COSC Learning Management System

Requirements and Design Document

Tennyson Demchuk - td16qg@brocku.ca - 6190532
Mutaz Fattal - mf17lg@brocku.ca - 6362156
Joel Gritter - jg17uy@brocku.ca - 6331763 (leader)
Kindeep Singh Kargil - kk17xg@brocku.ca - 6329817
Aditya Rajyaguru - ar18xp@brocku.ca - 6582282
Daniel Sokic - ds16sz@brocku.ca - 6164545

COSC 4P02
February 6, 2021

Table of Contents

Authors' Note **2**

Requirements **3**

Functional Requirements 3

1.1 User Authentication and Access 3

1.2 Student Abilities 3

1.3 TA Abilities 3

1.4 Professor Abilities 4

1.5 Functional System Behaviour 4

Non-Functional Requirements 4

User Story Equivalents 5

Development Workflow **7**

Branching Strategy 7

Jira 8

Automated Deployments 9

Database Design **10**

Context Diagram **10**

Deployment Diagram **11**

UI Design **12**

User Interaction Diagram 12

Authors' Note

Not all of the typical SRS documents are contained within this document, because we are utilizing an Agile methodology. The documents that we've provided in this document are the ones that logically make sense to make at this current point in time.

The rest of the SRS documents will be compiled as those respective components of the solution are developed, and then presented during the mandatory reports in March and April.

Requirements

Functional Requirements

1.1 User Authentication and Access

1.1.1 Users shall be able to sign into the system in such a way that users are uniquely identified and authenticated as such.
1.1.2 Users shall have different symbolic roles assigned to their accounts, such that they align with their actual campus roles.
1.1.3 Users shall be given access to different parts of the system dynamically according to the user role.
1.1.4 Users shall be able to edit their account details to ensure that their account information is accurate and up-to-date.
1.1.5 Users shall be given the ability to sign out of the system.

1.2 Student Abilities

1.2.1 Course Enrollments
1.2.1.a Students shall be able to see which courses they are currently enrolled in
1.2.1.b Students shall be able to enroll in new classes using a professor-provided code

1.2.2 Submissions
1.2.2a Students shall be able to submit submissions to assignments
1.2.2b Students shall be able to submit submissions to tests

1.2.3 Submission Feedback
1.2.3a Students shall be able to see automated test feedback on a given submission
1.2.3b Students shall be able to see feedback provided by a TA or professor on a given submission

1.3 TA Abilities

1.3.1 Course Enrollments
1.3.1a TAs shall be able to see which courses they are currently set as a TA for
1.3.1b TAs shall be able to be added to course by a professor

1.3.2 Course Administration
1.3.2a TAs shall be able to see all the students enrolled in a class to which the TA has been assigned
1.3.2b TAs shall be able to see each submission for each assignment in a course to which the TA has been assigned
1.3.2c TAs shall be able to see previously provided feedback on an assignment, whether provided by the automated tests, a TA, or a professor

1.3.3 TA Marking
1.3.3a TAs shall be able to provide feedback on any submission within any course to which the TA has been assigned

1.4 Professor Abilities

1.4.1 Course Enrollments
1.4.1a Professors shall be able to create new courses
1.4.1b Professors shall be able to add TAs to a course the professor oversees
1.4.1c Professors shall be able to generate a code with which students can join the course

1.4.2 Course Assessments
1.4.2a Professors shall be able to create new assignments within a course they oversee
1.4.2b Professors shall be able to create new tests for the courses they are in charge of

1.4.3 Assessment Feedback
1.4.3a Professors shall be able to created automated tests for assessments
1.4.3b Professors shall be able to see provided feedback on an assignment within a course they oversee, whether the feedback is provided by the automated tests, a TA, or a professor
1.4.3c Professors shall be able to provide feedback on an assignment for any course for which they oversee

1.5 Functional System Behaviour

1.5.1 The system shall acknowledge to the user when a submission has been received

Non-Functional Requirements

2.1.1 Under normal circumstances, a typical web-page request shall not exceed 5 seconds.
2.1.2 The system shall be able to handle enough concurrent sessions that 75% of the enrollment sum of all classes on the system can be handled at the same time.
2.1.3 The system shall never lose student submissions after acknowledgement of receiving the submission has been sent.
2.1.4 The system shall use TLS 1.2 and provide sessions over HTTPS.
2.1.5 The web-page should display correctly on mobile devices as well as desktops/laptops.
2.1.6 The system should have good security practices. Passwords should be hashed and salted; not stored in plaintext. Any input into the system shall be sanitized appropriately.
2.1.7 Automated testing of student submissions shall not exceed 120% of the time it would take to run manually.

User Story Equivalents

Since we are using Kanban, we'd thought we'd also include what these requirements could look like as user stories in the Kanban board:

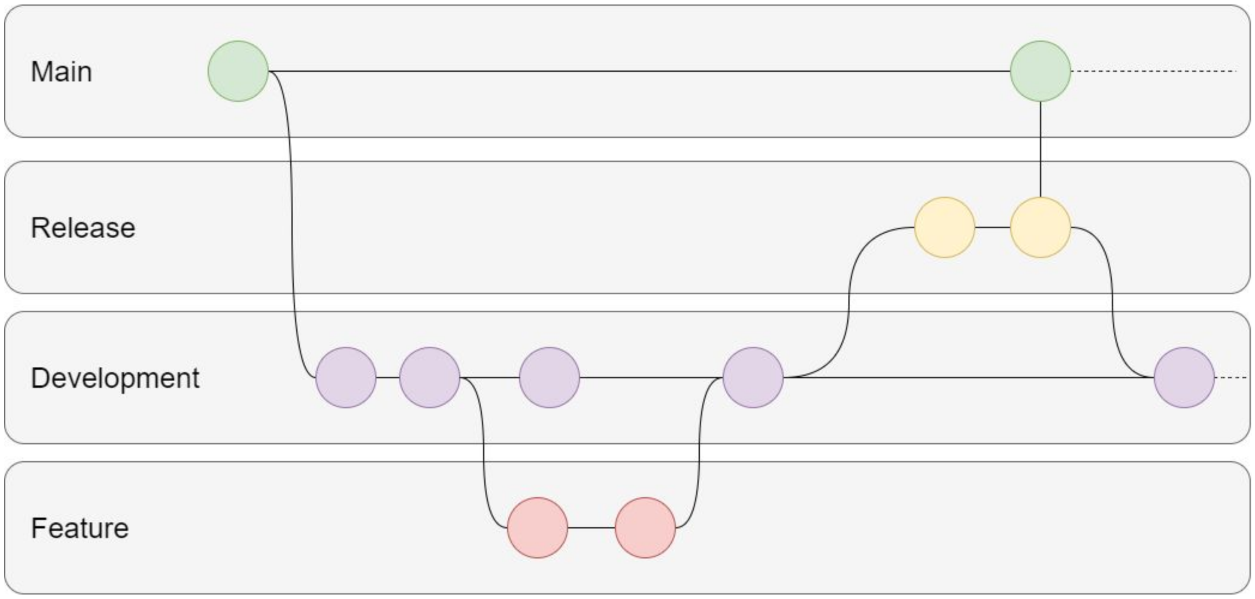
#	User story title	User story description	Priority	Notes
1	Login	As a user, I want to be able to login so that I can access my account.	High	Requires authentication with database
2	Assign permissions	As a sysadmin, I want to be able to assign permissions to different accounts.	Low	Need to be able to assign student, TA or professor permissions to an account
3	Profile Access	As a user, I want to be able to edit my profile details.	Low	
4	Upload submission	As a student, I want to be able to upload my assignment submission.	Medium	
5	See test feedback	As a student, I want to see what I got on the test so that I can understand my areas for improvement.	Medium	
6	See TA/Prof Feedback	As a student, I want to see written feedback from professors or TA's so that I can ask questions about the feedback.	Medium	
7	Add courses by name or id	As a student, I want to be able to add a new class to my dashboard, so that I can access the course homepage.	High	
8	See all students in class	As a professor or TA, I would like to see all students in my class.	High	
9	See submissions for assignment	As a professor or TA, I would like to see the submissions for an assignment	Medium	
10	See automated test feedback for submissions	As a professor or TA, I would like to see test feedback for individual students, so that I may assess and help my students in weaker	Medium	

		areas.		
11	Give Feedback	As a professor/TA, I want to be able to give feedback on a student submission, so that the student has a chance to improve	Low	
12	Create automated test cases	As a professor, I can create automated test cases for students.	High	
13	Create new classes	As a professor, I would like to be able to create new classes	High	
14	Create new assignment	As a professor, I would like to be able to create new assignments.	High	Be able to upload PDFs and write assignment description
15	Create test/quiz	As a professor, I would like to be able to create new tests	High	Ability to create, MCQ's, long answer, short answer etc.

Development Workflow

Branching Strategy

We will be following the GitFlow workflow model by Vincent Driessen.



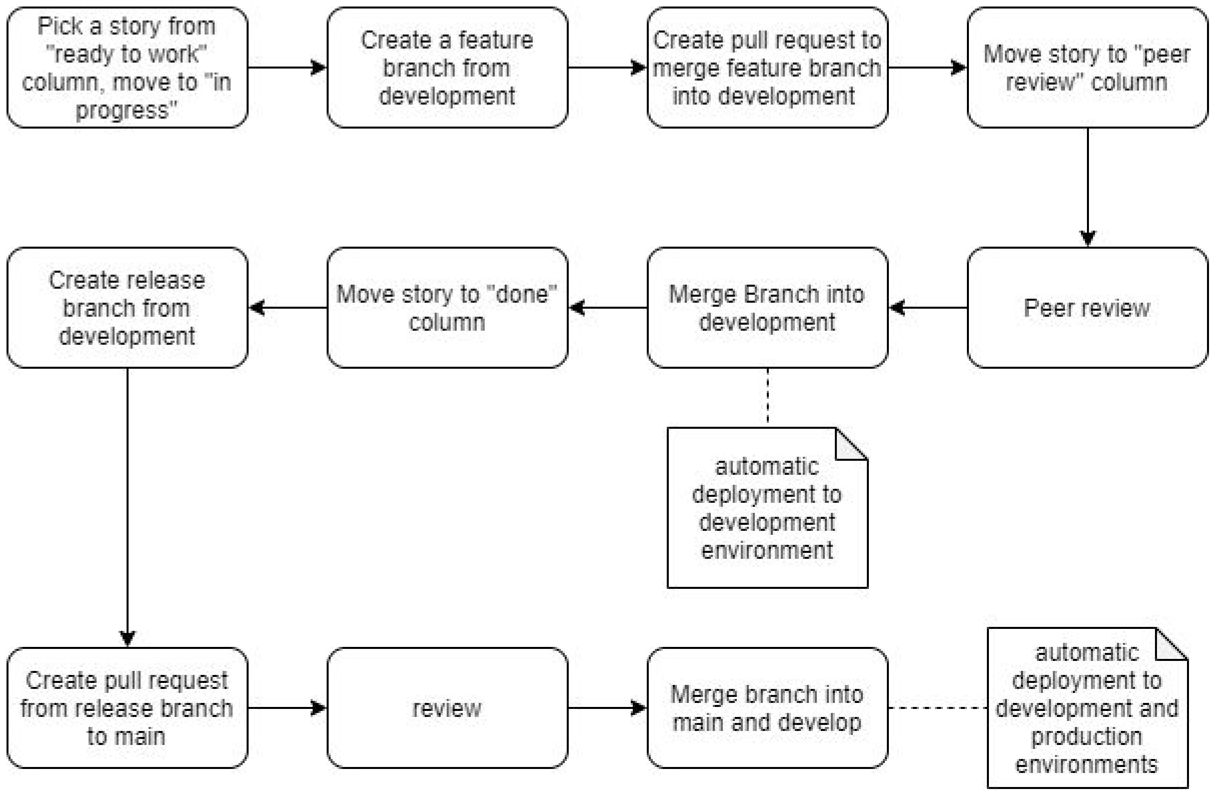
- All development will be done in feature branches
- Feature branches will start from the development branch, once a feature is completed, a pull request is made to merge the feature branch into development. The pull request goes through a peer review before getting merged into development.
- After merging, the feature branch is deleted.
- To create a release, a release branch is created from development. A pull request is made to merge this branch into main requiring one review. This branch is deleted after the merge.
- Changes to development and main branches will be automatically deployed to a development environment and a production environment through GitHub actions.
- Initially, it might be sufficient to only have the development branch and the respective environment and automation set up.

Jira

To make and track the progress of user stories, we will be using Jira. Our Jira board has 5 columns:

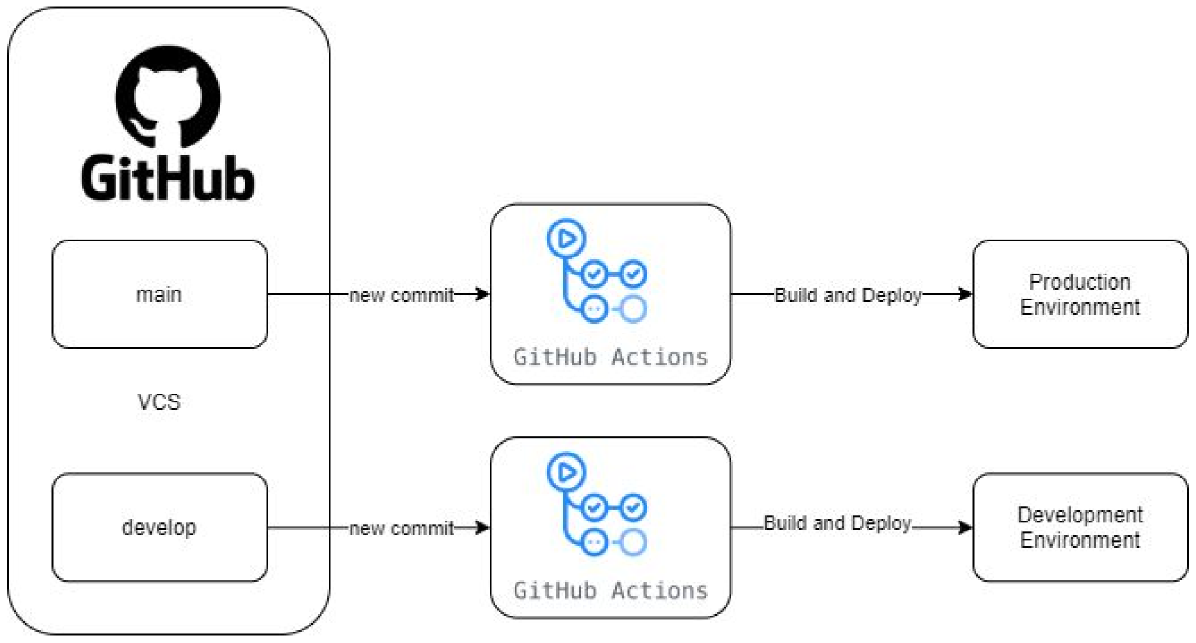
1. **To-Do** - New stories
2. **Ready for Work** - Stories ready to be picked up by a developer
3. **In progress** - Stories currently being worked on
4. **Peer review** - Stories with development done, waiting for a peer review before merging into the “develop” branch
5. **Done** - Merged into “develop”

This is how the development workflow will look like with this strategy:



Pick a story from "ready to work" column, move to "in progress"
Create a feature branch from development
Create pull request to merge feature branch into development
Move story to "peer review" column
Peer review
Merge Branch into development
Move story to "done" column
Create release branch from development
automatic deployment to development environment
Create pull request from release branch to main
review
Merge branch into main and develop
automatic deployment to development and production environments

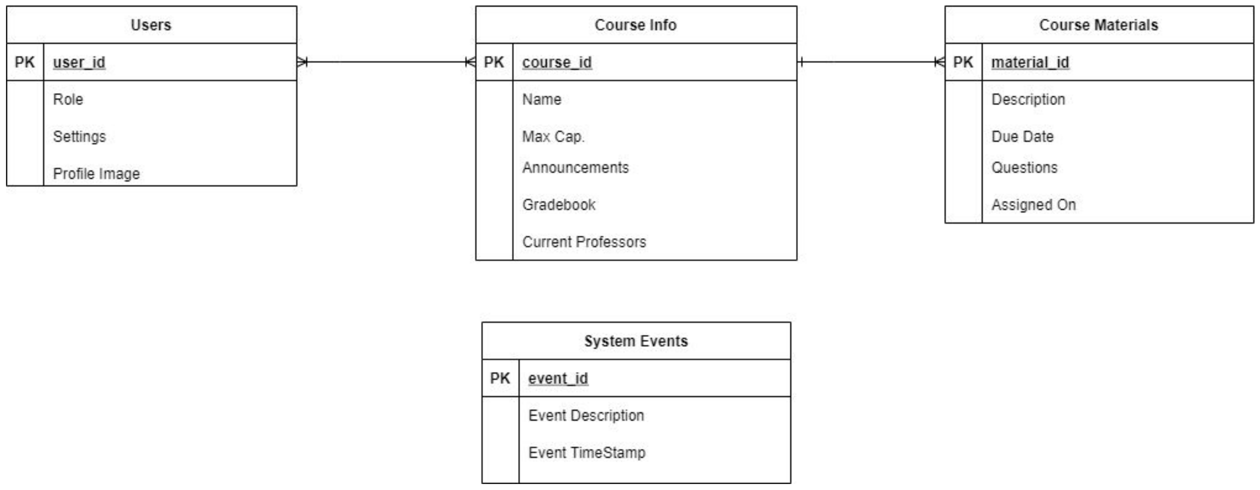
Automated Deployments



We will be using GitHub Actions for automated deployments. Two environments will be put in place, production and development, connected to branches main and develop respectively.

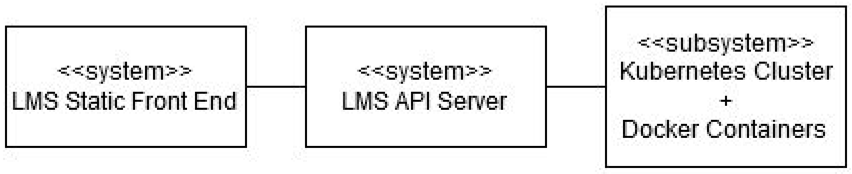
When a new commit is added to either branch, the code will be built with GitHub actions and then deployed to its respective environment.

Database Design



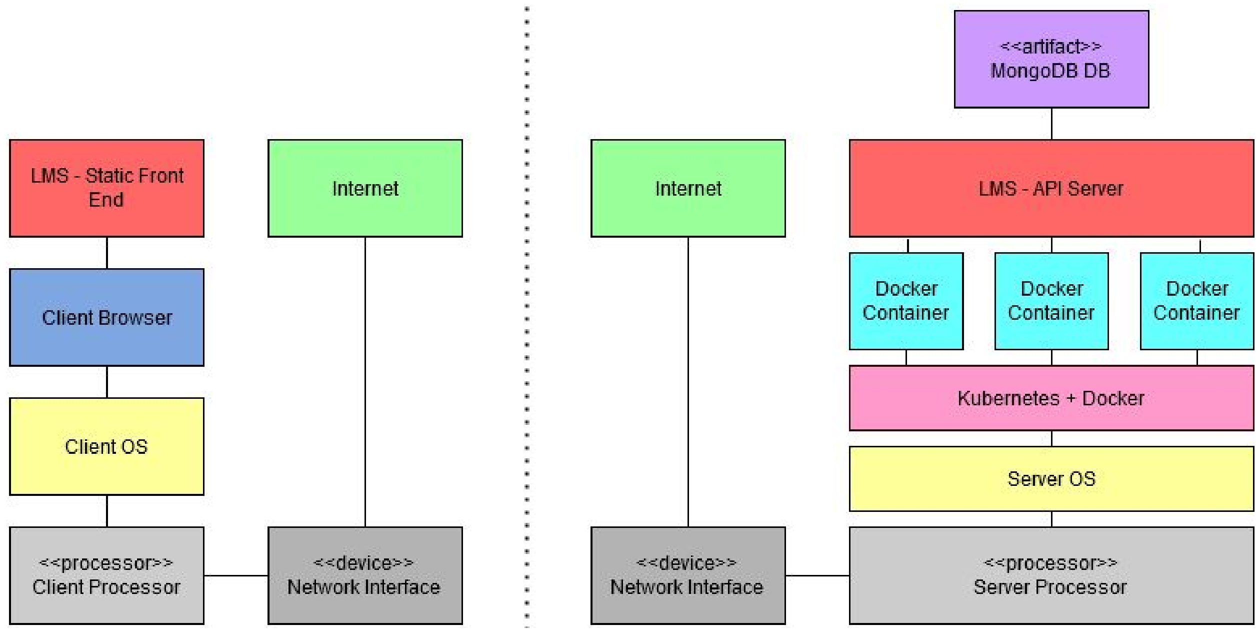
The database will consist of 4 main collections. The Users collection will hold all registered users and related information (i.e. classification, settings, etc.). The Course Info collection will hold each created course and the assigned information (i.e. announcements, gradebook, course documents, etc). The Course Materials collection will contain all assignments, tests, and quizzes related to a course. The System Events collection maintains all events that occur on the system with a description of the event and timestamp of when it occurred.

Context Diagram



The context of the system is defined within two main components: the client-side front end and the server-side API server. The API server (including Database) lies on top of a subsystem of Docker containers managed by a Kubernetes cluster.

Deployment Diagram



Deployment of the system on the client's end only requires a common modern browser (thus requiring an operating system and underlying hardware that will support this). Deployment on the server-side requires that a Kubernetes cluster and Docker Engine be set up for Docker containers. These containers allow the API server implementation to be platform-agnostic. A MongoDB database must also be set up to interface with the API server. Thus the underlying hardware and operating system of the server must be supported by Docker, Kubernetes, and MongoDB.

See also:

- https://www.docker.com/
- https://kubernetes.io/
- https://www.mongodb.com/

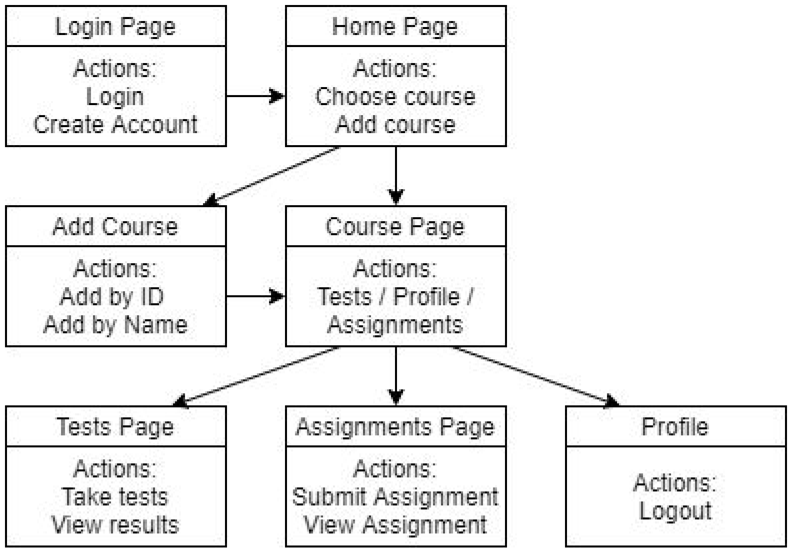
UI Design

For the front-end portion of our project, we will be utilizing React.js, an open-source JavaScript library developed by Facebook. The reasoning behind using such a library is that it helps our team build a reliable and well-performing web application. React utilizes client-side rendering, which boosts performance as it generates content dynamically as the user navigates through the application. This specifically benefits our use case where most of the application code can be cached. This also allows our application to work as a Progressive Web App. The library enables developers to build reusable components that can be built in isolation to be constructed into a complex UI.

With React.js the team will be using Material UI, a UI framework built for React. This framework enables teams to reuse prebuilt common components to help developers achieve their UI goals sooner and in a more reliable manner. Material UI has assisting tools to help design any complex application the developer desires. This framework enables the team to focus less on low-level components, and more on the overall UI.

See also:
- https://reactjs.org/
- https://material-ui.com/

User Interaction Diagram


Login Page
Actions:
Login
Create Account
Home Page
Actions:
Choose course
Add course
Add Course
Actions:
Add by ID
Add by Name
Course Page
Actions:
Tests / Profile /
Assignments
Tests Page
Actions:
Take tests
View results
Assignments Page
Actions:
Submit Assignment
View Assignment
Profile
Actions:
Logout