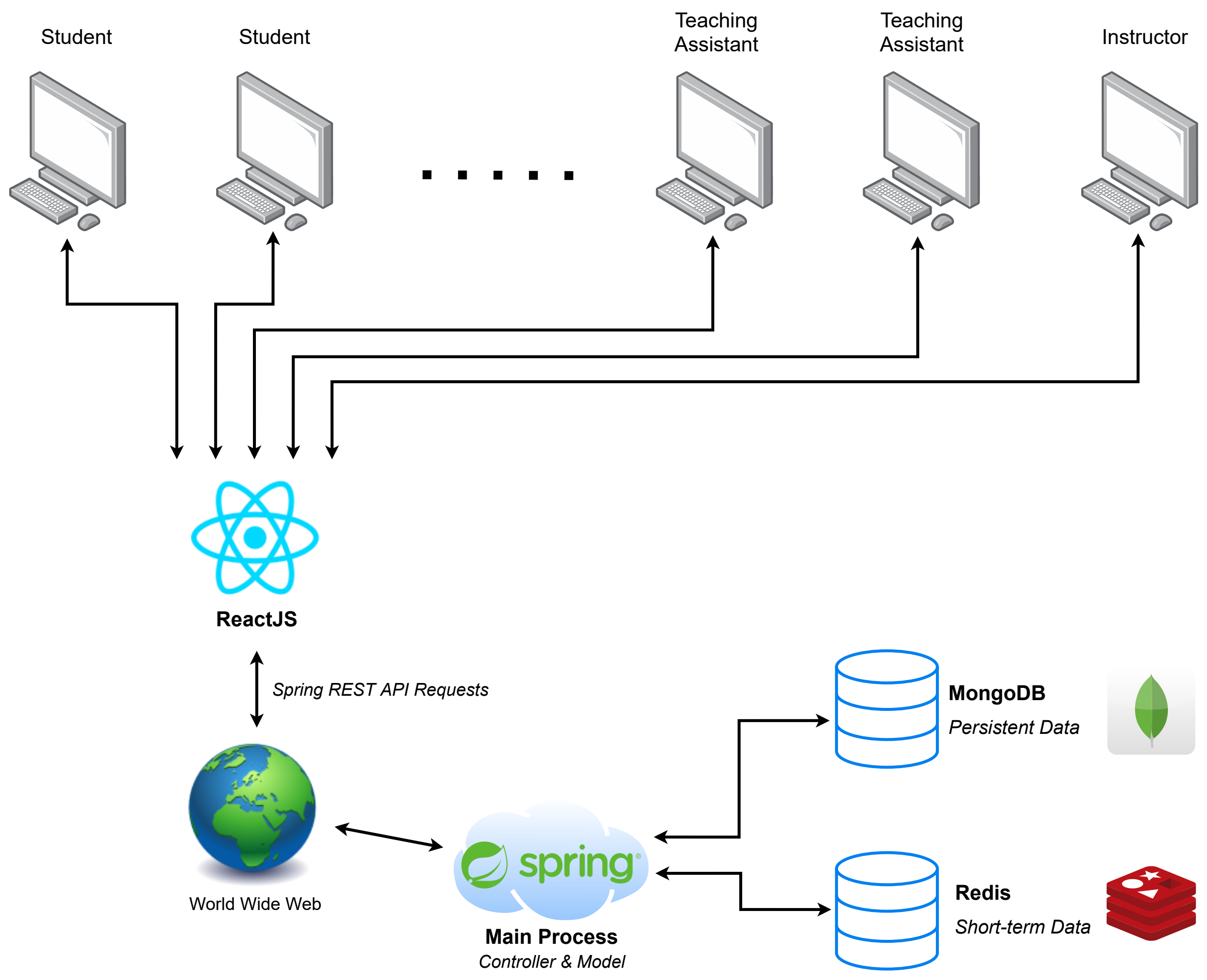


Figure 1: Overview of LabConnect's Organisation

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

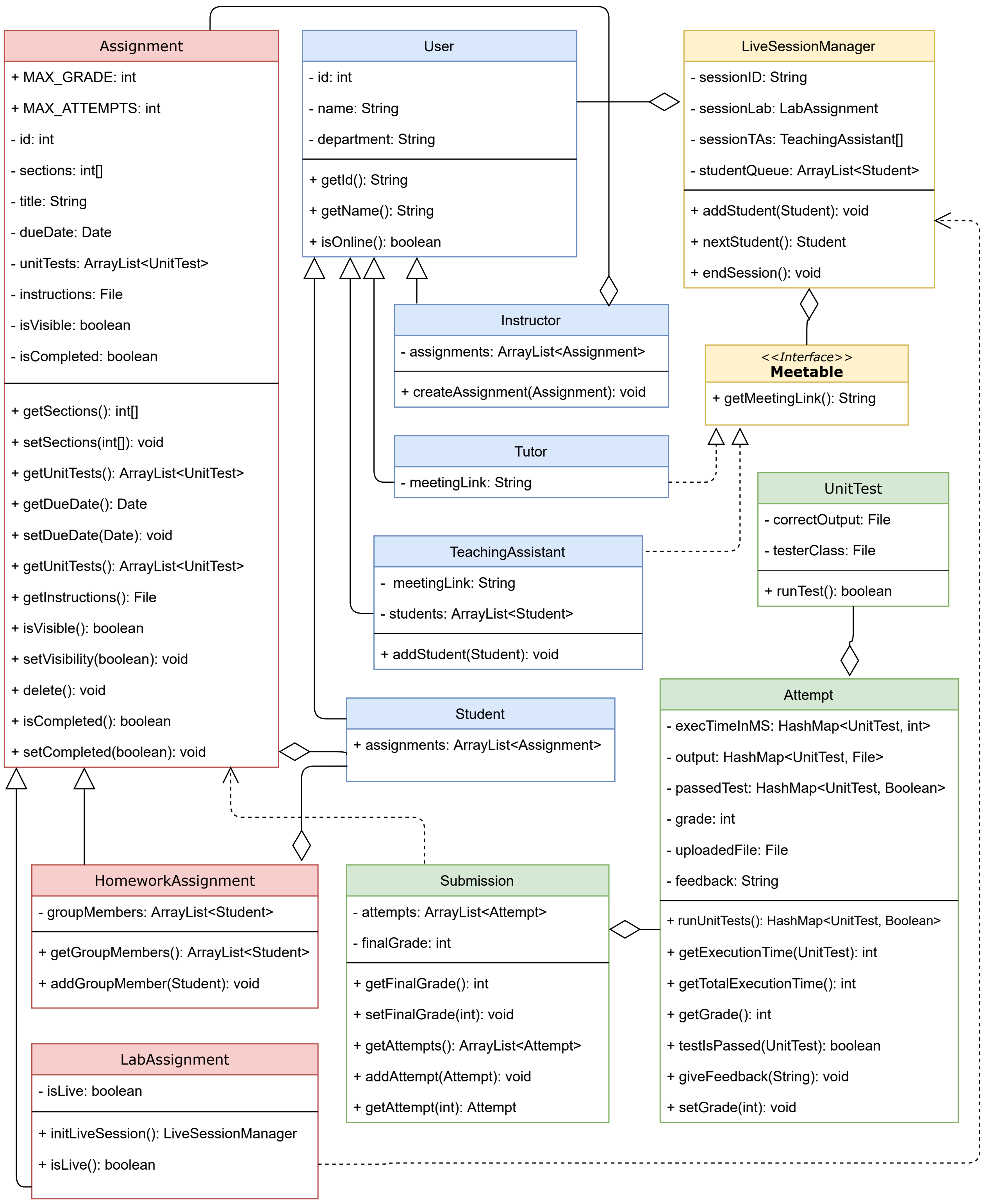
- **Spring Boot** - May be used to simplify the development of Spring components.
- **Maven** - Build automation tool, good for any medium to large scale project.
- **Docker** - Facilitates the deployment of the project, and it may also be viable to use *docker-compose* to deploy separate containers 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. In the class diagram below, the classes highlighted in red detail the assignment, blue detail the user account types, green detail submission-related data specific to an assignment, and yellow provide features for managing the live sessions.



Assignment
+ MAX_GRADE: int
+ MAX_ATTEMPTS: int
- id: int
- sections: int[]
- title: String
- dueDate: Date
- unitTests: ArrayList<UnitTest>
- instructions: File
- isVisible: boolean
- isCompleted: boolean
+ getSections(): int[]
+ setSections(int[]): void
+ getUnitTests(): ArrayList<UnitTest>
+ getDueDate(): Date
+ setDueDate(Date): void
+ getUnitTests(): ArrayList<UnitTest>
+ getInstructions(): File
+ isVisible(): boolean
+ setVisibility(boolean): void
+ delete(): void
+ isCompleted(): boolean
+ setCompleted(boolean): void
User
- id: int
- name: String
- department: String
+ getId(): String
+ getName(): String
+ isOnline(): boolean
LiveSessionManager
- sessionID: String
- sessionLab: LabAssignment
- sessionTAs: TeachingAssistant[]
- studentQueue: ArrayList<Student>
+ addStudent(Student): void
+ nextStudent(): Student
+ endSession(): void
Instructor
- assignments: ArrayList<Assignment>
+ createAssignment(Assignment): void
<<Interface>>
Meetable
+ getMeetingLink(): String
Tutor
- meetingLink: String
UnitTest
- correctOutput: File
- testerClass: File
+ runTest(): boolean
TeachingAssistant
- meetingLink: String
- students: ArrayList<Student>
+ addStudent(Student): void
Attempt
- execTimeInMS: HashMap<UnitTest, int>
- output: HashMap<UnitTest, File>
- passedTest: HashMap<UnitTest, Boolean>
- grade: int
- uploadedFile: File
- feedback: String
+ runUnitTests(): HashMap<UnitTest, Boolean>
+ getExecutionTime(UnitTest): int
+ getTotalExecutionTime(): int
+ getGrade(): int
+ testIsPassed(UnitTest): boolean
+ giveFeedback(String): void
+ setGrade(int): void
Student
+ assignments: ArrayList<Assignment>
HomeworkAssignment
- groupMembers: ArrayList<Student>
+ getGroupMembers(): ArrayList<Student>
+ addGroupMember(Student): void
Submission
- attempts: ArrayList<Attempt>
- finalGrade: int
+ getFinalGrade(): int
+ setFinalGrade(int): void
+ getAttempts(): ArrayList<Attempt>
+ addAttempt(Attempt): void
+ getAttempt(int): Attempt
LabAssignment
- isLive: boolean
+ initLiveSession(): LiveSessionManager
+ isLive(): boolean

4 Task Assignment

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

Borga Haktan Bilen	**Vedat Eren Arıcan**	**Berkan Şahin**	
AssignmentController	NewAssignment	NewAssignment	
SelfController	Note	AssignmentController	
AttemptService	AssignmentController	InstructorController	
Assignment	InstructorController	SelfController	
Feedback	SelfController	AssignmentService	
Meetable	AssignmentService	AttemptService	Bla
BlankLineAfterClassDeclerationChecker	AttemptService	SubmissionService	
ClassNInterfaceNamingChecker	Announcement	Assignment	
ForbiddenStatementChecker	Attempt	Attempt	
MethodNamingChecker	Course	Submission	
OperatorsSpaceChecker	Feedback	LiveSession	
ProgramHeaderJavadocChecker	ForbiddenStatementChecker	LiveSessionManager	
RegexHelper	MethodNamingChecker	TutoringSession	
Tester	RegexHelper	RegexHelper	
UserService	TestResult	StyleChecker	
Student	UserService	BadExampleException	
TeachingAssistant	Tutor	TestResult	
Tutor	User	TestState	
User	InstructorRepository	UnitTest	
InstructorRepository	StudentRepository	Instructor	
StudentRepository	SubmissionRepository	Student	
SubmissionRepository	TARepository	TeachingAssistant	
TARepository	TutorRepository	User	
TutorRepository	UserRepository	AssignmentRepository	
UserRepository	ObjectIdConverter	WebappSecurity	
	ObjectIdConverter	WebappConfiguration	
	WebappApplication	WebappApplication	
	InstructorService	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.