COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

FLORIDA ATLANTIC UNIVERSITY

PRINCIPLES OF SOFTWARE ENGINEERING

CEN4010 – Spring 2023

MILESTONE 3 – More Detailed Requirements, Architecture, & Vertical Software Prototype

(#23) TEAM NIGHTSHIFT:

Name	Email
Adam Clark	cclark36@fau.edu
Quang Le	leq2019@fau.edu
Nicholas Moussa	nmoussa2017@fau.edu
Mahmood Sakib	msakib2022@fau.edu
Kyla Tolentino	ktolentino2021@fau.edu

Revision History	Date
n/a	*n/a*

➢ **Submitted on 03/31/23 by Adam Clark**

Table of Contents

Executive Summary ... 4

Competitive Analysis ... 5

Data Definitions ... 7

Overview, Scenarios, and Use Cases ... 11

Use Case – Sign Up ... 11

Use Case – Login ... 12

Use Case – Logout ... 13

Use Case – Add Course ... 14

Use Case – Report Issue ... 15

Use Case – Grade Entry ... 16

Use Case – What If? ... 17

List of High-Level Functional Requirements ... 18

Unregistered User ... 18

Registered User ... 19

List of Non-Functional Requirements ... 22

Interoperability Requirements ... 22

Storage Requirements: ... 22

Security Requirements ... 23

Supportability Requirements ... 23

Availability Requirements ... 23

High-Level System Architecture ... 24

Visual Studio (IDE) ... 24

OpenWeatherMap ... 25

Node.js ... 25

GitHub/Git ... 26

Firebase 26
Jira 26
Discord........ 26
Canvas 26
High-Level UML Diagrams 27
Component & Deployment Diagram........ 27
Database Organization 28
Significant Algorithms........ 29
Risk Assessment 32
Schedule Risks 32
Technical Risks........ 33
Teamwork Risks........ 33
Legal/Content Risks........ 34
Team Nightshift Members 35
Project Links........ 35
GitHub Repository: 35
Team Web Page: 35
Jira Dashboard:........ 35
Vertical Demo:........ 35
Acknowledgements 35
Appendix........ 36

Figure 1 – Grade PlanAlyzer Logo

Executive Summary

As students, we all know from personal experience that there are always moments toward the end of any given semester in which we inevitably whip out our calculators to perform a series of "doomsday" GPA number crunchings. Questions arise such as "What do I need to score on this final to pass the course?" or, hopefully, "What's the absolute *worst* I can do and still retain my 'A'?" Such questions are important in prioritizing one's time and establishing goals, especially when coinciding project and exam deadlines roll in for those already juggling multiple courses and responsibilities from family to work! "But *where* are the answers to all these pressing issues?" you might ask. Why, the how, the what, the when, and the where are all within the Grade PlanAlyzer, of course – it just needs you!

The Grade PlanAlyzer application is our solution to students' woes by creating a platform tailored to providing them a more robust, proactive, and quantitative approach to academic planning throughout each semester. By retaining all of one's pertinent academic information, the student not only avoids having to manually transcribe large quantities of data on a case-by-case basis to calculate grading hypotheticals, but is also afforded the luxury of attaining more dynamic and impactful results via the application of data analytics. In other words, the student will remain keenly aware of one's academic standing throughout the entirety of a given semester such that they may make informed decisions to guide their path forward, rather than defensively access such information to backpedal a semester gone awry. The Grade PlanAlyzer will enable students to optimize their time management skills and dedicate a more appropriate amount of time to each assigned task. This approach will enable them to ensure that they are allocating their efforts efficiently and effectively, ultimately resulting in better overall performance and academic outcomes.

Competitive Analysis

The analysis of competitors' web sites will be quantitatively assessed based upon the following six primary features: User Experience, UI Design, Data Storage, Calculation Complexity, Customization, and What-If? functionalities. The criteria are scored on a 0 - 5 scale (0 = not present, 1 = awful, 2 = poor, 3 = fair, 4 = good, 5 = outstanding) and are applied to five software applications chosen for their thematic similarities involving grade calculations. The results of this assessment are displayed in Table 1, below.

Table 1 - Concept Scoring Matrix

	Grade PlanAlyzer	Canvas	Calculator.net	Fourpoint	Grades	The Grade Calculator
User Experience	4	5	2	4	4	2
UI Design	5	3	2	4	3	2
Data Storage	4	5	0	4	4	0
Calculation Complexity	5	4	3	4	3	1
Customization	5	3	0	5	4	0
What-If?	5	3	2	0	0	0
Mean	**4.67**	**3.67**	**1.50**	**3.50**	**2.83**	**0.83**

Canvas (3.67) - https://www.instructure.com/canvas

At present, Canvas falls short in providing students with essential features to optimize their academic performance. Specifically, the platform lacks the capability to calculate grades per assignment, limiting students' ability to allocate their time efficiently and accurately gauge their progress. Additionally, Canvas does not provide students with the ability to determine the minimum score required to achieve an A in a specific assignment, depriving them of essential information needed to set realistic goals and make informed decisions. By contrast, an advanced grade calculator app can address these shortcomings, empowering students to take a more proactive approach to their studies, optimize their learning experience, and achieve their full potential.

Fourpoint (3.50) - https://apps.apple.com/us/app/fourpoint-a-gpa-calculator/id383417299

Fourpoint is an efficient and user-friendly mobile application designed to make calculating one's GPA easy and accurate. It is beneficial for students who attend schools that use the supported GPA calculation system and allows for additional semesters and courses as one progresses throughout their academic career. Fourpoint has several useful features, including the ability to designate a course as Pass/Fail or not count toward one's GPA, setting the weight of grades, and determining one's major GPA. Users can also designate courses as in progress and add past GPA history with Q-Points and Hours. Fourpoint enables calculation of both a single semester GPA and a cumulative GPA for one's entire educational career. However, it may not be useful for students who have unique academic circumstances or situations that require more complex GPA calculations.

Grades (2.83) - https://apps.apple.com/us/app/grades-view-your-scores/id1434492452

The Grades App is a helpful tool for students who are looking to track their academic progress. With features such as automatic GPA calculation, the app can provide users with a quick and easy way to get an overview of their semester grades. However, it seems that the app may have some limitations. For example, the user interface may lack personality and customization options, which could be a disadvantage for some users who are looking for a more personalized experience. Additionally, the app may not account for customizable grading scales, which could be an issue for users who have unique grading systems in their schools or courses. Finally, the app's analytics may be somewhat inaccurate, which could impact the reliability of the data presented.

Calculator.net (1.50) - https://apps.apple.com/us/app/grades-view-your-scores/id1434492452

The visual appeal of Calculator.net is lacking and requires an excessive amount of manual data input due to the website's lack of dynamism. This means that each assignment grade must be manually selected and typed in for calculation, resulting in a time-consuming and frustrating process that can lead to errors and decreased efficiency. This is particularly problematic for individuals who need to calculate grades frequently, as it can impede their workflow and productivity. Calculator.net isn't user-based and requires complete manual entry for every grading query, also only represents a singular built-in grading scale so it is not customizable to the user.

TheGradeCalculator (0.83) - https://thegradecalculator.com/

The Grade Calculator tool has some useful features, but there are also some significant drawbacks that should be considered. Firstly, the user interface is outdated and not very user-friendly, which could make it difficult for some users to navigate. Additionally, the tool is not consolidated into one location and requires users to visit multiple websites to access all of the features they need. This can be time-consuming and inconvenient for users. Another major disadvantage of the tool is the lack of a login feature, which means that there is no user and data relationship. This could be frustrating for users who want to save their progress and access it later, as they will have to input their data every time they revisit the website. Finally, the tool lacks infographic capabilities, which could be a significant disadvantage for users who want to present their data in a more engaging and visual format.

Grade PlanAlyzer (4.67) - https://q-wrld97.github.io/cen4010_sp23_g23/
Planned Advantages:

The What-If? tool offered by the Grade PlanAlyzer is particularly advantageous in that its calculations will always be derived from the latest empirical dataset at one's disposal. Many competitors, on the other hand, simply attempt to provide a crude letter grade estimate based solely upon input of hypothetical score averages by weighted category. Moreover, such tools are further limited due to their complete lack of customization options to account for various calculation intricacies, such as a unique grading scale, extra credit, or even complex rules such as replacing the lowest deliverable score in a category with the average of the remainder. However, the Grade PlanAlyzer maintains the real-time authenticity of such What-If? estimates by automatically adjusting its calculations to display results under the context of remaining scoring opportunities after factoring in the weight of what events have already transpired.

Data Definitions

Table 2 - Table of Data Definitions

Name	Meaning	Usage	Comment
Registered User	Actor	*Use Case Scenarios*	A student who has registered with the system
Unregistered User	Actor	*Use Case Scenarios*	A student who has not registered with the system
System	Platform Hardware and Services	*Use Case Scenarios*	The collective intercommunicating software and hardware components forming a computer system, including front-end and back-end code, configuration files, documentation, and Firebase database
Website	User Interface	*User Interface*	Front-end display for user interaction
Landing Page	User Interface	*User Interface*	A standalone web page which serves as the entry point for the website
Homepage	User Interface	*User Interface*	Primary web page the user sees when not logged in, featuring a linked navigation bar
Navigation Bar	Service/User Interface	*Site User Service*	Fixed toolbar display containing a variety of site navigation pathways (Dashboard, Report Issue, Account Settings, Log Out, etc.)

Dashboard	Service/User Interface	*Site User Service*	Primary hub for displaying student information and tools
Account	Data	*Use Case Scenarios*	Refers to the location on a network server storing the user ID, password, and composite information associated with an individual user
Login	Service	*Site User Service*	Permits user to access their stored personal account data and grade analysis tools
Session	Service	*Site User Service*	Period of activity between a user logging in and logging out of a multi-user system
Logout	Service	*Site User Service*	Ends a user's login session and redirects to home page
Sign Up	Service	*Site User Service*	Process by which an unregistered user may become a registered user
User ID	Data	*Use Case Scenarios*	A unique logical entity and name of respective user's email address used to identify and distinguish between users
Password	Data	*Use Case Scenarios*	A confidential, user-defined string of characters used to authorize access to the account of an associated UserID
Forgot Password	Service	*Site User Service*	A self-service account recovery process by which a user may choose to create a new password after clicking the password reset link

Password Reset Link	Off-site Service	*Off-site User Service*	An automated link sent to a user's registered email address that contains a unique token identifying the user which is authenticated by second factor when the link is clicked
Confirmation Email	Off-site Service	*Off-site User Service*	An automated email triggered upon registration of a new user which must be addressed within 24 hours to verify user's identity and initiate a permanent account
Account Settings	Data/Service	*Use Case Scenarios/ Site User Service*	The configurations and settings associated with an authorized account which are established upon initial enrollment, but may also be changed at a later date
Course Form	Data	*Use Case Scenarios*	A formatted document containing blank fields in which users must input corresponding data necessary for site function, which is then stored in the database
FAQ	Service	*Site User Service*	Short for 'frequently asked questions', offers users a list of questions and answers relating to basic website operation
Report Issue	Service	*Site User Service*	Means by which a user may report bugs and issues experienced during website operation to its creators
Look Ahead	Service	*Site User Service*	A dashboard tool which contains a forecast of weather and to-do tasks

Infographics	Service	*Site User Service*	A dashboard tool which allows the user to view more advanced visual representations of their grading data, e.g. charts, diagrams, etc.
Grade Entry	Service	*Site User Service*	A dashboard tool whereupon a user may enter new or view/edit existing grade data
Course Info	Service	*Site User Service*	A dashboard tool which serves as the content hub for all actions related to course information within a given semester,
What If?	Service	*Site User Service*	A dashboard tool which allows for dynamic calculation of a user's potential grade based upon existing grade data
GitHub	Repository Hosting	*Use Case Scenarios*	An internet hosting service for software development and version control using Git whereupon the project code repository and documentation are stored and organized
GitHub Pages	Production Server	*Use Case Scenarios*	The website hosting service on which this project is published
Firebase	Database/Back-End Framework	*Use Case Scenarios*	A backend-as-a-service (Baas) which provides real-time database hosting and user authentication services
Node.js	Production Server	*Use Case Scenarios*	A persistent back-end server responsible for account management and dispatch of email notifications

Overview, Scenarios, and Use Cases

Use Case – Sign Up

A new unregistered user arrives at the homepage and wishes to create an account via the ‘Sign Up’ button on the navigation bar in order to gain access to the website’s many features.

1. Description:

Use case describes the process of how the unregistered user will become a registered account holder with the system.

2. Actors:

2.1 Unregistered User
2.2 System

3. Preconditions:

3.1 User has an active internet connection
3.2 System is available

4. Primary Flow of Events:

4.1 User arrives on website homepage
4.2 User clicks ‘Sign Up’ button from navigation bar
4.3 User enters new ‘UserID’, ‘Password’, and ‘Repeat Password’
4.4 User clicks ‘Submit’ button
4.5 Temporary account is created to expire after 24 hours without email confirmation
4.6 System sends automated confirmation email to address listed as UserID
4.7 User clicks confirmation email link to successfully establish a permanent account
4.8 Terminate Use Case – Sign Up

5. Alternate Flows:

5.1 User Enters Prohibited Format into UserID Field

If in Step 4.3, user enters a UserID without including ‘@’ and ending in a domain

1. Website notifies the user that a valid email address must be utilized
2. Return to Step 4.3

5.2 User Enters Prohibited Format into Password Fields

If in Step 4.3, user enters a Password that is less than six characters

1. Website notifies the user that passwords must be at least six characters
2. Return to Step 4.3

If in Step 4.3, user enters a Repeat Password that does not match Password

1. Website notifies the user that current passwords do not match
2. Return to Step 4.3

5.3 User Neglects to Address Confirmation Email

If in Step 4.7, does not complete confirmation email within 24 hours

1. The system deletes the user's temporary account
2. Return to Step 4.2

Use Case – Login

A registered user arrives at the homepage and wishes to log into their existing account via the 'Login' button in order to gain access to the website's many features.

1. Description:

Use case describes the process of how the registered user will access their account.

2. Actors:

2.1 Registered User
2.2 System

3. Preconditions:

3.1 User has an active internet connection
3.2 System is available
3.3 User has a permanent account

4. Primary Flow of Events:

4.1 User arrives on website homepage
4.2 User clicks 'Login' button from navigation bar
4.3 User enters their account 'UserID' and 'Password'
4.4 User clicks 'Login' button
4.5 Website successfully redirects to user's account dashboard
4.6 Terminate Use Case – Login

5. Alternate Flows:

5.1 User Performs Invalid UserID and Password Combination

If in Step 4.4, user submits any combination which is not an exact system match

1. Website notifies the user that the current UserID and Password pairing does not exist in the database
2. Return to Step 4.3

5.2 User Does Not Know Account Password

If in Step 4.3, user knows their UserID but is unable to recall its Password

1. User clicks 'Forgot Password' button beneath 'Login'
2. User enters UserID into the 'Email Address' field
3. User selects 'Submit' button
4. Website notifies the user that a password reset link has been sent
5. User clicks password reset link from their primary email address
6. Redirected user enters new matching 'Password' and 'Repeat Password'

7. Go to Step 4.1

5.3 User Performs First-Time Login

If in Step 4.5, user is logging into their account for the first time

1. Website successfully redirects user to complete 'Account Settings' form
2. User completes 'Account Settings' form and clicks 'Submit'
3. Website successfully redirects user to complete new 'Course Form(s)'
4. User completes 'Course Form(s)' and clicks 'Submit'
5. Go to Step 4.5

Use Case – Logout

A registered user who is currently logged in has accomplished their desired interactions with the system and wishes to terminate their current session to secure their personal information.

1. Description:

Use case describes the process of how the registered user will end their current session.

2. Actors:

2.1 Registered User
2.2 System

3. Preconditions:

3.1 User has an active internet connection
3.2 System is available
3.3 User has a permanent account
3.4 User is logged into the system

4. Primary Flow of Events:

4.1 User logs into their account
4.2 Website successfully redirects to user's account dashboard
4.3 User decides to end their current session
4.4 User clicks 'Logout' button from navigation bar
4.5 The system terminates the user's current login session
4.6 Website successfully redirects to the homepage
4.7 Terminate Use Case – Logout

5. Alternate Flows:

5.1 User Remains Inactive for a Prolonged Interval

If in Step 4.2, user remains inactive for 15 consecutive minutes

1. The system automatically terminates the user's current login session
2. Return to Step 4.6

Use Case – Add Course

A registered user is in possession of their course syllabi and wishes to input their course data into the system.

1. Description:

Use case describes the process of how the registered user will add courses.

2. Actors:

2.1 Registered User
2.2 System

3. Preconditions:

3.1 User has an active internet connection
3.2 System is available
3.3 User has a permanent account
3.4 User is logged into the system

4. Primary Flow of Events:

4.1 User logs into their account
4.2 Website successfully redirects to user's account dashboard
4.3 User selects 'Course Info' button ('ⓘ' icon) from the dashboard
4.4 User selects 'Add Course' tile ('**+**' icon)
4.5 User accurately inputs course information
4.6 User clicks 'Submit'
4.7 User assigns due dates for each course deliverable specified on prior form
4.8 User clicks 'Submit'
4.9 Course information is successfully stored in the database
4.10 Terminate Use Case – Add Course

5. Alternate Flows:

5.1 User Enters Prohibited Format into Numerical Fields

If in Step 4.5, user enters a percentage value which is not between 1 and 100

1. Website notifies the user that percentages must adhere to 1 ≤ % ≤ 100
2. Return to Step 4.5

If in Step 4.6, user has toggled a 'Custom' grading scale and left thresholds blank

1. Website notifies user that custom grading scale thresholds cannot be empty
2. Return to Step 4.5

If in Step 4.6, user has toggled a 'Custom' grading scale and entered a grade threshold which is higher than the letter grade which alphabetically precedes it

1. Website notifies user that subsequent letter grades must have lower bounds
2. Return to Step 4.5

If in Step 4.6, user has toggled a grading category but failed to specify a quantity

1. Website notifies user that numerical quantity value must be ≥ 1.
2. Return to Step 4.5

If in Step 4.6, user has toggled a grading category but failed to specify a weight

3. Website notifies user that a weight value between 1 ≤ % ≤ 100
4. Return to Step 4.5

If in Step 4.6, user has input weight category values which do not add up to 100

1. Website notifies user that cumulative weight category values must be 100
2. Form submission is rejected
3. Return to Step 4.5

5.2 User Neglects to Address Any Form Data Request

If in Step 4.6, user has failed to address any data request within the form

1. Website notifies the user that all requests for information must be addressed
2. Form submission is rejected
3. Return to Step 4.5

Use Case – Report Issue

A registered user encounters a website performance issue and wishes to communicate with developers, or to provide feedback on the design, functionality, and usability of the website.

1. Description:

Use case describes the process by which the user may communicate with developers.

2. Actors:

2.1 Registered User
2.2 System

3. Preconditions:

3.1 User has an active internet connection
3.2 System is available
3.3 User has a permanent account
3.4 User is logged into the system

4. Primary Flow of Events:

4.1 User logs into their account
4.2 Website successfully redirects to user's account dashboard
4.3 User selects 'Report Issue' button from navigation bar
4.4 User accurately inputs required correspondence information
4.6 User clicks 'Submit'
4.7 System sends incident report email to developers' communal inbox
4.8 Website successfully redirects to user's account dashboard
4.9 Terminate Use Case – Report Issue

Use Case – Grade Entry

A registered user has received their latest grade feedback and wishes to input its data into the system.

1. Description:

Use case describes the process of how the registered user will enter grade data.

2. Actors:

2.1 Registered User
2.2 System

3. Preconditions:

3.1 User has an active internet connection
3.2 System is available
3.3 User has a permanent account
3.4 User is logged into the system
3.5 User has added at least one course

4. Primary Flow of Events:

4.1 User arrives on website homepage
4.2 User logs into their account
4.3 Website successfully redirects to user's account dashboard
4.4 User selects 'Grade Entry' button ('📃' icon) from the dashboard
4.5 System automatically generates shorthand list of eligible score input fields entitled "Category Name> #<Number>:", grouped by course, for any deliverable which is both currently past its due date and lacking existing grade data
4.6 User inputs the associated score(s) as percentage value(s) up to 4 decimal places
4.7 User clicks 'Submit'
4.8 Grade data is successfully stored in the database
4.9 Terminate Use Case – Grade Entry

5. Alternate Flows:

5.1 User Enters Prohibited Format Into Grade Entry Field

If in Step 4.6, user enters a numerical score which is not between 0 and 100

1. Website notifies the user that scores must adhere to 0 ≤ SCORE ≤ 100
2. Return to Step 4.6

5.2 User Needs to Adjust Prior Grade Submission

If in Step 4.5, the user needs to adjust a prior grade submission, the desired field will not automatically generate due to its having existing grade data

1. User selects 'Show All Previous' toggle option within 'Grade Entry' dashboard
2. User inputs updated score(s) in place of existing scores for relevant field(s)
3. Return to Step 4.7

5.3 User Cannot Locate Desired Grade Entry Field

If after Alternate Flow 5.2, the user still cannot locate desired grade entry field

1. User selects 'Course Info' button from the dashboard
2. User selects the 'Edit' (✎ icon) button from the course tile corresponding to the missing grade field
3. User ensures the appropriate grade categories and quantities are designated for inclusion of the desired grade entry field
4. User clicks 'Submit'
5. User ensures an appropriate past deadline is attributed to the desired grade entry field
6. User clicks 'Submit'
7. Proper course configuration data is successfully stored in the database
8. Website redirects to user's account dashboard
9. Return to Step 4.4

Use Case – What If?

A registered user wishes to run potential overall grade calculations based upon hypothetical weighted category scores for the remainder of ungraded deliverables within a given semester.

1. Description:

Use case describes the process by which the user may run hypothetical calculations.

2. Actors:

2.1 Registered User
2.2 System

3. Preconditions:

3.1 User has an active internet connection
3.2 System is available
3.3 User has a permanent account
3.4 User is logged into the system
3.5 User has added at least one course

4. Primary Flow of Events:

4.1 User logs into their account
4.2 Website successfully redirects to user's account dashboard
4.3 User selects 'What If' button ('?' icon) from the dashboard
4.4 User adjusts weighted category sliders to desired grade values
4.6 User clicks 'Calculate'
4.7 System displays resulting overall course letter grade and score percentage
4.8 Terminate Use Case – What If?

List of High-Level Functional Requirements

Unregistered User

1. <u>Account Creation</u>

1.1 Procedural Steps

- I. User selects ‘Sign Up’ button from the homepage navigation bar
- II. User enters a new UserID (same as valid email address)
- III. User enters a Password:
    - A. System will check if password is *at least* six characters in length:
        - i. Meets Criteria – Borders of password input field box returns to default settings; Go to Step IV
        - ii. Fails Criteria – Borders of password input field box turn red and a notification appears informing the user that passwords must be *at least* six characters in length; Go to Step III
- IV. User re-enters Password:
    - A. System will check if both password fields match:
        - i. Passwords Match – Borders of password input field boxes return to default settings; Go to Step V
        - ii. Passwords Do Not Match – Borders of password input field boxes turn red and a notification appears informing the user of non-matching passwords; Go to Step III
- V. User confirms via ‘Submit’ selection
- VI. System will check if UserID is available:
    - A. Available –
        - i. System will send email prompting confirmation to user.
        - ii. A temporary account will be created in Firebase
        - iii. User will be notified to address confirmation email within 24 hours to prevent account deletion; Go to Step VII
    - B. Unavailable – borders of UserID input field box turn red and a notification appears informing the user of UserID unavailability and to try another UserID; Go to Step II
- VII. User may choose to confirm account validity via primary email:
    - A. User Confirms –
        - i. User clicks confirmation link within 24 hours
        - ii. System permanently saves UserID and password → End of process
    - B. User Does Not Confirm –
        - i. User does not click confirmation link within 24 hours
        - ii. System will delete account information at 24 hour mark → End of process

Registered User

2. Account Login

2.1 Login Procedure

I. User selects 'Login' button from the homepage navigation bar
II. User enters UserID
III. User enters Password associated with UserID
IV. User selects 'Login' button:
V. System will check if the entered UserID & Password pairing matches:
    A. Valid Login Combination – System shall redirect user to the dashboard → End of process
    B. Invalid Login Combination – System shall alert user that current UserID & Password combination is not found in the database; Go to Step II

2.2 Password Recovery

I. User selects 'Forgot Password?' button beneath the 'Login' button
II. System will redirect user to 'Password Reset' page
III. User enters UserID into the Email Address field
IV. User selects 'Submit' button
V. System will notify user that "a password reset link has been sent to the provided email address if the UserID exists within the system"
VI. System will check if UserID exists in the system:
    A. UserID Exists – System shall send a password reset link to the provided email address; Go to Step VII
    B. UserID Does Not Exist – System shall take no further action → End of process
VII. User clicks password reset link from their primary email address
VIII. System will redirect user to 'New Password' page
IX. User enters new Password:
    A. System will check if password is *at least* six characters in length:
        i. Meets Criteria – Borders of password input field box returns to default settings; Go to Step X
        ii. Fails Criteria – Borders of password input field box turn red and a notification appears informing the user that passwords must be *at least* six characters in length; Go to Step IX
X. User re-enters new Password:
    A. System will check if both password fields match:
        i. Passwords Match – Borders of password input field boxes return to default settings; Go to Step XI

ii. Passwords Do Not Match – Borders of password input field boxes turn red and a notification appears informing the user of non-matching passwords; Go to Step IX

XI. System redirects user to the application homepage to login with new password → End of process

3. Data Entry

3.1 Account Settings ('⚙' icon)

I. User selects 'Account Settings' button (cogwheel icon) from the navigation bar
II. User enters name of school
III. User selects current semester and year
IV. User confirms existing GPA scale or enters new parameters
V. User may opt to enter phone number for text notifications
VI. User clicks 'Submit' → End of process

3.2 Course Form ('+' Add Course icon)

3.2.1 Course Name

I. User inputs a three-letter acronym
   A. On input system will automatically capitalize and limit input length to 3 characters

3.2.2 Term Length

I. User clicks on field
II. Dropdown appears
III. User may choose from list of pre-designated options

3.2.3 Credit Hours

I. User clicks on field
II. Dropdown appears
III. User may choose from list of pre-designated options

3.2.4 Class Days

I. User selects either day(s) of the week or remote checkbox option
   A. System will check if remote option is selected:
      i. Selected – unchecks and prevents further interaction with options involving days of the week
      ii. Unselected – allows user to select day(s) of the week

3.2.5 Grading Scale

I. User selects either from two pre-made grading scales or a custom grading scale
   A. System will check if custom option is selected:
      i. Selected – custom grading scale input fields appear
      ii. Unselected – custom grading scale input fields disappear

3.2.6 Weighted Categories

- I. User selects the categories of weighted groupings
  - A. System will check if any options are selected:
    - i. For EACH Selected – two fields appear:
      - a. Quantity – User inputs the numerical total of items corresponding to category, and system will check:
        1. < 1 – notifies user numerical value must be greater than or equal to 1
        2. > 1 – nothing
      - b. Percentage – User inputs the numerical percentage of total grade corresponding to category, and system will check:
        1. < 1 or > 100 – notifies user numerical value must be between 1 and 100
        2. > 1 and < 100 – nothing
    - ii. For EACH Unselected – two fields disappear

3.2.7 Form Submission

- I. User clicks 'Submit' and input validation is executed
  - A. System will check if 'Custom' grading scale custom is selected:
    - i. True – System will check grading bounds for each field in hierarchal order from A-F for numerical legitimacy:
      - a. If ANY Above < Below, > 100, or < 1 – Notify user of invalid parameters; Go to Step 3.2.5
      - b. If ALL Above > Below and between 1 and 100 – Condition Passed; Go to Step B
    - ii. False – Go to Step B
  - B. System will sum Weighted Category percentage fields and evaluate if total equals 100:
    - i. True – Go to Step II
    - ii. False – Notify user that total percentages must equal 100; Go to Step 3.2.6
- II. User data is successfully stored in the database → End of process

3.3 Course Dates ('📅' icon)

- I. The Course dates form immediately follows submission of Course Form
- II. A new date field is generated entitled "<Category Name> #<Number>:" for each individual course deliverable based upon the item quantity designated for each weighted category on the previous Course Form
- III. User interacts with the date picker to assign the deadline for each item
- IV. User clicks 'Submit' and data is successfully stored → End of process

3.4 Grade Entry ('📋' icon)

I. User selects 'Grade Entry' button (report card icon) from the dashboard
II. A new grade field is generated entitled "<Category Name> #<Number>:" for each deliverable item with a deadline prior to the current date
III. For each deliverable item with known scoring feedback, the user may input the corresponding percentage value between 1 and 100 up to four decimal places
IV. User clicks 'Submit' and data is successfully stored → End of process

List of Non-Functional Requirements

Interoperability Requirements

Browser Compatibility – The system will be a we-based web app that operates on major browsers, including Google Chrome, Mozilla Firefox, Safari, Opera, Brave, and Internet Explorer.

Operating System Compatibility – The system will operate on multiple operating systems, including Windows, Linux, and OS X.

Mobile Operating Sytem Compatibility – The system will operate on all major mobile operating systems, including iOS and Android.

Storage Requirements:

Back-End Maximum Database Load:

- *Storage Capacity* – 1 GiB
- *Document Reads* – 50,000 per day
- *Document Writes* – 20,000 per day
- *Document Deletes* – 20,000 per day
- *Network Egress* – 10 GiB per month

Back-End Usage Load for Authentication and Authorization:

- *Daily Active Users* – 3000 per day
- *SMS Sent* – 10 per day per user
- *Multi-Factor Authentications* – 10 per day per user

Security Requirements

Our user authentication/authorization system will maintain the private integrity of site users' academic information and preferences on an individual basis.

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to any app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. The Firebase Authentication software development kit (SDK) provides methods to create and manage users that use their email addresses and passwords to sign in. Firebase Authentication also handles sending password reset emails.

Firebase Security Rules leverage extensible, flexible configuration languages to define what data users can access for Realtime Database, Cloud Firestore, and Cloud Storage. Firebase Realtime Database Rules leverage JSON in rule definitions, while Cloud Firestore Security Rules and Firebase Security Rules for Cloud Storage leverage a unique language built to accommodate more complex rules-specific structures. Firebase provides a rule-based access control system that allows developers to enforce fine-grained access control to data stored in Firebase services and restrict access to sensitive data to ensure that only authorized users can access.

Supportability Requirements

Coding Standards – Our system will be coded in a range of 75-80% of coding standards for HTML 5 and CSS3. The code will be routinely reviewed, tested, and validated via third-party compliance software.

Naming Conventions – HTML classes and id tags will be coded in lowercase except in naming situations involving multiple words, for which the camelCase convention will be applied. Firebase collections, documents, and fields will also adhere to this convention.

Availability Requirements

Accessible Times – Our system should be available for use 24 hours per day, 7 days per week, unless catastrophic unforeseen circumstances arise due to our application being hosted upon GitHub.

Downtime Impact – General downtime will be largely non-existent, save for brief events in which our team updates the software, which should only result in about 5 to 10 minutes of unavailability.

Report Issue – There will be a support feature accessible via the navigation bar which will pull up a template that allows the user to express concerns/report bugs which will automatically be routed to the team's email account.

High-Level System Architecture

Visual Studio (IDE)

- Hyper Text Mark-up Language (HTML) – will be the language that will allow the browser display the website

- Cascading Style Sheets (CSS) – will be the language used to decor the web pages

- JavaScript – language used for client side funcitonality that will be handled for User Interface (UI) needs to make the user experience enjoyable

- jQuery – a fast, small, and feature-rich JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, animation, and Ajax much simpler with an easy-to-use API (Application Programming Interface) that works across a multitude of browsers.

- Bootstrap – a front-end framework that is used to create responsive, flexible, and mobile-first websites, which means that web pages created with Bootstrap will automatically adjust their layout and content to fit different screen sizes and resolutions. It provides a set of pre-designed HTML, CSS, and JavaScript components, such as buttons, forms, menus, and modals, that can be easily customized and integrated into web pages.

- Chart.js – provides a set of frequently used chart types, plugins, and customization options. In addition to a reasonable set of built-in chart types, one can use additional community-maintained chart types. Furthermore, it's possible to combine several chart types into a singular mixed chart (essentially, blending multiple chart types into one on the same canvas).

- Moment.js – a JavaScript library that provides a simple and easy-to-use interface for working with dates and times in JavaScript. It allows the user to parse, manipulate, validate, and display dates and times in a wide variety of formats.

- Datepicker – a JavaScript library that provides a full-featured calendar solution for web applications. It allows developers to easily create and customize interactive calendars.

OpenWeatherMap

OpenWeatherMap is available as a web-based service and also provides APIs that developers can use to integrate weather data into their applications or websites. The APIs provide access to a wealth of weather data, including current conditions, hourly and daily forecasts, and historical weather data.

Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to build server-side applications using JavaScript. It utilizes an event-driven, non-blocking I/O model that makes it lightweight, efficient, and scalable. Node.js provides a vast library of modules and packages that can be easily integrated into the server-side application to perform various tasks, such as handling HTTP requests, working with databases, and implementing security features.

- <u>Express.js</u> – a JavaScript web framework that helps to build web applications in an easier and faster way. It simplifies the process of creating a web server, handling requests, and sending responses. Think of it like a toolbox that has all the necessary tools to build a web application. With Express.js, one can easily define routes, handle HTTP methods, and render dynamic HTML pages. It is a popular choice for developers because it is lightweight, flexible, and can be used with many other libraries and tools.

- <u>Node-Cron</u> – a module for Node.js that allows the user to schedule recurring tasks, such as running a script or a function, at specific intervals or times. It provides a simple and flexible API for defining when and how often a task should be executed, using a cron-like syntax. With Node-Cron, one can easily automate tasks such as sending emails, cleaning up data, or performing backups, without having to manually run the tasks oneself. It's a great tool for developers who need to perform repetitive tasks on a regular basis, and want to save time and improve efficiency.

- <u>PM2</u> – a process manager for Node.js applications that allows the user to manage and monitor their application's processes in a simple and efficient way. It provides features such as process monitoring, automatic restarting, load balancing, and error tracking.

- Nodemailer – a module for Node.js that allows the user to send emails from their Node.js application. It provides a simple and flexible API for sending email messages, with support for attachments, custom headers, and various transport methods.

GitHub/Git

A web-based platform for version control and collaboration that is widely used by software developers to manage and share their code. It provides a centralized location where developers can store their code, track changes, and collaborate with others on projects. At its core, GitHub is based on the Git version control system, which allows developers to create and manage multiple versions of their code. Developers can use Git to create "branches" of their code, which allow them to work on different features or aspects of a project independently. They can also use Git to "merge" changes from different branches back into the main codebase.

- GitHub Pages – a free web hosting service provided by GitHub, which allows users to host static websites directly from their GitHub repositories. It is a great way for developers to showcase their projects, documentation, and portfolios online, without having to worry about setting up a separate web server or paying for hosting.

Firebase

Backend framework developed by Google which allows for automated authentication, authorization, and database functionality using NoSQL.

Jira

Software application that allows teams to track issues, conduct milestone sprints, manage projects, and delegate tasks.

Discord

Primary internal communication application between team members. Discord is a VoIP (Voice over Internet Protocol) and instant messaging social platform wherein users may communicate via voice calls, video calls, text messaging, media and files in private communities called “servers” – among which one has been created expressly for the purposes of this project.

Canvas

Primary external communication application between team and instructor for upcoming Milestone assignments and deliverable requirements.

High-Level UML Diagrams

Component & Deployment Diagram

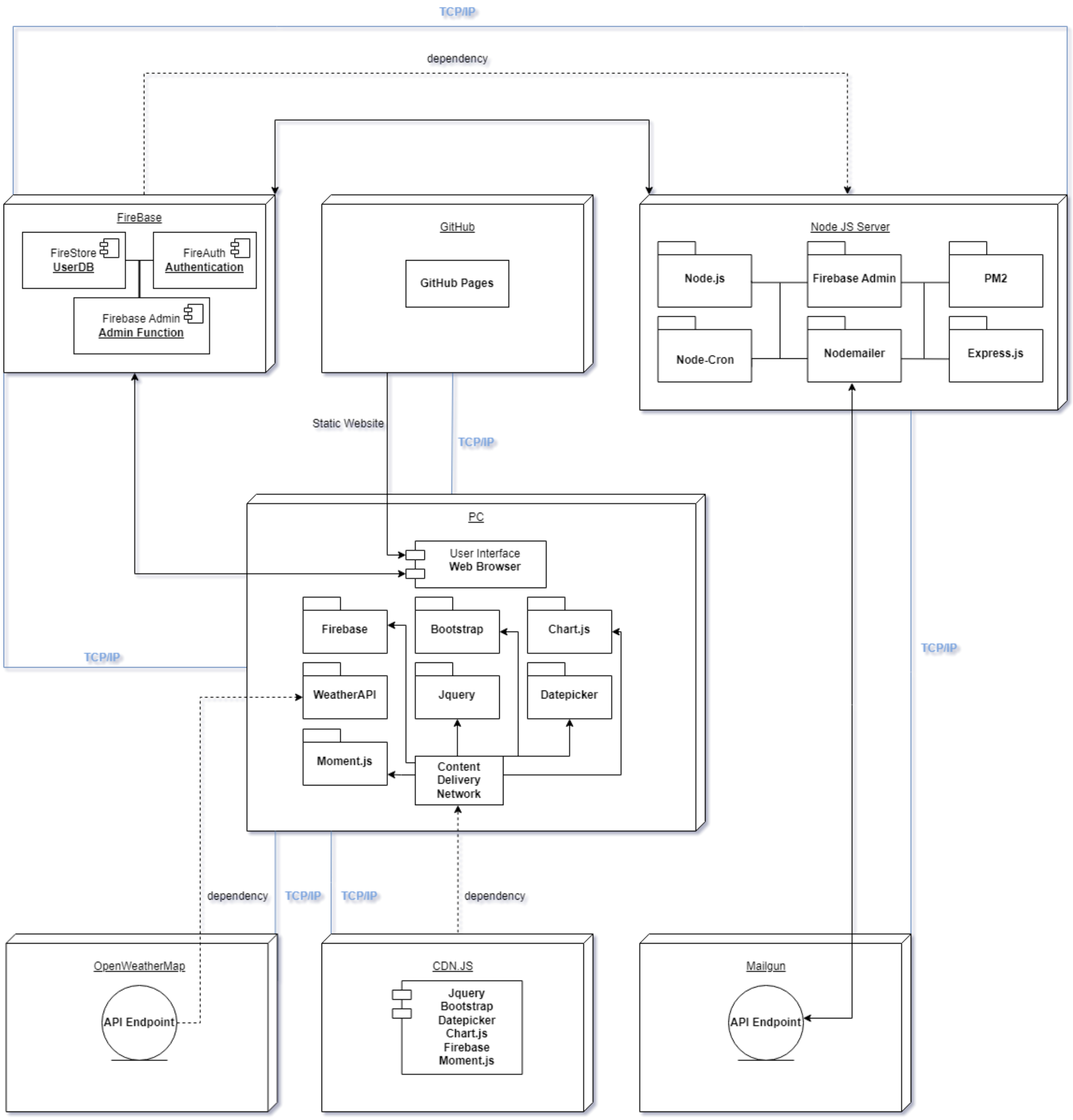


Figure 2 – System Component & Deployment Diagram

Database Organization

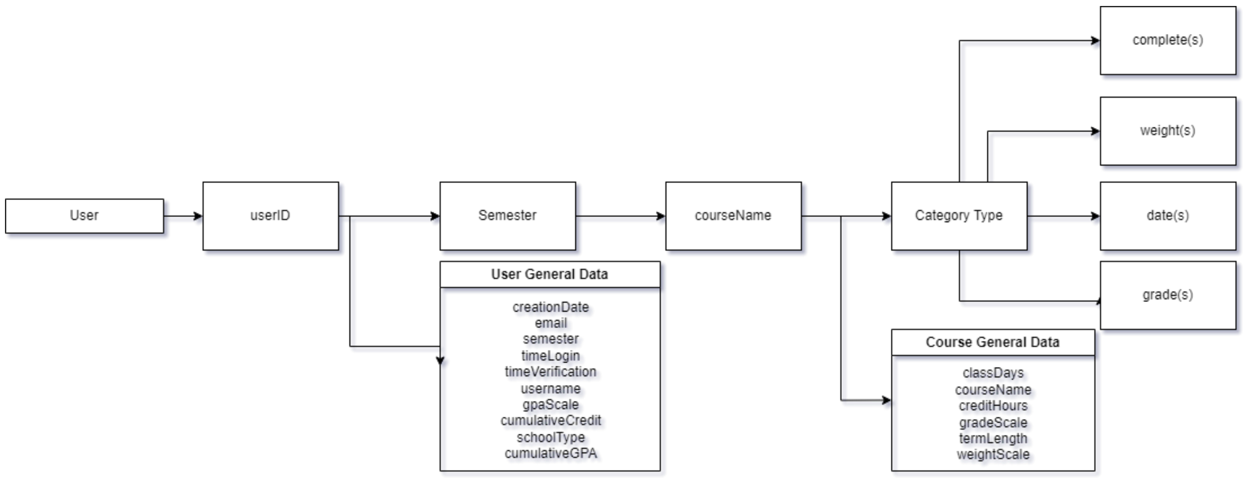


Figure 3 – System Database Organization (Firebase)

Users: The outermost database layer which holds a collection of all user profiles classified by userID.

userID: A document which holds the general user fields below & 'Semester' subcollections.

- creationDate: Time stamp of when user created the account.
- email: User email.
- semester: Array that store all user semester for querying.
- timeLogin: Time of last user login.
- timeVerification: Time Stamp in millisecond upon email verification sent.
- username: User display name.
- gpaScale: School lowest threshold GPA scale
- schoolType: Highschool or college grading curriculum
- cumulativeCredit: Total credit hours the user has taken
- cumulativeGPA: User's current GPA

Semester: Each semester subcollection will adhere to the naming convention of "<Fall/Spring/Summer>+<Year>" & hold 'courseName' documents.

courseName: Document(s) which hold the course-specific fields below & each courseName document will adhere to a naming convention of three capitalized letters which are stipulated by the user at the time of 'Course Form' submission alongside generation of 'Category Type' subcollections.

- classDays: array that indicates what days the user have class or if class is remote
- courseName: hold the course name
- creditHours: hold class credit hours amount
- gradeScale: hold user grading scale for the class in hash-map

- termLength: hold how long term length is
- weightScale: a hash-map nested in array that hold if a categories type quantity, weight and if it’s equally weighted

Category Type: Each Category Type subcollection may only take the name of dynamically generated subcollection based on user input and have 3 documents weight(s),date(s),grade(s)

complete(s): field(s) dynamically generated based on user quantity stored as Boolean values

weight(s): field(s) dynamically generated based on user quantity input example: exam1: 20%, exam2: 25%

date(s): field(s) dynamically generated based on user quantity input example: exam1: 2022/10/28, exam2: 2022/12/04

grade(s): field(s) dynamically generated based on user quantity input example: exam1: 30%, exam2: 90%

Significant Algorithms

Generating Semester Dropdown for Dashboard

1. Define an async function named getSemester() that does the following:
2. Get a reference to the HTML select element with an id of semesterSelect using document.getElementById().
3. Get the unique user ID of the currently authenticated user by accessing auth.currentUser.uid.
4. Use the user ID to retrieve the user's data from the Firestore database by calling db.collection('users').doc(userID).get().
5. When the data is retrieved, handle it using the .then() method.
6. Check if the document exists by calling doc.exists.
    a. If the document exists, retrieve the user's semester data from the doc.data() object.
        I. Loop through each semester data using a for loop with a counter variable i.
        II. Inside the loop, create an HTML option element using document.createElement().
        III. Set the value attribute of the option element to semester[i].
        IV. Set the text attribute of the option element to semester[i].

V. Append the option element to the semesterSelect select element using the .appendChild() method.

b. If the document does not exist, log the message "No such document!" to the console.

Generating HTML Elements for Course Form 2

1. Create an asynchronous function called "dataChecker" that takes four parameters:
    - a. "semester": a string representing the semester name
    - b. "courseName": a string representing the name of the course
    - c. "userID": a string representing the user's ID
    - d. "subCollectionName": an array of strings representing the names of sub-collections
2. Create an empty array called "previousKeys"
3. For each sub-collection name in "subCollectionName", do the following:
    - a. Try to get a snapshot of the sub-collection from the database using the user ID, semester, course name, and sub-collection name.
    - b. If the snapshot is empty, log "empty".
    - c. If the snapshot is not empty, do the following:
        - i. Create a label element with the sub-collection name and append it to the corresponding HTML element.
        - ii. For each document in the sub-collection, do the following:
            1. Get the data of the document.
            2. Sort the keys in the data array in ascending order based on the number in the key.
            3. For each key in the sorted keys array, do the following:
                - a. If the key is not in the "previousKeys" array, do the following:
                    - I. Create an input group element with a label and an input element.
                    - II. Append the input group element to the corresponding HTML element.
                    - III. Add the key to the "previousKeys" array.

IV. Add the input element to the "elements" array.

V. Initialize the datepicker for the input element.

4. Catch any errors that occur and log them to the console.

Validation Check for User Class Days

1. The function "classDayCheck" is called.
2. It checks whether the "Remote" option is checked by accessing the "day-7" checkbox element using the "getElementById" method and checking its "checked" property.
3. If the "Remote" option is checked, the function enters the "if" statement and executes the following steps:
    a. It initializes a "for" loop that runs from 0 to 6 (i.e., for each day of the week).
    b. Inside the loop, it accesses each day checkbox element using the "getElementById" method and unchecks it by setting its "checked" property to false.
4. If the "Remote" option is not checked (i.e., it's unchecked or another day checkbox is checked), the function enters the "else" statement and executes the following steps:
    a. It accesses the "day-7" checkbox element using the "getElementById" method and unchecks it by setting its "checked" property to false.

Generating HTML Elements for Weight Categories in Course Form

1. The function "generateFields" is called with a "category" parameter that specifies the type of assessment (Exam, Quiz, Assignment, Discussion, Project, or Participation).
2. The function creates an object "categoryIdMap" that maps each category to its corresponding HTML element IDs for the quantity container and the "Yes" checkboxes with values 2 and 4.
3. The function uses the "category" parameter to look up the corresponding IDs for the current category in the "categoryIdMap" object.
4. The function accesses the "Yes" checkboxes for the current category using their IDs and stores them in variables "categoryYes2" and "categoryYes4".
5. The function checks whether the value of the "No" radio button for the "categoryYes4" checkbox is selected and whether the value of the "Yes" radio button for the "categoryYes2" checkbox is greater than 1.
6. If the "No" radio button for "categoryYes4" is selected and the value of "categoryYes2" is greater than 1, the function enters the "if" statement and executes the following steps:

   a. It accesses the quantity container element using the "getElementById" method and sets its "display" style to "block" to make it visible.
   b. It clears the quantity container by setting its "innerHTML" property to an empty string.
   c. It creates a "for" loop that runs from 1 to the value of "categoryYes2".
   d. Inside the loop, it creates a new "input" element of type "number" and sets its "id", "name", and "placeholder" attributes to values based on the current iteration index and category.
   e. It attaches an "onchange" event listener to the "input" element that calls the "checkPercentage" function passing the current category as a parameter. f. It appends the "input" element to the quantity container.

7. If the "No" radio button for "categoryYes4" is not selected or the value of "categoryYes2" is less than or equal to 1, the function enters the "else" statement and executes the following step: a. It accesses the quantity container element using the "getElementById" method and sets its "display" style to "none" to hide it.

Risk Assessment

Schedule Risks

Description: *Team member chosen as Juror for an indeterminate timeframe*

Given the nature of jury duty, the exact duration of service requirement is unknown – especially in the context of a criminal trial which could last anywhere from days to months! If unaddressed, their absence may cause significant project delays and impact the quality of deliverables.

Probability: *High*

Impact: *Medium*

Response Plan: *Mitigate*

As a contingency plan for the member's potential prolonged absence, the team collectively agreed to proactively work on each of their individual Milestone 3 deliverable components earlier than previously anticipated in order to best allow the affected member to optimize their reduced free time for completion of their own integral contributions, without dependency delays. Furthermore, meeting notes were provided to the individual for any missed team collaborations, and a deadline was established by which the team would redistribute any remaining tasks amongst themselves.

Technical Risks

Description: *Lack of continuous back-end JavaScript runtime environment*

Due to hosting upon GitHub Pages, a static site hosting service, the team encountered two technical issues with respect to implementation of planned features. The first problem concerns the intended deletion of a temporary user account if the user neglects to verify within 24 hours. However, a static web page has no way of telling what time it is while not in use, nor does Firebase. As a temporary solution, a data field was stored in the back-end depicting the time of account creation which would then be compared to the user's local time upon each subsequent login attempt; if the account was found to be both unverified and older than 24 hours, the account would be deleted – this is not an ideal solution. By the same token, the second problem surrounds the desired feature of sending weekly batch email updates to users who opted into receiving them. Once again, the static server system provided no tangible means to automatically dispatch emails at a set time given it has no capacity to continuously track time when not in use. Moreover, any attempts to place the onus upon the user's system to constantly query the database to approve any pending email dispatch would induce two additional problems: A) the security vulnerability of a running JavaScript constantly querying the database which would undermine the established rule of only allowing authenticated users to read or access the database, and B) the performance cost of running such a high volume of scripting would be untenable due scaling exponentially with higher userbase.

Probability: *High*

Impact: *High*

Response Plan: *Avoid*

The team is currently researching a new hosting service that will enable node.js to run on the backend, thereby allowing for active 24/7 availability without incurring the excessive financial costs of either renting a cloud server or procuring new server hardware altogether for local hosting. With this new implementation, both of the aforementioned technical problems will be completely resolved. As an added benefit, this measure will further optimize client-side performance.

Teamwork Risks

Description: *Late addition of new team member*

Whenever a new member is introduced late into an ongoing project, there exists inherent risks to the team's productivity and working environment. Some of the more notable considerations may include disruption of current team dynamics, increased workload for existing members in bringing the new individual up to speed, or potential delayed onboarding for the new member which may greately reduce their ability to contribute effectively and efficiently.

Probability: *High*

Impact: *Low*

Response Plan: *Mitigate*

To mitigate these risks, the team immediately established a clear onboarding plan which included all pertinent project information and previous milestone coverage. The team then hosted a meet-and-greet session thereafter to welcome its new member, provide them a comprehensive scope of the project, and ensure they were assisted in the registration and downloading of all required project-related software. Furthermore, the individual was prompted for their preferred area of contribution based on their skillset and subsequently paired with an original member responsible for the chosen domain to provide adequate training for a smooth transition. The entire onboarding process went without a single hitch and the team is grateful to have a new addition who fits in seamlessly!

Legal/Content Risks

Description: *Data Breach of users' confidential grading information*

With consideration of the team's relative inability and inexperience in providing sufficient database security compared to more established entities, the team could not guarantee beyond a reasonable doubt that users' sensitive data would be impervious to data breaches if left to its own devices.

Probability: *Low*

Impact: *High*

Response Plan: *Transfer*

It was decided that the project's backend hosting services would be outsourced to Firebase, developed by Google. Renowned for its reliability, this household name has been a pioneer in the tech industry for decades and boasts an equally impressive arsenal of security measures to uphold its reputation. As such, the team felt confident in this risk transferrence to such a thoroughly vetted company. As an extra layer of mitigation, the Grade PlanAlyzer software stores no readily identifiable data, such as first name, last name, birthdate, university, city, etc., which may be maliciously used to link an individual's grading information to the user in the event of a data breach.

Team Nightshift Members

- **Adam Clark –** Back-End Lead/Scrum Master/Product Owner
- **Quang Le –** Team Lead/GitHub Master/Full-Stack Developer
- **Nicholas Moussa –** Front-End Developer
- **Mahmood Sakib –** Front-End Lead/Full-Stack Developer
- **Kyla Tolentino –** Front-End Developer/Graphic Designer

Project Links

GitHub Repository:

- **https://github.com/Q-Wrld97/cen4010_sp23_g23**

Team Web Page:

- **https://q-wrld97.github.io/cen4010_sp23_g23/**

Jira Dashboard:

- **https://teamnightshift.atlassian.net/**

Vertical Demo:

- **https://www.youtube.com/watch?v=wPiRiusQQnM**

Acknowledgements

We would like to express our utmost thanks to the following individuals for their unwavering support and subject-matter expertise provided throughout the entirety of our design project:

- ***Dr. Shihong Huang*** *– Course Instructor*

Appendix

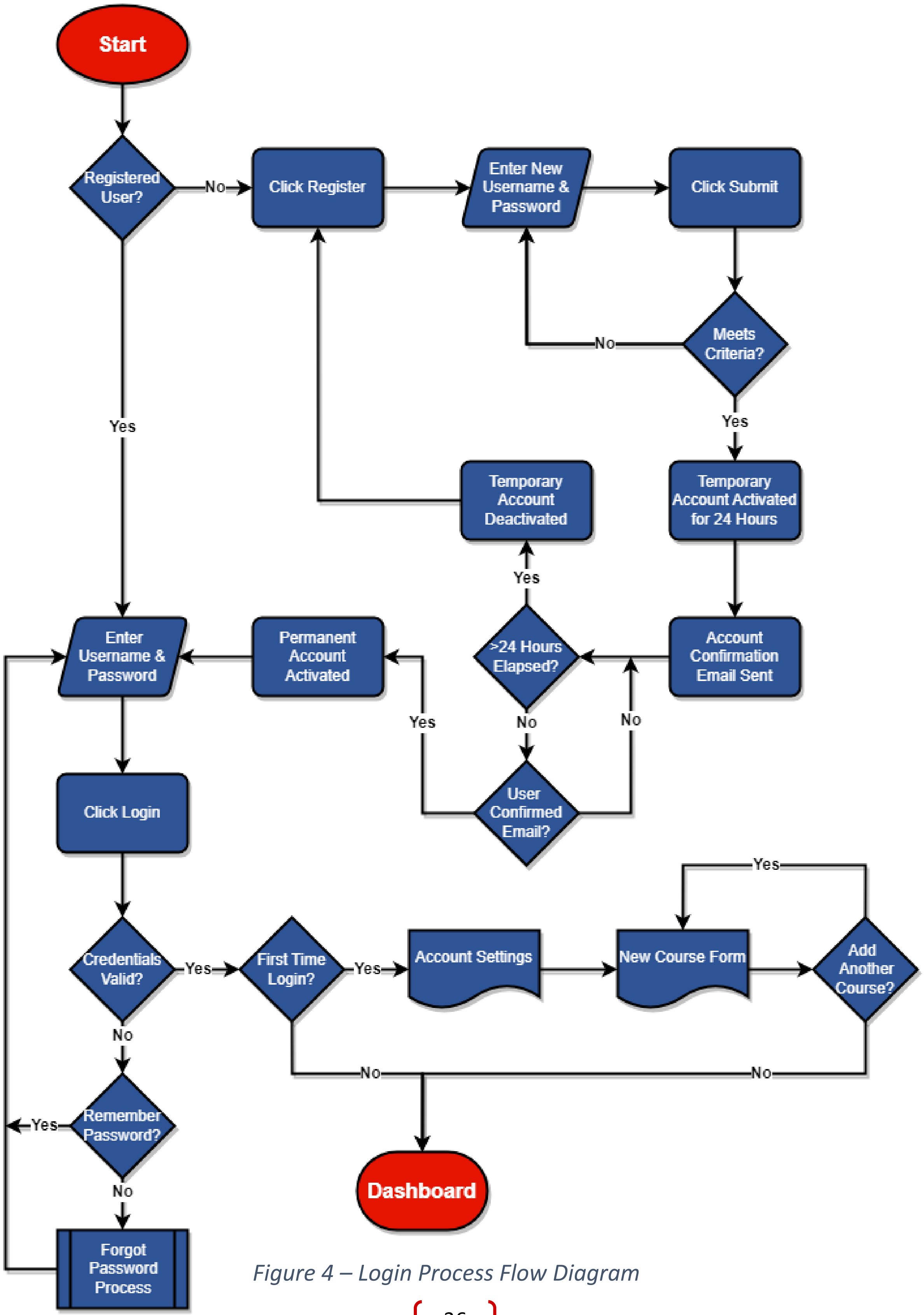
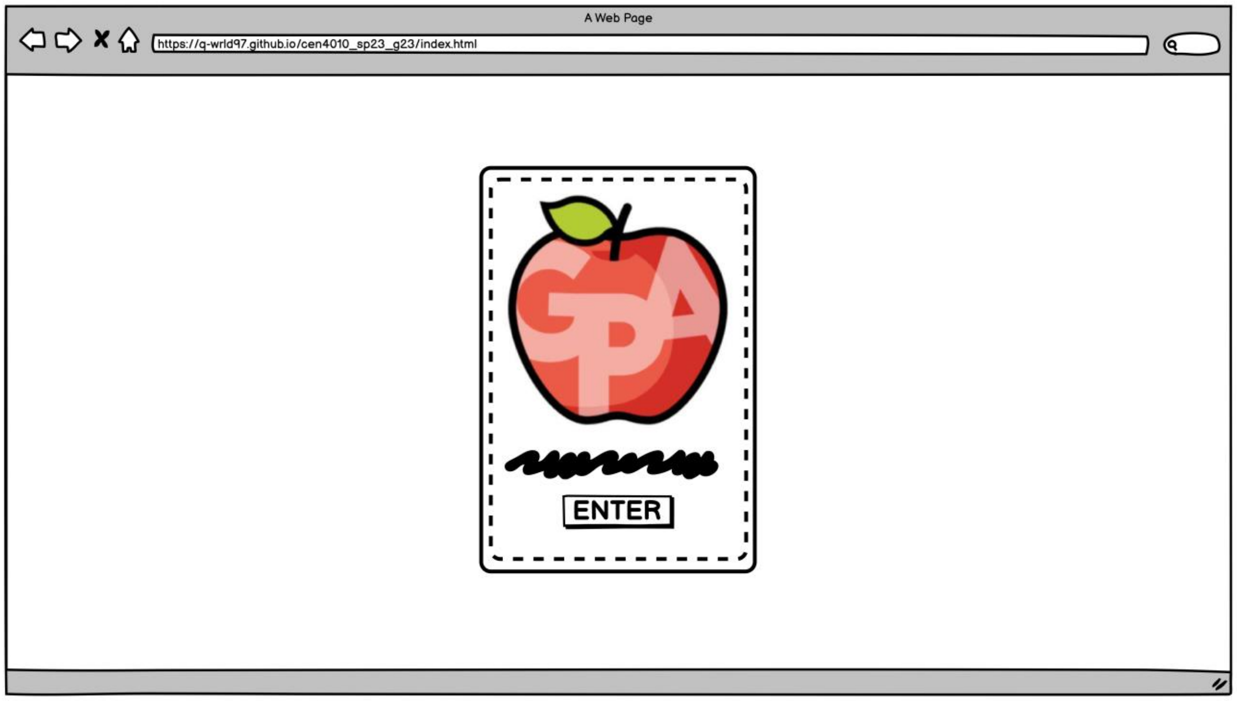
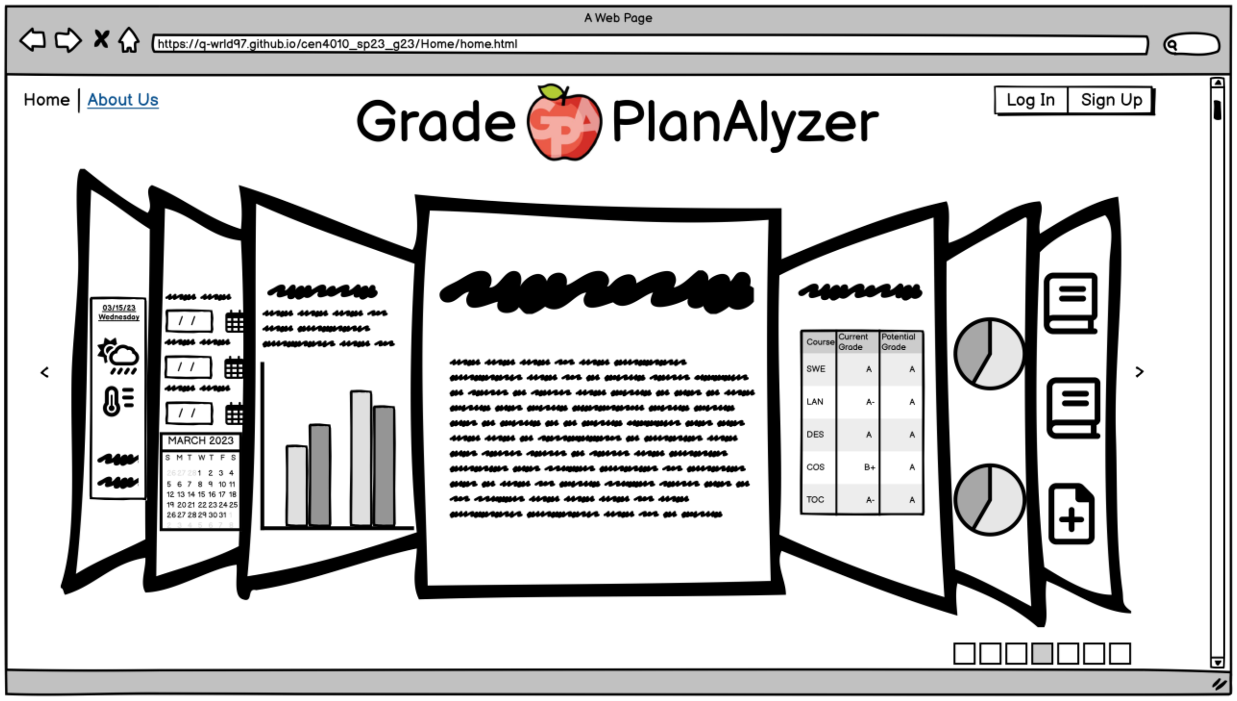


Figure 4 – Login Process Flow Diagram

Landing Page 1 / 14

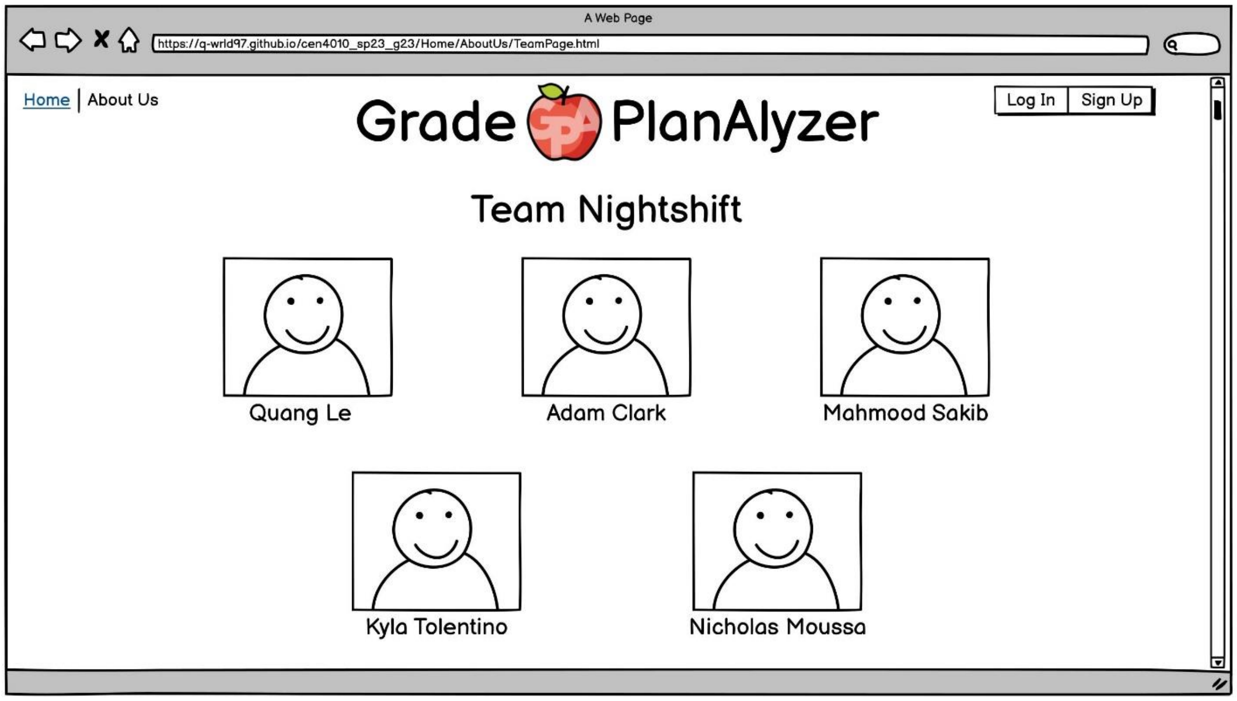


Home Page 2 / 14

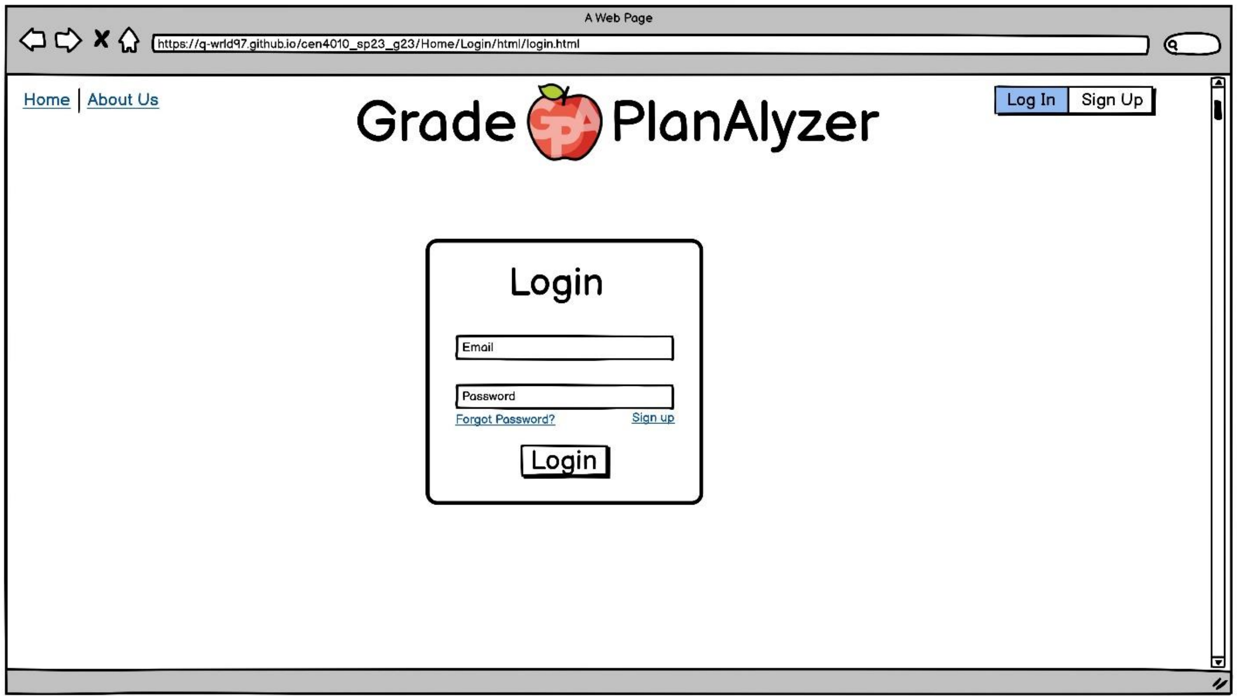


Course	Current Grade	Potential Grade
SWE	A	A
LAN	A-	A
DES	A	A
COS	B+	A
TOC	A-	A

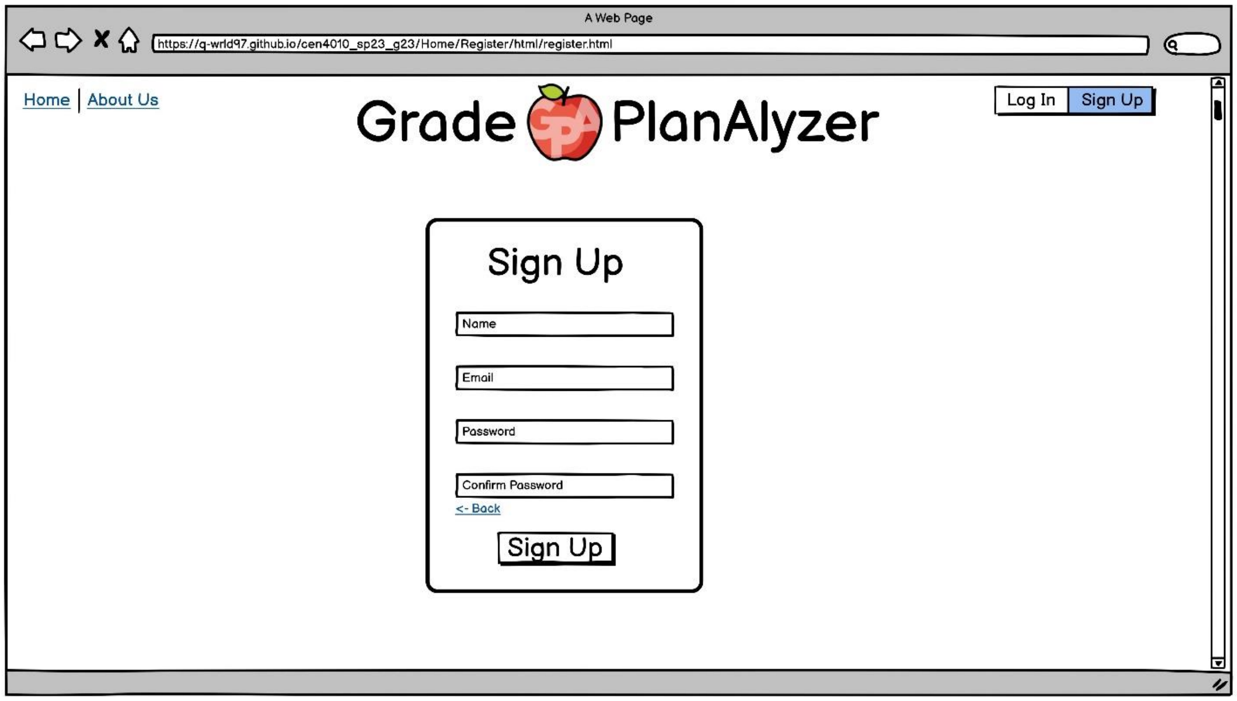
About Us Page 3 / 14



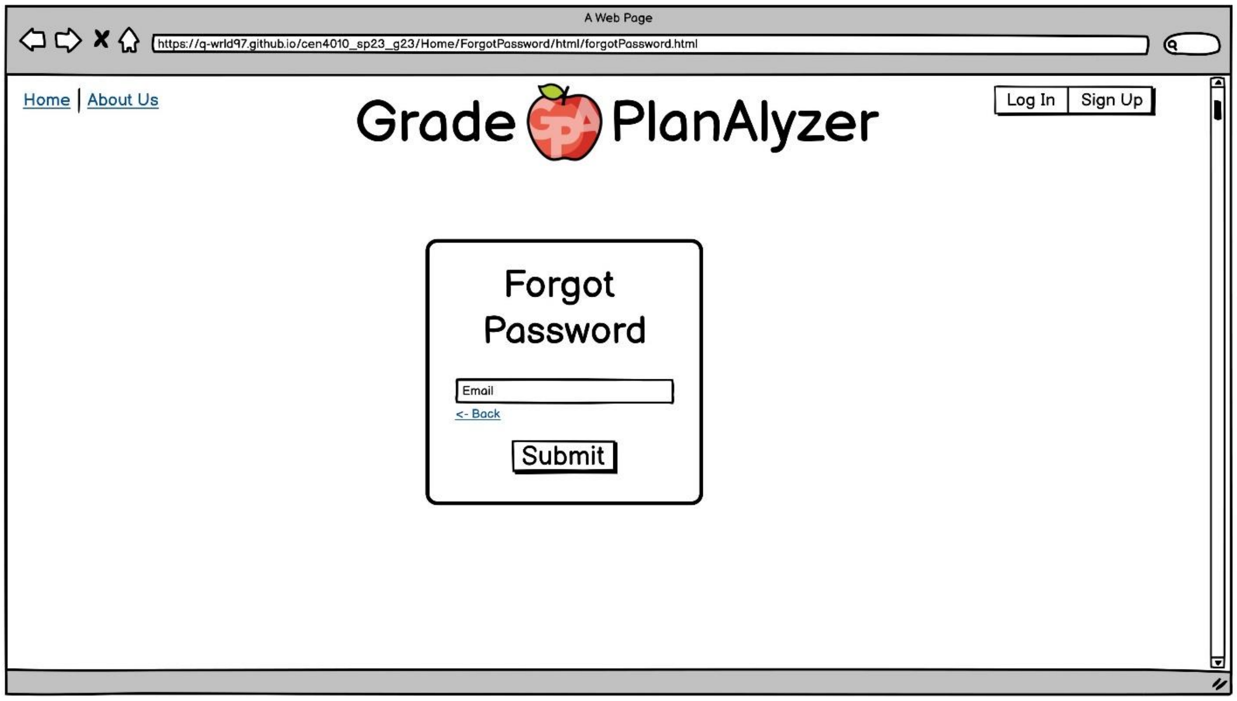
Login Page 4 / 14



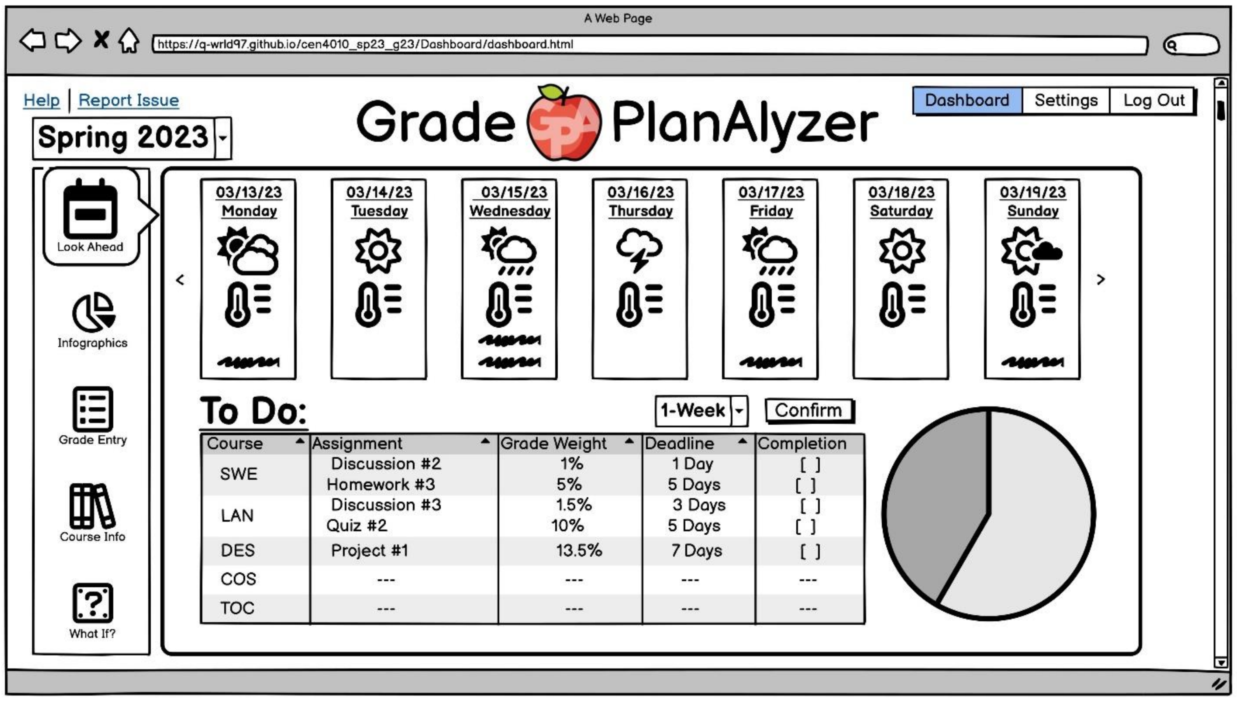
Sign Up Page 5 / 14



Forgot Password Page 6 / 14


A Web Page
https://q-wrld97.github.io/cen4010_sp23_g23/Home/ForgotPassword/html/forgotPassword.html
Home
About Us
Grade PlanAlyzer
Log In
Sign Up
Forgot Password
Email
<- Back
Submit

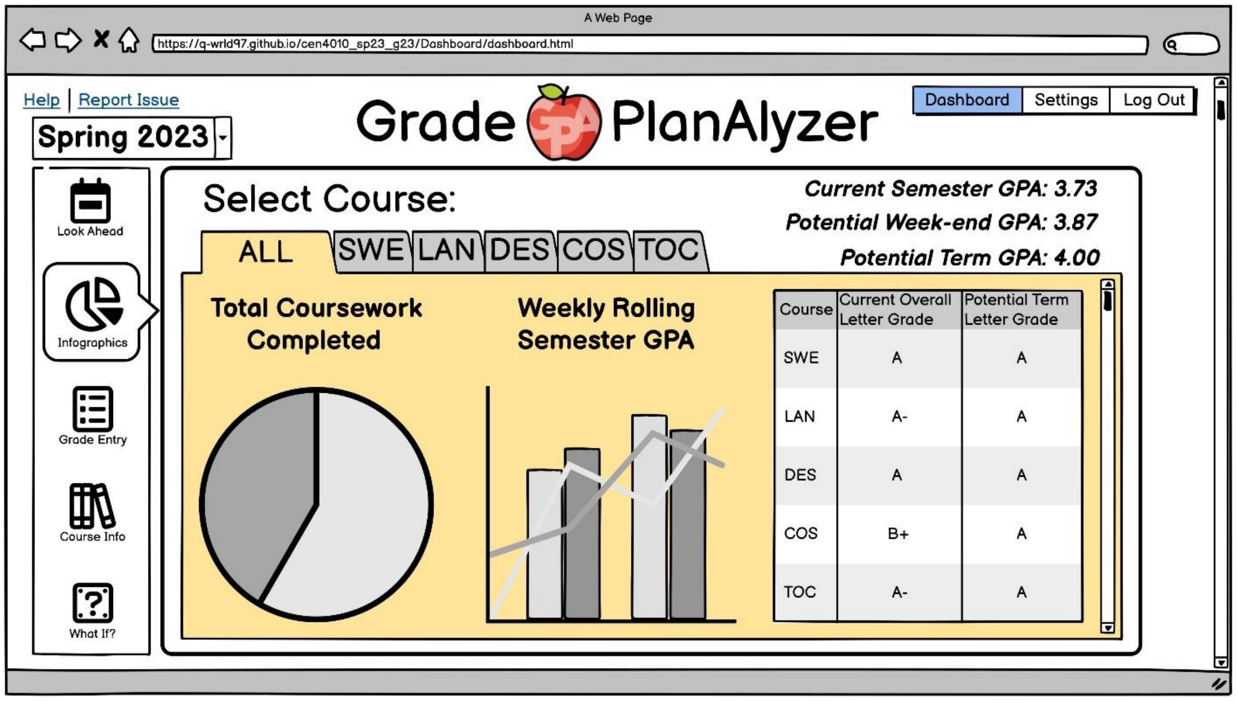
Dashboard - Look Ahead 7 / 14



To Do:

Course	Assignment	Grade Weight	Deadline	Completion
SWE	Discussion #2	1%	1 Day	[ ]
	Homework #3	5%	5 Days	[ ]
LAN	Discussion #3	1.5%	3 Days	[ ]
	Quiz #2	10%	5 Days	[ ]
DES	Project #1	13.5%	7 Days	[ ]
COS	---	---	---	---
TOC	---	---	---	---

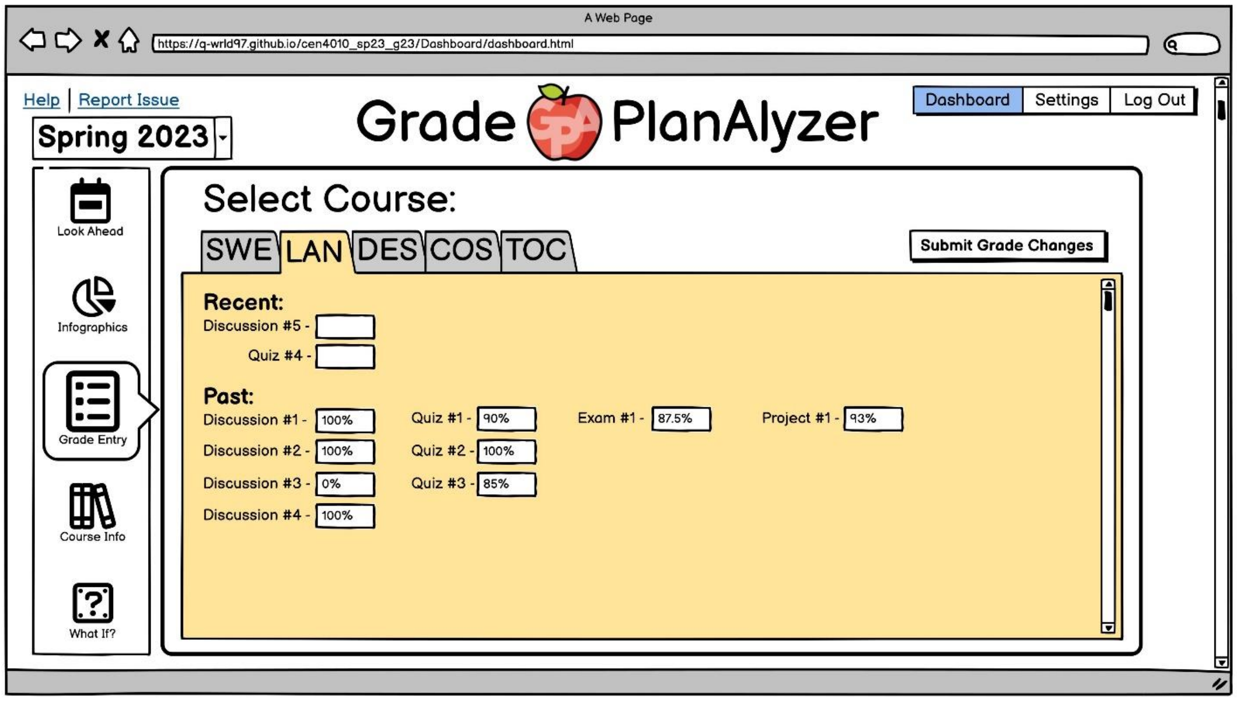
Dashboard - Infographics (All Courses) 8 / 14



Current Semester GPA: 3.73
Potential Week-end GPA: 3.87
Potential Term GPA: 4.00

Course	Current Overall Letter Grade	Potential Term Letter Grade
SWE	A	A
LAN	A-	A
DES	A	A
COS	B+	A
TOC	A-	A

Dashboard - Grade Entry 9 / 14



Recent:

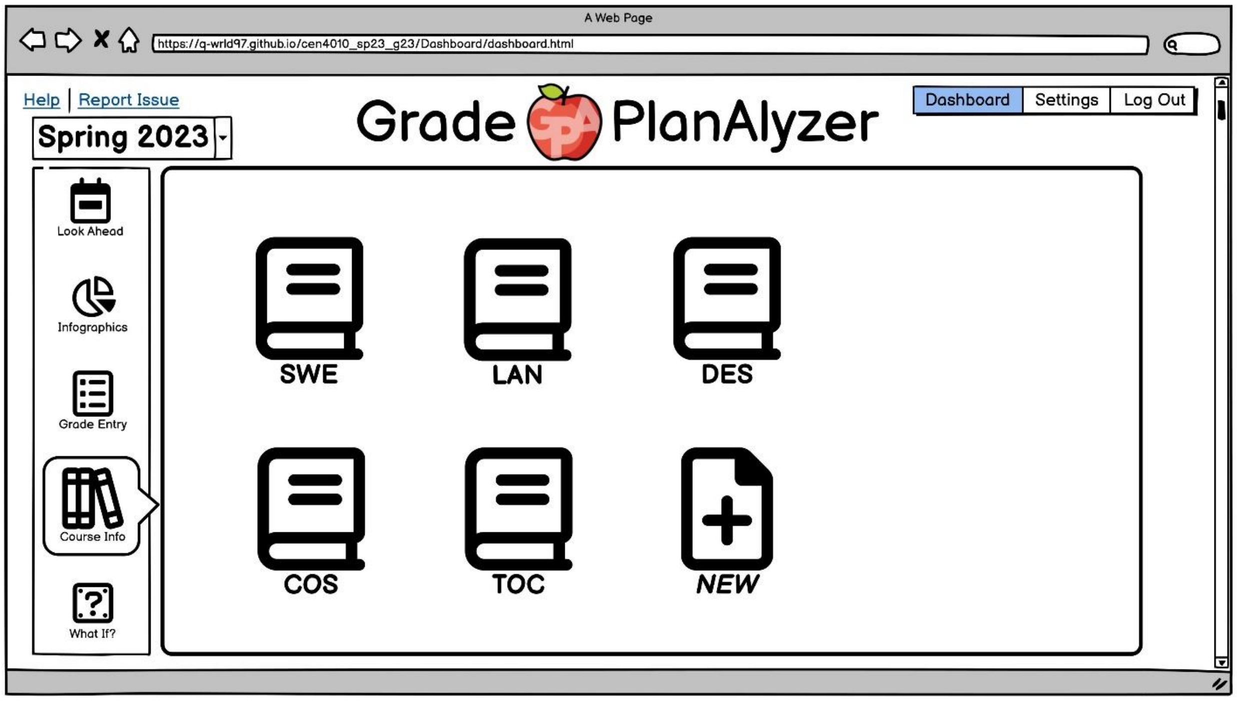
Discussion #5 -

Quiz #4 -

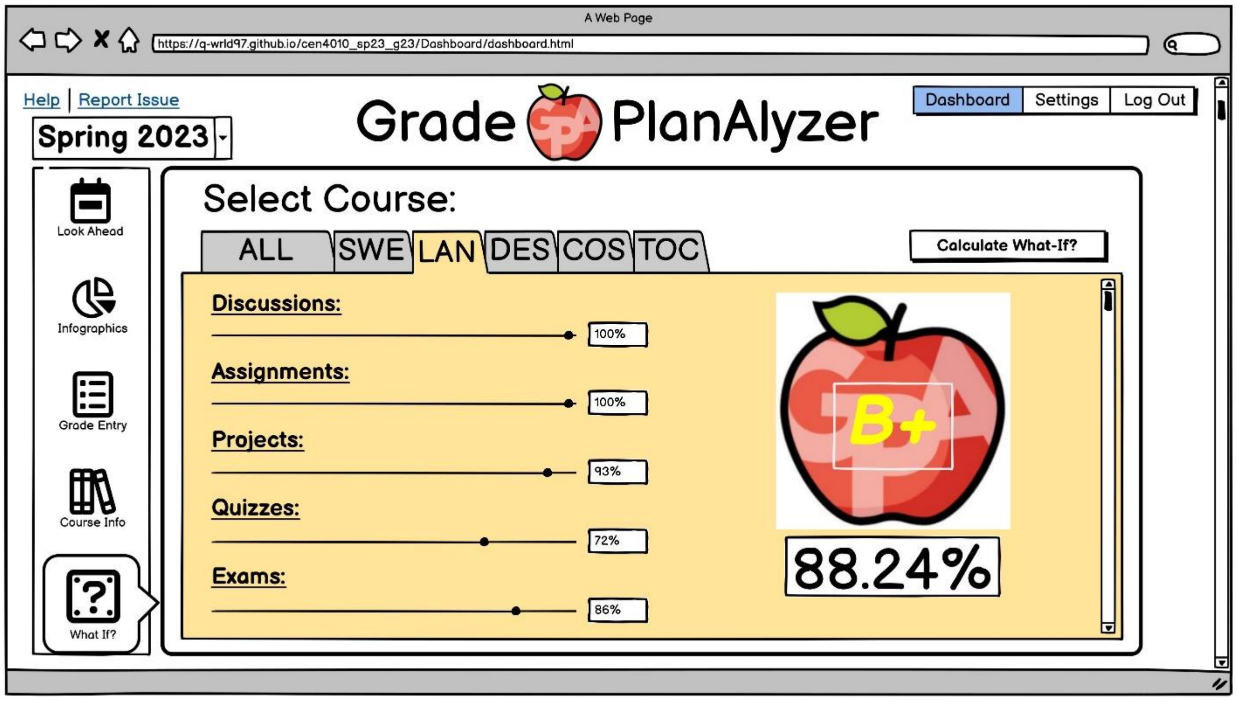
Past:

Discussion #1 - 100%	Quiz #1 - 90%	Exam #1 - 87.5%	Project #1 - 93%
Discussion #2 - 100%	Quiz #2 - 100%		
Discussion #3 - 0%	Quiz #3 - 85%		
Discussion #4 - 100%			

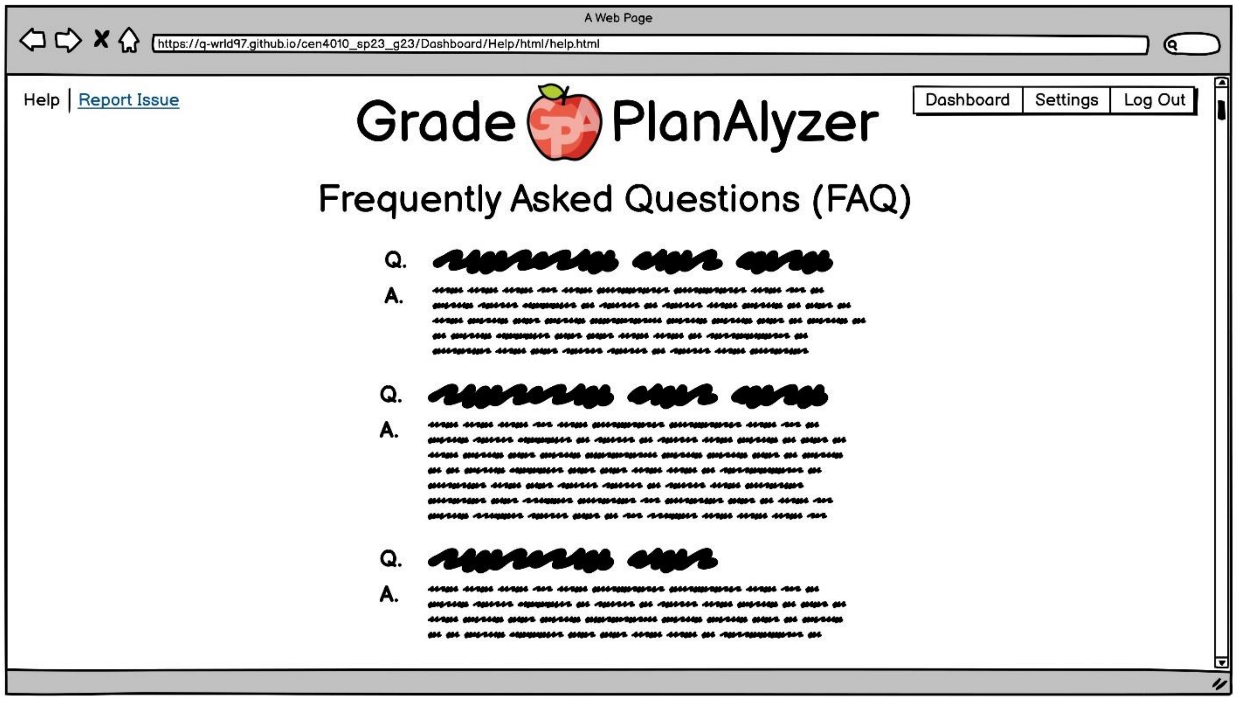
Dashboard - Course Info 10 / 14



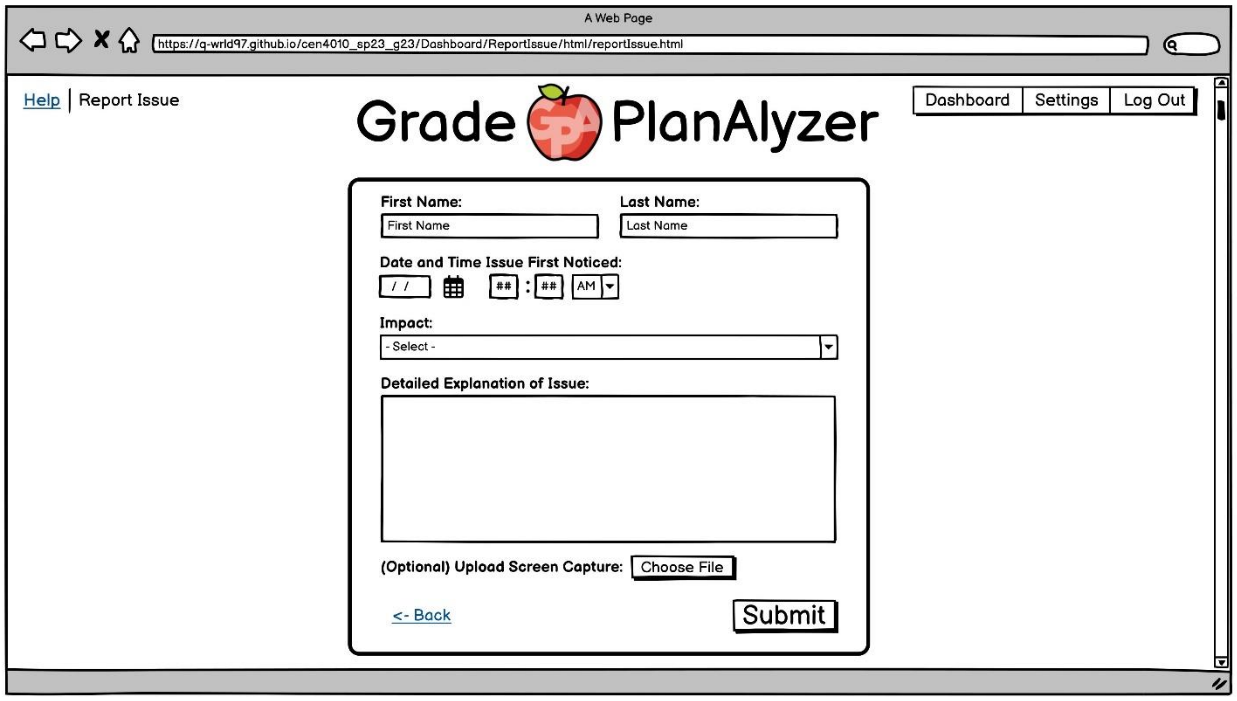
Dashboard - What If? 11 / 14



Help Page 12 / 14


A Web Page
https://q-wrld97.github.io/cen4010_sp23_g23/Dashboard/Help/html/help.html
Help | Report Issue
Grade PlanAlyzer
Dashboard
Settings
Log Out
Frequently Asked Questions (FAQ)
Q.
A.
Q.
A.
Q.
A.

Report Issue Page 13 / 14



A Web Page

https://q-wrld97.github.io/cen4010_sp23_g23/Dashboard/ReportIssue/html/reportIssue.html

Help | Report Issue

Grade PlanAlyzer

Dashboard | Settings | Log Out

First Name: First Name

Last Name: Last Name

Date and Time Issue First Noticed: / / ##:## AM

Impact: - Select -

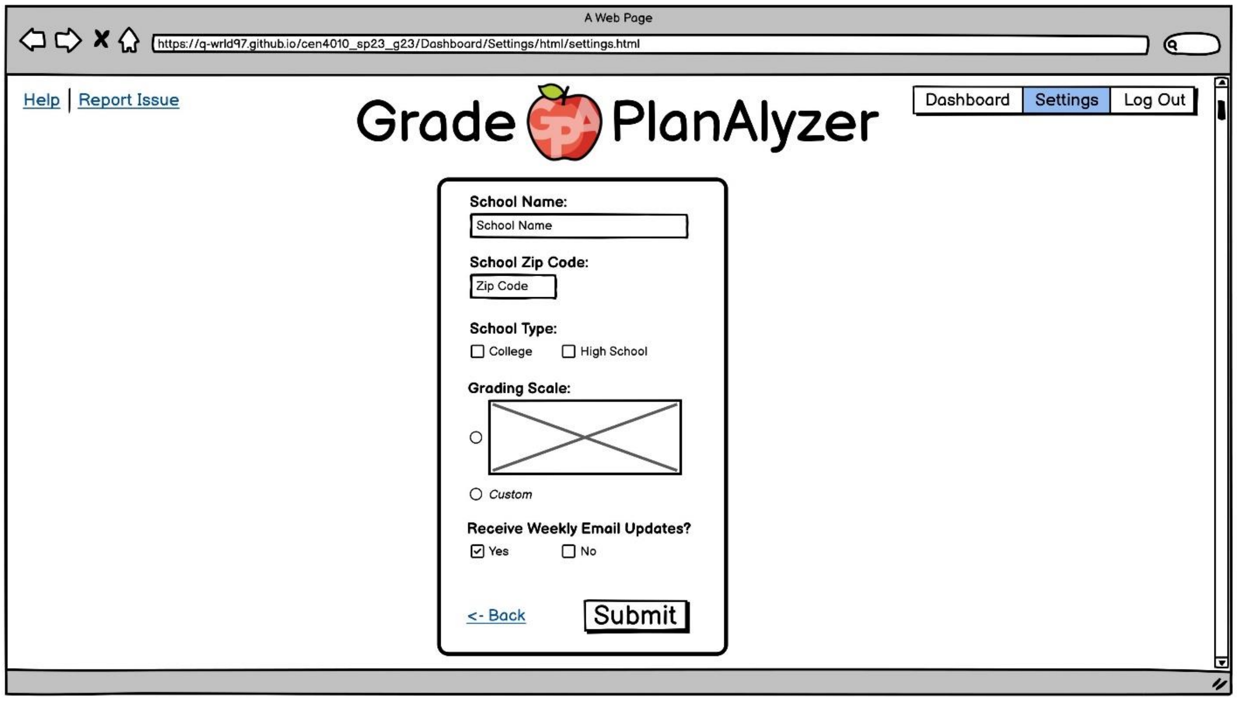
Detailed Explanation of Issue:

(Optional) Upload Screen Capture: Choose File

<- Back

Submit

Settings Page 14 / 14



A Web Page

https://q-wrld97.github.io/cen4010_sp23_g23/Dashboard/Settings/html/settings.html

Help | Report Issue

Grade PlanAlyzer

Dashboard | Settings | Log Out

School Name: School Name

School Zip Code: Zip Code

School Type: ☐ College ☐ High School

Grading Scale:

○

○ *Custom*

Receive Weekly Email Updates? ☑ Yes ☐ No

<- Back

Submit