Executive Summary

The application detailed in this report was designed to assist elementary school teachers, parents, and students in tracking student course progress virtually. Course progress was funnelled from three mock Learning Management Systems that replicated Google Classroom, Brightspace, and Canvas, and then integrated them into an analytics page. Unlike individual LMSs, the application was created to give teachers and parents the option to track students' progress with just a few clicks, instead of having to navigate through separate applications or a large number of pages.

With the recent increase in remote learning due to COVID-19, many elementary school classrooms were made virtual over the past year. This transition to virtual learning has proved challenging for elementary-aged students. It has also made tracking student progress challenging for parents and teachers. The application’s core goal was to provide a solution to that challenge by making student progress as easy to track as possible.

The application went through several stages before technical development began. These stages included research into competitors and market influencers, initial design, interviews and survey results, finalized mock-up design, and finally implementation research. By researching competitors and market influencers, it became clear that there was a gap in the market pertaining to amalgamating LMS information. The core idea was unique and topical as remote learning grew in prevalence, and so would be very viable in the market.

The initial research was then used to design early mock-ups presented to interviewees. The team interviewed several developers and teachers both through text chat and surveys. Developers offered design and implementation insights, and teachers offered feedback on mock-up designs as well as how virtual learning was influencing their classrooms. The feedback that was received indicated that the core functionality of the application needed to be the student progress analytics page, as all parties were most interested in that feature.

Agile was chosen as the project management framework for the application. The team created tasks in a software called Jira, which were then displayed on a scrum board and grouped based upon implementation goals. The three sprints used during the four-month development cycle were the frontend development sprint, the mock data ingestion sprint, and the final application refinement sprint. Each sprint had a final goal, which would be achieved through the successful completion of every task on the scrum board.

Angular was chosen as the web application framework that would be used to implement the application. Technical design was possible once the web application framework was chosen, as it meant that the frontend and backend could be merged into one codebase. Microsoft Visio was then used to develop class diagrams to represent the different key elements of the codebase and their relationships to one another.

The frontend application design criteria were for the application to be highly intuitive, for each element to complement the analytics, and for the analytics to be colour-coded and readable. The backend application design criteria were for data from Google

Classroom, Brightspace, and Canvas to be successfully mocked and funnelled into the different pages, and for the codebase to follow a standardized folder structure for easier developer readability. Each of these criteria was considered in the team's final application design.

The final design included five main elements: the login page, the top-bar menu, the course list, the student list, and the student analytics page. The login page allowed the user to input their username and password, and identify themselves as a teacher, student or parent. The top-bar menu was a feature at the top of each page that would allow the user to navigate to other pages. The course and student lists displayed their respective courses and students, with the key difference being that the student list provided a preview of the students' analytics. Finally, the analytics page displayed the student analytics.

The application was implemented using various IDE's, Git, GitHub, npm, NodeJS, Angular, and Bootstrap. Various JavaScript and TypeScript libraries were also imported into the project as needed in order to overcome implementation hurdles. The key imported library was chart.js, which was used to create clean graphics for the student analytics page.

The final analysis of whether or not the application matched the design criteria was based upon developer and alpha tester feedback. The team was unable to secure teacher alpha testing before the four-month development period ended. Each team member secured one or more developer testers and reported back their findings, which were then pooled together and analyzed.

In general, testers felt that the application was intuitive. They felt that navigation from the course list to the student list to individual analytics pages was seamless. They also felt that the colour-coded grades were recognizable as low or high based upon their colour. The main criteria that was not met was overall readability, as the progress tracking chart was deemed unclear compared to the grade analytics. More labels for the chart were requested.

There were two main obstacles faced during the project: remote communication and time restrictions. Having entirely remote communication prevented in-person meetings and led to asynchronous communication which increased the amount of time needed to complete certain tasks. The four-month time restriction made completing the application, unit testing, and finding testers difficult. The core application features were completed after three months, however refinements coincided with testing such that testers were sometimes testing pages prior to them being refined.

Recommendations for future development included adding more features and connecting to LMS endpoints rather than mock endpoints. Another recommendation was expanding the target audience to include higher education levels. Additional research and survey results were deemed necessary in order to implement new features tailored to the particular obstacles of remote learning in middle school, high school, and post-secondary education.

Overall, the application development was an exceptional learning experience, and the final application was successful based upon the design criteria. The team learned how to mock data ingestion obtained from LMS APIs, as well as improved their Angular

development skills. The application created by the team during this development period was a good foundation from which to expand into new features in the future.

Table of Contents

List of Tables and Figures

Introduction

The purpose of this report is to describe the development process of an application that aimed to improve virtual classroom environments for elementary school teachers, students, and parents. In the past year, classrooms migrated from in-person, physical spaces to virtual, technology-driven experiences. This changed student progress tracking for teachers as well as how parental check-ins occurred for elementary-aged students. By improving the coordination between parents, teachers, and students through an engaging user interface and progress-tracking analytics, the team sought to improve virtual classroom environments as a whole.

To respect healthcare guidelines, the application was developed remotely. The team targeted mobile development first, then transitioned to a cross-platform application that expanded usability to a greater audience. The application's initial features were informed by the team's various contacts in education and a survey where individuals provided feedback on the user interface and analytics services.

With this feedback, the team focused on two strategies to improve elementary school virtual classrooms. The first strategy was to ensure the frontend design was intuitive, colour-coded, and revolved around the analytics page. The second strategy was to implement task and grade tracking through mock learning management system endpoints. The assumptions embedded in successfully meeting these criteria were twofold. The first assumption was that additional features would be added swiftly following the mock endpoint development. The second assumption was that teachers,

parents, and students would integrate the finished product into their virtual learning environments to improve their remote learning experience.

This report was written to inform educational board members, teachers, and parents about the application. The team focused on the Ottawa school district as their target audience. The goal of this report was to inform Ottawa school district board members about the research, design, and production of an application developed to improve remote support for teachers, parents, and students. The report was also created with the intention of encouraging teachers and board members to reflect on what is lacking in their own e-learning environments and evaluate their own virtual classroom setups.

The first two sections of the report are the report introduction and the background section. The background section begins with the research that prompted the application's initial concept and development, including a problem statement, budget, and project management information. The section following that is the design section which details the layout of the application and the platforms used in the development process. The report then discusses the methodology of the application's development and concludes with the team's recommendations based upon an analysis of the final application's benefits to virtual classrooms and missing features. The report does not detail the application's codebase or backend logical flow. This report is an analysis of how the application came to fruition and whether or not it was a successful prototype for improving remote learning for elementary school teachers, parents, and students.

Background

The idea for an integrated learning management system application stemmed from reading about the challenges of remote learning during the 2020-2021 pandemic. The idea later evolved to target elementary school remote learning, as the team determined that keeping young children engaged in an online classroom environment would be especially challenging. This was further reinforced upon interviewing teachers working with kindergarten to sixth grade students and hearing about the challenges they faced.

The application began development in Summer 2020. The idea generation and planning phase occurred between August and December 2020, followed by technical development between January and April 2021. During this time period, the need for a centralized repository for student task monitoring increased exponentially as hundreds of millions of students' schoolings was disrupted by COVID-19 and transitioned into a remote learning environment. [1]

This global disruption led to new educational challenges that went beyond teachers and students. Upon interviewing teachers, the team found that keeping parents engaged in the learning progress of their children was proving extremely challenging during online schooling. As a result, the idea of a hybrid application that could assist both teachers and parents to monitor students was formed. The application would act as a digital parent-teacher conference, allowing for both the parent and teacher to stay up-to-date on student progress.

In 2020, there were many solutions to improve virtual classroom environments on the market. Education-related applications were the third most popular applications downloaded from the Apple App Store at the time. [2] As the learning landscape evolved to be more digital, technology became more and more relevant to classroom environments. [3] Despite the many educational applications on the market, the team found no direct competitors that amalgamated learning management systems for student tracking, and thus the application would be filling in a market gap.

The applications of interest to this project were Learning Management Systems. All of the LMS's on the market were separate from one another and did not have a centralized portal through which teachers could easily access student progress information. Popular classroom applications such as Google Classroom, Brightspace, and Canvas possessed individual student progress assessments, however teachers needed to swap between them in order to acquire student information. The team identified this separation between LMS's as a market gap.

Another core component missing from LMS's was the ability for parents to monitor their children. Parents did not have their own portal through which to monitor student task progress without going through the student's account. The application would offer not only a centralized task monitoring portal for teachers and students, but also for parents.

In order to successfully create this application, the team needed a strong backend and frontend programming background, which included experience in web development, mobile development, and REST HTTP protocols. Each member of the team possessed technical work experience and more than two years of programming experience. The

team's collective backgrounds allowed for the successful completion of this application on a technical level.

There were five criteria used in this report to determine whether or not the project was successful. The first was that the final design was intuitive for adult users. The second was that each page would complement the analytics and make user navigation easier. The third was that the analytics would be colour-coded and readable. The fourth was that the LMS data would be successfully mocked. And the final criteria were that the codebase would be organized into a standardized folder structure, thus ensuring the code would be readable for future development.

In the event of this application meeting the aforementioned criteria, it would then be marketed towards school boards in Ottawa. Important school boards in the city of Ottawa were the Ottawa-Carleton District School Board and the Ottawa Catholic School Board. These boards were identified as potential buyers of this application.

Project Management

The application's budgeting and project management was successful. In September 2020, the team decided on Agile Scrum as a project management framework. Tasks were arranged by the team using a scrum board and then grouped based upon their dependencies and similarities. Sprints were four weeks long and the scrum master ensured that sprint goals were achieved. The first and last sprint goals were met without issue, while the second sprint's length was increased to five and a half weeks in order to account for unexpected interruptions.

The budget for the application was $99 USD to purchase an Apple License for product deployment, and this budget was self-funded. [4] The team's primary distributions were through web-based free software and creating a google add-on, which did not add any additional costs. Costs for the project matched the initial budget exactly. The team executed the project within the budget guidelines decided upon prior to development.

The schedule for completing this project was eight months. The first four months were dedicated to market research, planning, and designing prototypes and mock-ups for the application. Many changes and discussions were had throughout this period that solidified the idea as filling in a market gap. The final four months were dedicated to application development. The team managed to execute five out of five criteria for core functionality within this timeframe.

Design

Early Concepts and Feedback

The core design objective of this application was to improve student task monitoring for teachers, parents, and students. The user interface was designed with this goal in mind, and tertiary features were stripped away throughout the design process so that the team could revolve the application around this one objective.

There were two tertiary features integrated into the initial application design: a colourful desktop layout and an assignment distribution repository combined with a task-reward

system. These two features were discarded during the four-month planning stage with the intention of implementing them following the four-month development stage.

Figure 1. A stock image example of the initial design idea for desk generation [5].

The first tertiary feature was a desktop layout that included auto-generated desks based upon the number of students [Fig. 1] that could be personalized by each student. This design also included a virtual blackboard. Based upon survey results about classroom sizes being too large to fit all of the desks on one screen, this layout feature was discarded.

The assignment distribution repository and task-reward system were discarded due to the fact that with these features, the application would become a direct competitor to

established Learning Management Systems. This market was very saturated and therefore not the correct approach for the application.

Based upon professional feedback, time management, and the usefulness of the feature, a student analytics page with easy-to-understand statistics and visuals [Fig. 2] took precedence over the tertiary features. Professional contacts using LMSs were far more eager to use the analytics design [Fig. 2] compared to the desk layout or an assignment repository with a task-reward system.

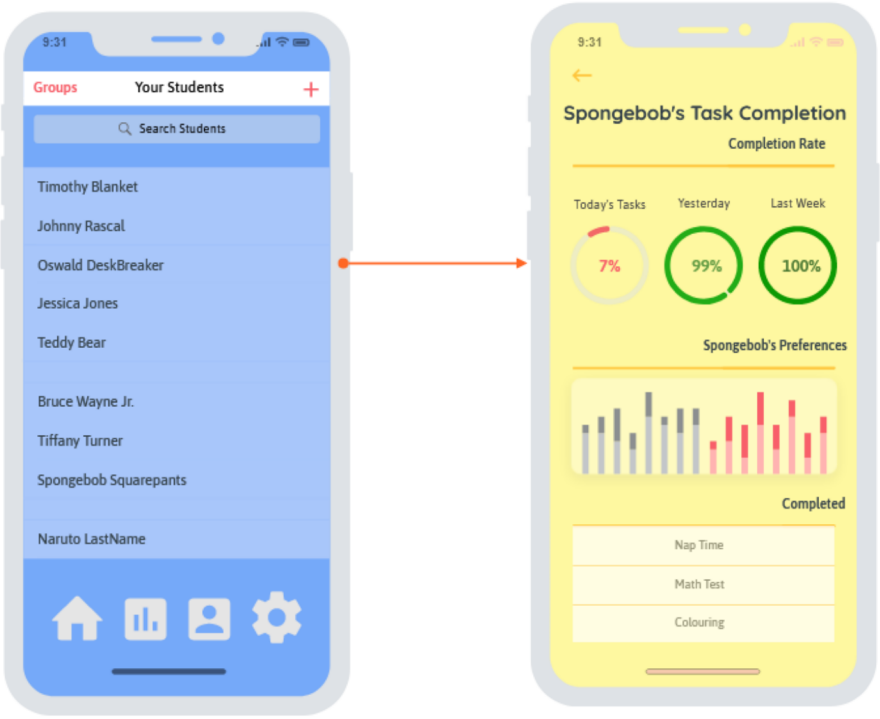

Figure 2. The mobile design featuring an analytics page, presented to professional contacts in education. Changes from earlier versions included the addition of a home bar for the Student List (left), as well as a red-yellow-green colour palette for the analytics.

When the team aggregated the responses received from teachers for each of the feature ideas, the analytics page was the clear favourite. When accounting for time

management and market viability, the analytics page was identified as the core of the application. Thus, the final design, design criteria, and application navigation center around the student analytics page.

Final Design

The application's final design met both the survey and time management criteria the team established and was deemed to have the highest chance of successfully meeting the design criteria. There were two groups of design criteria established, one for the frontend development and one for the backend development.

The criteria established for the frontend development were as follows: that the application would be highly intuitive, that each element would complement or navigate cleanly to the student analytics page, and that the analytics would be colour-coded and readable. The team determined whether or not the design matched these criteria, and then later validated this with testers.

The mobile design presented in the survey to professional contacts was too time intensive to create in four months, and so a web-based alternative was devised that could be shrunken to operate on mobile as well. In order to match the first criteria for the frontend navigation to be intuitive, two key elements were designed for the application. The key elements of this application were the top-bar navigation menu and the web pages. The web pages were the login, course list, student list, and student analytics page. Each page was then evaluated based upon the three frontend criteria first using a mock-up wireframe of the final design, and then the implemented final design.

The final design was implemented using Angular as a foundation for routing and page navigation, and Bootstrap as a design library to provide a CSS foundation. A central stylesheet was used for the entire application that provided a default font and colour scheme to ensure that the application's pages were in alignment. Chart.js was also used for the analytics page to provide chart designs that could animate programmatically, and to make the analytics page's design stand out from the other pages as it was the core feature.

The navigation bar was designed such that the homepage button routed back to the course list and the user could see their profile picture in the top-right, as well as change the theme and log out in a dropdown that appeared when they hovered over their profile picture. [Fig. 3] This was done with a combination of Angular's activated routing and local storage.

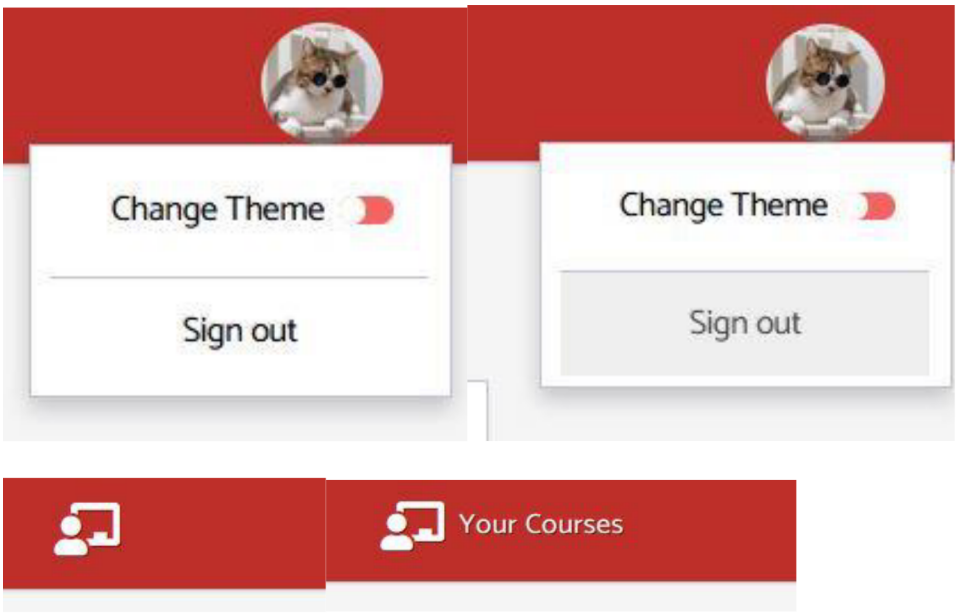

Figure 3. The final navigation bar's top-right dropdown (top-left), top-right dropdown with the cursor hovering on the "Sign out" button (top-right), the home page button (bottom-left), and the home page button with the user's cursor hovering over it (bottom-right). The navigation bar in dark mode can be found in Appendix A.

The login page was the first design that was created. In order to ensure that the page was intuitive, this page followed the standard login page design that includes: a login header, username and password text field, and a submit button. [Fig. 3] The only non-standard additions to this page were three buttons used to identify the user as a teacher, parent, or student. [Fig. 4] Once the submit button was activated, the page then successfully navigated to the course list using Angular's activated routing, which then linked directly to either the student list or analytics page dependent upon if the user was a teacher, student, or parent.

Figure 4. A Balsamiq wireframe of the final design for the login page.

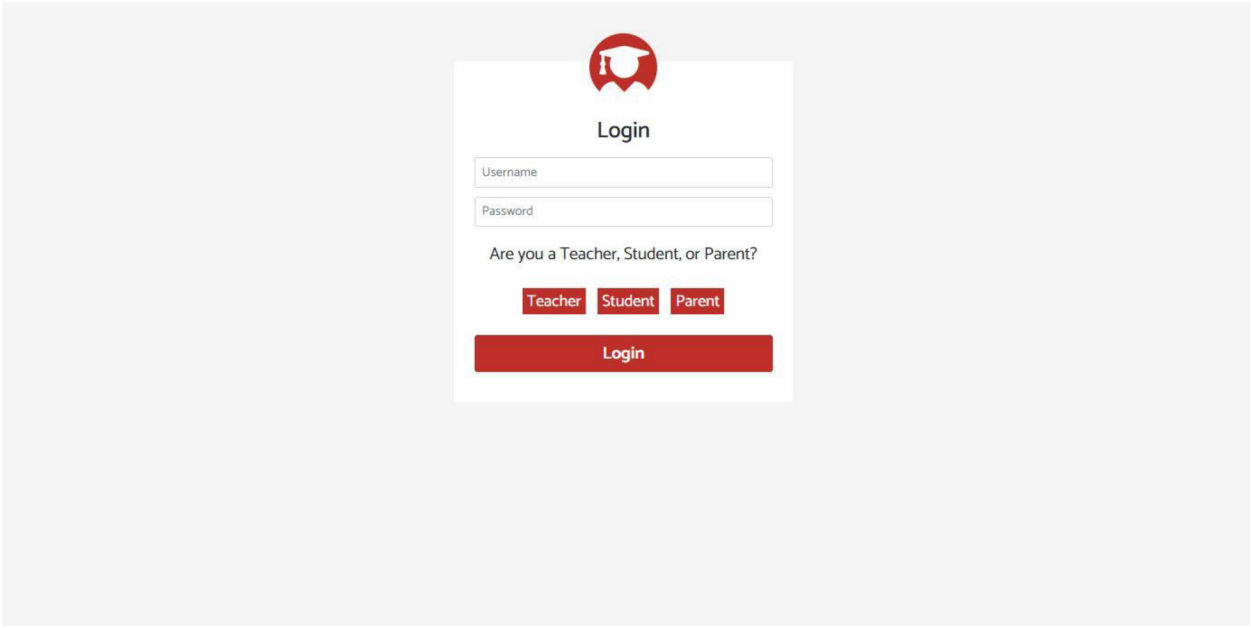

Figure 5. A screenshot of the final login page design in light mode. The login page's dark mode design can be found in Appendix A.

The course list was the second page that was designed. The course list followed a top-down, left-right model, wherein courses were listed on top of one another in two columns for easy readability. [Fig. 5-6] This list matched the navigation criteria as there were two routing options implemented that both led to the analytics page. The first was for students, who by clicking on a course, were routed to their own analytics page. The second was for teachers and parents, who would then choose a student from a student list and then be routed to the student's analytics page using Angular's activated routing. The course and student information were passed to the student list or analytics page as routing parameters that were then extracted to programmatically display the appropriate information.

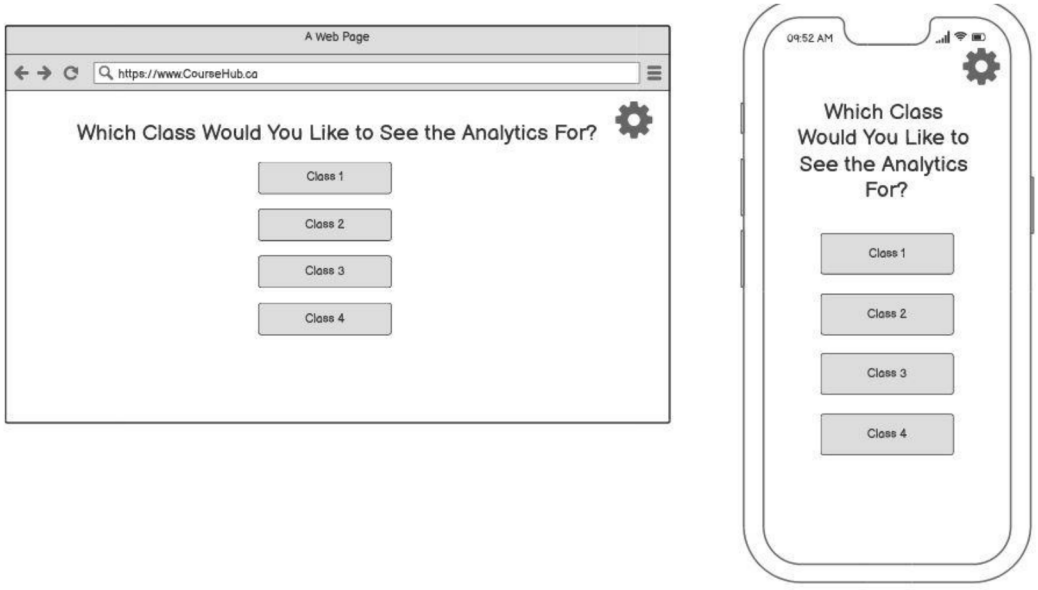

Figure 6. A Balsamiq wireframe of the final design for the class list page. This design was later altered to not include the text prompt above the class buttons.

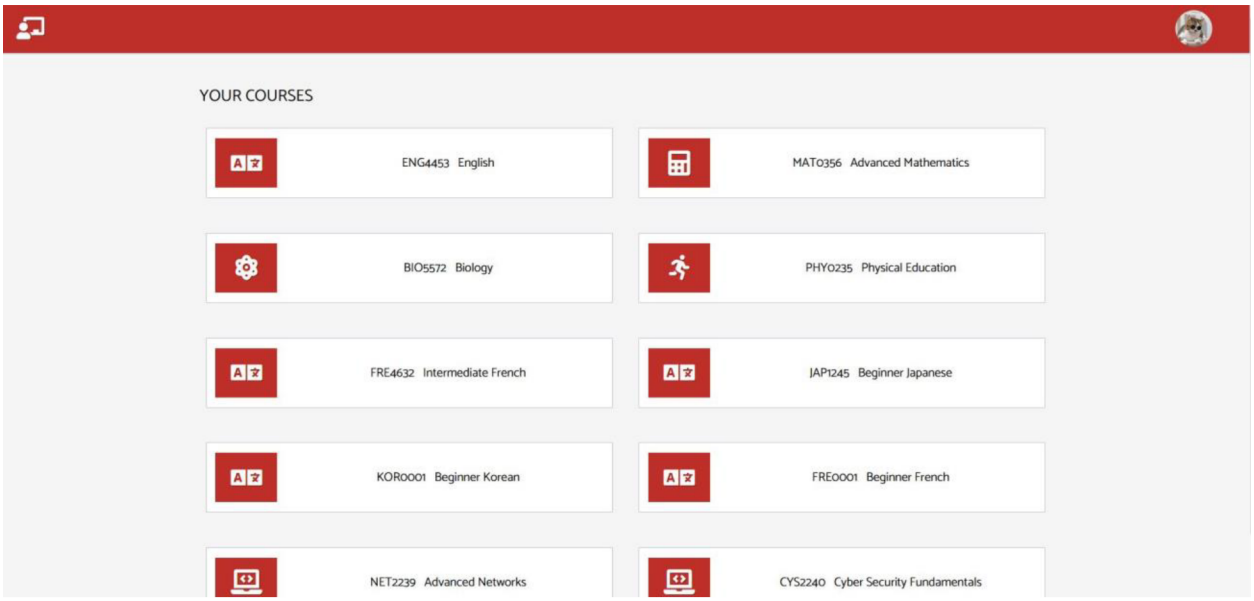

Figure 7. A screenshot of the Courses page in light mode. The course list's dark mode design can be found in Appendix A.

The student list was designed to follow a table layout, wherein student numbers, names, and grades were listed in a table. [Fig. 7] When the user hovered over a student row the student was highlighted and clicking on the student would navigate the user to that student's analytics page. The page matched the colour-coded analytics criteria by displaying the student's current grade with a surrounding ring of colour. [Fig. 8] The colour-coding is explained more in detail in the analytics page design.

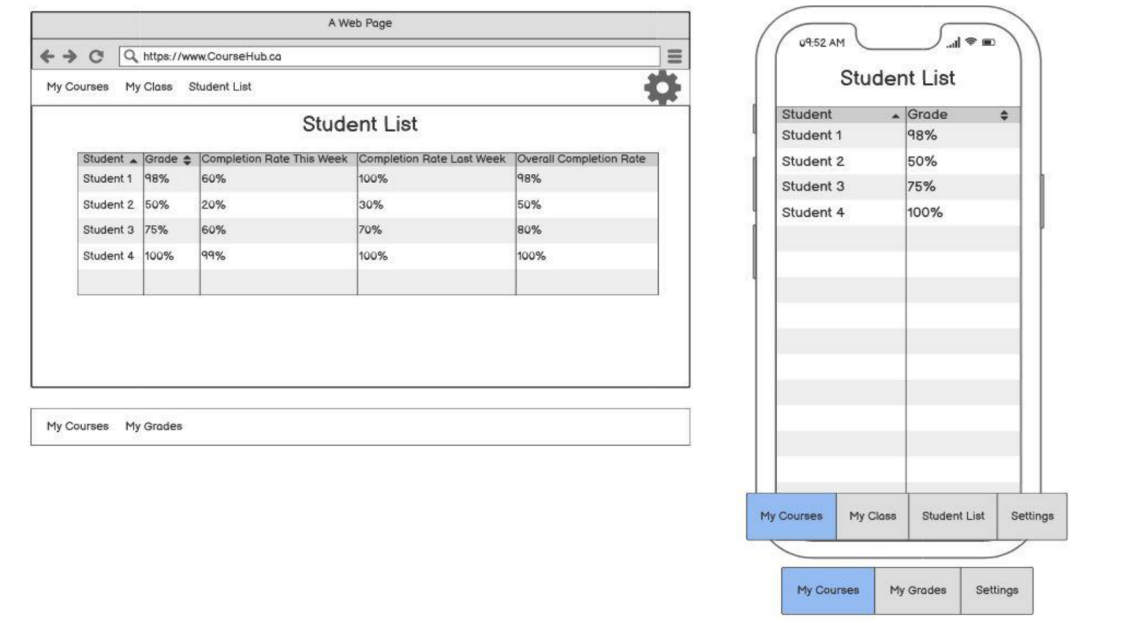

Figure 8. A Balsamiq wireframe of the final design for the student list.

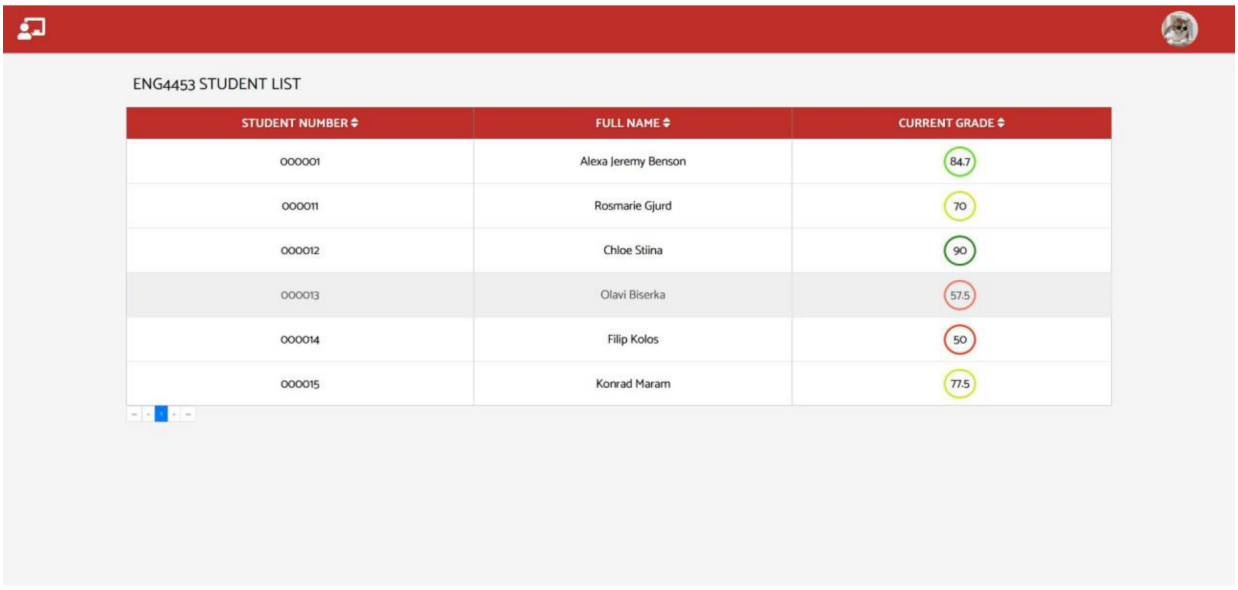

Figure 9. A screenshot of the Student List page in light mode with one student highlighted by hovering over them with the cursor. The student list's dark mode design can be found in Appendix A.

The final page design was for the analytics page. This page was the most challenging to make intuitive. It followed a top-down left-right layout, with student analytic graphics placed one on top of another or side-by-side in two columns. [Fig. 9] The overall grade was displayed in the first quadrant, and the task completion over the past week was displayed in the second with two different graphics. [Fig. 10]

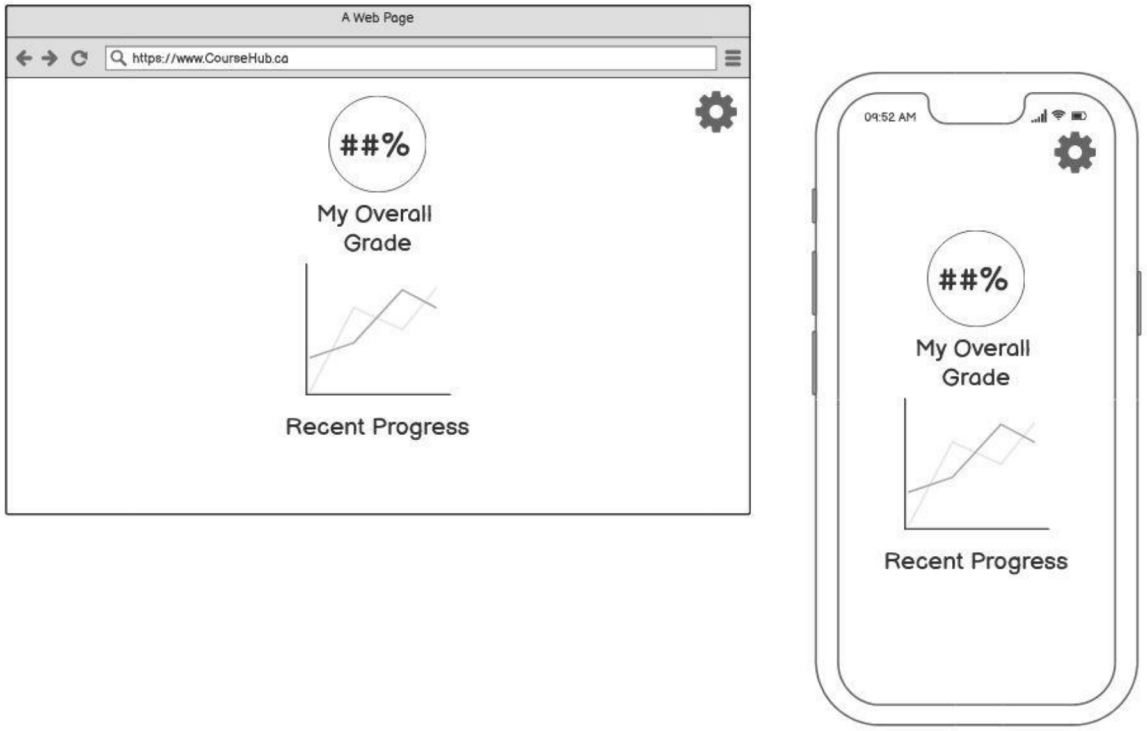

Figure 10. A Balsamiq wireframe of the final design for the student analytics page. This page was altered to include the student's first name and last name and was colour-coded to match the progress of the student.

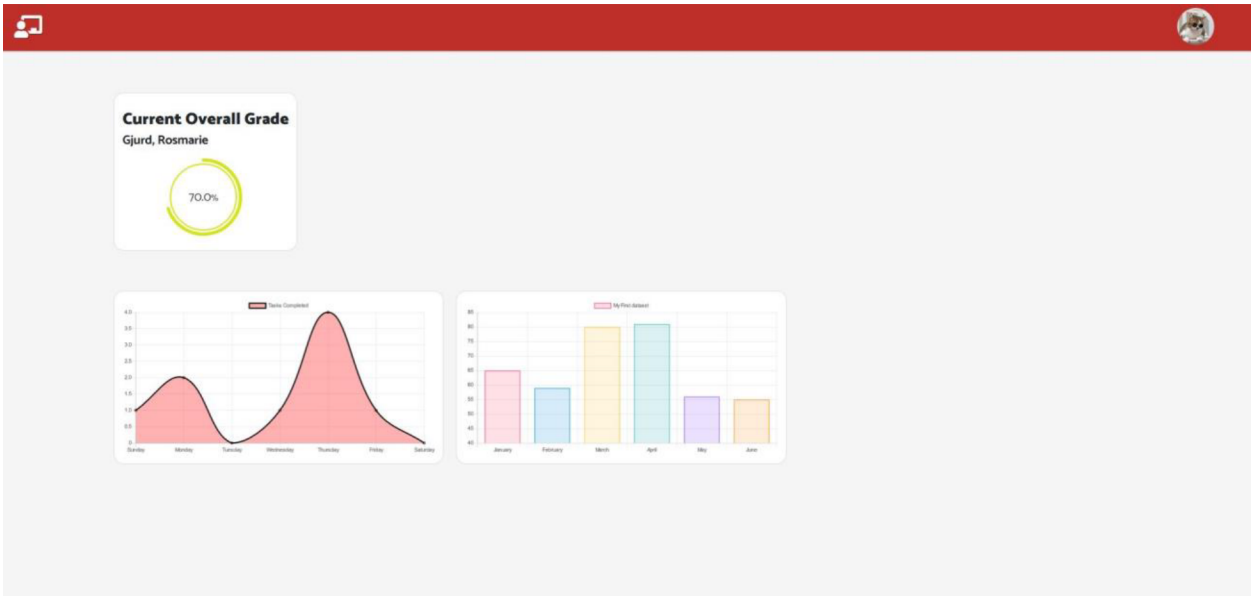

Figure 11. A screenshot of the Analytics page in light mode. The analytics page's dark mode design can be found in Appendix A.

Each analytic was colour-coded. Readability was improved by increasing the size and bolding the numbers, and by making the colours correspond to whether the grade was high or low. [Fig. 11] The redder the colour, the closer the student was to failing. The darker green the colour, the closer the student was to one hundred percent. These colours were programmed to change as the grade changes.

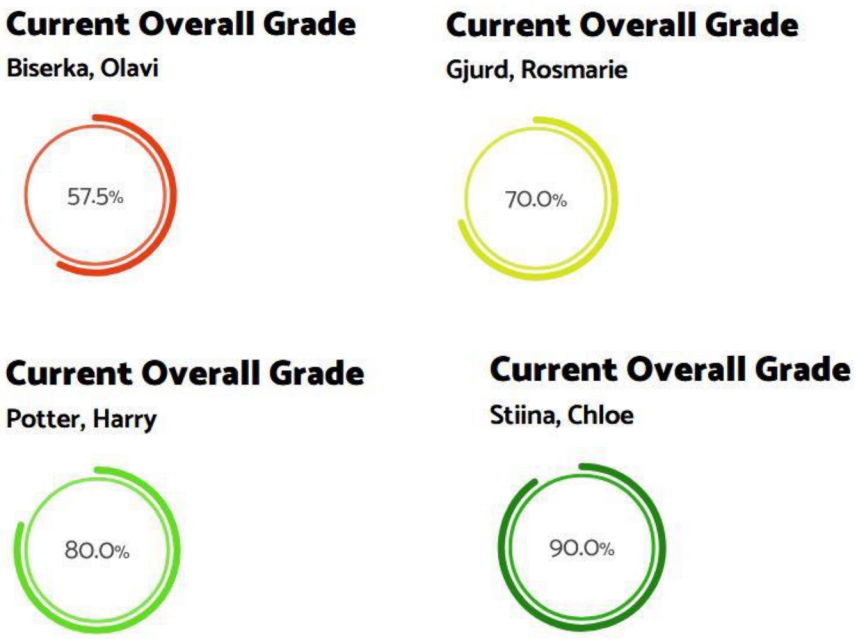

Figure 12. Examples of colour-coded grades in the Analytics page.

There were two key criteria for the backend design. The first backend design criteria were for Google Classroom, Brightspace, and Canvas to be successfully mocked and funneled into the different pages. The second backend criteria were for the codebase to follow a standardized folder structure for easier developer readability. In regard to the first criteria, a Class UML design was used to provide an early framework for mocking data. [Fig. 12]

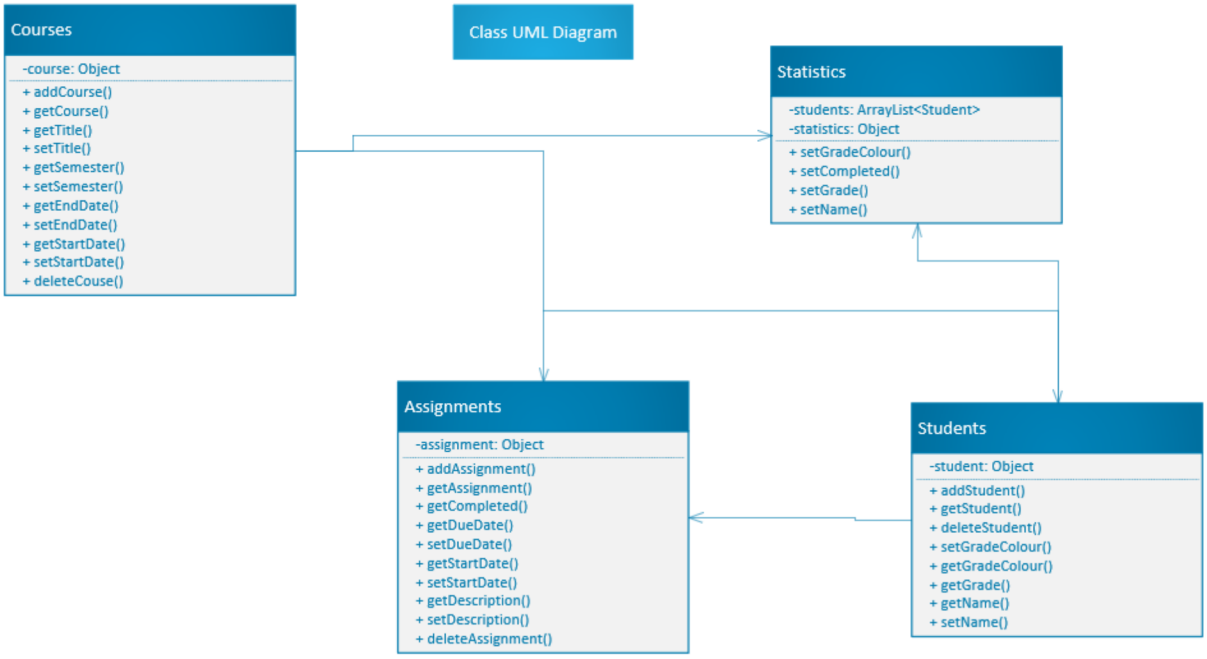

Figure 13. This is a Class UML Diagram for the application. It shows the members and functions in the code. It was developed using Microsoft Visio 2016.

The backend of the design gathered student statistics using mock LMS APIs. These mock APIs gathered information on students and teachers based upon the assignments they handed in and their grades and were then made accessible to the application through individual and combination pipelines that converted data types as needed. Each of the mocked LMS's was researched during the planning stage of the application's development and deduced to have sufficient documentation to proceed with the final design concept.

The second criteria of standardizing folder structure were a criteria that was continually evaluated during execution so that the codebase would not become too disorganized. The folder structure was recreated prior to mock data ingestion to reinforce these

criteria. That structure was based upon extensive research into professional folder structures when using Angular frameworks. [Fig. 13]

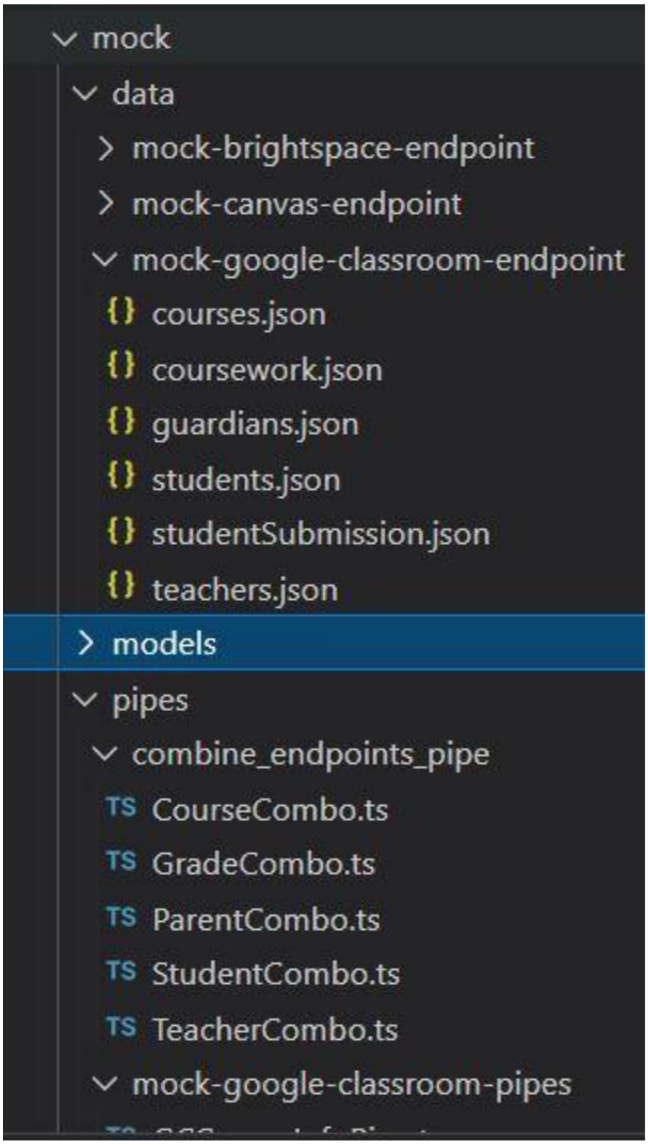

Figure 14. This is an overview of the folder structure used for mocking data and combining data endpoints in the backend.

The team evaluated the final backend and frontend designs based upon the criteria set and determined that they had a high likelihood of successfully meeting those criteria. Student tracking remained at the core of the design, and each feature or page was

introduced to complement student tracking and navigate to the student analytics page successfully. The backend was designed to match the mock ingestion criteria and then cleaned continuously to match the folder design criteria. These designs were then moved forward for implementation.

Methodology

Frontend Development

The frontend development used a combination of NodeJS, npm, Angular, and built-in browser development environments. Prior to coding, the team started this project by making user interface wireframes using Balsamiq, a web-based mock-up tool used for designing applications. The team used Jira software as an agile management tool, which was used to visualize the project's progress, task delegation, and outcomes. When the team developed the frontend, a local environment was built using NodeJS and npm to see what the application would look like in production.

Balsamiq software was the best UI design tool for the team because of its user-friendliness and the fact that the team could quickly make mock-ups using the samples the software provided. Balsamiq allowed the team to create widgets with a drag-and-drop tool. It allowed the team to go through built-in prototypes efficiently.

The team benefited from using Jira software as a development tool because it could be used to create an Agile scrum board which was responsive to changes during the

development process. Jira's software mapped to customized work processes that reflected both clients' requests for UI changes once testing began, and the developer's adjustments during ongoing development.

As the team implemented the foundational frontend code using Git and GitHub, team members were able to see the frontend design using npm commands in the command line. Npm was very useful for frontend development because the developers could code using any IDE while simultaneously running the code on localhost to display any changes before the code was deployed to production.

Each team member was responsible for different tasks and pages within the UI frontend, and through the first four-week Sprint the team was able to reproduce the wireframes from Balsamiq in each team member's local environment. For the final sprint, the frontend was improved to be more user-friendly and readable by taking on individual pages and improving them one by one.

The development process for each page followed similar steps. The page was first routed to through the main application, then the HTML layout was set, and then the CSS was completed to stylize the page. Originally, the data displayed was static in the pages. Later on, it was replaced with mock data. Coordination during page creation was made easy with a combination of Git, GitHub, and Jira, and development was successful by coordinating use of npm, NodeJS, and Visual Studio Code.

Mock Data Ingestion

Following the successful completion of the user interface using static data, application functionality was developed using mock data ingestion. This was accomplished using data hard-coded into the application in a mock store of JSON endpoints. Individual endpoints were created for each of Google Classroom, Brightspace, and Canvas. This mock store data was then piped into the application in order to populate user, student, and course information so that the application could be properly unit and integration tested.

The first step to integrating mock data ingestion was creating global teacher, student, and parent variables that altered the web page progression from the login page. From this altered progression, different data was available dependent upon the type of client accessing it. Teachers had access to all mock student data and their relevant course data, while students and parents had access to their relevant data only.

In order to test full functionality, there were three mock teacher accounts created, three mock parent accounts, and many mock student accounts. The three teacher accounts were one teacher without any students, one with multiple students across multiple classes, and one with students in only one class. The three parent accounts were one without any student children, one with a single student child, and one with multiple student children. The student accounts were some students without any classes, some with one, and some with several. Students were also given assignments with some incomplete and some complete grade metrics.

With the static data and preliminary static data testing complete, the next stage was to mock three-pronged data ingestion from Google Classroom, Brightspace, and Canvas, as the application would be integrating data from these three sources into one. To do this, the team had to mock the three injection sites that funnelled into one combination pipeline.

The original JSON object was converted into a JSON object using individual endpoint pipelines, then converted to a standardized object, and finally combined with the objects from the other endpoints. These combined Student, Teacher, Parent, Grade, and Assignment objects were then piped into the pages through the page's own programming logic, which would use a combination of grade calculators and a grade colour lookup table to mathematically calculate the proper values to display. These values were calculated using various String to Integer conversions, parsers, and formulas.

In the teacher and parent's case, their student list was populated such that teachers had access to all relevant student data, and parents saw a list of their children. In the student's case, they had access only to their courses and individual analytics pages. Any attempts to route to the student list would bring them back to their course list. The data piped into these pages for parents, teachers, and students was then adjusted and tested using various different hard-coded values, in order to test edge cases.

The final stage of mock data ingestion was unit testing. Each use case was tested using different unit tests. In order to ensure that the team did not miss any vital unit tests, the team designed a graphic that each team member could follow while unit testing so that

no use case would be missed. Once the unit testing was complete, the next stage was real-time LMS API injection using the funnels created with the mock data.

LMS API Development

Due to time constraints this stage of development was not reached during the January-April 2021 technical development term. However, in order to properly mock data ingestion, it was necessary to map out this stage of the development process. Understanding how real data ingestion worked was fundamental to mocking it accurately.

The mock endpoints acted the same as API endpoints in the application, with a few key differences. The first key difference was the login page. With mock endpoints, security and login was done by locally storing the username and password. All logins would go directly through LMS security endpoints once the APIs were connected. The benefit of this was that login and privacy would occur on the LMS side, thus preventing the application from having any security breaches of its own. The LMS APIs used token keys and end-to-end authentication as well as the users' login information to allow the application access to endpoints. This would guarantee the security of the login process and require reworking of the login page in the future.

The second key difference was that there would be no actual JSON data store within the application when linked to an actual LMS endpoint. That JSON would be piped into the individual conversion pipelines directly from the LMS endpoint resulting in no need to store data in the application.

The application would ping REST endpoints pertaining to student data, teacher data, and course data to populate tables and analytics across the application. With schools using many different LMS's, the team ensured through mock data testing that the application would work with many different API endpoints. These endpoints included D2L's such as Blackboard and Brightspace, as well as Google Classroom and Canvas. The information would be received through these endpoints, aggregated, and then displayed within the application.

With the combination of frontend development and API endpoints, the application would successfully obtain teacher, student, and class information dependent upon the type of account of the individual accessing it. The LMS APIs would make this a smooth process. Since the backend work was completed using the mock data injection, this portion of the development process was scheduled to continue following the January-April 2021 study term.

Testing, Results, Analysis

The application went through three testing stages: alpha testing, unit and integration testing, and finally outside developer testing. The time constraints prevented a full beta testing gambit, however the results of the three completed testing stages were promising. No application overhauls or feature retractions were deemed necessary once testing was completed, and the application was determined to be ready for beta testing.

Alpha testing was completed by the application's developers through raw clicking, typing, and stress testing of the application frontend during development. Alpha testing was chosen as the first method due to the fact that the developers on the application's team were familiar with different testing frameworks and methods. This would prevent major application breaking errors from being present for when outside sources were brought in for testing.

The primary methods used for alpha testing were the built-in developer features of Firefox and Chrome, npm, and Visual Studio Code's debugging features. All methods were implemented with the idea of attempting to break or circumvent application logic from the user's end and ensure that the application caught such use cases and continued to function as expected. As the team stress tested the application, the team began integrating unit and integration tests to programmatically test the robustness of the application. These tests proved successful, and thus the application moved on to alpha testing.

Outside developer testing was done using fellow student developers, close friends and family. They were given the criteria outlined in the design section and asked to rate the application based upon the criteria. These testers were able to test using a web client, wherein they navigated the application as students, teachers, and parents, and then tested the usability and robustness of the application. Testers were given logins and real environments to test the API interactions, ensuring the received data properly reflected the mock LMS data.

The feedback received from testers was categorized using the design criteria. For the first frontend criteria, the testers found that the application was intuitive. There was no feedback pertaining to routing confusion or navigation issues. While the testers found the application intuitive, two testers found the task progress charts unclear. They reported that the charts needed more labels and better colour-coding.

The second criteria received no feedback, as testers did not notice any issues navigating to the analytics page. As a result, the criteria were deemed to be successfully met. The third criteria were met successfully with the grade analytics, but not with the task progress charts. Thus, it was deemed to have been met in one out of two cases.

The backend criteria were not accessed by the outside developers as they did not have access to the team's codebase. Instead, alpha, integration, and unit testing were used in order to determine whether or not the two backend criteria were successfully met. For the second criteria, the developers of the team felt that the folder structure was easy to navigate and did not contain stray or mismatched file structures. Thus, the criteria were deemed to have been successfully met.

Two tests were used in order to check whether the first criteria of successfully integrating mock endpoints was met. These tests were unit tests and integration tests. Angular's built-in testing components were used for unit testing, and a testing component was assigned to each individual page in order to test it. Each page was navigated by a unit test that incorporated random clicks and scrolls, and also tested for when there was no data or large amounts of data present.

The overall website integration was tested manually with many different use cases. Each of the following tests was performed with parent, student, and teacher routing. The first was a situation where the user had no courses, students, or analytics to display. The second was a situation where the user attempted to route to invalid pages using invalid parameters, an invalid URL, or by injecting invalid data using the browser's built-in developer tools. These tests were all successful, and appropriate error pages were created to account for each edge case circumstance.

Overall, feedback from alpha, unit, and integration testing, as well as outside sources was positive, and showed promise for beta testing with a broader audience of students, parents, and teachers. Prior to completing beta testing, the analytics page charts feedback on colour-coding and overall readability would need to be addressed. Given that the analytics page was the core page of the design, its readability and design was the most important element to perfect.

Conclusions and Recommendations

The application detailed in this report gathered data from mock LMS endpoints and then displayed that data in a course list, student list, and analytics page for parent, teacher, and student users. The team's initial goal was to develop a solution to the challenges elementary-aged online learners and their teachers and parents were experiencing during remote learning. After speaking with teachers, researching, and reviewing survey

results, the team narrowed the scope of the application's goal to provide a readable, user-friendly layout that would improve student progress tracking.

Design criteria were established such that readability and understandable analytics would be the indicators of a successful application development process. The UI was determined to be readable and user-friendly, thus meeting the first frontend design criteria. Navigation between pages was easy to follow, and always ended with the student analytics thus indicating that the student analytics page was the central component of the application. The third frontend design criteria were met as the colour-coding was understandable to testers and the colours chosen were intuitive. The only part of the application that did not meet the intuition and readability criteria was the task tracking chart, which needed more labels and more intuitive colour-coding.

The two backend criteria were met successfully. Mock data ingestion was successfully implemented, and a clean, standardized folder structure was maintained during the backend development process. No stray files or unused code splices were retained for more than a week, and each model was routinely cleaned to match the specifications needed for each mock endpoint.

The end result of meeting these criteria was a fully functional, easy to follow grade analytics application to display grades and course progress in a trouble-free manner for users of all skill levels to understand. Beyond meeting criteria, there were two main challenges during application development and two recommendations identified for future development. Both challenges relate to the development process, and both recommendations involve increasing the scope of the application.

The two challenges were remote communication and time restrictions. Remote communication eliminated the ability to have in-person meetings, and also made communication asynchronous. This made design discussions more time-intensive and proved a challenge when coordinating tasks. The second challenge was time restrictions, which made it harder to implement more complicated features by April 2021. The time restrictions also influenced the outside source testing, as testers were testing the application while it was still in early development.

The two recommendations are feature expansion and expanding the target market to higher learning. The team was restricted to a four-month development period and therefore was unable to include certain useful features. Two examples of this are a reward system for successful task completion and personalized colour themes. Due to the time restrictions, the team was also unable to implement LMS APIs without mock data, and it is recommended that real LMS endpoints from Google Classroom, Brightspace, and Canvas should be integrated in the future. The second recommendation is that the application should also be expanded to include higher education, as the application could also be useful for more than elementary schools.

Overall, the application development was an exceptional learning experience, and the final application was successful based upon the design criteria. The team learned how to mock data ingestion obtained from LMS APIs, as well as improved their Angular development skills. The team is extremely thankful to the individuals who contributed to the development and marketing process including Laura McHugh, Howard Rosenblum, Margaret O’Brien, and the senior Algonquin Business Marketing teams.

Glossary

Abbreviations

API - Application Programming Interface

CSS - Cascading Style Sheets

D2L - Desire2Learn

IDE - Integrated Development Environment

JSON - JavaScript Object Notation

LMS - Learning Management System

UI - User Interface

Terms

Agile - A framework through which a team of developers can coordinate, collaborate, and self-organize the development of an application.

Apple License Pricing - Apple licenses are used in order to develop applications for Apple devices.

Application Programming Interface - Software that acts as a middleman between its mother application and another application. It allows for data to be extracted and shared without influencing the data integrity of the mother application.

Alpha Tester - Testing performed by users that are not the end users, usually within the organization developing the application.

Analytics - An analysis or integration of data that usually comes in the form of measurements or statistics about a particular topic.

Angular - A web application framework created by Google.

Balsamiq - A web application used to create user interface wireframes for web and mobile applications.

Cascading Style Sheets - Style sheets that make up how the application is presented to the user.

Class Diagram - A means of designing object-oriented modeling for an application.

Codebase - The complete body of code for the application.

Desire2Learn - The overarching company that created Brightspace and Blackboard.

Endpoint - A communication channel for a particular part of a software that allows an application to connect to that part of the software.

Git - A source control application to help with version managing and branching off of a codebase.

GitHub - The repository where the codebase can be found.

Integrated Development Environment - A software that provides all the facilities necessary for software development, including a source code editor, the ability to build and compile, and a debugger.

JavaScript - A programming language.

JavaScript Object Notation - A way of formatting data so that it can be sent and received between endpoints.

Learning Management System - Applications that are used to manage virtual classroom environments through assignment uploading, grading, discussion boards, and more.

Mock Data Ingestion - Faking information that is funnelled into the application.

Scrum - An agile framework that emphasizes software development as the core of the development process.

Scrum Board - Where the tasks for a sprint and backlog items are created and visualized.

Sprint - A time-boxed period wherein a scrum team will complete an amount of work.

TypeScript - A programming language that integrates and builds upon JavaScript.

Use Case - A potential scenario in which an application is used.

User Interface - The frontend matter of an application. The application that the user sees is the user interface, as it is where the user can interface with the application.

Web Application Framework - A software that supports the development of a web application.

Bibliography

[1] Unesco. (2020). *Education: From disruption to recovery*. Unesco. [Online]. Available: https://en.unesco.org/covid19/educationresponse [Accessed: Mar. 12, 2021]

[2] S. Martin (2020, Aug. 11). *How Much Does It Cost To Make A Educational Mobile App in 2020*. Medium. [Online]. Available at: https://medium.com/flutter-community/how-much-does-it-cost-to-make-a-educational-mobile-app-in-2020-27f93cf3cc83 [Accessed: Mar. 13, 2021].

[3] J. Heyck-Williams (2020, Apr. 8). *Connection, Core Content, and Creativity: Three Values for Distance Learning.* NextGenLearning. [Online]. Available at: https://www.nextgenlearning.org/articles/three-values-distance-learning-connection-content-creativity [Accessed: Mar. 12, 2021].

[4] Apple. (2021). *Purchase and activation.* Apple. [Online]. Available at: https://developer.apple.com/support/purchase-activation/#:~:text=Pricing,in%20local%20currency%20where%20available.[Accessed: Apr. 4, 2021].

[5] VectorStock. (2021). *School Classroom Scene Icon Stock Illustration.* VectorStock. [Online]. Available at: https://www.vectorstock.com/royalty-free-vector/school-classroom-scene-icon-vector-23474840 [Accessed: Apr. 3, 2021].

Appendices

Appendix A

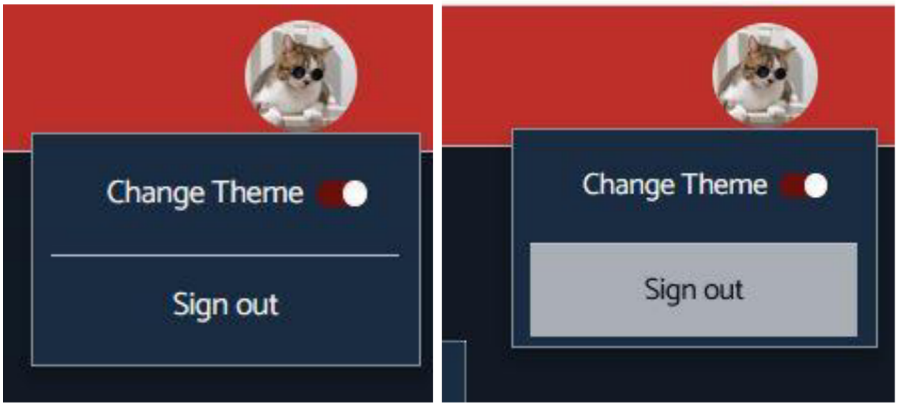

Figure 15. The top navigation dropdown in dark mode.

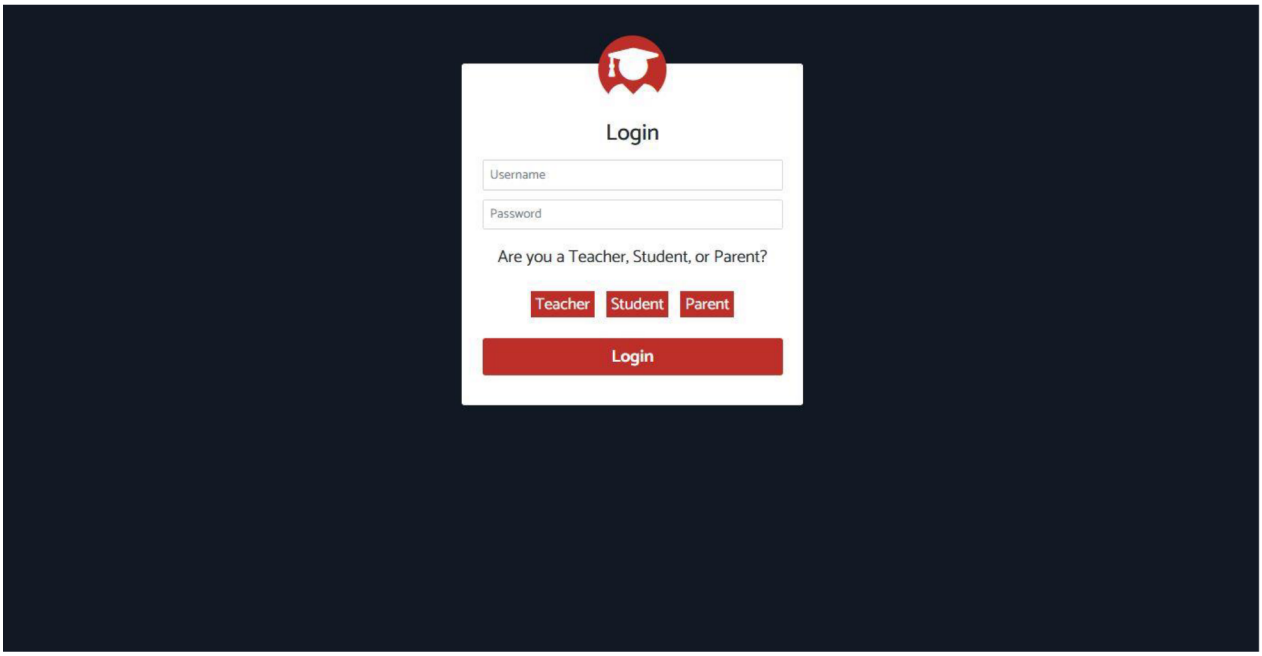

Figure 16. The login page in dark mode.

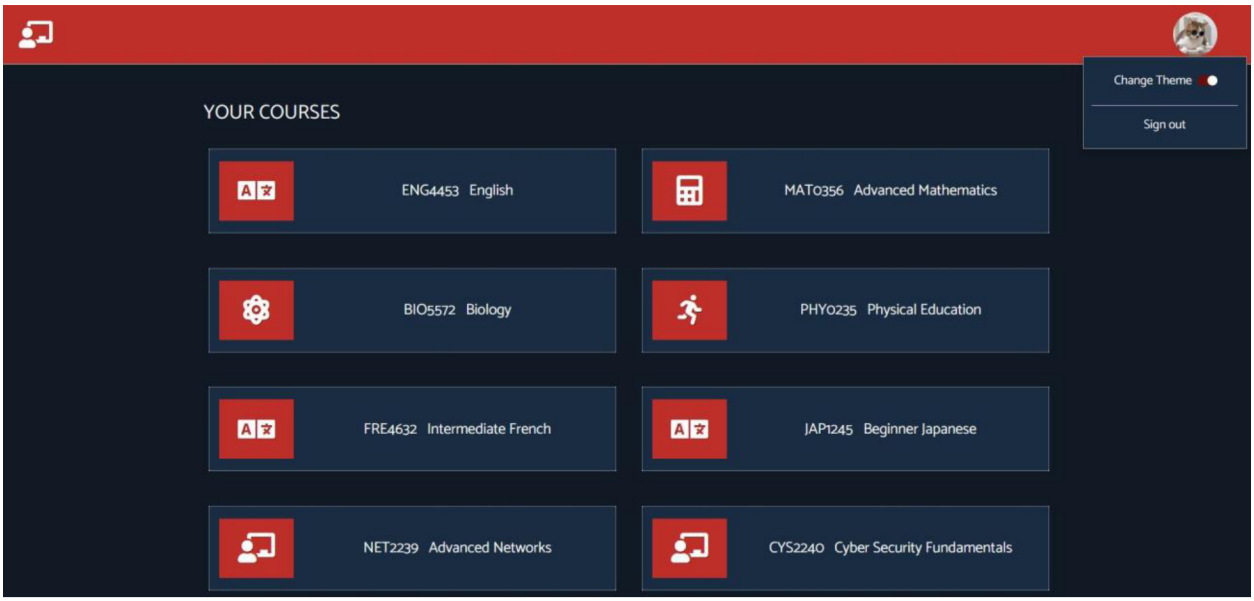

Figure 17. The course list in dark mode.

Figure 18. The student list in dark mode.

Figure 19. The student analytics page in dark mode.

Rubric (marked out of 100)

	0 / 10 to 4.9/10	5/ 10 to 5.9 / 10	6 / 10 to 6.9 / 10	7 / 10 to 7.9 / 10	8 / 10 to 10 / 10
Professionalism *Each element is worth one mark, except for the letter of transmittal (worth seven marks) and IEEE reference page (worth two marks)* **25**	1. Overall report appearance **(2 marks)** 2. Letter of Transmittal **(7 marks)** 3. Cover page **(1 mark)** 4. Title page **(1 mark)** 5. Declaration of Sole Authorship **(1 mark)** 6. Table of Contents **(1 mark)** 7. List of Tables and Figures **(1 mark)**			8. Illustrations / Appendices have captions and references **(4 marks)** 9. Illustrations / Appendices are linked to report content **(2 marks)** 10. Reference page is in IEEE **(2 marks)** **11.** Glossary **(1 mark)** 12. Report is in the right order **(1 mark)** 13. Rubric is provided **(1 mark)**	
Language (10 marks) Use of correct language conventions (grammar, spelling, punctuation), tone and style	Insufficient evidence for assessment	Uses language with limited accuracy; errors impede comprehension	Uses language with some accuracy; errors sometimes impede comprehension	Uses language with considerable accuracy; errors do not affect meaning	Uses language accurately all or most of the time, to enhance meaning.
Summary (10 marks) Stand-alone document that reflects the content of the report (executive summary)	Insufficient evidence for assessment	Would require a lot of additional reading of the actual report	Would require some additional reading of the actual report	Would require little additional reading of the actual report	Is a stand-alone document that accurately reflects the actual report
Introduction (10 marks) Contains all elements from the format discussed in-class, captures interest, has good paragraph flow	Insufficient evidence for assessment	Missing many elements, lacks structure, weak opening, fails to catch attention	Missing some elements, some structure problems, somewhat engages reader	Ranges from missing one element, to all included. Structure and engagement are generally good, some suggestions	All elements included, structure and engagement are excellent, few to no minor suggestions for improvement
Background (15 marks) Contains all elements from the provided guidelines. Should provide context for the reader and a thorough examination of the design process and final design presentation. Well-organized, with clear detail.	Insufficient evidence for assessment	Work has little structure and is difficult to follow, either missing content and / or extremely vague Limited detail – lacks clarity or depth comprehension Reader provided with little context for report: how a design was inspired / created / finalized	Work has some structure and is somewhat difficult to follow, missing some content and / or somewhat vague Some detail – needs more clarity and depth Reader provided with some context for report: how a design was inspired / created / finalized	Work has considerable structure. Errors do not affect readability, all elements included, all are generally well-covered. Detail is provided and is understandable for reader Reader provided with considerable context for report: how a design was inspired / created / finalized	Work demonstrates an exceptional ability to structure and organize information. Few suggestions for improvement. Use of detail is exceptional, few suggestions for improvement. Reader provided with exceptional context for report: how a design was inspired / created / finalized
Mix Design/Methodology (10 marks) How was the design executed? Should be a step-by-step process description that guides a reader through building / coding / testing. Well-organized, with clear detail.	Insufficient evidence for assessment	Work has little structure and is difficult to follow, either missing content and / or extremely vague Limited detail – lacks clarity or depth comprehension	Work has some structure and is somewhat difficult to follow, missing some content and / or somewhat vague Some detail – needs more clarity and depth	Work has considerable structure. Errors do not affect readability, all elements included, all are generally well-covered. Detail is provided and is understandable for reader	Work demonstrates an exceptional ability to structure and organize information. Few suggestions for improvement. Use of detail is exceptional, few suggestions for improvement.

		Methodology is poor and it would be hard to re-create the process	Methodology is somewhat done and could be somewhat re-produced	Methodology is considerably done and could be re-produced	Methodology is exceptionally done and could be easily re-produced
Testing, Results and Analysis (10 marks) What were the results? Present them clearly and logically. Make sure that the reader understands what the results are, what they mean / how to interpret them, and how they affect the goal of the project (success or not). Well-organized, with clear detail.	Insufficient evidence for assessment	Work has little structure and is difficult to follow, either missing content and / or extremely vague Limited detail – lacks clarity or depth comprehension Results are unclear and lack explanation. It's hard to interpret the results and unclear what they mean for the project.	Work has some structure and is somewhat difficult to follow, missing some content and / or somewhat vague Some detail – needs more clarity and depth Results are somewhat well-presented. They are somewhat explained. The reader can somewhat interpret them for himself. It's somewhat listed what the results mean for the project.	Work has considerable structure. Errors do not affect readability, all are generally well-covered. Detail is provided and is understandable for reader Results are considerably well-presented. They are considerably well-explained. The reader can interpret them for himself. It's listed what the results mean for the project.	Work demonstrates an exceptional ability to structure and organize information. Few suggestions for improvement. Use of detail is exceptional, few suggestions for improvement. Results are exceptionally well-presented. They are exceptionally well-explained. The reader can easily interpret them for himself. It's clear what the results mean for the project.
Conclusions and Recommendations (10 marks) Required elements: What did the project set out to do? What was actually achieved? Successes / limitations – how could they project have gone differently? Show reflection and understanding. Provide concrete recommendation / next steps with clear course of action.	Insufficient evidence for assessment	Work has little structure and is difficult to follow, either missing content and / or extremely vague Limited detail – lacks clarity or depth comprehension Limited presentation of conclusion and recommendations.	Work has some structure and is somewhat difficult to follow, missing one to two elements and / or somewhat vague Some detail – needs more clarity and depth Some presentation of conclusion and recommendations. Action and reflection provided but unclear.	Work has considerable structure. Errors do not affect readability, all elements included, all are generally well-covered. Detail is provided and is understandable for reader Considerable presentation of conclusion and recommendations. Action and reflection.	Work demonstrates an exceptional ability to structure and organize information. Few suggestions for improvement. Use of detail is exceptional, few suggestions for improvement. Exceptional presentation of conclusion and recommendations. Clear action and reflection.