Software Requirement Specifications

CodeClassy

Version: 1.0.0

Project Code	*F210701*
Supervisor	*Dr. Muhammad Rafi*
Co Supervisor	*NA*
Project Team	*Saif Ul Islam - 18K 0307* *Muhammad Hassan Zahid - 18K 0208* *Tashik Moin Sheikh - 18K 014*
Submission Date	*7th December, 2021*

Document History

Version	Name of Person	Date	Description of change
0.0.1	Saif Ul Islam	6th Dec, ‘21	Document Initialized
1.0.0	Saif Ul Islam	11th Dec, ‘21	Document finalized

Distribution List

Name	Role
Dr. Muhammad Rafi	Supervisor
NA	Co- Supervisor

Document Sign-Off

Version	Sign-off Authority	Sign-off Date
	Muhammad Rafi	Dec 07, 2021

Table of Contents

1. Introduction

1.1. Purpose of Document

The SRS documentation provides an insight into the development of the project, the requirements that the project is built upon, the interactions between its different modules and concerns of stakeholders. This SRS serves to provide a timestamp in the timeline of this project that can clarify to interested parties the aspects on which the project is built on.

1.2. Intended Audience

The intended audience of this document are faculty members, such as the FYP coordinators, the jury, and the team's supervisor, to give them a broader perspective over the project's facets and conventions.

1.3. Abbreviations

This section will contain abbreviations used throughout the document, if applicable.

1.4. Document Convention

This document is written in Arial, with font-size 10.

2. Overall System Description

2.1. Project Background

The project ideas and use cases stem from an observation in the current set of educational oversights presented in many of the approaches currently or previously utilized by educational Teachers due to a lack of efficient tools that would better improve their aspects of (a) Flexibility, (b) Interactivity & Communication, (c) Easiness & "A Ductile Feedback Process". It is the opinion of the team that these aspects can be, and should be, targeted and amalgamated in a web application based product, able to be universally used by individual educational institutes, from small teams to larger faculties.

The team has primarily kept in its interests some particular features it believes are crucial in the development of an MVP,

1. *Learning groups*
2. *Course outline management*
3. *Quiz Analysis/Feedback*
4. *Assimilated results/accomplishments - a dashboard or task view capable of displaying progress interactively*
5. *A programming platform, primarily, can be individual or pair programmed tasks*
6. *Group activities, possibly over the idea of 'assignments' and 'projects'*
7. *Effortless assignment assigning process with an integrated coding environment*

2.2. Project Scope

The product scope aims to be much like an MVP - taking on requirements that can be covered within a 10 month period, and no more. Thus the product is currently defined in terms of feasibility and time limit constraints over how the product is to be evolved over the coming months. As such, the product has a certain idea that it wants to cover and go over. These include steps such as,

1. *Covering the requirements, use cases, and user stories initially defined in theProject Proposal*
2. *Developing a complete front-end and a back-end to accompany the project*
3. *Project deliverable*

For FYP I, these deliverables include,

1. *Requirement Analysis*
2. *Design Phase*
3. *Data Modelling*
4. *Architecture & Infrastructure Setup*
5. *Implementation Phase (WIP) & Boiler-plate setup*
6. *Quiz Management System Requirement*

For FYP II, these deliverables include,

1. *Finalizing "Remote Code Execution"*
2. *Finishing FYP touches*
3. *Generating FYP report*

2.3. ***Not In Scope***

At the time of writing this document, and due to a constrained time and technical frame, CodeClassy does not aim to divert too much from the initial requirements, as we believe it is enough for the duration of the project. This does not mean the project will stop itself from working on further milestones goals once the MVP goals have been accomplished, it simply refers to ideas that the team are not taking into consideration during development, but might possibly gather requirements and research over if the target goals are achieved before deadlines, with opportunity to pursue more features if the team can plan and work out a proper schedule for the process involved.

Our project intends to deliver on the deliverables promised. As such, our time is rigid and inflexible because we realize the complexity from both a development and a design perspective, and maintaining a large code can be quite difficult for a group of 3 students who are studying at the same time. Because of limited time and resources in terms of workforce, we intend our best to not divert away from the promised deliverables, at the same time, to be careful to not include any unnecessary features or requirements, either small or large.

2.4. ***Project Objectives***

The purpose of this project focuses heavily on integration, development, and a smooth user experience that comes together under one platform. The aim is to avoid many of the pitfalls that are often seen with a cluster of technologies being used together to facilitate communication, tracking, progress, and analysis of the classroom as a single communication channel.

***CodeClassy** is thus a measure towards, "presenting a flexibly adequate hallmark towards many innumerable inconsistencies and haphazard workflows in the EdTech domain". The Purpose overall encompasses functionalities dealing with organization, management, progress tracking, and feedback process between teachers and student in a virtual learning environment*

2.5. ***Stakeholders***

The target audience for this project includes students and faculty staff as educational institutes to help in better facilitating communication between Teachers and their students. The students will be using this application to visualize and comprehend their academic progress, perform and progress with their assignments, and compare themselves with other students.

For both parties involved, the teachers and students, an important point is to facilitate the idea of maintaining an organized set of workload by removing away the burden of having to keep up with deadlines and notifications from teachers for the students, and for the teachers to provide an ease in conducting online sessions much more efficiently and effectively by helping them scale easily the number of students and utilize the best that these platforms offer to the advantage of online learning.

This by no means is a project intended to deprecate traditional classrooms, but one that intends to build on the flexibility, space, and freedom of online systems and play with them to the best of the platform's advantage. As a result, while there is mimicking of mechanisms on the platform that represent traditional methods of learning, it does not necessarily have to be the case. An underlying motivation is to use and build on the strengths of online learning to make the platform unique and popular in the niche of EdTech.

2.6. Operating Environment

Both the users in the context of the use cases and requirements will be working in environments where they have access to internet, ranging from low to high bandwidth, and in standing or sitting positions in quiet or empty places, and where there is not movement.Entirely, the assumptions about the user operating in their environment can be considered as the same as the for applications such as Google Classroom or Wikipedia.

2.7. System Constraints

A few specific constraints, such as the project suffering from not being able to accomplish its full potential. Thiscan mean not having enough time to factor in future features and expansion related ideas. While it is in hopes that the primary user concerns will be completed, the "completeness" aspect of how "well prepared"and "tested" a solution is can vary.These lacking can be described in two ways,

1. *Project Management*
2. *Technical Aspects*

*For "**Software Constraints**",*

1. *Language: Our primary use is of JavaScript and TypeScript. Thus, our Software Constraints include all constraints in these languages as well*
2. *Framework/Libraries: we are in the process of using server-side rendering for our main interface, with a module-based architecture for our backend. This defines our workflow as it forces us to compartmentalize logic, services, connections, and such*
3. *Databases: We are using MySQL, an RDBMS. Our database constraints envelope all the constraints that come with MySQL*
4. *Timescales: Our primary limitations are "Schedule", "Resources", "Quality", which affect our "Timescales". We are severely limited by how much can and is done or should be done as a team of 3, we are multidimensional in terms of families, work hours, pace, social circles, education, assessments and deadlines*
    a. *Schedule - we would like to deliver on time what we have promised to deliver for our deliverables*
    b. *Resources - in man power, we are 3 students trying to handle a large code base and feature requirements/engineering, along with discussions and external influences on our time*
    c. *Quality - we would like to deliver the quality of software to the best of our abilities and capabilities. Simply delivering something "that works" is not good enough for us. Our focus is on design, architecture, well written and maintained code as well*

*For “**Hardware Constraints**”,*

1. *As our project is a web application, it does not exactly have fixed hardware requirements. Most if not all of modern devices, on mobile, desktops, and laptops support web browsers capable of handling our web application*

*For “**Cultural Constraints**”,*

1. *We only deal with an interface designed for the English language. While we would like to cater to multiple languages to increase the reach for our audience, we have little, if not, no time to develop a strategy to do so*

*For “**Legal Constraints**”,*

1. *Privacy, SLA, User Agreement, Ensuring Security for end users to guarantee a safe environment to operate in is vital for us, but not the main aim of an FYP*

*For “**Environmental Constraints**”,*

1. *The software will be used primarily by students or teachers in classrooms. Hence, we would like to make our webpage as responsive as possible to cater to different screen sizes*

*For “**User Constraints**”,*

1. *The interface aims to be simple for both experienced and inexperienced users*

*For “**Off The Shelf Components**”,*

1. *We are using an ORM - thus, we are not writing direct queries to the database for any CRUD operations, and are relying on the ORM to do it on our behalf. Whether this is a constraint or not is subjective according to the requirements.*
2. *Most other aspects of these components are more so related to the ideas of “design” and “architecture”, and less so about “constraints”, “limitations”, or “rules”. We could have chosen other technologies to do what we liked, but the decisions were made on the basis of what would lead us to better software principles rather than always “ease of use”.*

2.8. Assumptions & Dependencies

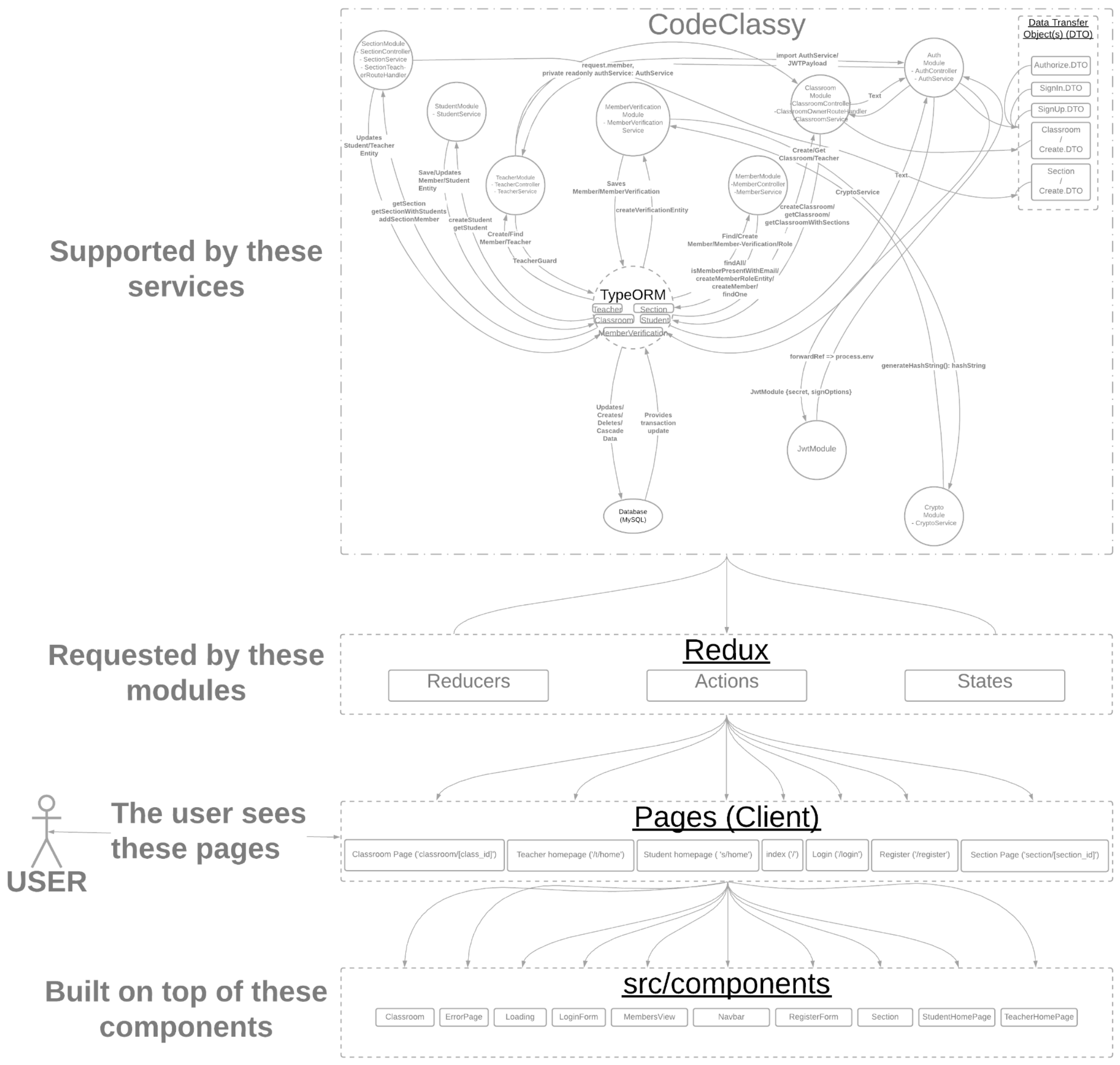
Going with the assumption (and generally the will) of putting the application finally into production once everything is completed and set up, we would like to cater to certain architectures to better adjust to our economic needs and finance.

For example, we would like to develop a system that is easy to modularize and break into components and microservices to be deployed on machines. As such, we would like to write a backend that supports such service related breakdown and adjustment.

As for the environment, it is a fact that users on different platforms and screens will use our platform.

Our dependencies will rely on the packages we use for development. We are assuming that they are long supported, lasting, and are easy enough to develop, modify as per our needs, and quick to learn to use in our project directly.

3. External Interface Requirements


CodeClassy
SectionModule - SectionController - SectionService - SectionTeacherRouteHandler
StudentModule - StudentService
MemberVerification Module - MemberVerification Service
TeacherModule - TeacherController - TeacherService
MemberModule -MemberController -MemberService
Classroom Module -ClassroomController -ClassroomOwnerRouteHandler -ClassroomService
Auth Module - AuthController - AuthService
Data Transfer Object(s) (DTO)
Authorize.DTO
SignIn.DTO
SignUp.DTO
Classroom / Create.DTO
Section / Create.DTO
request.member, private readonly authService: AuthService
import AuthService/ JWTPayload
Text
Updates Student/Teacher Entity
getSection getSectionWithStudents addSectionMember
Save/Updates Member/Student Entity
createStudent getStudent
Create/Find Member/Teacher
TeacherGuard
Saves Member/MemberVerification
createVerificationEntity
Create/Get Classroom/Teacher
Text
CryptoService
createClassroom/ getClassroom/ getClassroomWithSections
Find/Create Member/Member-Verification/Role
findAll/ isMemberPresentWithEmail/ createMemberRoleEntity/ createMember/ findOne
TypeORM
Teacher
Section
Classroom
Student
MemberVerification
forwardRef => process.env
generateHashString(): hashString
JwtModule {secret, signOptions}
JwtModule
Updates/ Creates/ Deletes/ Cascade Data
Provides transaction update
Database (MySQL)
Crypto Module - CryptoService
Supported by these services
Requested by these modules
Redux
Reducers
Actions
States
USER
The user sees these pages
Pages (Client)
Classroom Page ('classroom/[class_id]')
Teacher homepage ('/t/home')
Student homepage ('s/home')
index ('/')
Login ('/login')
Register ('/register')
Section Page ('section/[section_id]')
Built on top of these components
src/components
Classroom
ErrorPage
Loading
LoginForm
MembersView
Navbar
RegisterForm
Section
StudentHomePage
TeacherHomePage

3.1. Hardware Interfaces

As such, there is not really a place for hardware interfaces as all of the system lies entirely on the developer's machine for now. Thus, as a result, the client will only need a web browser to view the pages on-demand. The only hardware in use is just the machine that the user is using to run the web application.

3.2. Software Interfaces

There are 3 primary flows of data between the system,

1. ***Services*** *- Services provide the backend logic needed to connect with the database (MySQL) and offer modular services to the pages that need them*
2. ***Redux*** *- Calls the back-end resources on the front-end and maintains proper states for the pages*
3. ***Pages*** *- Pages are the views that the user actually sees*
4. ***Components*** *- Components are the parts of a view that are developed individually and integrated with each of the pages to provide a complete view of the page*

Externally, we are depending on NPM packages to support our application. They help in reuse as in a library, but do not exactly act as a way for us to store the data. All the data is stored on the database. Since the library does not store or transfer data (to make a data flow), they are not well suited here in these diagrams.

The database in question is MySQL, connected with an ORM called TypeORM.

There are no commercial components as of such.

Each of the services either talks to other services OR the ORM to provide services/responses based on the requests given to it. The purpose of these data items is to provide business logic on how to deal with incoming requests and to manage the transfer of data between the database and the actual views.

The services needed are described in the diagram above,

1. ***SectionModule*** *- For all services related to sections, such as updating student/teacher entities, getting section information, student information in sections, adding section members*
2. ***StudentModule*** *- For all services related to students, it saves/updates member/student entities, creates students, and fetches student information*
3. ***TeacherModule*** *- For all services related to teachers, it creates/finds member/teachers, and protects teacher related information/resources from unauthorized access using the AuthService*
4. ***MemberVerificationModule*** *- For all services related to member verification, it saves information about each member, and creates a verificationEntity*
5. ***MemberModule*** *- For all services related to members, it finds/creates member/member-verification. Some of the current services right now include, findingAll members, checking member presence, creating members, finding only one member, creating member entities*
6. ***ClassroomModule*** *- For all services related to classrooms, it creates, get classroom/teacher entities, it creates classrooms, gets classrooms, ge all classrooms with sections and uses the AuthModule*
7. ***AuthModule*** *- Connects with TypoORM for members and memberVerification related services*
8. ***CryptoModule*** *- Generates hashes for the module*
9. ***JwtModule*** *- For authentication related services and queries*

10. ***TypeORM*** *- For a connection with the databases. Provides updation, creation, deletion, and cascading of data across the database based on the entities provided to it such as Teacher, Section, Classroom, Student, MemberVerification*

All these services provide an end point that with the right information provide a response. It also has built-in integration of error handling for different error codes based on the request given.

3.3. Communications Interfaces

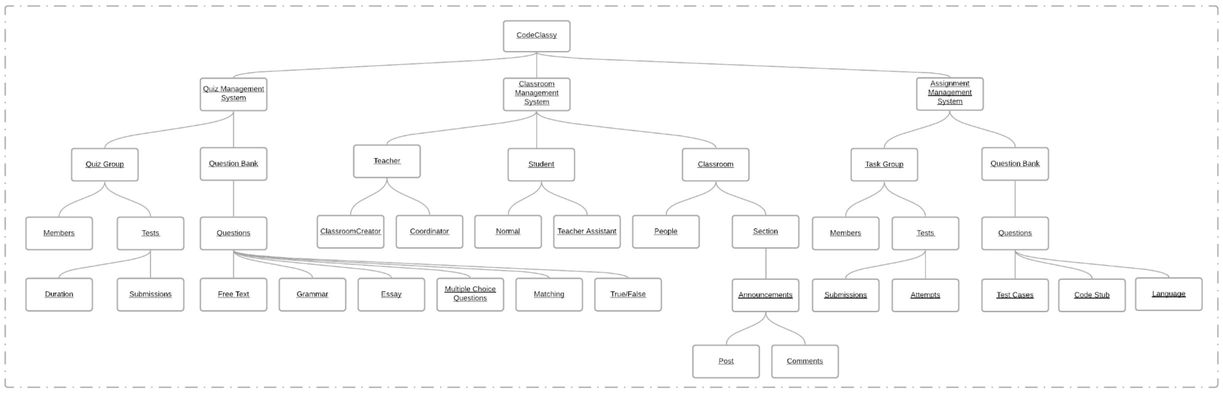
The underlying system uses the HTTP protocol for all data transfer. Communications on the front-end side are performed using the `axios` package found on NPM for JavaScript. There are no integrated packages right now for e-mail packets and messaging. The web-browsers that can be used are standard browsers such as FireFox, Chrome, Opera, Edge.

There isn't a specified pertinent message format.

There isn't work done right on the secure Communication Security side of things, or encryption. We do have an emphasis on synchronization as we are not currently dealing with distributed systems. Data Transfer rates of performance are not a critical issue as of right now.

4. Functional Requirements

4.1. Functional Hierarchy



As per above, there is a functional hierarchy described by each of the levels of hierarchy. In terms of user stories, these can be written as the following,

1. *Teachers should be able to create new classrooms*
2. *Teachers should be able to create a course online for the classroom*
3. *Teachers should be able to publish posts on the classroom stream*
4. *Teachers should be able to invite students to the created classrooms using a referral link*
5. *Teachers should be able to add students to the created classrooms*
6. *Teachers should be able to create quizzes*
7. *Teachers should be able to schedule quizzes according to time constraints*
8. *Teachers should be able to assign quizzes to students*
9. *Teachers should be able to create coding assignments for both individuals and in groups*
10. *Teachers should be able to assign coding assignments both individually and in groups*
11. *Teachers should be able to track progress of students for quizzes assigned to students*
12. *Students should be able to track progress with respect to the tasks assigned to them in the form of quizzes and assignments*
13. *Students should be able to access an integrated coding environment to solve the coding assignments with automated testing on provided test cases*
14. *Students should be able to take quizzes assigned to them by teachers*
15. *Students should be able to comment on the posts published on the classroom stream*
16. *Students should be able to implement the coding assignments assigned to them using the integrated coding environment*
17. *Students should be able to collaborate together in groups for coding assignments*

4.2. Use Cases

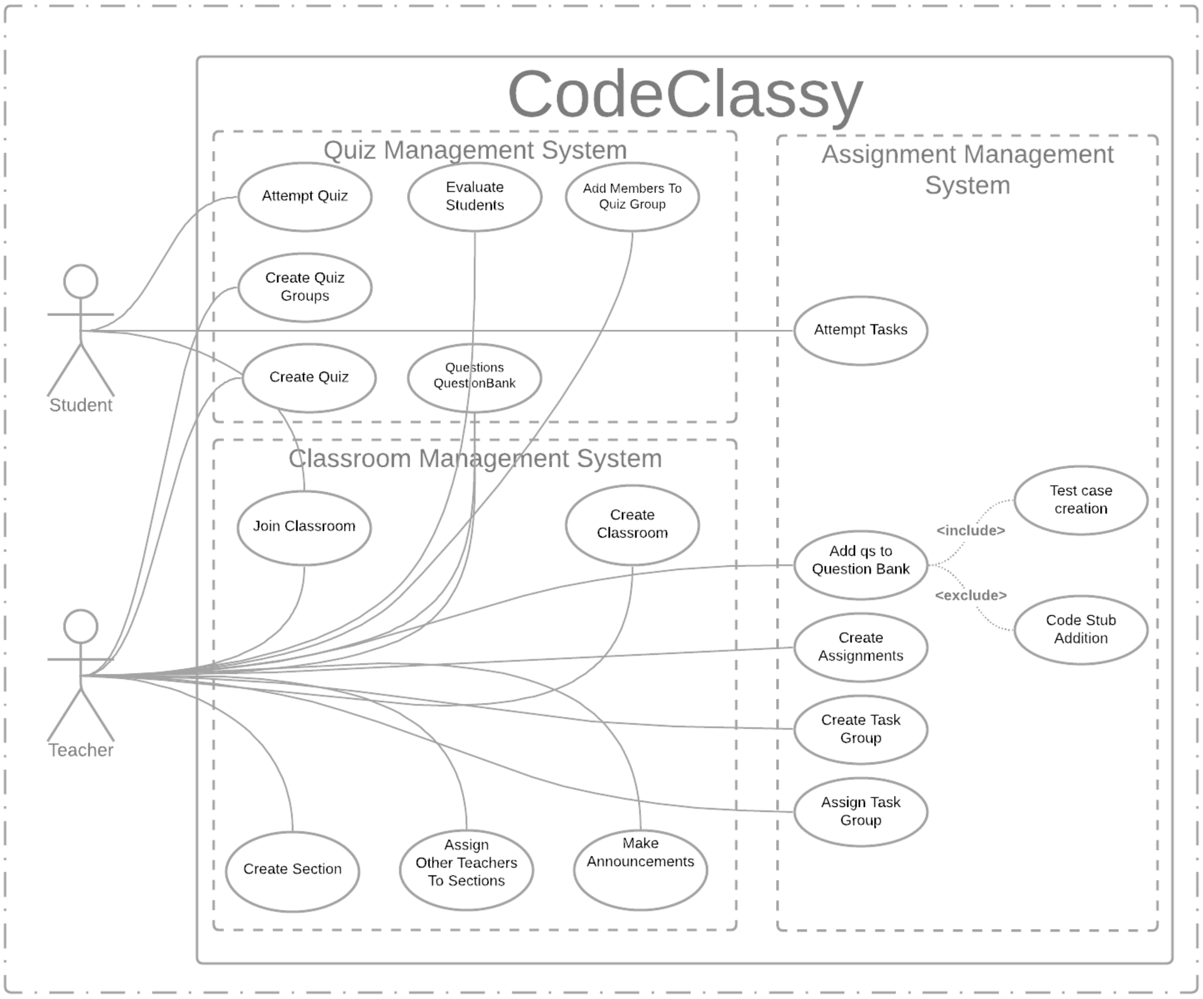
The primary uses of concern right now fall mainly into three categories,

1. *Quiz Management System*
2. *Classroom Management System*
3. *Assignment Management System*

Each of these three use cases fall further into their own categories (sub-functions) as the following,

1. *Quiz Management System,*
    a. *Quiz Group*
        i. *Members*
        ii. *Tests*
            1. *Duration*
            2. *Submissions*
            3. *Attempts*
    b. *Question Bank*
        i. *Questions*
            1. *Free Text*
            2. *Grammar*
            3. *Essay*
            4. *Multiple Choice Questions*
            5. *True/False*
2. *Classroom Management System,*
    a. *Teacher*
        i. *ClassroomCreator*
        ii. *Coordinator*
    b. *Student*
        i. *Normal*
        ii. *Teacher Assistant*
    c. *Classroom*
        i. *People*
        ii. *Section*
            1. *Announcements*
                a. *Post*
                b. *Comments*
3. *Assignment Management System*
    a. *Task Group*
    b. *Question Bank*
        i. *Questions*
            1. *Test Cases*
            2. *Code Stub*
            3. *Language*

4.2.1. Use Cases


CodeClassy
Quiz Management System
Attempt Quiz
Evaluate Students
Add Members To Quiz Group
Create Quiz Groups
Create Quiz
Questions QuestionBank
Classroom Management System
Join Classroom
Create Classroom
Create Section
Assign Other Teachers To Sections
Make Announcements
Assignment Management System
Attempt Tasks
Add qs to Question Bank
<include>
<exclude>
Test case creation
Code Stub Addition
Create Assignments
Create Task Group
Assign Task Group
Student
Teacher

1: Add Members To Quiz Group		
Use case Id:	1	
Actors: *Teacher*		
Feature: *Quiz Management System*		
Pre-condition:	A quiz group exists	
Scenarios: *A teacher wants to add members to a quiz group*		
Step#	***Action***	***Software Reaction***
1.	Teacher first creates a quiz group	A quiz group is either successfully created or is denied based on some certain conditions.
2.	*Teacher opens dialog box and enters emails of members to add to quiz group*	*Members are successfully added based on their role on the platform (students), or are rejected if that email does not match a member, or if that member is NOT a s*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: *Members are added or denied to be added to a quiz group depending on the role and the email if it verified to be a member*		
Step#	***Description***	
1.	Email(s) are checked if they actually belong to (a) member(s)	
2.	*If yes, add that member to the specified quiz group*	
3.	*If no, reject the request (throw an error) and display and error message*	
Use Case Cross referenced	NA	

2: Create Quiz		
Use case Id:	2	
Actors: *Teacher*		
Feature: *Quiz Management System*		
Pre-condition:	A quiz group exists with members	
Scenarios: *A teacher will create a test that can be attempted by students*		
Step#	***Action***	***Software Reaction***
1.	Teacher first creates a quiz group	A quiz group is either successfully created or is denied based on some certain conditions.
2.	*Teacher creates questions that are added to the question bank*	*Questions are successfully created and added to the question bank*
3.	*After creating questions in the question bank or by reusing previous set of questions, the teacher can select questions from the question bank for a "Test"*	*A test is created by assigning it to a quiz group, and selecting questions from the question bank.*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: *Members are added or denied to be added to a quiz group depending on the role and the email if it verified to be a member*		
Step#	***Description***	
1.	A test is created	
2.	*The group is assigned the test*	
Use Case Cross referenced	NA	

3: Add Questions To Question Bank		
Use case Id:	3	
Actors: *Teacher, Student*		
Feature: *Quiz Management System*		
Pre-condition:	NA	
Scenarios: *A teacher will add questions to the question bank to choose from for the test*		
Step#	***Action***	***Software Reaction***
1.	Teacher adds questions to the question bank	Question is added
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: *A question is added and can be viewed the next time the quiz dashboard is seen*		
Step#	***Description***	
1.	Teacher visits quiz page to add questions	
2.	*Teacher adds a question*	
Use Case Cross referenced	NA	

4: Evaluate Students		
Use case Id:	4	
Actors: *Teacher*		
Feature: *Quiz Management System*		
Pre-condition:	A test submitted by student(s) has now to be evaluated by the teacher	
Scenarios: *A student attempts a quiz that will now have to be evaluated (checked) by a teacher*		
Step#	***Action***	***Software Reaction***
1.	Student submits a test	A test is submitted as an "attempt"
2.	*Teacher "evaluates" a submission*	*After "evaluation", a number has been assigned to the test attempt*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: *An attempt has been evaluated, and score has been assigned to a student in a student group*		
Step#	***Description***	
1.	A teacher walks through each of the questions, "evaluates" it	
2.	*After evaluation, the submission is "finished" and the correct number can be assigned to that student for that attempt for a quiz*	
Use Case Cross referenced	NA	

5: Create Quiz Groups		
Use case Id:	5	
Actors: *Teacher*		
Feature: *Quiz Management System*		
Pre-condition:	NA	
Scenarios: *A teacher wants to create quiz groups to which students can be assigned so the teacher can assign quizzes to that group*		
Step#	***Action***	***Software Reaction***
1.	A teacher creates a quiz group	A quiz group is added
2.	*The teacher adds members to that quiz group*	*Members are added to that quiz group*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: *Quiz groups are created and members are added to that quiz group*		
Step#	***Description***	
1.	*Quiz groups are created and members are added to that quiz group*	
Use Case Cross referenced	NA	

6: Attempt Quiz		
Use case Id:	6	
Actors: *Student*		
Feature: *Quiz Management System*		
Pre-condition:	NA	
Scenarios: *A student wants to "attempt" a "quiz" assigned to him by a "teacher" through a "quiz group".*		
Step#	***Action***	***Software Reaction***
1.	A student attempts the quiz	An attempt for that student is recorded
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: *Attempt results are submitted and saved on the database*		
Step#	***Description***	
1.	*Attempt results are submitted and saved on the database*	
Use Case Cross referenced	NA	

7: Create Classroom		
Use case Id:	7	
Actors: *Teacher*		
Feature: *Classroom Management System*		
Pre-condition:	A teacher should be able to create a classroom	
Scenarios: *A teacher would like to create a classroom where he/she can add sections and teacher coordinators to that classroom*		
Step#	***Action***	***Software Reaction***
1.	A teacher creates a classroom	A classroom is created
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: *A classroom is created*		
Step#	***Description***	
1.	A classroom is created	
Use Case Cross referenced	NA	

8: Assign Other Teachers To Section

Use case Id:	8	
Actors: *Teacher*		
Feature: *Classroom Management System*		
Pre-condition:	A teacher should be able to assign other teachers to a section	
Scenarios: *A teacher would like to create a section within a classroom and assign another teacher to be a "coordinator" for that classroom*		
Step#	***Action***	***Software Reaction***
1.	A teacher creates a classroom	A classroom is created
2.	*The teacher creates section within that classroom and assigns it to another teacher account*	*Section is created and assignation is performed*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: *A section with an assigned coordinator is created*		
Step#	***Description***	
1.	A section with an assigned coordinator is created	
Use Case Cross referenced	NA	

9: Addition To Sections		
Use case Id:	9	
Actors: *Student*		
Feature: *Classroom Management System*		
Pre-condition:	A teacher wants to add students to his/her classroom	
Scenarios: *A student has made an account on the platform, and a teacher now wants to add that student to their classroom*		
Step#	***Action***	***Software Reaction***
1.	A teacher adds students via a dialog box for students in a section	Students are added to the section
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: Students are added to the section		
Step#	***Description***	
1.	Students are added to the section	
Use Case Cross referenced	NA	

10: Create Section

Use case Id:	10	
Actors: *Teacher*		
Feature: *Classroom Management System*		
Pre-condition:	A classroom creator should be able to add sections to that classroom	
Scenarios: *A classroom can have multiple sections to emulate real world teaching scenarios*		
Step#	***Action***	***Software Reaction***
1.	Clicking on the “Create Section” button on the Classroom page should give the ClassroomCreator the option to create a section	A section is created
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: NA		
Step#	***Description***	
Use Case Cross referenced	NA	

11: Make Announcements		
Use case Id:	11	
Actors: *Teacher*		
Feature: *Classroom Management System*		
Pre-condition:	A teacher assigned to a section should be able to add announcements to a classroom	
Scenarios: *A teacher assigned to a section will want to share resources with his/her section for updates about the section's progress*		
Step#	***Action***	***Software Reaction***
1.	A teacher assigned to a section will visit the section's view.	Section page is rendered
2.	*The teacher clicks on the "Post announcement" bar on the section page*	*A textfield will open which will let the teacher write content to share with the section*
3.	*Clicking on the "Post" button will share the announcement with the section*	*The post is added to the list of announcements for that section*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: Announcement is shared with the classroom		
Step#	***Description***	
1.	Post is added to the list of announcements for that section	
Use Case Cross referenced	NA	

12: Attempt Assignments		
Use case Id:	12	
Actors: *Student*		
Feature: *Assignment Management System*		
Pre-condition:	A student assigned an assignment should be able to perform the given task(s) and perform an "attempt"	
Scenarios: *A teacher assigns a task to a student that he/she must "attempt"*		
Step#	***Action***	***Software Reaction***
1.	A teacher will assign the student an assignment	An assignment is assigned to a student
2.	*A student will attempt that "task" or "assignment"*	*The submission will be noted*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: An attempt is noted		
Step#	***Description***	
1.	An attempt is noted	
Use Case Cross referenced	NA	

13: Add Question To Assignment		
Use case Id:	13	
Actors: *Teacher*		
Feature: *Assignment Management System*		
Pre-condition:	NA	
Scenarios: *A teacher wants to add questions to the question bank that he/she can reuse everywhere*		
Step#	***Action***	***Software Reaction***
1.	A teacher will open the question creation option	The option is now displayed to the user
2.	*The teacher adds information about the assignment such as the name, the content, the code stub (if available), test cases, set the language, and more related information and submits*	*The new question is submitted to the user for the end user to use*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: A question is added to the question bank		
Step#	***Description***	
1.	A question is added to the question bank	
Use Case Cross referenced	NA	

14: Assign Assignments To Sections		
Use case Id:	14	
Actors: *Teacher*		
Feature: *Assignment Management System*		
Pre-condition:	Questions exist within the Question Bank	
Scenarios: *A teacher wants to create an assignment. He/she will want to do with the questions created and added to the Question Bank*		
Step#	***Action***	***Software Reaction***
1.	A teacher will open the assignment creation option	The option is now displayed to the user
2.	*The teacher adds information about the assignment such as the name, the content, the code stub (if available), test cases, set the language, and more related information and submits*	*The new question is submitted to the user for the end user to use*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: An assignment is created		
Step#	***Description***	
1.	An assignment is created	
2.	*Students will be able to see assigned assignment*	
Use Case Cross referenced	NA	

14: Create Assignments		
Use case Id:	14	
Actors: *Teacher*		
Feature: *Assignment Management System*		
Pre-condition:	Questions exist within the Question Bank	
Scenarios: *A teacher wants to create an assignment. He/she will want to do with the questions created and added to the Question Bank*		
Step#	***Action***	***Software Reaction***
1.	A teacher will open the assignment creation option	The option is now displayed to the user
2.	*The teacher adds information about the assignment such as the name, the content, the code stub (if available), test cases, set the language, and more related information and submits*	*The new question is submitted to the user for the end user to use*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: An assignment is created		
Step#	***Description***	
1.	An assignment is created	
2.	*Students will be able to see assigned assignment*	
Use Case Cross referenced	NA	

15: Create Task Group		
Use case Id:	15	
Actors: *Teacher*		
Feature: *Assignment Management System*		
Pre-condition:	Students have registered on the platform	
Scenarios: *A teacher wants to create a Task Group where he/she can add students so as to assign work to them*		
Step#	***Action***	***Software Reaction***
1.	A teacher creates a task group	A group is created on the backend
2.	*The teacher starts adding students for that task group*	*Members are added to the task group*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: Task Group(s) is/are are created and Members are added to that group		
Step#	***Description***	
1.	Task group is created	
2.	*Members are added*	
Use Case Cross referenced	NA	

16: Assign Task Group		
Use case Id:	16	
Actors: *Teacher*		
Feature: *Assignment Management System*		
Pre-condition:	A task group has been created and members have been added to that group	
Scenarios: *A teacher wants to assign tasks/assignments to a task group*		
Step#	***Action***	***Software Reaction***
1.	A teacher creates a task group	A group is created on the backend
2.	*The teacher starts adding students for that task group*	*Members are added to the task group*
3.	*The teacher creates a task and assigns to it that task group*	*The task group is assigned the task created by the teacher on the back-end*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: Tasks are allocated to the task group		
Step#	***Description***	
1.	Tasks are allocated to the task group	
Use Case Cross referenced	NA	

17: Test Case Creation		
Use case Id:	17	
Actors: *Teacher*		
Feature: *Assignment Management System*		
Pre-condition:	A question is created for the question bank	
Scenarios: *Test cases need to be added to evaluate the results on automation*		
Step#	***Action***	***Software Reaction***
1.	A teacher creates a question in the question group	A question is created on the back-end
2.	*The teacher adds test creation that will be evaluated once the code is submitted*	*Inputs and output are matched to score the "Attempt" or "Submission"*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: Test cases are successfully added to the question for automated evaluation		
Step#	***Description***	
1.	Evaluations are performed automatically via matching inputs and outputs	
Use Case Cross referenced	13	

18: Code Stub Creation		
Use case Id:	17	
Actors: *Teacher*		
Feature: *Assignment Management System*		
Pre-condition:	A question is created for the question bank	
Scenarios: *A code stub should be added to provide a starting point for attempts by students*		
Step#	***Action***	***Software Reaction***
1.	A teacher creates a question in the question group	A question is created on the back-end
2.	*The teacher adds a code stub which the student can use to quickly get to the main logic of the task*	*A code stub is provided to all students attempting the task to get them a head start*
Alternate Scenarios: NA		
1a: ***2a:***		
Post Conditions: Code Stub is added to the question in the question bank		
Step#	***Description***	
1.	Code Stub is added to the question in the question bank	
Use Case Cross referenced	13	

5. Non-functional Requirements

5.1. Performance Requirements

The system should be prepared to handle large amounts of incoming traffic and not falter. It should cater to increasing and decreasing demand, and provide a solution that does not prevent users from being unable to browse and use the features smoothly. The application should be able to perform requests and services as intended, providing a robust and correct set of inputs and outputs, and being able to cater to user mistakes. It should be fast in loading the correct pages and the data in a smooth fashion, and should not overwhelm the user's machine with its software requirements.

The application should, thus, provide,

1. ***Speed** - fast for loading, reloading, content fetching, and not putting limitations on the client end and instead on the user's end*
2. ***Precision** - the software should be able to tolerate mistakes, user problems, and deliver the right results without compromise, and discrepancy in the output provided*
3. ***Concurrency** - the software should be able to handle multiple requests and multiple "activated" functionationalities concurrently, where needed*
4. ***Capacity** - the software should be able to handle and deal with multiple concurrent users requesting for the same resources*
5. ***Safety** - the complexity of the software is minimal, for both safety and security purposes, for administrative and normal user procedures. It does not pose harm to the end user. Failure modes, including hardware, software, human, and system are addressed in the design of the software*
6. ***Reliability** - The software should exhibit failure free functionality and work with correctness and reliability.*

5.2. Safety Requirements

The software does not include functionalities or operations that would result in possible loss, damage, or harm from the use of our system. It does not pose a physical, financial or otherwise cause of harm to the end user. It does not put interface in such a way that would affect the user negatively, or put any user in any risk or case of discrimination through poor interface design and color sensitivity.

5.3. Security Requirements

Obvious issues such as data privacy, integrity, confidentiality, protection of data generated by user through their content, protection of internal private information, and exposure to data to only what is needed when is the primary concern of the software project. The data modelling aspect concerns itself with what is populated by what, and who has access to what kind of information and at what level. We also define what resources are defined under what users and at what capacity to ensure un-authorized users are redirected accordingly. Unauthenticated users are prevented from accessing any part of the *system through the use of guards, proper token payload with JWT, and confidentiality is ensured by the proper management of sessions and through password encryption.*

The system should focus on setting up it's probable infrastructure as securely as possible, and making sure that communication between different services is done securely and data is trusted to be stored as expected. Authentication, authorization, are all important concerns.

Because the scope of an FYP is limited, we have not considered putting time into identifying security, or privacy policies or certifications that the system must satisfy.

5.4. User Documentation

Our requirements do not yet specify the development of user documentation to act as helping guides. We assume that the user interfaces will be simple and enough to be used at an MVP level. This is because the scope of an FYP is limited to development as a main priority. If time allows and if needed, user documentation can be developed after the development has been completed and has ceased.

6. References

Not applicable

7. Appendices

Not applicable