Classroom Features

Project # 19
Project TA: Chandini Palem

Team #: 10b
Division: Emerald

Date Started: 10/25/2023
Date Delivered: 12/13/2023

Product Manager: Zachary Miller, (607) 542-0005, millerz@ufl.edu
Scrum Master: Kyle Hoang, (305) 322-6034, kyle.hoang@ufl.edu

Development Team:
Zackary (Brock) Denson, (904) 673-3640, zackary.denson@ufl.edu
Theodros Amde, (904) 480-7378, tamde@ufl.edu
Manny Modrono, (305) 934-2635, manny.modrono@ufl.edu
Prajuna Venkatesan, (813) 469-6798, venkatesanp@ufl.edu

Thank you for sponsoring a project with

University of Florida CEN 3031 Introduction to Software Development.

For more info on this course, please contact Dr. Christina Gardner-McCune gmccune@ufl.edu

Table of Contents

1. Product Overview

1.1 Brief Description of the Project

This project allowed a teacher to queue lessons for students to work through. It also allowed mentors and content creators to add due dates and open/close dates to assignments. In addition, notifications were sent out to students about due dates, as well as custom announcements sent by the mentor.

A classroom page was added, with sections for assignments, a calendar, syllabus, discussion board, and notifications. These sections are linked to pages that the mentor can create.

A discussion board was created that allows students to communicate with mentors and other students. Mentors have the ability to delete posts and replies if needed.

1.2 Major Features Proposed & Completed

Description of prioritized features for this development effort

- Our team prioritized getting features functional before focusing on making features integrated with one another.
- The basic features listed as project features were as follows:
    - Allowing teachers to queue lessons
    - Creating release, closure, and due dates
    - Adding past due/upcoming notifications for students
    - Creating a lesson creator with a rich text editor
    - Adding a classroom view with lessons and the syllabus
    - Creating a discussion board with a rich text editor

All Completed tasks were proposed unless stated otherwise

Proposed	Completed
● Due dates on calendar ● Creating an improved lesson	● Notification Center ● Discussions

creator ● Creating automatic notifications for due dates	● Student Dashboard ● Syllabus ● Due Dates ● Lesson Queue ● Content Creator

1.3 Links to Useful Project Information

GitHub https://github.com/CEN3031-Team10b/Emerald-Project19-10b

GitHub Project Board link: https://github.com/orgs/CEN3031-Team10b/projects/1

Live Web Application URL: N/A

Web Application Admin Information

Web Server Hosting Provider - N/A

Document or Image Storage Provider - N/A

There is no live version of the website at the current moment

2. Product Planning

2.1 About the User Stories Creation Process

a. Process: Describe your initial process for creating your user stories.

Brainstorming activities conducted as a group where we conducted an idea storm of what the product should deliver based on the position of the intended user (i.e. teachers and students). User stories were refined based on the Epic assigned in the project and through communication with the team and the client.

Reflection: User stories broken into smaller subtasks in the project board and then assigned points and developers.

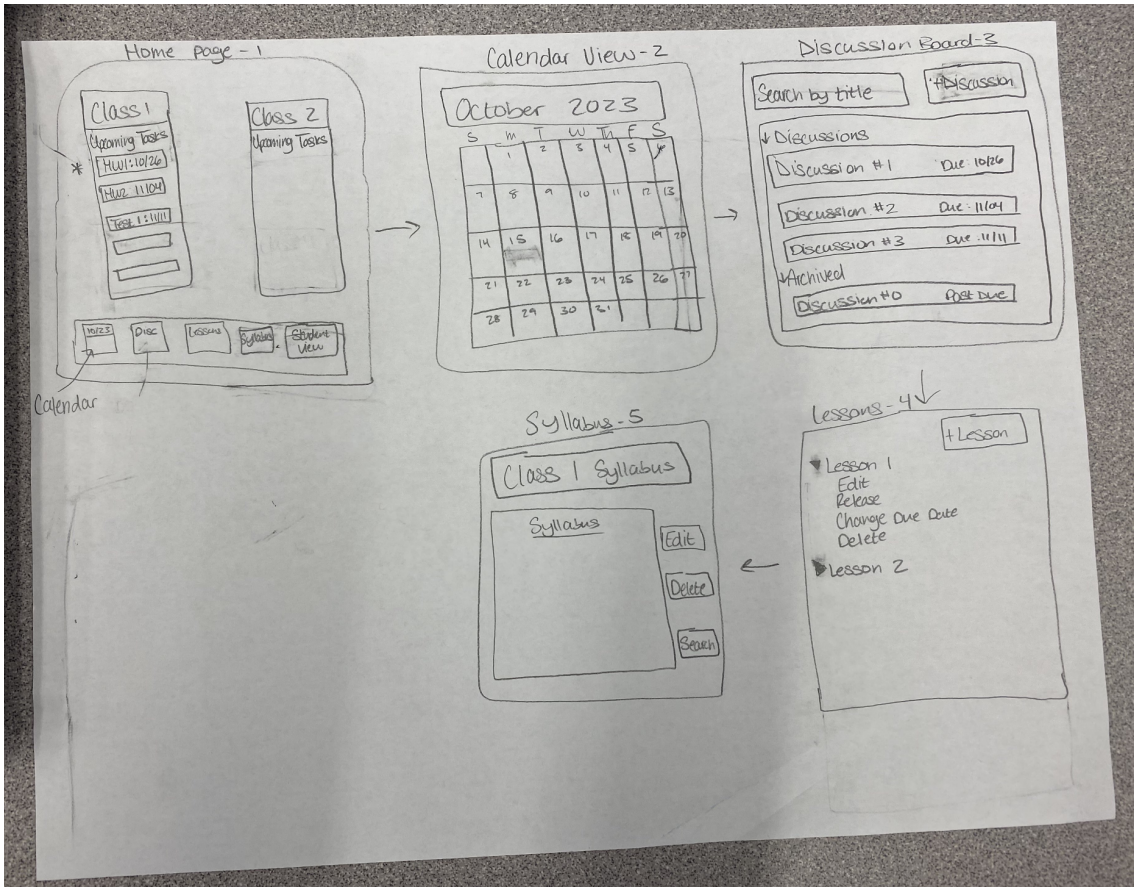
b. Reflection: In what ways did your user stories change over the course of the project and Why?

Our user stories became more specific over the course of the project as more details were introduced which made us be more specific with our user stories. Additionally, the client had shown us in greater detail exactly what they wanted.

Following the additional information directly from the client, we were able to formulate additional stories that were more in line with what the client wanted.

See section 1.3 for the Github Project Board

2.2 About the Wireframes Creation Process



a. Process: Describe your initial process for creating your wireframes.

Looking at the project specifications and client requirements, we thought about the different features that students and teachers should have access to in order to satisfy those requirements. Using Canvas as a backbone, we thought about which pages should encompass the various features, and designed the wireframe with the minimal number of pages possible to enhance easy access for students/teachers.

b. Reflection: In what ways did you use your wireframes and how did they change over the course of the project and Why or why not?

We used the wireframe to figure out what features would be the most difficult to implement. For example, we saw that the home page would need to link up with every other page. Therefore, we were able to prioritize certain tasks.

In addition, as the project progressed, we realized how certain features needed to change. One example is the discussion board(s). At first, we thought that we would include discussion assignments to the application. However, we realized that it would be better if it was turned into an open channel of communication between mentor and student.

We also decided to change the pages that we created. Instead of having a separate page for the calendar, we instead created a widget that fit inside the student view.

This wireframe was a good basis for the application. However, as the client clarified their needs and we understood our assignment, we changed details within this wireframe.

See section 1.3 for Github project board

3. Technical Specifications

MEAN.JS Development Technology Stack/ REACT, STRAPI, NodeJS, Docker, Vite, Angular

This web application was built using MEAN.js Version 1.5.0 <http://meanjs.org> which uses Node,js Version 21.4.0, Express Version 4.18.1, Angular Version 1.8.2. Vite 4.5.0. Docker 4.25.0.

We cloned the repository to GitHub Organization CEN3031-Team10b and Project Emerald-Project19-10b https://github.com/orgs/CEN3031-Team10b/projects/1

Database

The current version of this program operates without a database, as it stores all information locally. Because of this, adjustments will have to be made if future versions are to be used with a database.

External Libraries & APIs

ReactQuill - https://github.com/zenoamaro/react-quill#usage

User Interface Styling (CSS & Bootstrap/other framework)

Your web app was styled using CSS v. x.x.x & Bootstrap/other framework v. x.x.x.

Location of CSS & Styling for this project

<name of folders/files> - link to Location in GitHub Repository - Developer(s) names
/client/src/components/NotificationCenter/NotificationCenter.less - Zackary Denson
/client/src/views/Mentor/Classroom/SendNotificationForm.less - Zackary Denson
/client/src/views/StudentDashboard/StudentDashboard.less - Kyle Hoang
client/src/views/Mentor/Classroom/Discussion/Discussion.jsx - Manny Modrono
client/src/views/Mentor/Classroom/Discussion/NewPost.jsx - Manny Modrono
client/src/views/Mentor/Classroom/Discussion/Post.jsx - Manny Modrono
client/src/views/Mentor/Classroom/Syllabus/Syllabus.jsx - Manny Modrono
client/src/view/Mentor/Classroom/Home/Home.jsx - Zachary Miller
client/src/view/ContentCreator/ActivityEditor/components/ActivityDetailModal.jsx - Zachary Miller
client/src/view/ContentCreator/LessonModuleCreator/LessonModuleCreator.jsx - Zachary Miller
client/src/view/ContentCreator/ContentCreator.jsx - Zachary Miller

Storage

We have no large files or videos that we need to store.

4. Product & Feature Design & Documentation

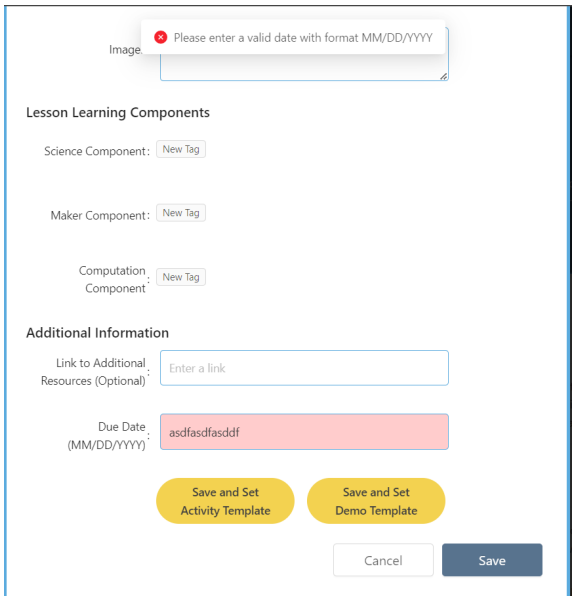
(incl. Screenshots, wireframes, and diagrams)

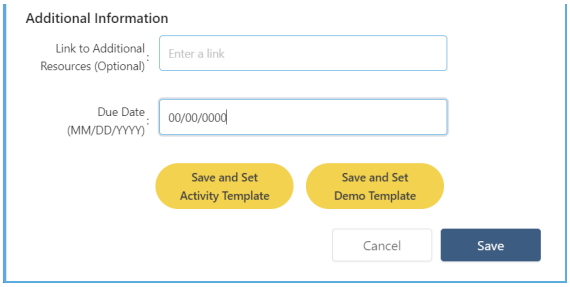
User Features

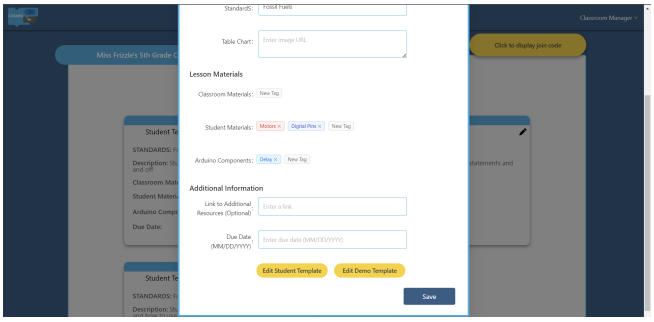
We developed these features for this project:

Due Dates

Due dates are now possible to add in the activity creator and editor. Both content creators and mentors can do this. The date is stored as text, but the text is checked to ensure that the date is in proper format. This is to ensure compatibility with other features when they are added.

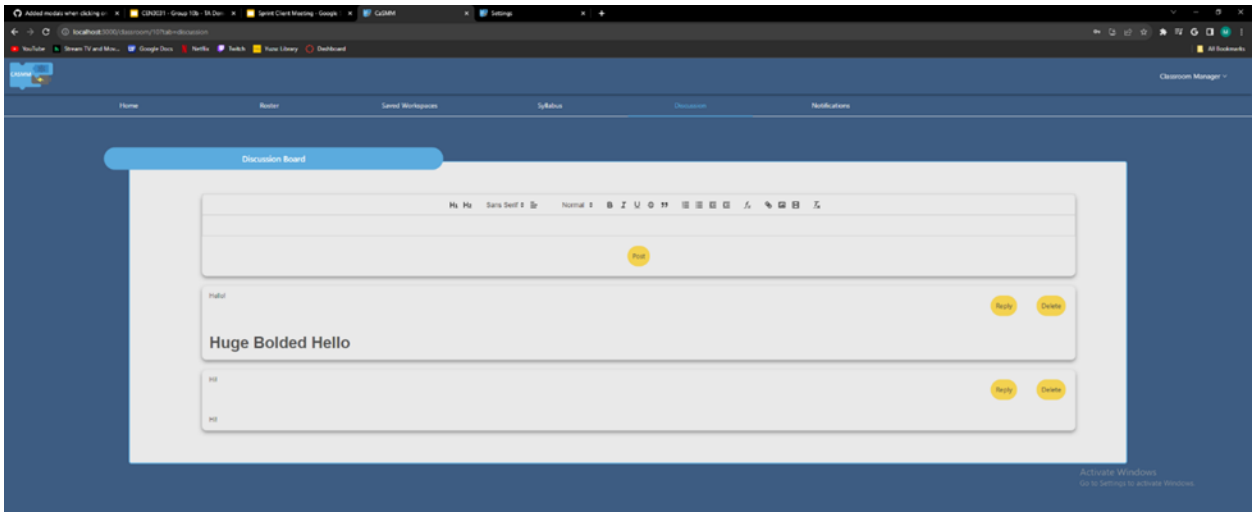






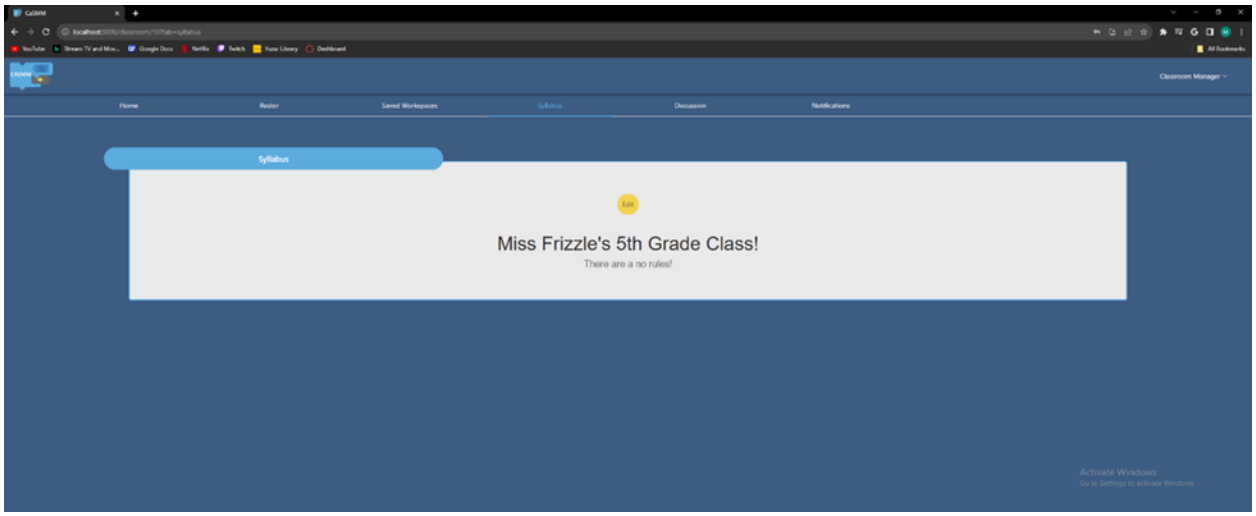
Discussion Board

The discussion board allows students and teachers to communicate. It allows for posts and replies. It is fully integrated into the back end and keeps each class's discussions saved.



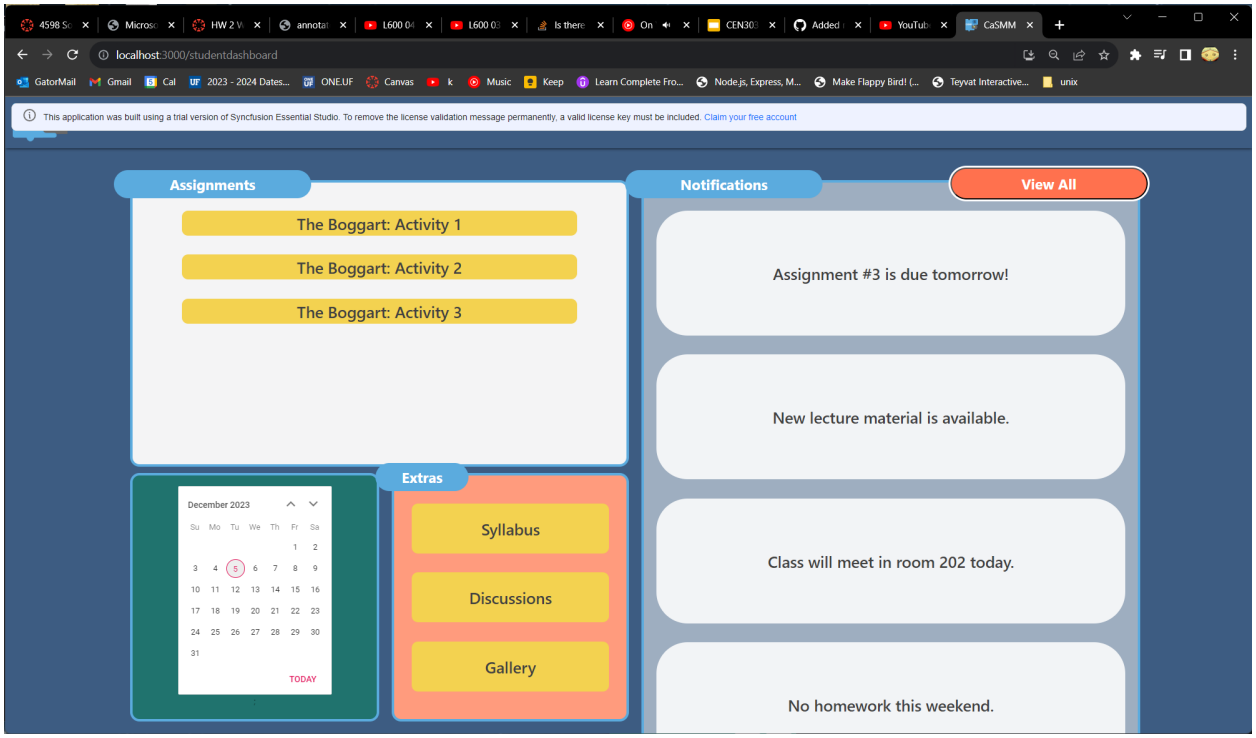
Syllabus Page

The syllabus page allows teachers to specify the rules for a class. Students have easy access to a class's syllabus from their student dashboard. The syllabus is stored in a classroom's collection.



Student Dashboard

The student dashboard is a new view that improves on the existing view and incorporates new features from other parts of the project for easy access for the student. Previously, students were only able to see assignments and had a lot of wasted space. The new view allows students to see assignments, notifications/announcements, due dates, discussions, and the syllabus all from one screen. Currently, this new view is a separate page which the user will have to navigate by going to "/studentdashboard". By default, it will go to the original view on login for legacy purposes.

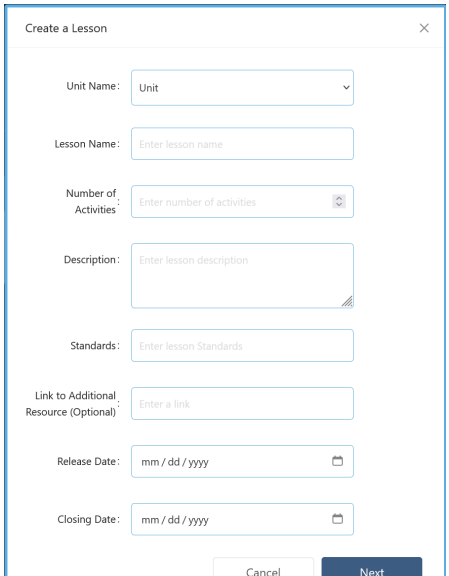


Database Models & Schemas

In order to store discussion boards, two collections were added. First, a collection named “Discussions”, which stores a discussion’s posts and a relation to the classroom it is connected to. Another collection, “Discussion-Posts”, stores all discussion posts and their replies. To store a classroom’s syllabus, an additional text field was added to the classroom collection.

For the Notification Center, a ‘Notification’ collection was made containing the fields for text, date, and the read/unread/flagged status of each notification. Additionally, a relation to the ‘Student’ collection was made to link each notification with students using a many-to-many relation.

Release and Closing Dates (Content Creator)

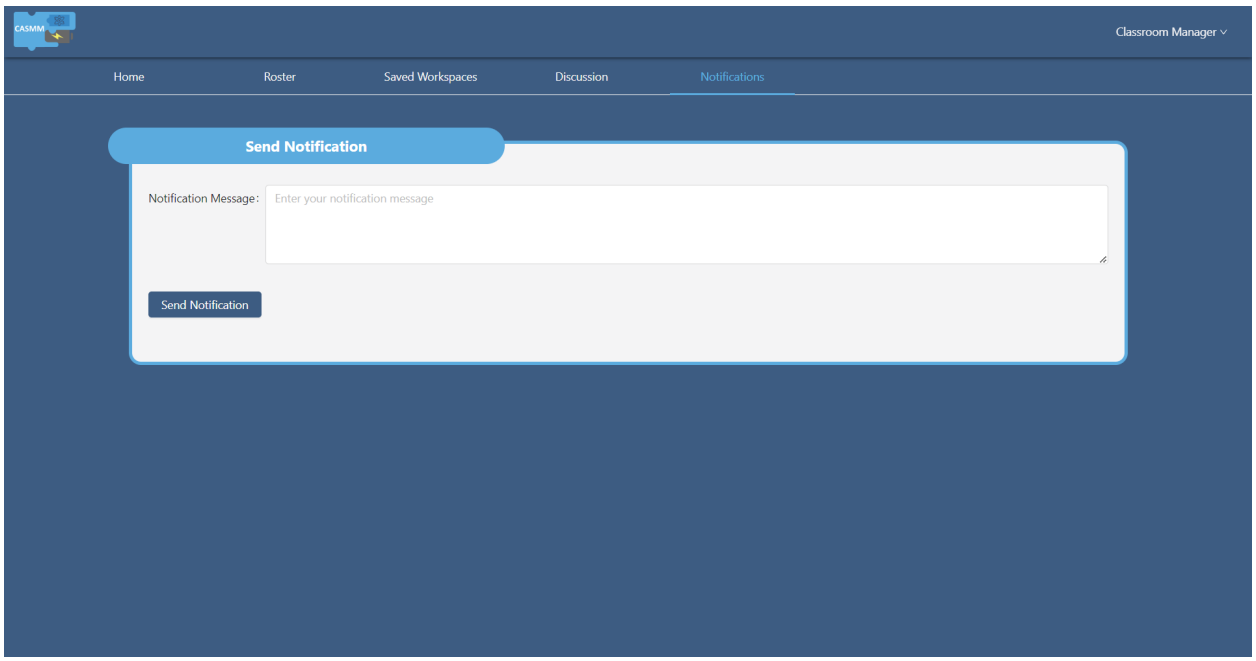


- When creating a lesson in the content creator, release closing dates can be selected. The dates cannot be before the current date, and the closing date cannot be before the opening date.
- The second screenshot shows a section for the dates to be shown but was unable to work on the backend to get it to show.

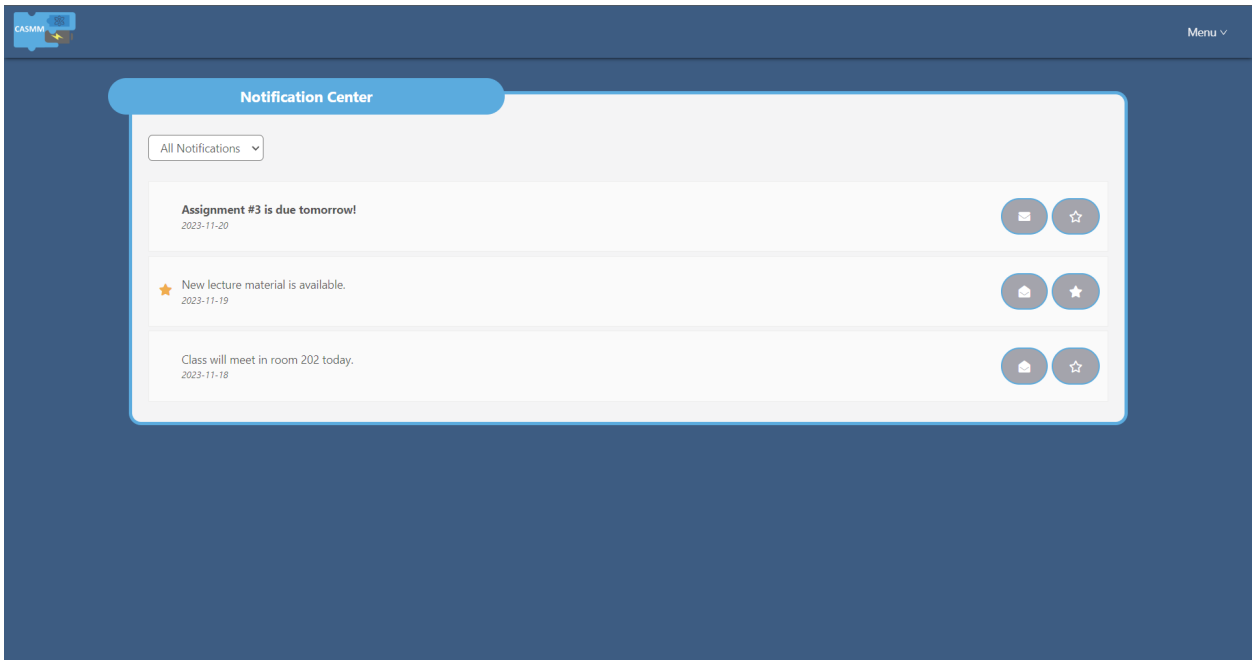
Notification Center

Teachers can send notifications from their classroom view to students. Students can view the notifications from their dashboard and mark them as read, unread, and flagged. Students can also filter notifications by these tags.

Teacher View:



Student View:



5. Testing Information

5.1 Description of Testing incl. Manual & Automated Tests

Our goal in testing this web application was to cover all the expected user interactions with the main features developed for the project by the development team.

Highly Tested Features	Still in Need of Extensive Testing
• Inserting invalid dates and inputs for dates • Correct UI scaling for design elements • Creating and deleting discussion posts	• Security of teacher view sending information to backend

5.2 Locations of tests

There are no tests that have to be run from the inside of the repository, so they are not included here.

5.3 Description of Frontend Test

Frontend implementations were manually tested by the individuals who implemented them. Incremental and end-to-end testing were primarily used in development of the UX and cross-examined with other team members to ensure the proper vision was implemented.

Student Dashboard

UI elements were sketched out crudely and implemented into easily distinguishable blocks and were continually updated with better visual design elements to fall in line within CASMM's design language. Visual testing across different browser environments was performed to ensure a consistent visual design.

Notification Center

The notification center frontend was tested using incremental integration testing whilst implementing the various features. Each button's functionality was implemented and tested before moving on to the next. After all the features and functionalities were properly implemented, the styling, formatting, and structure of the Notification Center was improved using CSS.

Release and Closing Dates (Content Creator)

The release/closing dates frontend was tested with UI testing. This was done by viewing other aspects of CaSMM and ensuring that the design matched up across the board and that it provided a good user experience. Cross-browser testing was also done to ensure it works on the most common web browsers (Chrome, Safari, Edge, Firefox).

Due Dates

The due dates frontend was tested manually. This was done by viewing the changes made in the website. This was done by making sure the visual design matched the design of features already present within CASMM.

Discussion Board

The discussion boards were manually tested. This was done by ensuring that new posts would be visible, deleted posts would no longer be visible, and that all buttons worked as intended.

Known Frontend Bugs:

Description of bugs and ideas of possible sources. Include links to user story/bugs in GitHub Project Board.

In the student dashboard, the first time a student views the syllabus, the text formatting does not load. When the modal is closed and reopened, the problem is fixed.

Calendar widget in the student dashboard does not scale properly with browser window size.

5.4 Description of Backend Test

Describe your process for backend testing and who was responsible for doing this part of the project.

We tested the back end manually by adding data in the Strapi backend and verifying important information. After confirming that the frontend can retrieve this data, we knew that functionality was successfully added.

Known Back-end Bugs:

Description of bugs and ideas of possible sources. Include links to user story/bugs in pivotal tracker.

Due Date Errors

Activities do not store due dates in the Strapi backend when created. This has been extensively tested, and solutions have been offered by other students and TAs. In this project, none of these methods have functioned. Whenever a mentor or content creator inputs a date, the application reads it properly. However, once it's sent via a HTTP request, its value is set to "null". Therefore, the issue must be with the HTTP requests.

5.5 Description of Integration Test

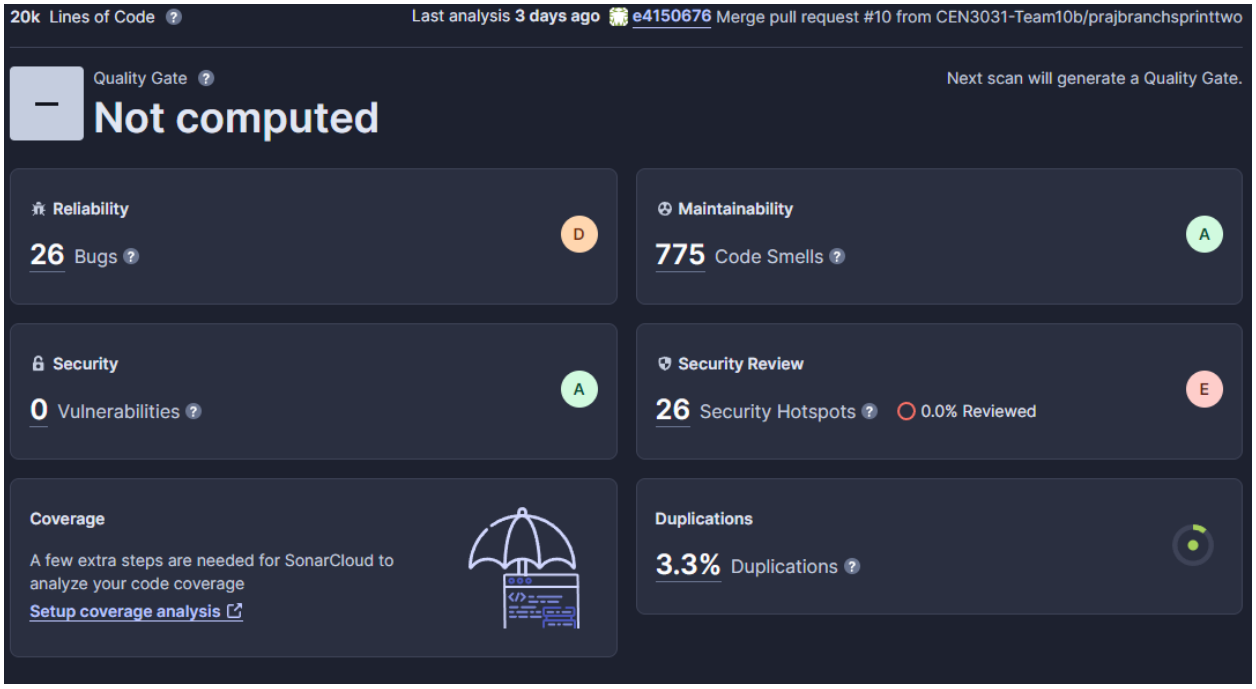
We did not do any integration testing, as most features could be tested manually.

Known Integration Bugs:

Description of bugs and ideas of possible sources. Include links to user story/bugs in GitHub Project Board.

5.6 Static Testing Results

Describe your process for static testing, how code smells, security vulnerabilities and hotspots were resolved if at all. Who was responsible for doing this part of the project. Readable Screenshoots of static test reports especially code smells, security vulnerabilities and hotspots.



6. General Instructions for Deployment or Running of your application

All Instructions for how to run your application up and running need to be on your GitHub ReadMe. Link https://github.com/CEN3031-Team10b/Emerald-Project19-10b/tree/develop#readme.

There are individual branches for each of our developers and at the end of each sprint the primary branch “develop” is merged into following peer-reviewing of code to ensure any conflicts are resolved. The develop branch is then tested to ensure functionality from which the individual branches will rebase their code off of.

Database

We are not using a database, as most information is held locally.

File Storage

Our project does not require us to store any information, so file storage is not a concern.

Other Integration or Third Party Tools

Node.js and node package manager must be installed. Docker must be running and using the command “docker compose up” in the root of the project will start the backend of the web app. Using “yarn” in the client folder will install all dependencies and running “yarn start” in client will begin Vite for the frontend rendering of the website.

Login Information or Accounts created to interact with your features

- **There are no new accounts. To access CASMM, use the login passwords provided**

All the secret Git Ignore or configuration information.

N/A