

Chamblee High School Extended Learning Time Optimization Project

Reid Bryant, Project Manager

Quinn Tobin, Software Developer - UI Lead

Tristan Brown-Mulry, Software Developer - Optimization Lead

Advised by Dr. Andrew Milne

Submitted April 27th, 2022

Kennesaw State University

Executive Summary

The following is a report on the design and documentation of a program meant to assist Chamblee High School. The definition of the problem, front and back-end development, analysis of the program, integration between programs, final results and conclusions contained within the report contain multitudes of information about the system and recommendations for the continued development of the program after the involvement of the Kennesaw Industrial Education Team.

The basic problem of the project is the organization of information flow. To solve problems with the basic solution to organization within the school, the team must develop a method to creating a software that will assign students and teachers within the new flexible learning period that Chamblee intends to integrate into its block schedule in the fall semester, 2022. In response to the problems with the baseline solution, the team developed a new flowchart that would alleviate the problems presented by it and found a method of testing the program quality to check if the final product was sufficient or if new tools would need to be recommended.

The front-end development was handled entirely through the preferred system, Coda.io, at the recommendation of the client. The development of the software was temporarily halted due to the legal ramifications behind student anonymity and the school's inability to share information, but the development of the software served to establish the future for a new software. The front-end accomplished the creation of a teacher and student view that accomplished most requirements of the interface. The back-end development accomplished the stated goals of the algorithm and servers but was not optimal because of the difficulties integrating with Coda.io and involved the use of SQL, MySQL, R-Script, Python, and Amazon Web Servers.

Integration proved to be the most intense challenge of the project. Maintaining the data structure and working with the suboptimal routines for data transfer in Coda.io made the process incredibly difficult. Employing the use of API, third-party servers, and Python after the initial solution recommended by the developers at Coda.io, Zapier, failed to be capable of the data transfer that was necessary.

After the analysis of the program revealed a less than optimal software quality, and that the program on review did not meet all requirements of the software due to limitations in Coda.io, the team is forced to make recommendations that involve the abandonment of Coda.io as a platform and strategies to rework the current back-end development and servers to create a smoother overall program.

Table of Contents

Executive Summary	2
Chapter 1: Initial Presentation and Assignment.....	6
1.1 Introduction:.....	6
1.2 Overview:.....	6
1.3 Objective:.....	6
1.4 Justification:.....	7
1.5 Project Background:.....	7
1.6 Problem Statement:.....	7
1.7 Baseline Flow Chart:.....	8
Chapter 2: Logistical Assignment.....	12
2.1 Responsibilities:.....	12
2.2 Gantt Chart and Schedule:	12
2.21 Initial and Preliminary Schedule.....	12
2.22 In Progress Schedule.....	13
2.23 Critical Design Schedule.....	14
2.24 Final Design Schedule	15
2.3 Budget:.....	15
Chapter 3: Literature Review:.....	17
3.1 Software Quality Literature, McCall’s Model:.....	17
3.2 Coda Literature	17
3.3 Backend System Design Literature.....	18
3.4 Flexible Learning Literature	18
Chapter 4: Problem Solving Approach:	20
4.1 Why DMAIC?.....	20
4.2 Requirements:	20
4.3 Proposed Flow Chart Solution:.....	21
4.4 Materials Required:.....	22
4.5 Resources Available:	22
Chapter 5: Creation in Coda	23
5.1 Existing templates regarding Month View + initial navigation elements.....	23

5.2 Student View + Mobile View	24
5.3 Teacher View	26
5.4 Teacher View Subpage	27
5.5 Data Importation	28
5.6 Data Display.....	28
5.7 Document Locking.....	29
5.8 Document Filtering	29
Chapter 6: Back End Development.....	31
6.1 Synthetic Data Generation	31
6.2 Assignment Algorithm.....	32
Chapter 7: McCall’s Analysis.....	38
7.1: Software Quality Factor.....	38
7.2: Correctness.....	38
7.3: Reliability.....	39
7.4: Efficiency.....	40
7.5: Integrity.....	40
Chapter 8: Results and Discussions:	43
Chapter 9: Conclusions:	46
References.....	47
Appendix A: Acknowledgements	49
Appendix B: Contact Information	50
Appendix C: Work Table.....	50
Appendix D: Reflections.....	51
Appendix E: Backend Code.....	53
Python	53
RStudio	55

Table of Figures

Figure 1: Overview of system requirements	6
Figure 2: Baseline flow chart.....	8
Figure 3: McCall Model [23].....	9
Figure 4: Product operation attribute chart [22]	10
Figure 5: IDR and PDR schedule.....	12

Figure 6: IPR schedule.....	13
Figure 7: CDR schedule.....	14
Figure 8: FDR schedule	15
Figure 9: Proposed solution flow chart.....	21
Figure 10: Calendar view.....	23
Figure 11: Calendar view 2.....	24
Figure 12: Student view	25
Figure 13: Mobile student view	26
Figure 14: Teacher view main page.....	27
Figure 15: Teacher view class subpage	27
Figure 16: Chart Example.....	28
Figure 17: Document locking	29
Figure 18: Student Assign.....	30
Figure 19: System data flow chart	31
Figure 20: Assignment algorithm variables.....	32
Figure 21: Assignment algorithm	33
Figure 22: Student preference distribution	33
Figure 23: Student preferences and weight function	34
Figure 24: Objective function value calculations	35
Figure 25: Student assignment distributions.....	35
Figure 26: Computation time for number of students in the model.....	36
Figure 27: Software Quality Factor	38
Figure 28: Correctness	38
Figure 29: Reliability	39
Figure 30: Efficiency [22].....	40
Figure 31: Integrity	41
Figure 32: Useability [22].....	41

Table of Tables

Table 1: Project budget.....	15
Table 2: Projected Implementation budget.....	15
Table 3: Computation time at different student base sizes	37
Table 4: Work division table.....	50

Chapter 1: Initial Presentation and Assignment

1.1 Introduction:

Chamblee High School will be instituting a flexible learning period in the upcoming fall semester, 2022. The school is within the public education system, and the project has been spearheaded by Andrew Milne, a teacher within the mathematics department at the school. There are an estimated 1800 students and an estimated 100 teachers at the school, all of whom must be scheduled for this new portion of time, a projected one hour and thirty-minute block, every Wednesday, in the fall.

1.2 Overview:

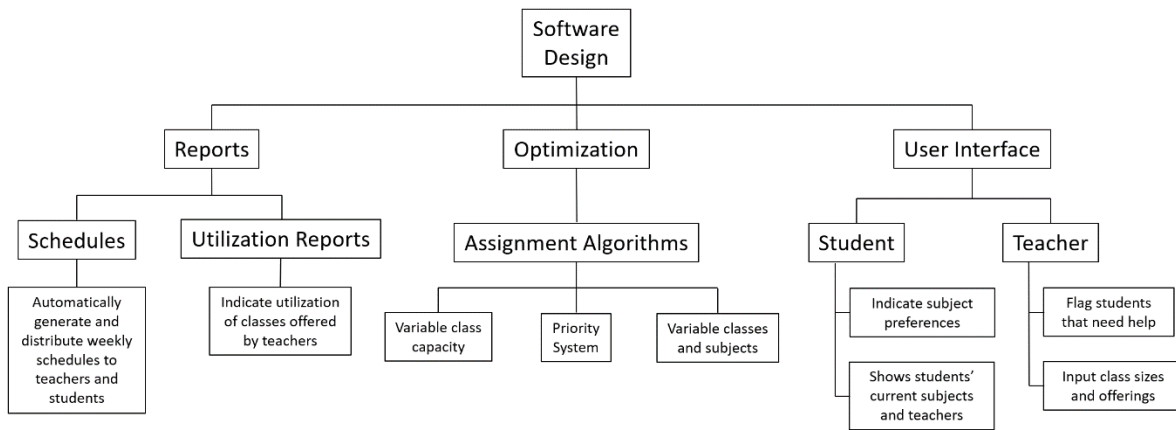


Figure 1: Overview of system requirements

Figure 1 shows an overview of the requirements for the system. There will be a student and teacher side to the user interface which will offer different features to users depending on their requirements. The assignment algorithm will allow for variable classes and subjects with variable capacities. Users with high priorities will be placed into supplementary classes from their own teachers before users with low priorities. The software will generate reports for users including schedules and utilization of offered classes to help teachers tailor their class offerings to student needs.

1.3 Objective:

The objective of the project is to design a program that is capable scheduling students and teachers and enable easy feedback to assist in the decision-making process for scheduling. Implementation of the program should make communications between departments easier and improve the efficiency of time used within the flexible learning block that will be enacted in the fall.

1.4 Justification:

Chamblee high school will require a structure in place that allows the scheduling of the projected 1800 students and an estimated 100 teachers to be quickly and easily scheduled, without excessive overlap or waste. To achieve this, Chamblee needs a flexible scheduling platform to facilitate the flexible learning period they are implementing in the fall semester. Without a platform that is capable of the databasing and is accessible to those not proficient in advanced data analysis the overall project of flexible learning will see waste and is likely to encourage infractions by the students.

1.5 Project Background:

During the pandemic, the shift to virtual learning showed teachers at Chamblee High School that a more flexible system for learning allowed students to become more involved in their education and get help in subjects they were struggling with. Dr. Andrew Milne, a teacher at the school, approached Kennesaw State University with an idea for a project to create a piece of software to automatically assign students to supplementary classes for the subjects they're struggling with.

1.6 Problem Statement:

The Kennesaw Industrial Education Team is tasked with the design and creation of a flexible scheduling platform that accounts for teacher and student preferences for Chamblee High School using the Coda software.

1.7 Baseline Flow Chart:

Below is figure 2, which describes the flow of information should the school not utilize any programming to organize the student body. Teachers would have a sign-up sheet within their classes after deciding what they are offering for that week, and students would register within that class until the sheet is at the maximum capacity. If a student cannot sign up for a course, the student will be sent to an independent study session.

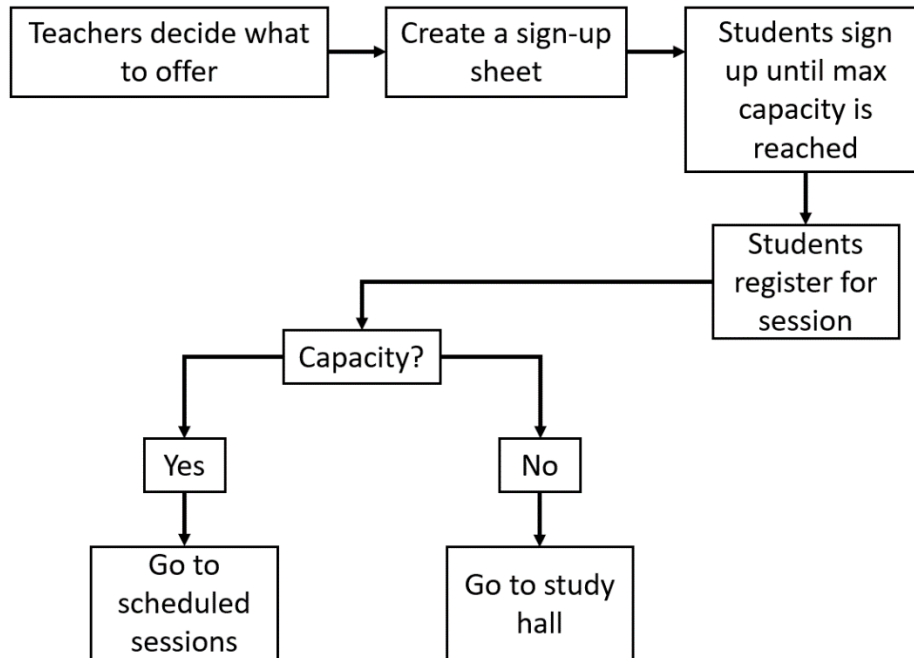


Figure 2: Baseline flow chart

The problems presented with the flow of information are numerous, and it presents logistical issues to the goal of the flexible period. To consider the flexible period a successful project there are three things that must be attained. Every student must have an assigned place to be within the schedule as it is necessary for administration to know where students are. Teachers must be helping as many students as possible. Teachers without students registered for their course need to be helping with the independent area to maximize utilization. Finally, there needs to be an increase in time that students in need are receiving the help they need. The baseline solution flow of information cannot solve these problems. The system inherently favors students to register in their earlier period classes, making registration about availability, not priority. Since the flow of information surrounding teachers will be slow, it cannot be guaranteed that the class does not become a free period for teachers who have not had students register for their session. The system would not allow for a one area allowance for independent study, so keeping track of students that have not registered for a session becomes near impossible if there are even enough independent areas to allow it.

The role of the project software is to mitigate if not eliminate the issues of this flowchart. The software has the goals of reducing communication time about scheduling to allow more room for identifying and resolving issues, identify classes that meet minimum capacity or have reached

maximum capacity to allow students to find a new session or default to independent study with notice, and assign students according to proper prioritizations. Proper prioritization includes student input and preference, as well as academic input from teachers and administration.

1.8 Project Software Analysis:

Any software analysis must be focused based on the project at hand. Within the current project, feedback is limited, and analysis is based on an exceedingly small sample size. While there are ISO models available for Software Analysis, the model selected for development of the system that will move through the transition of developers to users is McCall's Model. McCall's Model was originally introduced in 1977 to focus on "harmony between users and developers." [23] The model has been used for several studies on "web-based platform testing" [22] which should translate well to CODA.io, an entirely web-based sharing platform.

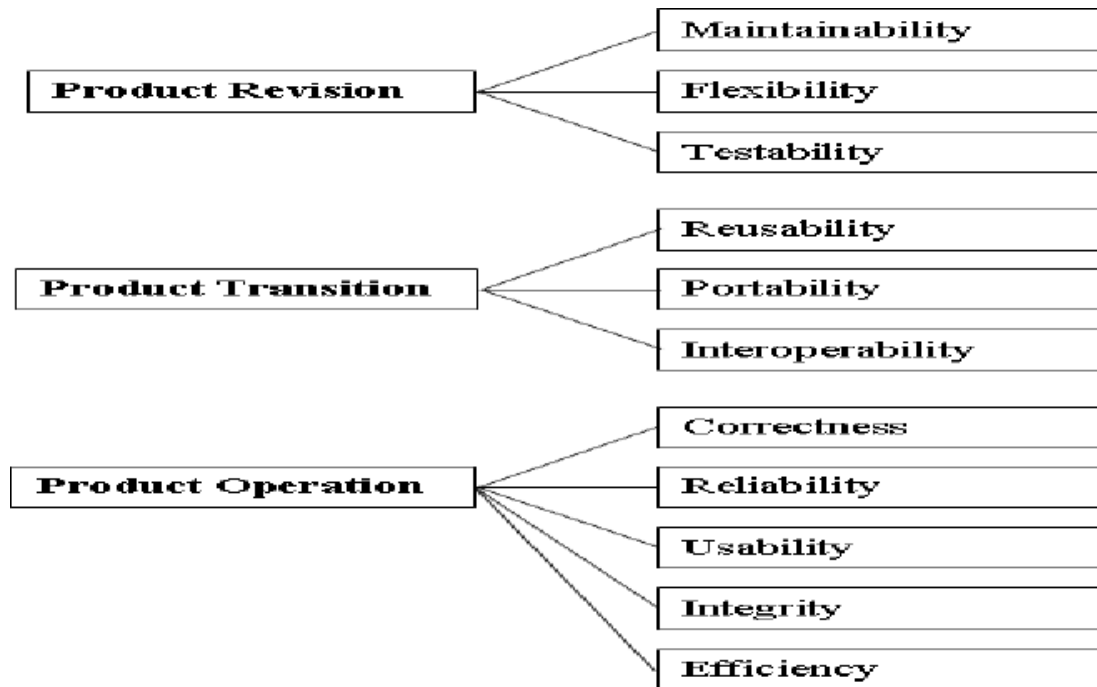


Figure 3: McCall Model [23]

Figure 3 is a representation of McCall's model, listing the primary functions of the model, which are then segmented into different attributes that contribute to those functions. McCall's Model has three essential functions portrayed in figure 3. Product Revision, which contains Maintainability, Flexibility, and Testability, Product Transition, which contains Portability, Reusability and Interoperability, and Product Operation, which includes Correctness, Reliability, Efficiency, Integrity, and Usability, all have philosophy or formulas that lend credence to the value of the software at hand.

If not for time constraints and the logistical launch of the project, the team would be involved through the completion of Product Operability, but the legal constraints involving data transfer and student protection prevent the team from being a part of the totality of this operation.

Within Product Revision, Maintainability is defined by its applicability to users. The attribute considers what will be required of maintenance personnel to identify and address software errors, and subsequently correct them. Flexibility lists the capabilities to support adapting the software in maintenance. This could include the ease of altering the software itself to varied applicability without altering the software itself. Testability details the capability of the system to return test data, as well as automatic testing to ensure the program is in working order on startup. [24]

Within Product Transition, Portability requirements are defined by what the software must be able to adapt to in terms of different hardware and operating systems. Reusability factors include the software’s capability to be used within a new project or be adapted and enable a new project. Interoperability requirements are focused on what the software can integrate with including different software and firmware. [24]

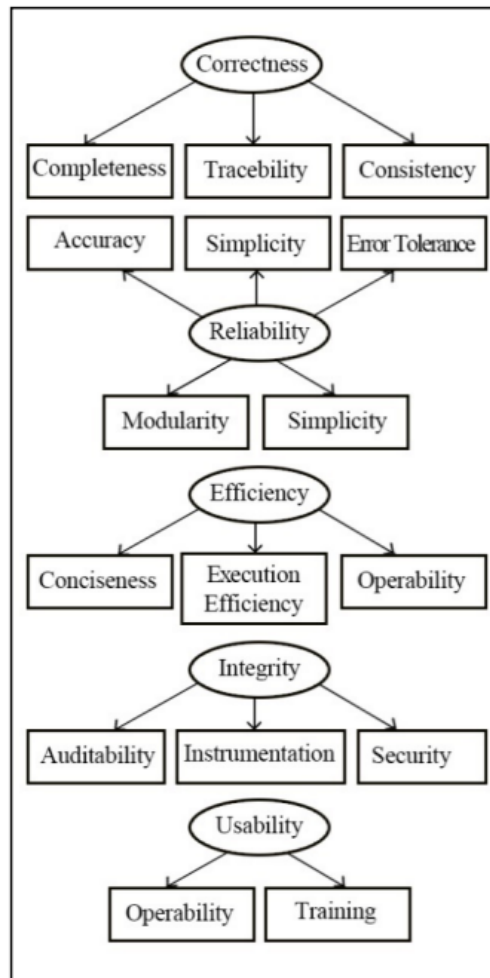


Figure 4: Product operation attribute chart [22]

Figure 4 is an in-depth breakdown of the attributes that define Product Operation. Product Operation carries the bulk calculations of analysis. Typically, Operation is the defining trait that determines a project's total success. Figure 4 [22] further details the attributes of Product Operation and how they relate to one another in the original study for the figure and will be a guide for analysis within this project. Correctness in this study was defined by Completeness, Traceability, and Consistency. Reliability is defined by Accuracy, Simplicity, Error Tolerance, Modularity, and Simplicity. Efficiency is defined by conciseness, execution efficiency, and operability. Integrity is defined by Auditability, Instrumentation, and Security. Usability is defined by Operability and Training. [22]

Calculations involving these attributes will be finished in Chapter 7. Normally, in the data collection stage, a random sample of the users would be given a questionnaire. With no access to students, and limited access to teachers, this type of collection may be limited. The sample size may be limited to the primary contact, Andrew Milne, and whatever teachers are willing to volunteer data from the estimated 100 teachers.

Chapter 2: Logistical Assignment

2.1 Responsibilities:

Acting Project Manager for the project is Reid Bryant. Primary responsibilities as acting Project Manager include communications with client, both in person and over any messaging platform, scheduling task completion and interviewing dates, methods for initial design ideas, and budgeting. As project manager, it is Reid Bryant's responsibility to ensure formatting is correct in the report, as well as complete literature review to make a comprehensible analysis of the final program.

Software Developer and Optimization Lead for the project is Tristan Brown-Mulry. Primary responsibilities include developing the server infrastructure and data structures for the system backend and assignment algorithms. Tristan is also responsible for helping to integrate back-end programs with Coda and they interact smoothly.

Software Developer and User Interface Lead is Quinn Tobin. Primary responsibilities include creating a user interface that students and teachers will interact with, as well as testing for usability and ensuring that the system functions as needed. Additionally pulling and displaying any data from the MySQL database is another task which will need to be undertaken.

2.2 Gantt Chart and Schedule:

2.2.1 Initial and Preliminary Schedule

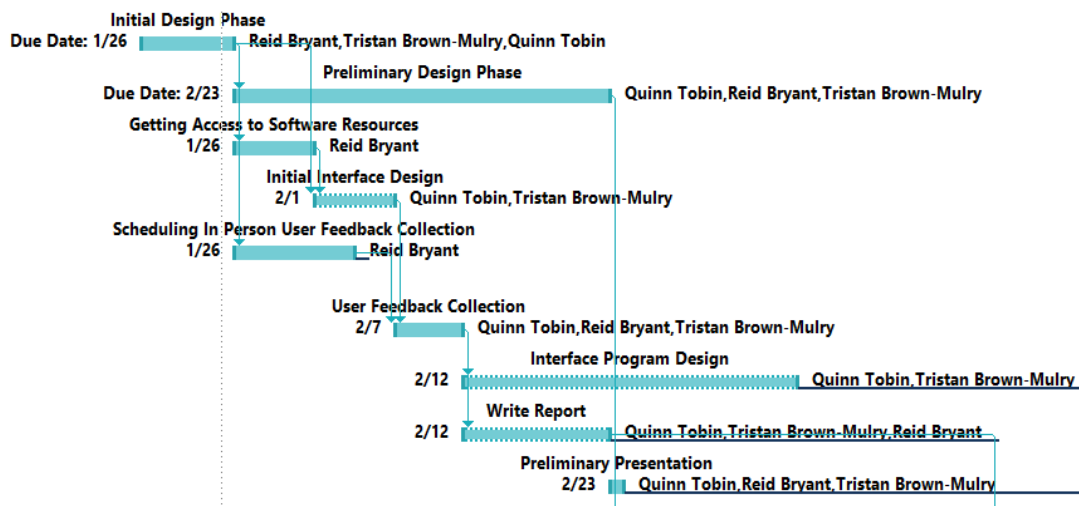


Figure 5: IDR and PDR schedule

Figure 5 represents the schedule for the Initial Design Phase and Preliminary Design Phase. The Initial Design Phase was late for start-up, and meeting with the client took longer than would have been ideal. During this time, the initial requirements were made, and the software to be used for the project was selected, with consideration that additional software may be necessary in the future. The Preliminary Design Phase involved development of the interface and gathering

feedback to better define the algorithm so the initial design would be pointed in the direction that Teachers could easily access and understand. Reid Bryant, Project Manager, scheduled a day to see the teachers between February 7th and the 11th to ensure that there was time to use that feedback to guide the initial solution. The feedback was collected using cardstock to represent an idea for an interface, with clarification that this was not the design, but a device to direct thought, and was used to further define the problem and goals of the project. After this we began the Interface Program design, which would continue into the In Progress Review, directed by Quinn, and the finish the Report, which included a literature review of both programs being used at the time, an analysis of flexible learning, and the finalization of including DMAIC as part of the process development.

2.22 In Progress Schedule

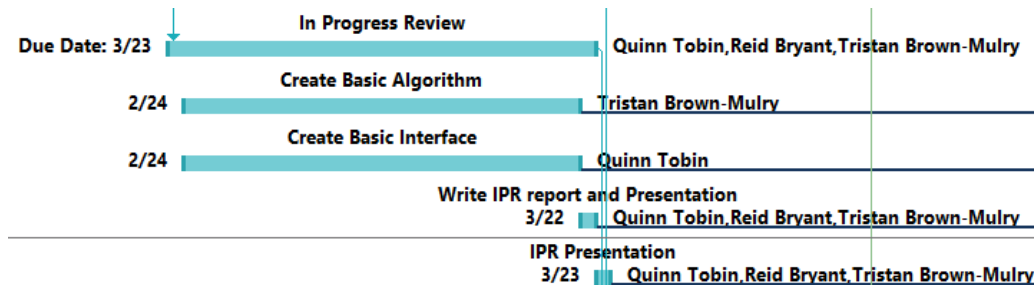


Figure 6: IPR schedule

The above schedule in figure 6 illustrates the bulk tasks involved in the preparation for the In Progress Review. For the In Progress Review, the Team had a large focus on developing the basic User Interface and Algorithm separately before integration of the two. In an ideal world, the Team would have had a larger focus on analysis during this stage, but were lost without data that represented what the project would largely be for the school. Tristan Brown-Mulry, Optimization Lead, created the back end algorithm from examples that may be similar to the data structure of the school, while Quinn Tobin, User Interface Lead, focused on creating a user interface that met the needs of the proposed flowchart in figure 9. It was discovered in this time period that the original viewpoints behind interface design are not feasible within the CODA system, but a new interface that would be later proposed to Andrew Milne, the client contact, could be developed. Creation of both the Basic Interface and Algorithm both involved literature review, and will be discussed in further detail in chapters 5 and 6.

2.23 Critical Design Schedule

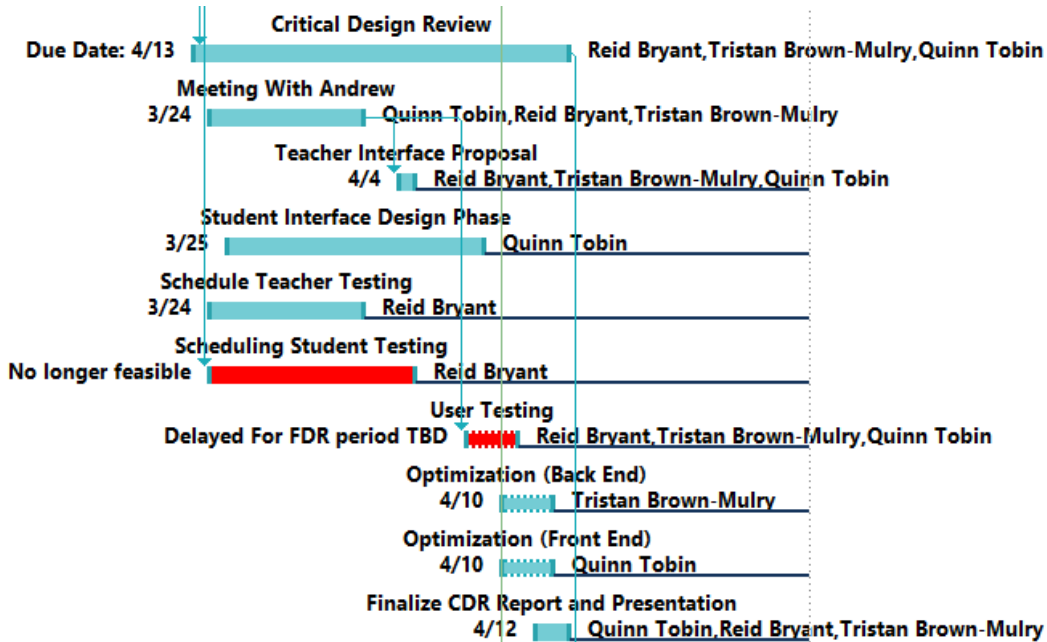


Figure 7: CDR schedule

Figure 7 Represents the task schedule for the Critical Design Review Phase. The Critical Design Phase started with a meeting with Andrew Milne. Quinn Tobin Presented the new interface design, as was feasible in Coda, while Tristan Brown-Mulry again requested the information of data structure to finalize the back-end servers for integration with the user interface. During the meeting, integration between the servers and interface was discussed, and the team has decided to use a program called Zapier, which will be further defined in the Final Design Phase. After the meeting with Andrew, it was agreed that the project needed to move in a new direction than it had been going to satisfy the requirements quickly and by the deadline. To regroup, reorganize, and redefine the problem, plans, and analysis, Reid Bryant met with Professor Lois Jordan, who has had contract work in Software Development as a consultant and project manager. This involves new literature review, as well as defining a flowchart with primary contact, Andrew Milne, which shows the flow of information without the development of the software and writing on the faults within the flow chart as discussed in Chapter 1 with figure 2. At this point, it was decided that meeting with Students, even though contact outside the school was no longer feasible within the timeframe of the project. User testing with teachers has also been further delayed to a point in the schedule to be determined when more information about scheduling is available from Andrew Milne. Information on data structure was received from Andrew Milne, which will be kept discrete for the purposes of anonymity. This has allowed Tristan Brown-Mulry to further develop the back-end servers to fit the data structures within the school to deliver a useable product.

2.24 Final Design Schedule

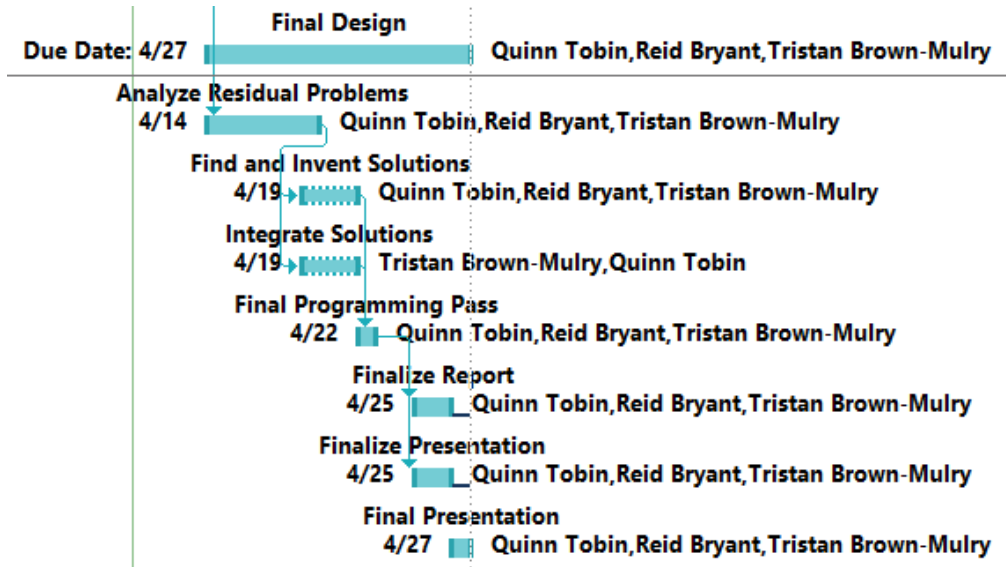


Figure 8: FDR schedule

Figure 8 represents the Final Design Phase of the project. The problems dealt with in this phase was integration between Coda, servers, and the algorithm, as well as analysis Calculations using McCall’s Method. Troubleshooting was done on the final issues in the Front End and Back End development to ensure the program was functioning to complete the video for presentation.

2.3 Budget:

Table 1: Project budget

Necessary Software	Cost	Duration	Total Cost
CODA.io	108\$/Mont	3 Months	324\$
Zoom Pro	15\$/Month	3 Months	45\$
Zapier	20\$/Month	1 Month	20\$

Table 2: Projected Implementation budget

Implementation SW	Cost/Month	Cost/Year	C/Y in bulk
Coda.io	100\$/Month	1200\$/Year	1020\$/Year

We have included the software licensing and subscription costs for our budget. Table 1 shows the individual costs of each program. We will be using the developer license for SQL and as such do not need to include the cost of licensing for SQL. The total cost of the software will come to

an estimated 389\$. A projected budget to maintain the project within the school will be created for the Final Design Review now that the software is more definitely locked.

Table 2 is a representation of the projected budget for the school, should they implement the developed Software. The budget for the school, as far as can be determined, is only the enterprise level of access to Coda.io. The team does not have access to this pricing but estimates that the cost would be near 100\$ per month. This would be an estimated annual cost of 1200\$ per year, or 1020\$ per year if Chamblee High School decides to pay annually.

Chapter 3: Literature Review:

3.1 Software Quality Literature, McCall's Model:

Our project is working with delivering a product in transition period, which can be seen in figure 3 that McCall's Model has direct consideration for. "What is Software Quality?" [23] is a site that compares various models in Software Quality and will be utilized to present figure 4 with some brief history about the Model. The bulk of the literature will come from "E-Voting Software Quality Analysis with McCall's Method," [22] which goes well into the application of McCall's Method in specific reference to a project, including calculations and formulas. Both references include definitions of the attributes within McCall's model, which is discussed in Chapter 1, section 1.8, Project Software Analysis. "Software Quality Factors" [24] shall be referenced for succinct definition of each of the attributes within the major aspects of the McCall Model.

3.2 Coda Literature

The group must learn how to utilize the resources the project calls for. In this section, we will be reviewing the literature that allows the group to both learn and innovate on the Coda.io platform. The first piece of literature is the "Learn by Watching-The Learn Doc" [1]. This document contains multiple tutorials for the basics of Coda, from the syntax and structure to how documents are designed. As well as information on how different codes interact with the documents in Coda and how to troubleshoot any issues that you come across when making a project. [1] This document gives the group the basic structure of how Coda functions before we investigate the history with, "Designing the CODA Formula Language" [2] which explains the design behind Coda and how the developers can use it in their projects. We knew the project would require servers to organize the data and find resources that assist us with integrating MySQL with Coda. [3] The other requirements involve scheduling and resourcing availability. An example can be found at "Availability" [4] about scheduling software used by a company for co-ordination, while "How to Create a Smart & Automated Employee Scheduling App?" has more information about starting from scratch. [5] These are the primary Literature that have been used to begin the design of the User Interfaces for the Project. There is also reserve literature for API referencing that explains how Coda can interact with other software languages as potential solutions to issues with errors between the utilization of Coda and MySQL. [6] The group has also found assistance in Coda information about "Refreshing Table from External Data" [13], "Automation with Buttons" [14], and "Automation Failed to Run" [15], that is being utilized to solve issues that have been encountered discussed in further detail in Chapter 4. [17] "Subpage menus - replicate across subpages" pertains to navigation elements across the document as a whole and explores different ways for users to navigate through the pages and subpages of their documents. [18] "Doc Locking" describes how to implement editing restrictions into a document, lists details pertaining to access levels as well as the different methods of restricting editing. It also warns against treating document locking as a form of data security. [19] "Using charts in Coda" explores the various charts that can be implemented in a document as well as information on how

to troubleshoot some common issues. Additionally talks about issues encountered with trying to automate data display with graphs and charts in coda. [20] “Members and roles” lists the roles that members of a document can possess, as well as their relationship to editing restrictions. [21] “How to coda permissions?” explores another way to restrict document access and limit what changes different members can make. [25] “” contains information on how to synchronize columns containing email addresses. [26] “” contains information on how to filter the views of tables based on different controls. This is useful for restricting content based on a user. [27] “” lookups are one of the ways to get data to transfer between different tables in coda. [28] “” contains information on how to automatically update text boxes in columns across different tables using formulas. [29] “” contained information pertaining to moving data between tables using formulas. [30] “” was a community tutorial on how to copy data from one column to another and have it automatically updated when the value is changed. [31] “” detailed different methods for assigning users to different imported data.

3.3 Backend System Design Literature

This project requires the team to create a robust backend for the software which will allow its use by Chamblee High School without constant maintenance. Due to this, one important aspect of the project has been to research data engineering practices and pitfalls. One approach to real time data processing that has been used increasingly often in modern industries is known as data streaming. This approach involves updating datasets as actions occur rather than in large batches. The book, “Streaming Systems: The what, where, when, and how of large-scale data processing” [8], provides information on its benefits, drawbacks, methods to implement it, and alternatives. “Architecting Modern Data Platforms: A guide to enterprise hadoop at scale” [9] and “The Enterprise Big Data Lake: Delivering the promise of Big Data and data science” [10] serve as guides to modern data platforms and best practices. Much of the information in these books is outside the scope of this project but they provide some context to the field that is helpful when building a platform that hundreds of people will rely on each week. “Practical Synthetic Data Generation: Balancing Privacy and the broad availability of data” [7] is a guide to synthetic data generation. It is often impossible to collect data for projects due to privacy or feasibility concerns, so synthetic data generation is a common practice in fields like education or healthcare. This process varies between situations, but this book contains information on ways to approach it. “Assigning students to courses,” from R for Operations Research [16] contains code written by Dirk Schumacher that serves as a starting point to begin development of the assignment algorithm for this project. It is much simpler than the final algorithm that Chamblee High School needs but the core assignment model with variable capacity for class scheduling is ideal to work from and its license allows users to modify and use it with very few restrictions.

3.4 Flexible Learning Literature

This section describes the literature for justification of flexible learning. “Why Schools Should Embrace Flexibility and Innovation Beyond COVID-19?” details multiple case studies on

flexible learning implementations in schools across the country and the students most impacted by them. It includes information on the motivation behind these programs, informing how the current system is designed for the middle class, leaving behind gifted students who have home responsibility and labeling them bad students unjustly, and how flexible learning benefits these students and others. [11] “What are flexible pathways for learning?” provides details on the Connecticut Department of Education’s opinion and approach to successful flexible learning. Not all the information is directly relevant to the project, but it provides a valuable insight into some of the aspects of flexible learning that should be avoided or highlighted. [12]

Chapter 4: Problem Solving Approach:

Define: This step of the project has included initial contact with the client, Andrew Milne. The group was given the opportunity to see what had already been developed by Andrew so that the project would have some amount of defined direction. We then developed the minimum requirements of the system as defined in the next section, and established a plan to gather data, and schedule to maintain during the process.

Measure: This step of the process has included in person interviews with the staff of Chamblee HS to see if the direction taken would be well received. The data was a freeform answer but gave significant insight into where we should focus the project in future efforts. Additionally, the group met with Cindy Shaw, database admin with Kaiser Permanente, to gain ideas and experience about what techniques could be deployed and what questions should be asked in the future.

Analyze: This step is unexplored at this time but is projected to include using synthetic generated data to evaluate the effectiveness of the program, some new user testing with specific guideline instructions to gauge useability and accessibility of the finished program with teachers as volunteers for testing. With this data, the group should be capable of establishing what needs to be changed for the Improve section of DMAIC.

Improve: Here we will be finding solutions to problems that will arise during the measure and analyzation stage and implementing them so that the database can support the data necessary for the program to function and the user interface has the features required for the project to be successfully helpful.

Control: The Control Portion of this project will be managed outside of the group's development. While it is projected that we may be able to create automatic reports to show improvements or detriments to student performance, the program will not be implemented until after the time our project is finished, and utilization of such data will be outside of the scope of the project.

4.1 Why DMAIC?

DMAIC has been selected due to the longevity of the project, and the lack of reliability for multiple trials. The analyzation in the order of DMAIC has assisted to keep focus on the critical tasks throughout the specific steps of the project, while keeping the end goal of a functional User Interface with an integrated database and algorithm to assist with organization of the student body in mind. Keeping in mind that the Control Section of DMAIC is largely out of the group's hands, as the project will be given to the school post final presentation. Providing proof of concept, and a functional interface that can be redesigned to fit control preferences.

4.2 Requirements:

1: The software must allow students to select their desired subjects.

- 2: The software must allow teachers to select which students need additional help.
- 3: The software must generate utilization reports for students and teachers.
- 4: The software must generate and distribute weekly schedules for students and teachers.
- 5: The software must allow teachers to select capacity for tutoring or review sessions.
- 6: The software must assign all students to either individual study or review/tutoring sessions.
- 7: The software must display each students' current subjects and teachers.
- 8: The software must be accessible on a phone or computer.

4.3 Proposed Flow Chart Solution:

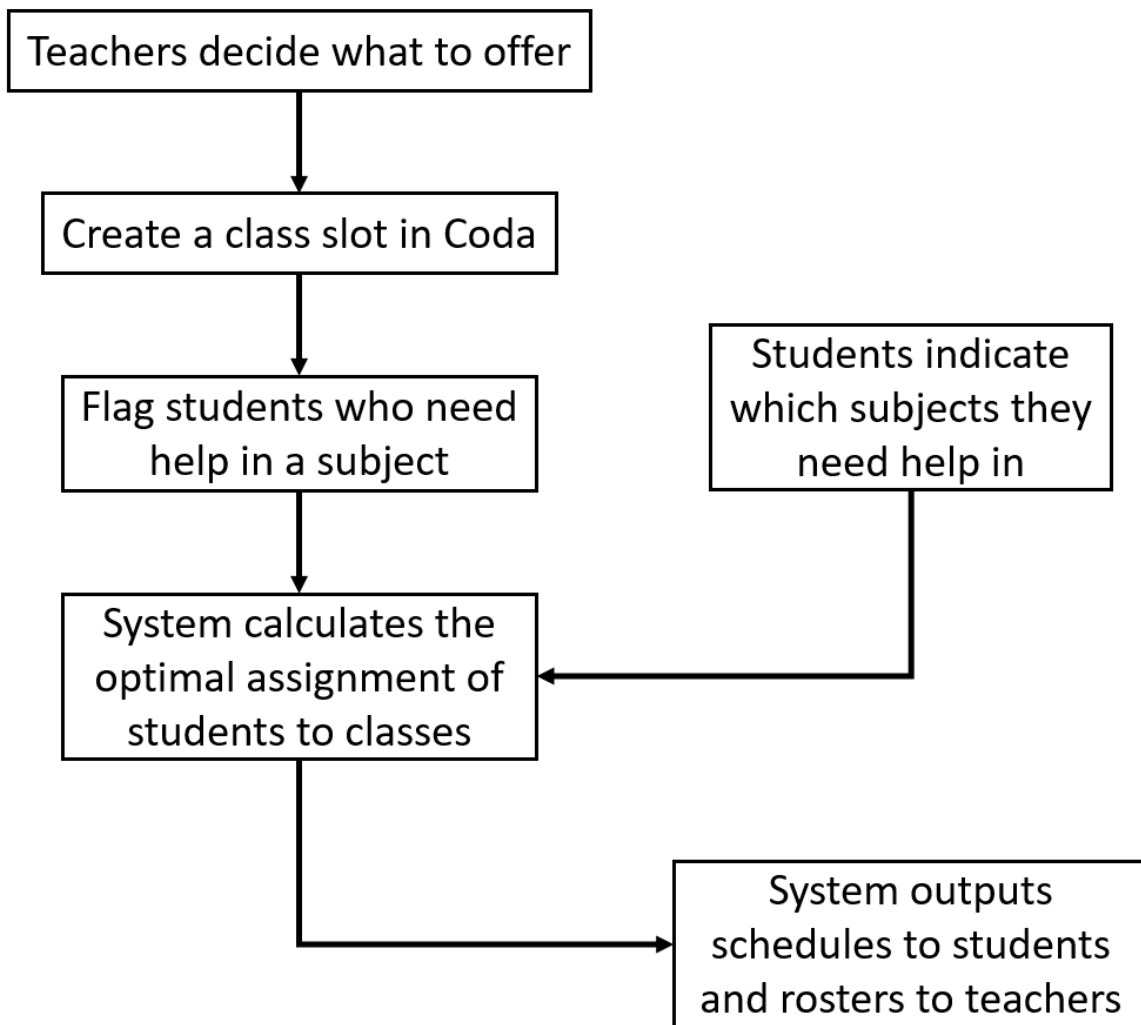


Figure 9: Proposed solution flow chart

Figure 9 shows a flow chart for the proposed solution. Compared to the baseline solution shown in figure 2, the proposed solution simplifies the work that students and teachers are required to do. Teachers will log into Coda to create their class offerings and flag any students they believe need help in their subject. Students will log into their view of Coda and see their classes listed. Here, they can choose their top three preferences for that week's flexible learning period. Information from the teachers and students will update datasets in the system which will be fed into the assignment algorithm. This algorithm will find the optimal assignment for each student and update the student and teacher interfaces with their corresponding schedules and rosters. This system helps to ensure that students are assigned to classes based on their needs instead of who gets to the sign-up sheet first. It also means that teachers can focus on teaching instead of coordinating and advertising flexible learning sessions. Many of the teachers consulted about the flexible learning program were worried that this would become another area for them to worry about so the proposed solution is aimed at making the process as easy as possible for all users to work with.

4.4 Materials Required:

This is a list of the materials needed to operate with the school and complete the project.

- Zoom Pro
- Coda Software
- MS Project
- Data Reports
- SQL Server
- Python
- R Studio
- MySQL
- Amazon Web Server

4.5 Resources Available:

Cindy Shaw and related contacts at Kaiser Permanente have made themselves available to the group for consultation. Andrew Milne, the primary contact at Chamblee, is available for consultation and is vital to scheduling in person visits with the school. Lastly, volunteers can be found in the staff of Chamblee High School for testing and feedback.

Chapter 5: Creation in Coda

5.1 Existing templates regarding Month View + initial navigation elements

While creating the user interface for this project Coda posed some unique issues with the creation of a UI. The requirements given to us by the client gave a clear preference for working on this platform and Dr. Andrew Milne used Coda to format some data previously and suggested we work with Coda. Coda is most useful when employing a template for different topics or processing data from tables [2], however most of its applications are business based so there are no templates that we could rely upon for the entirety of this project, additionally, while the user input is intuitive there are many parts pertaining to the creation of documents [1]. This resulted in us combining multiple elements of different templates so that we could have a provisional UI. The template dealing with creating a scheduling application was most useful as the calendar system serves a key role in our own work [5]. Figure 10 is the month view page during the UI's development, and Figure 11 is an alternate way to view when the session is scheduled for. As can be seen in Figures 10 and 11 the calendar template has been adapted and modified for this project. Navigation elements to other pages have been added at the top and the bottom of the page, as well as prompts on how to interact with and use the calendar. Additional details about the class are located at the bottom of the page, and the navigation elements allow easy access to changing any details.

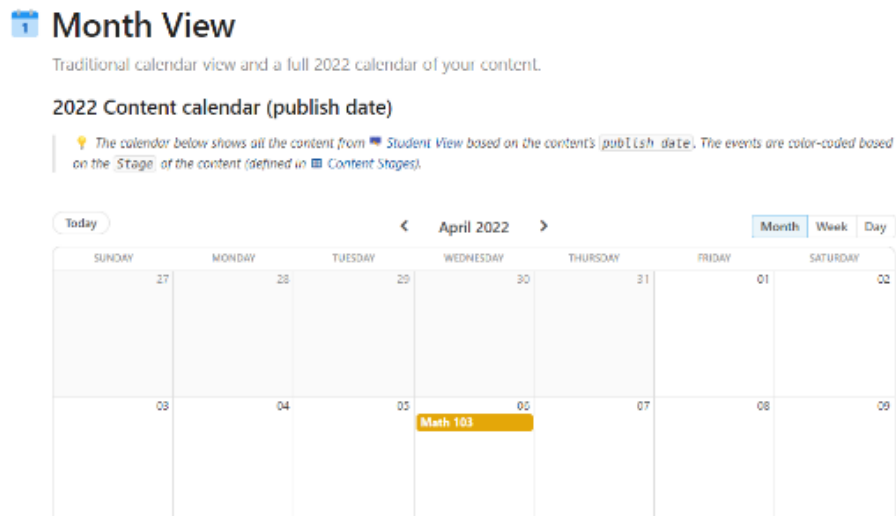


Figure 10: Calendar view

Full 2022 calendar view (estimated publish month)

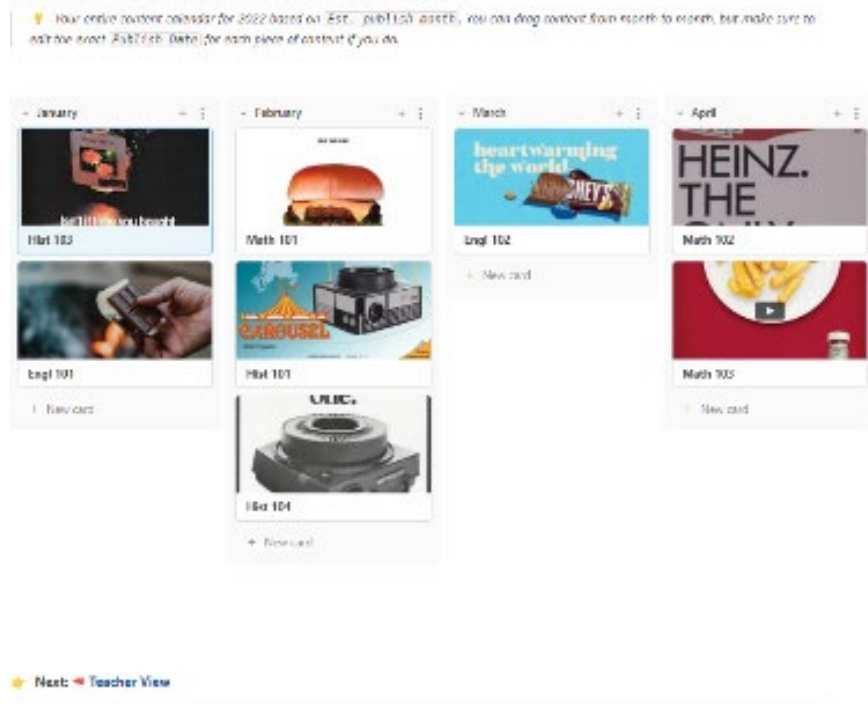


Figure 11: Calendar view 2

5.2 Student View + Mobile View

Dr. Milne gave a rough outline for this project that included both a teacher's view and a student's view. Other pages were created to fulfil our needs, such as importation of data, or to demonstrate coda's capabilities like the month view. Navigation elements are located not only on the page/subpage bar but within each of the views as well. Additional navigation elements are located at the top of the page which allow the student to access student subpages and view more information. These elements in particular make for easy navigation of the program and meet requirement 8 for ease of navigation from page to page when used from a mobile or web browser. The student view allows the students to view the different class sessions for each subject as well as any relevant information such as the teacher, session type, materials, and date. Figure 12 is the student view page during the UI's development. The students are also able to interact with the materials needed for their sessions from this page as can be seen in Figure 12, The students can click on the materials in the resource column and be taken to the relevant subpage where teachers will have uploaded their content. Students can navigate from the main page to the month view and see what sessions have been scheduled by the teachers in advance as well as check on what sessions are currently in progress.

Student View

A list of all your content to be published. Click sub-pages below to see this table of content viewed different ways.

By stage By type Ongoing Session

+ Add Content

View	Title	Resources	Stage	Type	Subject	Learning Material	Month	Date	Teacher	Approvers	Class Setting
View	Math 101	Doc 1	Uploaded	Blog post	Math		February	2/17/22	Alan Chowansky	Adam Davis Buck Dubois Felix Marlin	Blog Twitter LinkedIn
View	Math 102	Doc 2	Producing	Blog post	Math		April	4/16/22	Polly Rose	Adam Davis Polly Rose	Instagram
View	Math 103	Doc 4	Approved	Podcast episode	Math		April	4/6/22	Felix Marlin	Adam Davis Buck Dubois Felix Marlin	Community forum
View	Math 104	Doc 3	Editing	Infographic	Math		June	6/8/22	Alan Chowansky	Lola Tsoudorym Maria Marquis	Community forum In-product promo
View	Hist 101	Behind the wheel	Producing	Whitepaper	History		February	2/6/22	Buck Dubois	Lola Tsoudorym Maria Marquis	Blog In-product promo
View	Hist 102	Dependable at gravity	Editing	Video	History		May	5/12/22	Alan Chowansky	Maria Marquis	Website Chatbot Community forum
View	Hist 103	Isn't it time you bought a carousel	Editing	Infographic	History		January	1/29/22	Adam Davis	Adam Davis Buck Dubois Felix Marlin	Community forum
View	Hist 104	A zillion slides	Uploaded	Webinar	History		February	2/2/22	Maria Marquis	Adam Davis Buck Dubois Felix Marlin	In-product promo
View	Engl 102	Doc 2	Idea	Podcast episode	English		March	3/1/22	Maria Marquis	Buck Dubois Maria Marquis	Email newsletter Community forum
View	Engl 104	Doc 3	Producing	Blog post	English		May	5/1/22	Buck Dubois	Adam Davis Buck Dubois Felix Marlin	Blog
View	Engl 101	Doc 1	Idea	Infographic	English		January	1/21/22	Felix Marlin	Lola Tsoudorym Maria Marquis	LinkedIn Email newsletter
View	Engl 103	Doc 4	Approved	Video	English		June	6/14/22	Maria Marquis	Alan Chowansky	Twitter
View			Live								
View	Write title here						January				


Customize columns in this table like Stage, Type, and Distribution Channels in Customize columns. Clear dummy data above by clicking here [Clear data](#)


View your: Month View

Figure 12: Student view

Student View

A list of all your content to be published.
Click sub-pages below to see this table of content viewed different ways.

 By stage  By type

 Ongoing Session

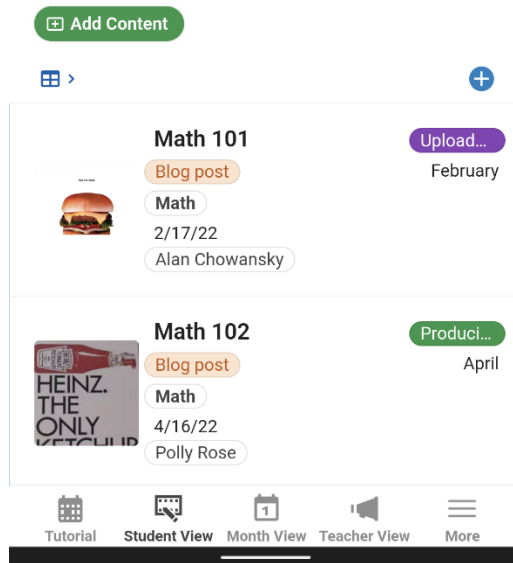


Figure 13: Mobile student view

5.3 Teacher View

The main teacher view allows the teachers to access content by subject as well as add additional subjects if they wish to do so. Teachers are also able to assign different teachers and edit the content on the subject from this main page. The teachers can view and add materials by class by interacting with the draft folder column. Figure 14 is the teacher view page displaying details about the UI during its development. In Figure 14 the materials in this column are linked to their respective folders. They are also able to interact with the asset's column, this is where teachers can import images which are linked to their classes. There are navigation elements at the top and bottom of the page that allow the teachers to access the class view subpage as well as the resources in each subject. Figure 13 is the student view page during the development of the UI's navigation elements. Figure 13 illustrates how these navigation elements can be helpful when navigating through mobile use.

Teacher View

A list of all the campaigns your team is creating content for.

+

Subject	See drafts folder	Assets	Status	Teacher	Team	Start Time	Content in Class	Notes
Math	Class Math		Live	Adam Davis	Alan Chowansky Maria Marquis Lola Tseudonym	Q1	Math 101 Math 102 Math 103 Math 104	Notes 1
English	Class English		Idea	Alan Chowansky	Buck Dubois Adam Davis	Q1	Engl 101 Engl 102 Engl 104 Engl 103	Notes 2
History	Class History		Done	Lola Tseudonym	Alan Chowansky Adam Davis	Q2	Hist 101 Hist 102 Hist 103 Hist 104	Notes 3

+

Clear dummy campaigns by clicking here [Clear data](#)

Next: [Resources](#)

Figure 14: Teacher view main page

5.4 Teacher View Subpage

From the class view subpage teachers can create sessions as well as add documents and other resources. The teachers can also specify session type date and what stage the session is currently in. Like other subpages there is navigation back to the main page. Figure 15 is a teacher view subpage during the development of the user interface. As can be seen in Figure 15, the teachers now interact with their materials on a class-by-class basis as opposed to the main teacher view where they are able to import data and assets. This subpage is where the teachers can make changes to the stage, type, and date of their sessions.

By Class

Grouped by campaign.

See details about each campaign by hovering over the campaign or go to [Teacher View](#)

Subject	Title	Resources	Stage	Type	Teacher	Month	Date
Math	Math 101	Doc 1	Live	Blog post	Alan Chowansky	February	2/17/22
	Math 102	Doc 2	Producing	Blog post	Polly Rose	April	4/16/22
	Math 103	Doc 4	Approved	Podcast episode	Felix Marlin	April	4/8/22
	Math 104	Doc 3	Editing	Infographic	Alan Chowansky	June	6/8/22
History	Hist 101	Behind the wheel	Producing	Whitepaper	Buck Dubois	February	2/6/22
	Hist 102	Dependable as gravity	Writing	Video	Alan Chowansky	May	5/12/22
	Hist 103	Isn't it time you bought a carousel	Editing	Infographic	Adam Davis	January	1/29/22
	Hist 104	A zillion slides	Uploaded	Webinar	Maria Marquis	February	2/2/22
English	Engl 101	Doc 1	Idea	Infographic	Felix Marlin	January	1/21/22
	Engl 102	Doc 2	Idea	Podcast episode	Maria Marquis	March	3/1/22
	Engl 104	Doc 3	Producing	Blog post	Buck Dubois	May	5/1/22
	Engl 103	Doc 4	Approved	Video	Maria Marquis	June	6/14/22

Go back to: [Teacher View](#)

Figure 15: Teacher view class subpage

5.5 Data Importation

The main area of difficulty is importing data [13], there are numerous cases of errors being thrown when there are issues with the importation of data. The customize columns page is where the data from the MYSQL server will be imported into tables and accessed by the rest of the coda document. This portion of the document is also where other information used by other pages is kept such as the various categories for content as well as what setting that content is in. The columns page is key to making the rest of the document run smoothly through the data that will be imported and other data referenced by other pages. Lastly is the automation of the user interface. While there are many potential errors in this section of the code, it is ideal to have an automated [15] script for the UI and this will be an ongoing issue for the remaining duration of the project.

5.6 Data Display

Additional features requested by Dr. Milne such as data display and locking capabilities have been implemented. Figure 16 is an example of a coda chart and the customization ability that coda has. As can be seen in Figure 16 charts can be configured to display data associated with any variables kept by coda. While coda keeps track of these different variables, oftentimes charts must be manually configured by a user with an access level above Doc Maker to display the desired data.[19] There are many ways that charts can be set up based on your data and how you want to view it as can be seen in the figure below.

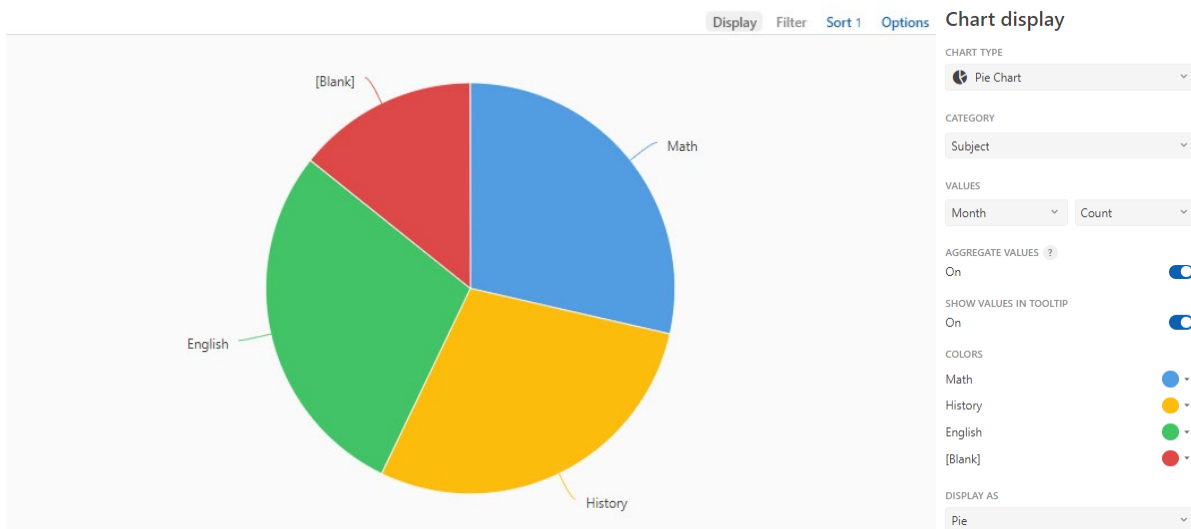


Figure 16: Chart Example

5.7 Document Locking

Restriction of editing access was another feature requested by Dr. Milne. In general, there are 3 different access levels for a given document, editor, which can make simple changes when permissions allow, Doc Maker which can add new things to documents as well as create new ones, and Admin which can access the underpinnings of the system [18]. Figure 17 is an illustration of coda's locking system and the different access levels within it. Additional changes to access can be made to individual pages as illustrated by Figure 17 which would restrict the ability to make changes to elements within pages. One important note is that Doc locking is not a security feature, and it would be irresponsible to treat it as such. We are aware that our client is concerned with access to student data, and they would need to explore a solution that would suit their unique needs.

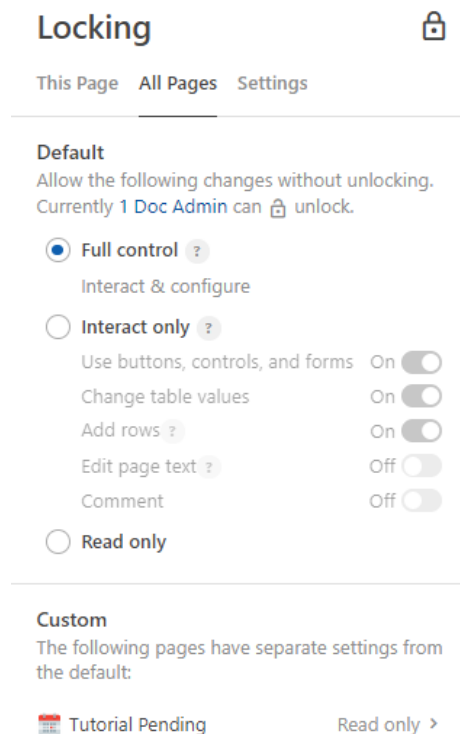


Figure 17: Document locking

5.8 Document Filtering

Coda offers alternate ways to restrict a user's ability to interact with pages and the tables that they contain. This is frequently done with the expression `User()=thisrow.User()`. However, this can result in difficulties transferring data from columns between the different tables in coda. Because of this we ended up using a slightly more complex expression, `student_df.Filter(thisRow.User.Email.Contains(Student_Email)).Student_Email`, this references

the student data frame table and checks if the email imported from the server matches the email the account is registered with. It then restricts the view to the StudentAssign table and allows the user to input their class selections and observe their class assignments without having to view other users' data. Figure 18 is a view of the of the StudentAssign table in the Student View page. Figure 18 only displays one row out of the possible 100. This dramatically streamlines the process by which students select and receive their classes and cuts down on the possible errors that could be encountered.

StudentAssign								
First Name	Last Name	Student_ID	Subject_1	Subject_2	Subject_3	AssignedClass	User	Email
Taylor	Harmon	df1f9574	Math 110	Engl 109	Math 109	Engl 111	Quinn Tobin	quinntobin01@gmail.com

Figure 18: Student Assign

Chapter 6: Back End Development

6.1 Synthetic Data Generation

The synthetic data generation for the project was done in R Studio. The first stage of this process was to create datasets that match the format of reports that can be generated by Chamblee High School. Due to privacy laws that protect the information of students in public schools, we were unable to directly access any of the school's data. They use a reporting API for Infinite Campus called Clever, so the format of the datasets was matched to the default settings generated by that system.

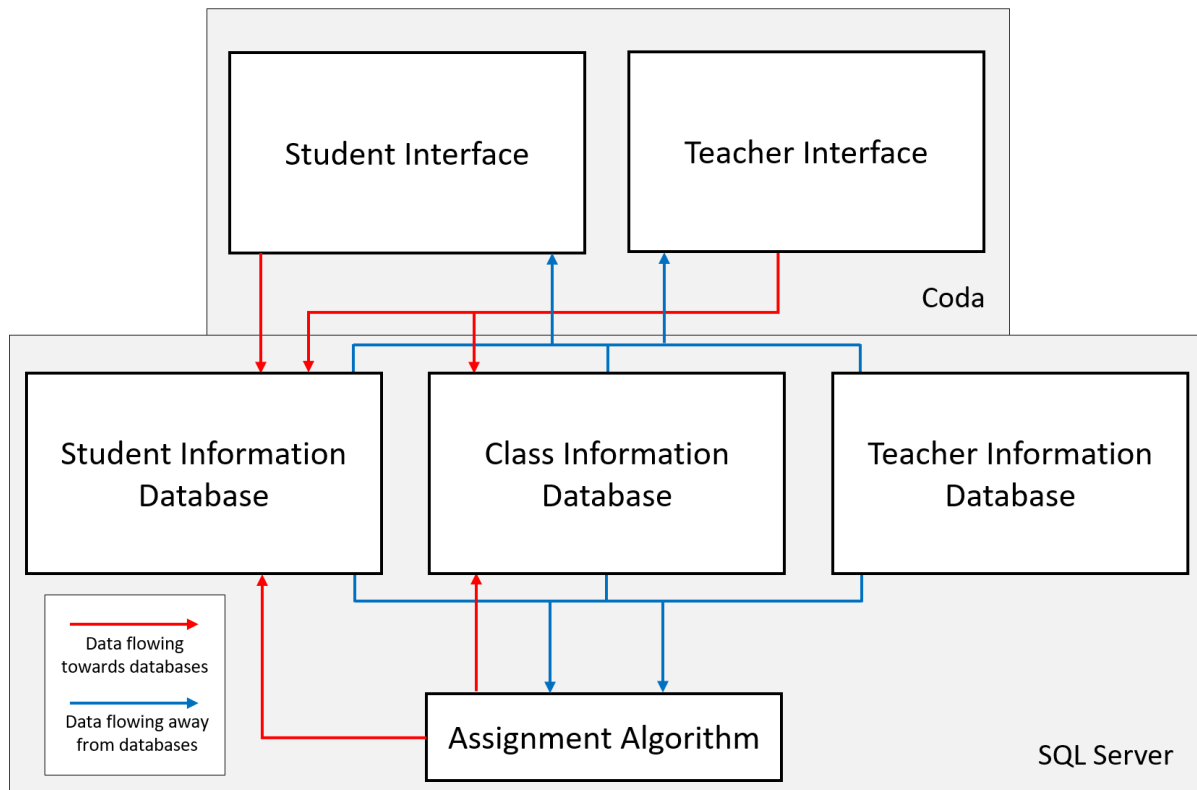


Figure 19: System data flow chart

Figure 18 shows how data will flow within the system. Student and teacher interfaces will be populated with information from the student, class, and teacher information databases. From these, students and teachers will be able to make selections which will update entries in the student and class information databases. These pieces of data will be fed into an assignment algorithm to divide students between supplementary classes and these results will be fed back into the student and class information databases. This information will update the student and teacher interfaces to display their supplementary schedules and class rosters. For the initial backend development only 100 students and 10 teachers were generated but the code that was used can be scaled to replicate the organization of the school itself. The data structure of the system allows it to satisfy the backend needs for requirement 7. When students input their ID

number into Coda, their corresponding data will be retrieved and shown to them in the user interface. There were several other datasets generated to contain information used by the algorithm. These include datasets such as a student preference dataset that contains information collected through the user interface to be used in the algorithm. These datasets are generated from the three primary databases and user data, so they do not need to match the structures of any existing school data.

6.2 Assignment Algorithm

The assignment algorithm was based on code developed by Dirk Schumacher at R for Operations Research [16]. His code served as an ideal starting point for the problem which could be altered and developed to suit Chamblee High School's needs.

$$x_{ij} = \begin{cases} 1 & \text{if student } i \text{ is assigned to class } j \\ 0 & \text{otherwise} \end{cases}$$

w_{ij} = weight of assigning student i to class j

c_j = capacity of class j

Figure 20: Assignment algorithm variables

Figure 19 shows the core variables used in the assignment algorithm. For each of these, students are organized with the variable 'i' and classes are organized with the variable 'j'. There are three main pieces of data that are fed into the algorithm and used to calculate the optimal distribution of students. 'X_{ij}' is a set of identity matrices that display the possible assignments of each student. It is identical for each student but allows the algorithm to check the weight of each class assignment. 'W_{ij}' is a matrix of weights for each possible class for each student. Figure 22 shows how this weight function works in more detail. Finally, 'C_j' is a list of the capacities of each class. This is set by the teachers that are offering each study session and allows the system to be

used for large or small group settings. The addition of this variable satisfies requirement 5 for the project.

$$\begin{aligned}
 1) \quad & \textit{Maximize} \quad \sum_{i=1}^n \sum_{j=1}^m x_{ij} \cdot w_{ij} \\
 2) \quad & \textit{Subject to} \quad \sum_{j=1}^m x_j \leq c_j \quad j = 1, \dots, m; \\
 3) \quad & \sum_{i=1}^n x_i = 1 \quad i = 1, \dots, n;
 \end{aligned}$$

Figure 21: Assignment algorithm

Figure 21 shows the assignment model expressed in mathematical notation. It is similar to most conventional assignment algorithms except for a weight function replacing the standard cost associated with each assignment and a variable capacity for each class. The value of the weight for each possible class assignment is maximized in line one. Line two ensures that the number of students assigned to each class does not exceed its capacity. Line three ensures that each student is only assigned to one class. This combination of objective function and constraints allows the algorithm to work as desired and take priorities and flags for each student to assign them to the optimal class.

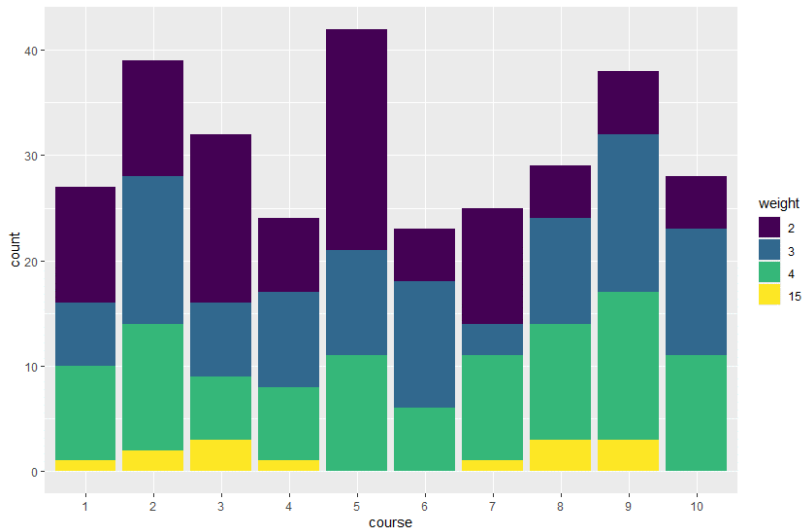


Figure 22: Student preference distribution

The graph in figure 22 shows the distribution of synthetically generated student preferences between the ten main classes. Yellow indicates that it is a student's top preference followed by green then blue. The synthetic data was randomly distributed but it is likely that real-world data will indicate preferences for students at Chamblee High School so reports of raw student preference data will be available to teachers and administrators.

	<i>Class</i>				
	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>Overflow</i>
<i>Student 1 Preferences</i>	<i>3</i>	<i>4</i>	<i>0</i>	<i>2</i>	<i>1</i>
	<i>x = class j flagged by teacher</i>				
	<i>Class</i>				
	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>Overflow</i>
<i>Student 1 Weights</i>	<i>3</i>	<i>4</i>	<i>-100,000</i>	<i>15</i>	<i>1</i>

Figure 23: Student preferences and weight function

The student preference dataset was populated with artificial student preferences. The code used for this will only be included during the development of the software and in the final version data will flow in the opposite direction. Students and teachers will enter their preferences which will update the datasets and be fed into the assignment algorithm. An example of student preference data is shown at the top of figure 23. This example is simplified to only four classes in addition to the study hall overflow session. In this example, student number one had a preference generated with a top priority for class two followed by class one and class four. Each student has the study hall class as their lowest priority. The effects of the weighting function are shown in the lower half of figure 23. It checks whether a proposed class is on the student's preference list and if it is, it assigns that priority level as its weight. If it is not, the class is assigned a weight of -100,000 to force the algorithm not to consider it. Class four is shown in red to indicate that its teacher has flagged the student as needing to attend the scheduled review session. The weighting function sets the weight to 15 for this class to increase its favorability to the algorithm. The teacher input weighting occurs before -100,000 is assigned to irrelevant classes so it will be weighted highly even if a student has not indicated a preference for it. The student preference dataset and weighting function satisfy requirements one and two which were laid out at the start of the project.

Figure 24 shows how the objective function calculates the possible values for each student. The weights for student one's preferences, generated in figure 23, are on the left and the student one assignment matrix is shown on the right. 'X_{1,j}' is an identity matrix where 1 indicates that the student is assigned to that class and 0 indicates they are not. Every possible assignment is included in this matrix. The objective function multiplies these together to create a list of the possible weights that correspond to each assignment for each student. In this case, it would be the list (3, 4, -100000, 12, 1). Assigning student one to class four would result in the highest weight and assigning them to class three would result in the lowest weight. This is repeated for every student and the overall objective value from the sum of each combination of assignments is compared. The model uses an internal algorithm to find the global optimal maximum value and the assignments for each student are recorded to a dataset which is used to update the Coda interface.

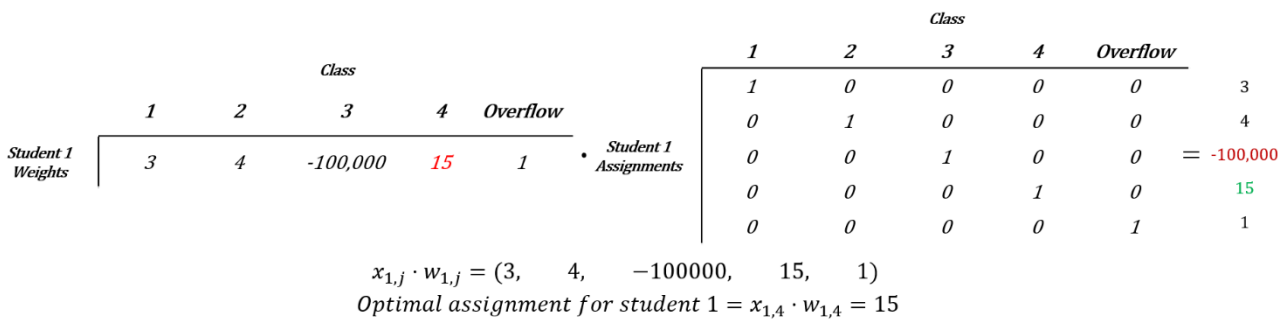


Figure 24: Objective function value calculations

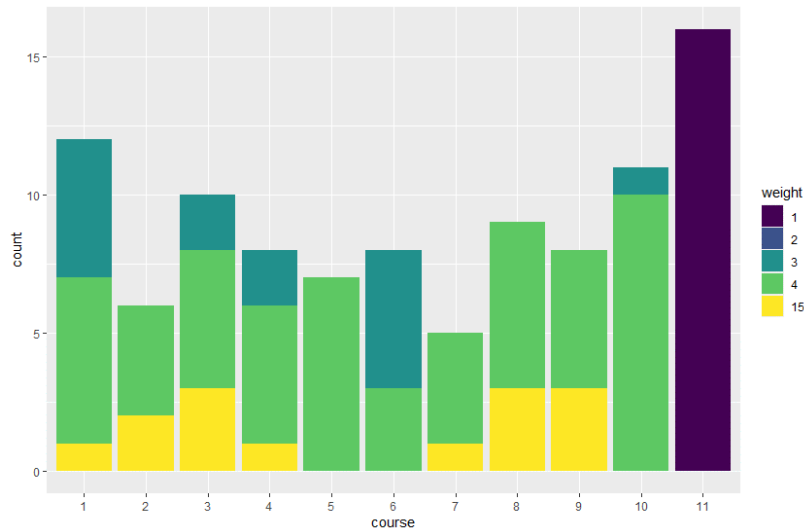


Figure 25: Student assignment distributions

The graph in figure 25 shows the assignment of students to each of the ten classes and the overflow class shown in purple. It shows that, based on the current parameters, most of the students were assigned to their highest priority classes shown in yellow. Some students were

assigned to their second or third priority classes and 16 were assigned to the overflow class. As the figure shows, this model fulfills requirement 6 and assigns all students to review sessions or the independent study session. The integration with Coda will satisfy requirement 4 to generate and distribute weekly schedules to students. This algorithm has been tested up to 1200 students and 120 classes and continues to perform well. Teachers at Chamblee High School indicated that it was important for them to be assigned their own students when possible. This will be included in the weight function to increase the value of matching students with their teachers to the algorithm and disincentivize assigning students to other teachers. From these assignments, the system will record utilization percentage for each class that teachers offer. This is calculated by dividing the number of enrolled students by the total capacity and the system will be configured to allow this to be sorted by subject, grade, and teacher. This feature allows teachers to determine how to best structure their classes to help students and allows administrators to ensure teachers are not creating undesirable sessions to avoid teaching. This was requested by many of the teachers that were consulted about the project. Integrating these reports with the Coda interface will satisfy requirement 3 of the system. After each time the assignment algorithm is run, the utilization of each session is calculated and added as a column to a copy of the offered session dataset. This is then uploaded to the server with a record of when the report was generated and can be retrieved through a server query or set to update a page in Coda.

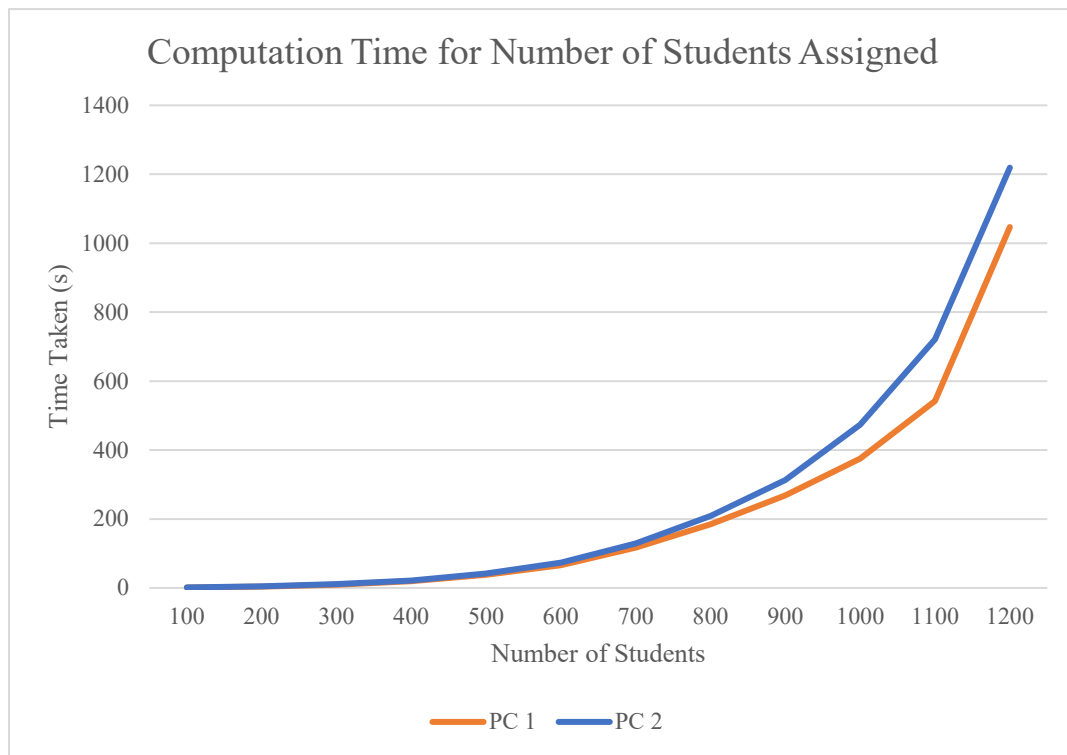


Figure 26: Computation time for number of students in the model

Figure 25 shows a graph of the processing time on two computers as the number of students and classes increase. Student base sizes ranging from 100 to 1200 were tested with each having ten percent of the number of students set as the number of classes. This is not realistic to the school

itself, but it gave valuable insight into how the algorithm performs in extreme situations. Testing the algorithm with 1700 students and 40 classes took less time to process than 1200 students and 120 classes so it can be assumed that the main source of computation time in this test was the number of classes. These 12 levels were run on two computers with different levels of performance. Computer one, shown in orange, is a desktop with an AMD Ryzen 7 3700X at 3.6GHz. It has 8 cores with 16 logical processors. Computer two, shown in blue, is a laptop with an Intel i5-8300H at 2.3GHz. It has 4 cores and 8 logical processors. The graph shows that as the number of students and classes increases, the time it takes to build and solve the model increases exponentially. This confirms that dividing the student body by grade level and running the algorithm four times would take less time than combining them and running the entire group at once. Even the maximum time the algorithm took would be acceptable but running four smaller models would reduce the risk of a problem occurring during the process and causing issues for the system. A major requirement for this project is reliability as the group will not be able to maintain the system once it is handed off to the school.

Table 3: Computation time at different student base sizes

Number of Students	Number of Classes	PC 1 Time (s)	PC 2 Time (s)	Difference (%)
100	10	1.25	1.43	114%
200	20	3.83	4.45	116%
300	30	9.66	10.75	111%
400	40	19.33	21.5	111%
500	50	37.75	41.76	111%
600	60	66.11	73.2	111%
700	70	117.08	128.9	110%
800	80	184.81	208.93	113%
900	90	267.99	313.16	117%
1000	100	375.14	473.05	126%
1100	110	542.56	721.68	133%
1200	120	1046.89	1219.21	116%

The numerical results from figure 26 are shown in table 3. Despite the significant difference in performance between these two systems, they do not show a significant difference in processing time. On average, computer two took 116% of the time that computer one did. Rather than purchase a new system, Chamblee High School is likely to put the SQL server containing the algorithm on an unimportant computer, so this is an important consideration. These results suggest that the performance of the computer running the algorithm is not very impactful to the amount of time it takes to run. All of the code for the backend algorithm is included in Appendix E under RStudio.

Chapter 7: McCall's Analysis

7.1: Software Quality Factor

$$Fq = \frac{\sum m_n}{T}$$

Where:

Fq = Software Quality Factor

m_n = the nth metric that affects the software quality factor

T = The total number of metrics that effect total software quality

Figure 27: Software Quality Factor

Figure 27 shows the equation that will represent the total software quality for the project. The formulas have been designed to measure most accurately to the purposes of the project, to create a software that serves as the baseline software for an assignment system based on priority for Fall, 2022. The sum of the metrics, m_n , are composed of Correctness, Reliability, Efficiency, Integrity, and Useability. [22] These factors have been altered based on the needs of the project, as not all attributes associated with these metrics can apply. In an ideal world, all attributes would be included and modified, but lack of access to real sample data prevents this from happening.

7.2: Correctness

$$\text{Correctness} = \frac{C+T}{2}$$

Where:

$$C = \text{Completeness} = \frac{\text{requirements met}}{\text{total requirements}}$$

$$T = \text{Traceability} = \frac{\text{User Traceable Features}}{\text{Total Features}}$$

Figure 28: Correctness

Figure 28 gives the created Correctness equation based on equations found through literature and altered for best fit to the project, as was done in literature review. Completeness is defined by completion of the requirements, to the team’s knowledge, all requirements of the system have been met. With 7/8 requirements fulfilled, Completeness has a rating of .875 or 87.5%.

Traceability is a factor based on the “ease of referring back to the component of the program to the needs of the user.” [22] In order to define this, the Team has analyzed the traceable features within the program that the user can easily define. Of the applicable features, 7/9 features were found to be easily traceable, leaving a percentage of 77.77%

Based on these factors, the following equation gives the correctness metric of the project.

$$\text{Correctness} = \frac{87.5+77.77}{2} = 82.635\%$$

7.3: Reliability

$$\text{Reliability} = \frac{A+S+ET}{3}$$

Where:

$$A = \text{Accuracy} = \frac{\text{Fully Compatible Features}}{\text{Total Features}}$$

$$S = \text{Simplicity} = \frac{\text{Simple Code Features}}{\text{Total Code Features}}$$

$$ET = \text{Error Tolerance} = \frac{\text{Features Accounting for Error}}{\text{Total Features}}$$

Figure 29: Reliability

Figure 28 gives the created Reliability equation based on equations found through literature review and altered for best fit to the project. Reliability is defined by Accuracy, Simplicity, and Error Tolerance for the purposes of the project. Accuracy is defined by how many of the features are fully compatible within the system. [22] Of the features tested, 15/15 are fully compatible and integrated, so the rating of Accuracy is equal to 1 or 100%. Simplicity is defined by the ease of manipulation of the source code that defines the features. [22] Of the features tested 14/16 features were considered simple, while 2 were considered complex. Complex features include linking tables to share data between pages in the interface automatically, and integration between the interface and the back-end development. Simplicity has a rating of 0.875 or 87.5%. Error Tolerance is defined by how many features account for error within the system. For this project,

error will occur with human input or data transfer. 7/16 features account for error in the project, giving error tolerance a rating of .4375 or 43.75%.

The following calculation gives the metric for Reliability.

$$\text{Reliability} = \frac{100+87.5+43.75}{3} = 77.083\%$$

7.4: Efficiency

$$\text{Efficiency} = \frac{O+EE}{2}$$

Where:

$$O = \text{Operability} = \frac{\text{Operable Features}}{\text{Applicable Features}}$$

$$EE = \text{Execution Efficiency} = 1 - \frac{\text{Average Ram Used}}{\text{Total Ram Available}}$$

Figure 30: Efficiency [22]

Figure 30 gives the created Efficiency equation based on equations found through literature review and altered for best fit to the project. For the sake of this project, Efficiency will be defined by Operability and Execution Efficiency. Execution Efficiency would normally be calculated with the hardware that will use the program, but without access to the hardware the school will be using Execution Efficiency will be estimated. Execution Efficiency has been tested on more than one system as can be seen in Chapter 6 represented by figure 25. With an average ram used of 1.55 GB and expected ram available of 4 GB, it is reasonable that the Execution Efficiency should be near 61.25% based on the systems available for analysis. The Operability is defined by “ease of operation of the software.” [22] 7/7 applicable features are considered operable by the client. Operability is given a rating of 1 or 100%.

The following calculation gives the metric for Efficiency.

$$\text{Efficiency} = \frac{100+61.25}{2} = 80.625\%$$

7.5: Integrity

$$\text{Integrity} = \frac{\text{Sec} + \text{C}}{2}$$

Sec = Security = 0

C = Completeness

Figure 31: Integrity

Figure 31 gives the created Integrity equation based on equations found through literature review and altered for best fit to the project. Security is considered a 0 for this project. There are no systems for adding security to the features within CODA. This prevents the team from being able to create a system with robust protection against attack. This limitation will be further discussed in conclusions. Auditability and Implementation are also often included in Integrity. [22] For this project, measuring the success of Implementation is not possible. There is no availability for trial runs of the system, so the factor has been removed. Auditability has also been deemed to have definition that coincides with Completeness for this project, so Completeness will be used in place of Auditability for this Software. Completeness was previously deemed to have a rating of 1 or 100%, so the calculation for Integrity would be the following.

$$\text{Integrity} = \frac{87.5 + 0}{2} = 43.75\%$$

$$\text{Useability} = \frac{O + Tr}{2}$$

Where:

O = Operability

$$\text{Tr} = \text{Training} = \frac{\text{Menus with Training}}{\text{Total Menus}}$$

Figure 32: Useability [22]

Figure 32 gives the Useability equation found through literature review and altered for best fit to the project. Operability was found previously in section 7.4 and has a rating of 1 or 100%. Training is defined by how many of the menus that exists within the program come with training on how to use them. [22] 7/7 menus have training meaning our Training factor has a rating of 1 or 100%.

The calculation for Useability is the following.

$$\text{Useability} = \frac{100+100}{2} = 100\%$$

7.6: Final Software Rating

The final software quality rating is in the next calculations based on the equation from figure 26.

$$F_q = \frac{82.635+77.083+80.625+43.75+100}{5} = 76.8186\%$$

These calculations show that the software is lacking in Product Operation for Integrity, due to lack of programmable security in Coda.io. That Reliability could be improved on and is lacking in Error Tolerance. Currently, the system lacks capability to prevent improper inputs from breaking the system. The strongest areas of the program are Useability, Correctness, and Efficiency in order. Overall, the program meets the requirements given by the client, but fails in robustness due to the limitations of the medium Coda.io, which was primarily meant to be a document sharing program.

Chapter 8: Results and Discussions:

Integrating Coda and the backend algorithm for the system proved to be one of the most difficult stages of the project. The limitations of Coda resulted in the team having to make changes to the structure of datasets in Coda. Coda does not have the ability to use whole columns with its automation. This resulted in having to make importation columns in the student data frame and the preference data frame. These columns are where the data is stored to calculate which class corresponded with what subject code. The student data frame takes the preference data from the 'StudentAssign' table on the Student View page and converts that by row to subject codes. The preference table then takes those same codes by column and copies them. The preferences are then updated in the table based on the codes. Because of this the data frame server side had to be updated to reflect this for the student and preference data frames.

A MySQL regional database server hosted on Amazon Web Services was used to contain the datasets. The structure of the server is very simple, it has a single database which contains all of the datasets used by both Coda and the backend. The team initially planned to use a service called Zapier to interface between Coda and the backend, but it proved significantly more limited than it gave the impression. Zapier requires a new link to be made for every row that needs to be automatically updated. This would take a prohibitive amount of time to institute and is not feasible for a system which can scale in size. Our test datasets only used 100 students and ten classes, but the proper implementation would require around 1700 students and 100 classes. Due to these limitations, the team decided to work with the Coda API directly to create a script which would pull the datasets from the user interface and update them after the algorithm had been run. This script was written in Python though the API itself is written in and works best with JSON. The documentation for the API [33] was very poor and made pulling data from the interface difficult. The API allowed information about tables to be retrieved in HTML, but they would have had to be manually reconstructed into dataframes. Instead, the team found a python package which makes Coda API calls significantly more usable. This package, called Blasterai or Codaio [32], let the team write single lines of code which would import the data from tables in Coda. These could then be converted into Pandas dataframes which could be uploaded to the server or interacted with directly.

The script is run through a signal dataset which contains only a single value. This value is automatically changed to a one on a specific day of the week but can also be manually changed to test or run the scripts early. The Python script waits until this value is set to one, then downloads the student preference and class offering datasets. This data is uploaded to the MySQL server, and the backend algorithm is executed using a subprocess call within Python. This algorithm was written in R, so the script had to be converted into an executable using RScript. This allows it to be run without opening RStudio. If the limitations of Zapier had been known previously, the assignment algorithm could have also been written in Python to streamline the system. This approach allowed the team to finish the integration without having to discard a significant amount of work though. Once the assignment algorithm finishes running, it uploads all of the datasets it outputs to the MySQL server and closes. The Python script downloads these datasets and uploads student assignments to Coda. A major limitation of the

Coda API is that entire tables cannot be updated automatically. Instead, they have to be updated one row at a time and too many row updates in a short time will cause the API to send an error and crash the script. Due to this, the team had to iterate over each row of the student assignment dataframe and include a 0.2 second delay between them to prevent errors.

The team was able to meet 7 requirements listed, and the interface meets the needs to fix the problems with the original flow chart in figure 2. Assignment is based on student preference and teacher recommendation, as discussed in chapter 6, preventing first come first serve from being the deciding factor behind scheduling. Teachers can check what sessions are active for them and what students are meant to be there, helping ensure that all students are where they are meant to be. Teachers that are not used for sessions can be found and reassigned to aid with independent study, helping ensure teacher utilization. The proposed flow chart in figure 9 fits with our final flow chart for the flow of information and works well as a model for what can be developed.

The following is the requirements for the system reiterated, and how the system meets them.

1: The software must allow students to select their desired subjects.

The interface in Coda allows an area in the student view page where student's select their desired subjects.

2: The software must allow teachers to select which students need additional help.

In the teacher class view, teachers are capable of using the student flag column to select a specific student to flag in the system and change the weight to make them more likely to be assigned to the subject or session they are flagged for.

3: The software must generate utilization reports for students and teachers.

The software generates utilization reports that can be pulled from the server, which also has in the title when the report was generated so that they are organized by descending order based on time by year, month, and day.

4: The software must generate and distribute weekly schedules for students and teachers.

The software uses the algorithm to generate the schedule available to students and teachers when the program is executed at the discretion of the administrator. The teacher class view makes it easy for teachers to see who is assigned to them, and the student view gives not only a calendar view with assigned sessions but informs the student of their assigned subject in the student view.

5: The software must allow teachers to select capacity for tutoring or review sessions.

In Coda, the capacity for sessions can be set to a specific capacity by typing in the capacity that they want for the session in the student capacity column in the teacher class view.

6: The software must assign all students to either individual study or review/tutoring sessions.

The software has successfully assigned students using the algorithm and has not left any generated students without an assigned subject or session for the generation.

7: The software must display each student's current subjects.

This requirement is the only requirement unfulfilled. The restrictions within Coda and the lack of a sample dataset from the school meant we were not able to integrate a display that showed their current courses and classes into the student view.

8: The software must be accessible on a phone or computer.

Navigation elements were included in Coda's pages and subpages to allow for navigation in the mobile browser.

Chapter 9: Conclusions:

The project implemented several types of software and development methods. The problem the team had to solve was how to create a web-based platform that assigned students to a session during the extended learning period that will be integrated into the fall. The major problems that the team ran into were co-operation regarding data from the client, Chamblee High School, integration between Coda.io, the servers, and the algorithm, and the inherent restrictions within Coda.io.

The analysis of the software gave a quality software rating of 76.8186%. McCall's method proved an effective analysis of the software, it can be concluded from the analysis that the software is not sufficient for implementation into the school system as is. One of the leading issues that lowers the quality of the software is security, for which Coda.io has no option. Data security is one of the most important features for any system being integrated into a school due to the massive amount of personal information that could be exposed from a leak. Overall, most requirements for the system were met, and the flow of data still alleviates the problems with the baseline solution, but the software quality shows that meeting these requirements and fixing these problems are still insufficient, as even had all requirements been met, the rating of the software quality would be equal to 79.803%. The restrictions within Coda.io lead the team to recommend that the school step away from Coda.io as a system for interface development.

Overall, with the difficulty of integration between Coda.io and other systems, as well as the security issues and program limitations, these are the recommendations the team would make for further development toward the program post development .

- A platform with better data security than Coda and individual views should be used for the user interface.
- The backend scripts need exception handling to deal with unforeseen errors from sources like the Coda API.
- The assignment algorithm should be written in Python to reduce the number of interactions with the server.
- Data collection in the user interface needs stricter controls to prevent inputs which will break the system.
- A server structure that can run the backend scripts should be used.
- The backend scripts should be called with a server variable rather than a signal dataset.
- Deeper collaboration with the customer is necessary to implement some of their desired features such as automatically displaying the classes each student is enrolled in.

References

- [1] Coda_hq, “Learn by watching · the learn doc,” *Coda*, 2019. [Online]. Available: <https://coda.io/learn/learn-by-watching-62>. [Accessed: 22-Feb-2022].
- [2] Coda_hq, “Designing the CODA formula language,” *Coda*, 2019. [Online]. Available: <https://coda.io/@coda/designing-the-coda-formula-language>. [Accessed: 23-Feb-2022].
- [3] G. L. Golfe, “Connect coda-mysql database?,” *Coda Maker Community*, 10-Aug-2020. [Online]. Available: <https://community.coda.io/t/connect-coda-mysql-database/17681>. [Accessed: 23-Feb-2022].
- [4] M. Olson, “Availability,” *Coda*, 2020. [Online]. Available: https://coda.io/d/Availability_dh6oN7oaYSZ/Availability-Tracker_suYW4?utm_campaign=embed&utm_medium=web&utm_source=h6oN7oaYSZ#_lupXl. [Accessed: 23-Feb-2022].
- [5] V. Hebert, “How to create a Smart & Automated Employee Scheduling App?,” *Coda Maker Community*, 18-Dec-2018. [Online]. Available: <https://community.coda.io/t/how-to-create-a-smart-automated-employee-scheduling-app/5368>. [Accessed: 23-Feb-2022].
- [6] Coda_hq, “API reference (1.2.4),” *Coda*, 2019. [Online]. Available: <https://coda.io/developers/apis/v1#section/Using-the-API/Resource-IDs-and-Links>. [Accessed: 23-Feb-2022].
- [7] E. K. Emam, L. Mosquera, and R. Hoptroff, *Practical Synthetic Data Generation: Balancing Privacy and the broad availability of data*. Sebastopol, CA: O'Reilly Media, Inc., 2020.
- [8] T. Akidau, S. Chernyak, and R. Lax, *Streaming systems: The what, where, when, and how of large-scale data processing*. Sebastopol, CA: O'Reilly Media, Inc., 2018.
- [9] J. Kunigk, *Architecting Modern Data Platforms: A guide to enterprise hadoop at scale*. Sebastopol, CA: O'Reilly, 2019.
- [10] A. Gorelik, *The Enterprise Big Data Lake: Delivering the promise of Big Data and data science*. Sebastopol, CA: O'Reilly Media, Inc., 2019.
- [11] M. M. Scott, J. Shakesprere, and K. Porter, “Why schools should embrace flexibility and innovation beyond covid-19,” *Urban Institute*, 28-Oct-2020. [Online]. Available: <https://www.urban.org/features/why-schools-should-embrace-flexibility-and-innovation-beyond-covid-19>. [Accessed: 22-Feb-2022].
- [12] “What are flexible pathways for learning?,” *CT.gov*. [Online]. Available: <https://portal.ct.gov/SDE/Mastery-Based-Learning/What-Are-Flexible-Pathways-for-Learning>. [Accessed: 22-Feb-2022].

- [13] T. Robbs, "Refreshing table from external data file," *Coda Maker Community*, 02-May-2018. [Online]. Available: <https://community.coda.io/t/refreshing-table-from-external-data-file/663>. [Accessed: 22-Mar-2022].
- [14] Raul_San_N.H, Dalmo_Mendonca, and Paul_Danyliuk, "Run an automation when a button is clicked," *Coda Maker Community*, 07-Aug-2019. [Online]. Available: <https://community.coda.io/t/run-an-automation-when-a-button-is-clicked/9663>. [Accessed: 22-Mar-2022].
- [15] T. Meissner, "Automation failed to run - but testing the rule works (Gmail automation)," *Coda Maker Community*, 27-Jan-2019. [Online]. Available: <https://community.coda.io/t/automation-failed-to-run-but-testing-the-rule-works-gmail-automation/6017>. [Accessed: 22-Mar-2022].
- [16] D. Schumacher, "Assigning students to courses," *R for Operations Research*, 2018. [Online]. Available: <https://www.r-orms.org/mixed-integer-linear-programming/practicals/problem-course-assignment/>. [Accessed: 22-Mar-2022].
- [17] B. Woithe, "Subpage menus - replicate across subpages," *Coda Maker Community*, 27-May-2020. [Online]. Available: <https://community.coda.io/t/subpage-menus-replicate-across-subpages/16255>. [Accessed: 12-Apr-2022].
- [18] L. Webster, "Doc Locking," *Coda Help Center*, 2022. [Online]. Available: <https://help.coda.io/en/articles/3388799-doc-locking>. [Accessed: 12-Apr-2022].
- [19] L. Webster, "Using charts in Coda," *Coda Help Center*. [Online]. Available: <https://help.coda.io/en/articles/2043825-using-charts-in-coda>. [Accessed: 12-Apr-2022].
- [20] L. Webster, "Members and roles," *Coda Help Center*, 2022. [Online]. Available: <https://help.coda.io/en/articles/3388781-members-and-roles>. [Accessed: 12-Apr-2022].
- [21] C. Huizer, "How to coda permissions?," *Medium*, 14-Sep-2021. [Online]. Available: <https://medium.com/geekculture/how-to-coda-permissions-54f6c838bf50>. [Accessed: 12-Apr-2022].
- [22] F. Panca Juniawan *et al.*, "E-Voting Software Quality Analysis with McCall's Method," *2020 8th International Conference on Cyber and IT Service Management (CITSM)*, 2020, pp. 1-5, doi: 10.1109/CITSM50537.2020.9268854.
- [23] "What is software quality? and how to achieve it?," *What is software quality*. [Online]. Available: <https://www.testbytes.net/blog/what-is-software-quality/>. [Accessed: 13-Apr-2022].
- [24] "Software Quality Factors," *Software quality factors*, 2018. [Online]. Available: https://www.tutorialspoint.com/software_quality_management/software_quality_managem

ent_factors.htm#:~:text=McCall's%20Factor%20Model,%2C%20Efficiency%2C%20Integrity%2C%20Usability. [Accessed: 12-Apr-2022].

- [25] T. Robbs, “User().Email and Crossdoc syncing of text column that contains email address,” *Coda Maker Community*, 11-May-2020. [Online]. Available: <https://community.coda.io/t/user-email-and-crossdoc-syncing-of-text-column-that-contains-email-address/15844>. [Accessed: 26-Apr-2022].
- [26] Z. Ooi, “Filtering tables and views based on controls,” *Coda Help Center*, 2022. [Online]. Available: <https://help.coda.io/en/articles/1224633-filtering-tables-and-views-based-on-controls>. [Accessed: 26-Apr-2022].
- [27] L. Webster, “Using lookups,” *Coda Help Center*, 2022. [Online]. Available: <https://help.coda.io/en/articles/1385997-using-lookups#the-lookup-column-format>. [Accessed: 26-Apr-2022].
- [28] P. Cicala, “How can I update textbox based off another column,” *Coda Maker Community*, 02-Aug-2020. [Online]. Available: <https://community.coda.io/t/how-can-i-update-textbox-based-off-another-column/17560>. [Accessed: 26-Apr-2022].
- [29] Heather_Hart, Connor_McCormick, and Iv_Bako, “Help with formula from one table to another,” *Coda Maker Community*, 06-May-2021. [Online]. Available: <https://community.coda.io/t/help-with-formula-from-one-table-to-another/23288>. [Accessed: 26-Apr-2022].
- [30] Alireza_karimi and joost_mineur, “How to copy One column data into another with desired duplications,” *Coda Maker Community*, 22-Aug-2021. [Online]. Available: <https://community.coda.io/t/how-to-copy-one-column-data-into-another-with-desired-duplications/25185>. [Accessed: 26-Apr-2022].
- [31] S. Ramaswamy, “How do I copy A ‘people’ field from one table to another?,” *Coda Maker Community*, 28-Mar-2019. [Online]. Available: <https://community.coda.io/t/how-do-i-copy-a-people-field-from-one-table-to-another/7242>. [Accessed: 26-Apr-2022].
- [32] Blasterai, “Blasterai/codaio: Python wrapper for coda api,” *GitHub*, 2020. [Online]. Available: <https://github.com/Blasterai/codaio>. [Accessed: 26-Apr-2022].
- [33] “API reference (1.2.5),” *Coda*, 2020. [Online]. Available: <https://coda.io/developers/apis/v1>. [Accessed: 26-Apr-2022].

Appendix A: Acknowledgements

Andrew Milne, Teacher at Chamblee High School

Cindy Shaw, Senior Oracle Database Administrator at Kaiser Permanente

Lois Jordan, provided contact with Cindy Shaw

Christina Scherrer, Chair of the Department of Industrial and Systems Engineering

Appendix B: Contact Information

Quinn Tobin:

quinntobin01@gmail.com

Reid Bryant:

reidmitchellbryant@gmail.com

Tristan Brown-Mulry:

tbrownmulry.tbm@gmail.com

Appendix C: Work Table

Table 4: Work division table

Names:	Quinn Tobin	Tristan Brown-Mulry	Reid Bryant
Chapters	Sections Worked		
Chapter 1: IPA	1.1, 1.3-1.6	1.1, 1.2-1.6	1.1, 1.3-1.8
Chapter 2: LA	2.1	2.1	All Sections
Chapter 3: LR	3.2	3.3, 3.4	All Sections
Chapter 4: PA	4.2, 4.4	4.2, 4.3	4.0, 4.1-4.2, 4.4, 4.5
Chapter 5: CIC	All work done	NA	NA
Chapter 6: BED	NA	All work done	NA
Chapter 7: MA	NA	NA	All work done
Chapter 8: R&D	All work	All work	All work
Chapter 9: C	All work	All work	All work

Appendix D: Reflections

Reid Bryant

The base line flow chart and proposed flow chart should have been the first task completed to better make comparisons and establish goals and requirements in the Initial Design Review. Not having this done made the development of the project during the In Progress Review incredibly difficult.

The team did not receive dataset structures until nearing the end of the project, but integration testing between Coda and Servers should have been done during the In Progress Review to ensure the programs worked as believed.

Literature review to determine a software quality model that would fit the project should have been completed before the Preliminary Design Review. Completion of this review would have made the process during the In Progress Review smoother.

Quinn Tobin

Dr. Milne suggested that we work with Coda for this project as it was the platform that best suited his needs, working in Coda turned out to be suboptimal for a few reasons. Primarily while there is detailed description of Coda's syntax on their website, there is a shocking lack of information or examples on how to develop the project. The few topics that did exist tended to yield little useful information due to Coda's site side conversation rules. Additionally, Coda does not have security features unless you are operating at the enterprise level, and what security features exist for that level would not be sufficient for our client.

Coda is primarily geared towards team meetings and automated mailing. There was only one reference to another user attempting to create something for a school database. Unfortunately, this user did not publish their document or make a template from it, so I was forced to construct the UI without an idea of what it should be modeled on.

For our data integration Coda recommended an affiliate service called Zapier. There are references to it being an effective way to push and pull data between Coda and MySQL. However, Zapier ended up being total insufficient for this project on multiple levels. Zapier ended up only able to pull by row as the integration between the two platforms is only in beta. This meant that we would have to spend upwards of \$300 per month just to have the capacity to send all the data we needed to. Even more critically too many requests to the Coda API causes it to time out resulting in a failure to transfer the data.

Tristan Brown-Mulry

This project required me to learn a number of skills I didn't have previous experience in. I had experience coding, particularly for things like operations research or statistical models, but

creating a system that could run continuously in the background to perform a task was a unique challenge. This was only made more difficult by the lack of real data. The core of the system had to be built around data structures which we did not know the format of, and we did not have contacts in the school administration to work with. This made it challenging to design a system which could be passed to them and retain functionality with minimal upkeep as the school is unlikely to devote many resources to the project.

Thanks to its challenges, I learned a lot from this project. I'm much more confident with general programming and problem solving now than I was before it. Integrating Coda and our backend algorithms forced me to learn how to work with its API and debug errors written in JSON. Setting up the MySQL database server required me to learn about how server database's function, and how to set up security, permissions, and connections to them. Working with the assignment algorithm taught me how to alter them to meet different needs and I feel confident in my ability to work with them in future situations.

Some of its flaws, like Coda's lack of data security, will likely prevent it from ever being fully implemented at the school. Dr. Milne and our team are hopeful that it can serve as a proof of concept for the idea though and lead to the school investing further in their flexible learning program. Based on our literature review, we believe that flexible learning would be a huge benefit to the students at Chamblee High School. A system like the one we designed would alleviate the administrative demands for it and allow teachers to fully focus on helping their students.

Appendix E: Backend Code

Python

```
import pandas as pd
from codaio import Coda, Document, Cell
import mysql.connector as mysql
import sqlalchemy
from sqlalchemy import create_engine
import subprocess
import time

coda = Coda('██████████')

doc = Document('██████████', coda=coda)

table = doc.get_table('grid-94eTxtPTK6')
execute = pd.DataFrame(table.to_dict())
print(execute)

while 1 == 1:
    if execute['Signal'].values[0] == 0:
        print('Sleeping for 10 seconds...')
        time.sleep(10)
        execute = pd.DataFrame(table.to_dict())
    else:
        print('Executing')
        break

    # Get tables from Coda
table = doc.get_table('grid-_GMgCzCBFW')
offer = pd.DataFrame(table.to_dict())

table = doc.get_table('grid-6mwzDx0WgU')
pref = pd.DataFrame(table.to_dict())
print('Datasets pulled from Coda')

# Connect to MySQL
cnx = mysql.connector(user='dbaccess', password='██████████',
                      host='██████████',
                      database='coda')

# create sqlalchemy engine
engine = create_engine("mysql+pymysql://{user}:{pw}@██████████/{db}"
                      .format(user="dbaccess", pw="██████████",
                              db="coda"))
print('Connection to server established')
```

```

execute.to_sql('execute', con=engine, if_exists='replace', index=False)
pref.to_sql('pref_df', con=engine, if_exists='replace', index=False)
offer.to_sql('offer_df', con=engine, if_exists='replace', index=False)
print('Datasets pushed to server')

print('Running R Script')
command = 'C:/Program Files/R/R-4.1.1/bin/Rscript.exe'
arg = 'R --vanilla'
subprocess.Popen([command, '--vanilla', 'C:/Users/.../servercheck.R'], shell=True)
print('Waiting...')
time.sleep(6)
print('R Script finished')

# Update pref_df assignments
table = doc.get_table('grid-6mwzDx0WgU')
connection = engine.connect()
metadata = sqlalchemy.MetaData()
prefs = sqlalchemy.Table('pref_df', metadata, autoload=True, autoload_with=engine)
query = sqlalchemy.select([prefs])
ResultProxy = connection.execute(query)
ResultSet = ResultProxy.fetchall()
prefs = pd.DataFrame(ResultSet)
print('Downloaded pref_df from server')
x = 0
for row in table.rows():
    y = int(prefs.iloc[x]['Assignment'])
    compperc = int(((x+1)/len(prefs))*100)
    perstr = f'{compperc}% complete.'
    print(perstr)
    name_cell_a = Cell(column='c-7sy-zL9sUO', value_storage=y)
    value_cell_a = Cell(column='c-7sy-zL9sUO', value_storage=y)
    table = doc.get_table('grid-6mwzDx0WgU')
    table.update_row(row, [name_cell_a, value_cell_a])
    x = x+1
    time.sleep(0.2)
print('Updated pref_df in Coda')

# Update execute table
name_cell_a = Cell(column='c-wqGj7GCZ99', value_storage='0')
value_cell_a = Cell(column='c-wqGj7GCZ99', value_storage='0')
table = doc.get_table('grid-94eTxtPTK6')
row = table['i-SrgjWb45G_']
table.update_row(row, [name_cell_a, value_cell_a])
print('Updated execute in Coda')
print('Code finished')

```

RStudio

```
#!/ C:/Program Files/R/R-4.1.1/bin/Rscript.exe
```

```
# install.packages('dplyr')
# install.packages('tibble')
# install.packages('lpSolve')
# install.packages('ggplot2')
# install.packages('purrr')
# install.packages('viridis')
# install.packages('ompr')
# install.packages('ompr.roi')
# install.packages('ROI.plugin.glpk')
# install.packages('tictoc')
# install.packages('RMySQL')
```

```
library(RMySQL)
```

```
mysqlconnection <- dbConnect(MySQL(), user='dbaccess', password='[REDACTED]', dbname='
'coda', host='[REDACTED]', port=3306)
```

```
dbListTables(conn=mysqlconnection)
```

```
execute <- dbReadTable(conn=mysqlconnection, name='execute')
```

```
# Loop to check whether to execute code
# while (0 == 0)
# {if (execute$signal == 0)
# {
#   print('System waiting')
#   Sys.sleep(10)
#   execute <- dbReadTable(conn=mysqlconnection, name='execute')
# }
#
# else
# {
#   print('Executing')
#   break
# }
# }
```

```
student_df <- dbReadTable(conn=mysqlconnection, name='student_df')
```

```
teacher_df <- dbReadTable(conn=mysqlconnection, name='teacher_df')
```

```
offer_df <- dbReadTable(conn=mysqlconnection, name='offer_df')
```

```
prefs <- dbReadTable(conn=mysqlconnection, name='pref_df')
```

```
pref_misc = subset(prefs, select = c(Student_ID, Code_1, Code_2, Code_3, Teacher_Code))
```

```

pref_df = subset(prefs, select = -c(Code_1, Code_2, Code_3, Teacher_Code))

n <- nrow(student_df) # 100

m <- (nrow(offer_df)-1)

capacity <- offer_df$Capacity

pref_df$p11 <- 1

# Convert pref_df to list
preference_data <- vector("list", nrow(pref_df))
names(preference_data) <- pref_df$Student_ID
preference_data[] <- 11

for (x in 1:nrow(pref_df)) {
  sid <- pref_df$Student_ID[x]
  for (y in 2:4) {
    cl <- which(pref_df[x,2:(2+(m-1))] == y)
    preference_data[[sid]] <- c(preference_data[[sid]], cl)
  }
}

preferences <- function(student) preference_data[[student]]
flagged <- function(student) pref_df$Flag[[student]]
which(as.numeric(6) == flagged(as.numeric(7)))

# Weight function
weight <- function(student, course) {
  p <- which(as.numeric(course) == preferences(as.numeric(student)))
  f <- which(as.numeric(course) == flagged(as.numeric(student)))
  as.integer(if (length(p) == 0 & length(f) == 0) {
    -100000
  } else if (length(f) >= 1) {
    15
  }
  else {
    p
  })
}

library(ggplot2)
library(purrr)
library(dplyr)
library(viridis)
plot_data <- expand.grid(

```



```

course = seq_len(m),
weight = c(2, 3, 4, 15)
)%>% rowwise() %>%
mutate(count = sum(map_int(seq_len(n), ~weight(.x, course) == weight))) %>%
mutate(course = factor(course), weight = factor(weight))
ggplot(plot_data, aes(x = course, y = count, fill = weight)) +
geom_bar(stat = "identity") +
viridis::scale_fill_viridis(discrete = TRUE)

library(ompr)
model <- MIPModel() %>%

# 1 if student i is assigned to course m
add_variable(x[i, j], i = 1:n, j = 1:(m+1), type = "binary") %>%

# maximize the preferences
set_objective(sum_expr(weight(i, j) * x[i, j], i = 1:n, j = 1:(m+1))) %>%

# we cannot exceed the capacity of a course
add_constraint(sum_expr(x[i, j], i = 1:n) <= capacity[j], j = 1:(m+1)) %>%

# each student needs to be assigned to one course
add_constraint(sum_expr(x[i, j], j = 1:(m+1)) == 1, i = 1:n)

# model

library(ompr.roi)
library(ROI.plugin.glpk)
result <- solve_model(model, with_ROI(solver = "glpk", verbose = TRUE))

matching <- result %>%
get_solution(x[i, j]) %>%
filter(value > .9) %>%
select(i, j) %>%
rowwise() %>%
mutate(weight = weight(as.numeric(i), as.numeric(j)),
preferences = paste0(preferences(as.numeric(i)), collapse = ",")) %>% ungroup

print(matching)

matching %>%
group_by(weight) %>%
summarise(count = n())

plot_data <- matching %>%
mutate(course = factor(j), weight = factor(weight, levels = c(1, 2, 3, 4, 15))) %>%

```

```

group_by(course, weight) %>%
summarise(count = n()) %>%
tidyr::complete(weight, fill = list(count = 0))
ggplot(plot_data, aes(x = course, y = count, fill = weight)) +
geom_bar(stat = "identity") +
viridis::scale_fill_viridis(discrete = TRUE)

for (x in 1:nrow(pref_df)) {
  rn <- matching$i[x]
  pref_df$Assignment[rn] <- matching$j[x]
}

for (c in 1:m) {
  offer_df$Utilization[c] <- (table(pref_df$Assignment)[c]/offer_df$Capacity[c])
}

year <- format(Sys.Date(), '%Y')
month <- format(Sys.Date(), '%m')
day <- format(Sys.Date(), '%d')
name <- paste('offer_df', year, month, day, sep='_')

dbWriteTable(conn=mysqlconnection, name=name, value=offer_df, row.names=FALSE,
overwrite=TRUE,
  field.types = c(Class_Code="INT(11)", Teacher_ID="TEXT",
Subject_Code="INT(11)",
Capacity="INT(11)", Utilization="DOUBLE(3, 2)"))
prefs$Assignment <- pref_df$Assignment
dbWriteTable(conn=mysqlconnection, name='pref_df', value=prefs, row.names=FALSE,
overwrite=TRUE)

```