

Santa Clara University

## Scholar Commons

---

Computer Science and Engineering Senior  
Theses

Engineering Senior Theses

---

6-11-2020

### Smart Quizzes

Andres Cisneros-Munoz

Cole Sanchez

Follow this and additional works at: [https://scholarcommons.scu.edu/cseng\\_senior](https://scholarcommons.scu.edu/cseng_senior)



Part of the [Computer Engineering Commons](#)

---

**SANTA CLARA UNIVERSITY**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Date: June 11, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

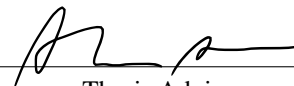
**Andres Cisneros-Munoz**  
**Cole Sanchez**

ENTITLED

**Smart Quizzes**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



---

Thesis Advisor

*N. Ling*

---

Department Chair

# **Smart Quizzes**

by

Andres Cisneros-Munoz  
Cole Sanchez

Submitted in partial fulfillment of the requirements  
for the degree of  
Bachelor of Science in Computer Science and Engineering  
School of Engineering  
Santa Clara University

Santa Clara, California  
June 11, 2020

# Smart Quizzes

Andres Cisneros-Munoz  
Cole Sanchez

Department of Computer Science and Engineering  
Santa Clara University  
June 11, 2020

## ABSTRACT

Standardized testing, while efficient for testing a large amount of people, is not an effective or holistic method of evaluating test takers. They fail to take into consideration factors such as stress, which can inhibit a student's ability to perform well on a test. Our goal is to create a framework for creating tests that are dynamic, adaptive assessments, minimize stress levels, and overall improve student performance. This solution will produce in tests that even out the academic playing field while being more informative in their assessments.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Problem . . . . .	1
1.3	Solution . . . . .	1
<b>2</b>	<b>Requirements</b>	<b>3</b>
2.1	Design Constraints . . . . .	3
2.2	Critical Requirements . . . . .	3
2.3	Recommended Requirements . . . . .	3
2.4	Suggested Requirements . . . . .	4
<b>3</b>	<b>Concept</b>	<b>5</b>
3.1	Use Cases . . . . .	5
3.2	Activity Diagram . . . . .	6
3.3	Technologies Used . . . . .	6
3.4	Architectural Diagram . . . . .	7
3.5	Design Rationale . . . . .	7
	3.5.1 Architecture . . . . .	7
	3.5.2 Technologies . . . . .	7
3.6	Conceptual Model . . . . .	9
	3.6.1 Quiz Makers . . . . .	10
	3.6.2 Quiz Takers . . . . .	13
<b>4</b>	<b>Development Process</b>	<b>16</b>
4.1	Development Timeline . . . . .	16
4.2	Risk Analysis . . . . .	17
<b>5</b>	<b>Test Plan</b>	<b>18</b>
5.1	Unit Testing . . . . .	18
5.2	Client Testing . . . . .	18
<b>6</b>	<b>Social Implications</b>	<b>19</b>
6.1	Ethical . . . . .	19
6.2	Social . . . . .	19
6.3	Political . . . . .	19
6.4	Economic . . . . .	19
6.5	Health and Safety . . . . .	20
6.6	Manufacturability . . . . .	20
6.7	Sustainability . . . . .	20
6.8	Environmental Impact . . . . .	20
6.9	Usability . . . . .	20

<b>7</b>	<b>Difficulties Encountered</b>	<b>21</b>
7.1	System Bugs . . . . .	21
7.2	COVID-19 Pandemic . . . . .	21
<b>8</b>	<b>Suggested Changes</b>	<b>22</b>
<b>9</b>	<b>Lessons Learned</b>	<b>23</b>
<b>10</b>	<b>Installation Guide</b>	<b>24</b>
10.1	Getting the Code from Github . . . . .	24
10.2	Running the Code . . . . .	24
<b>11</b>	<b>User Manual</b>	<b>25</b>
11.1	Quiz Maker . . . . .	25
11.1.1	Logging In . . . . .	25
11.1.2	Creating a Quiz . . . . .	25
11.1.3	Looking at Completed Quizzes . . . . .	25
11.2	Quiz Taker . . . . .	25
11.2.1	Logging In . . . . .	25
11.2.2	Taking a Quiz . . . . .	26

# List of Figures

3.1	Use Case diagram . . . . .	5
3.2	Quiz Maker Activity Diagram . . . . .	6
3.3	Quiz Taker Activity Diagram . . . . .	6
3.4	Architectural Diagram . . . . .	7
3.5	Initial Login . . . . .	9
3.6	Sign Up Page for Assessment Creators . . . . .	10
3.7	Home Page for Assessment Creators . . . . .	11
3.8	Editing Surveys . . . . .	12
3.9	Student SignIn Page . . . . .	13
3.10	Sample Question . . . . .	14
3.11	Assessment Completion Page . . . . .	15
4.1	Development Timeline . . . . .	16

# List of Tables

4.1 Risk Analysis . . . . .	17
-----------------------------	----



# Chapter 1

## Introduction

### 1.1 Motivation

Stress is a major issue that affects students and their ability to succeed on tests, and the way tests are currently conducted only aggravates this problem. The process of taking standardized tests are extremely stress-inducing; students have to sit for several hours, answering as many questions as correctly as they can. Standardized tests have been shown to not be accurate indicators of who will succeed and who will not. The value of a standardized test score can be so distorted and skewed by outside factors that it is no longer useful. The SAT and GRE exams, for example, are only used by university programs to see which students surpass a certain floor score. The stress of these tests detracts both student performance and the ability of universities to accurately evaluate potential candidates. When students' futures are determined by their scores on standardized tests, it is urgent to reduce stress to maximize student performance and to level the academic playing field.

### 1.2 Problem

There are currently no good solutions to this problem. The most obvious option is to hand-write tests for each student; this is not efficient in terms of time or human resources due to the sheer number of students taking any sort of test. The test would also be static during the test period, and would not be able to respond dynamically to the student taking it. The only other option is to radically restructure the way we evaluate students, moving away from standardized tests. While this would broaden student evaluation to be more holistic, the transition would be costly and slow to implement. Even if this type of evaluation was the end goal of the education system, our solution would serve to augment traditional standardized testing between these phases.

### 1.3 Solution

To combat this issue, we propose a framework for creating dynamic, adaptive assessments. An assessment made using our framework would adjust in real time while a student takes it, changing the order and difficulty of questions to best

respond to the student's current performance. For example, rather than giving many difficult or easy questions in a row, the framework would determine the difficulty of the next questions on a test using how well the student had answered the current question.

No traditional solution can currently match the benefits of ours. Handcrafted tests are not feasible at the scale that standardized testing demands, and even then, a handcrafted test is static. Our framework is dynamic, compensating for optimal student performance in real time. Its flexibility and adaptability also means it could even extend outside of academia to job interviews, psychology, and therapy.

## Chapter 2

# Requirements

In this section, we list the design constraints for our project, as well as listing the requirements that fall under critical, recommended, and suggested requirements.

### 2.1 Design Constraints

In no particular order, the following is a list of design restraints for our application.

- The system must be delivered before the end of Spring Quarter 2020.
- The solution should be a web application.

### 2.2 Critical Requirements

In no particular order, the following is a list of requirements our project must meet or exceed.

- Users are able to create assessment templates.
- Users should be able to assign weights to questions in their assessments.
- Users should be able to distribute their assessments.
- The assessments should be generated dynamically based on the performance of the user taking them.
- Assessments responses should be stored after submission.
- Assessments responses should not be editable after submission.

### 2.3 Recommended Requirements

In no particular order, the following is a list of requirements our project should meet or exceed.

- The ability to start and stop a collection period—that is, a date or time to start collecting and a date or time to stop collecting responses to the assessment—at will should exist.
- Our product is as user-friendly and as simple to use as possible given the time constraint and other system functionality priorities.

## **2.4 Suggested Requirements**

In no particular order, the following is a list of requirements that would be nice if our projects meets or exceeds.

- A non-authenticated user should not be able to access admin functionality.
- The ability to set a collection period by setting a start date or time in the future and an end date or time after the start date should exist.
- Assessment results will be made available to the assessment creator for export after the collection period.
- Admin functionality that should exist includes the ability to edit questions in the assessment.

# Chapter 3

## Concept

### 3.1 Use Cases

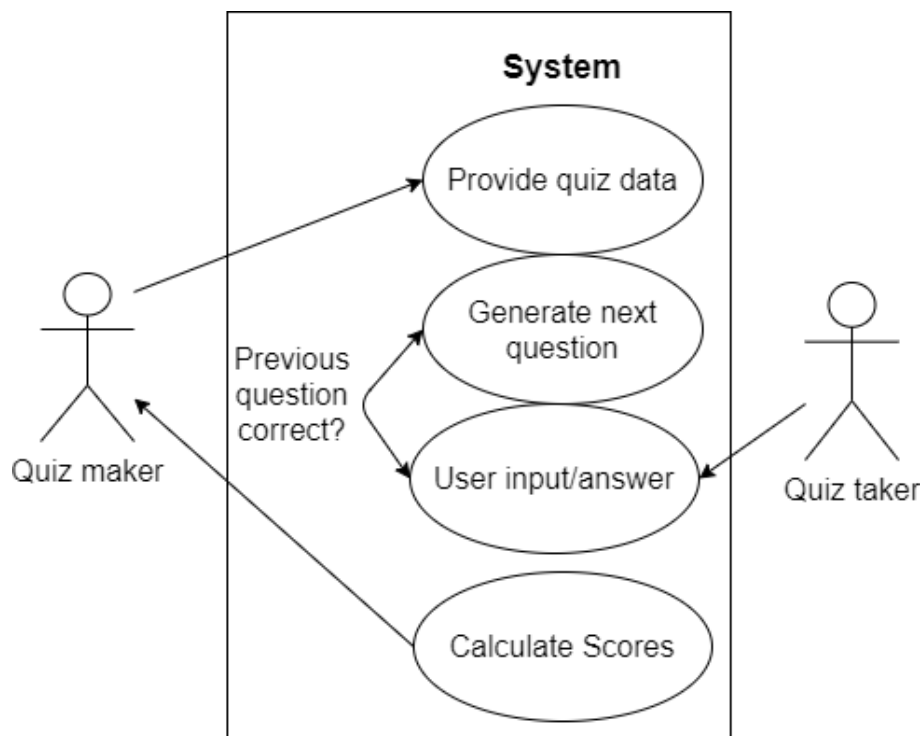


Figure 3.1: Use Case diagram

Figure 3.1 shows a typical user experience for using our application. There are two users, an quiz maker and an quiz taker. The creator is responsible for populating their quiz with questions and providing difficulty levels (weights) to each question. After creating their quiz, they will share and distribute it to the users who will take it. The quiz taker receives the assessment and is responsible for providing inputs/answers to the questions. The system will adjust what the next question will be based on whether the assessment taker got the current question right or wrong. Lastly, the system calculates the scores and sends a report to the quiz maker.

## 3.2 Activity Diagram



Figure 3.2: Quiz Maker Activity Diagram

Figure 3.2 shows the flow of actions of the person who is using our product to create assessments. First, the creator initializes a quiz template, which will be the base on which individually produced quizzes will be generated. This user will enter questions or edit in a pool. They will then attach tags to these questions which the system will use to organize individual assessments and analyze test taker performance.

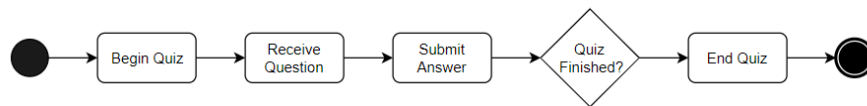


Figure 3.3: Quiz Taker Activity Diagram

Figure 3.3 shows the flow of actions the person taking the assessments. It is simple; this user starts the quiz and receives one question at a time, selecting and answering each question until the quiz is over.

## 3.3 Technologies Used

1. Figma
2. HTML, CSS, Javascript, Node.js
3. Github

## 3.4 Architectural Diagram

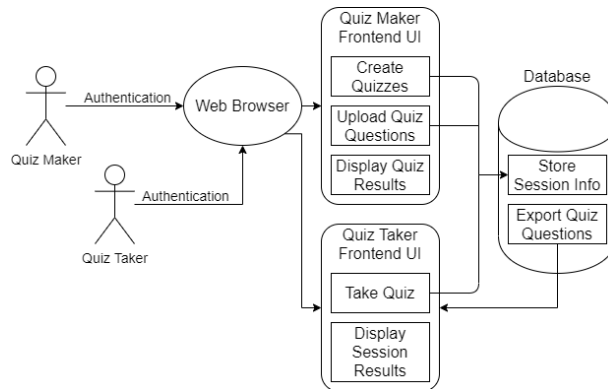


Figure 3.4: Architectural Diagram

The architectural diagram seen in Figure 3.4 shows the concepts and elements that are part of the Smart Quizzes web application. It is service-oriented. Small databases are used to temporarily hold Test Taker data and Quiz data from the Test Maker. Data is stored from their respective services provided: Test Taker data taken from Test Taking service, and Quiz data stored from Test Making service. Both the Test Maker and the Test Taker interact with our application via the Smart Quizzes Framework.

## 3.5 Design Rationale

### 3.5.1 Architecture

We chose a service-oriented architecture because we offer two different essential services, one for a user who creates assessments and one for a user who fills them out. A service-oriented architecture allows us to provide functionality that has been abstracted away. The users need only to be concerned with filling in inputs, while our system interacts with data held in both databases to dynamically generate quizzes.

### 3.5.2 Technologies

- HTML is the most basic language used for creating web pages. It will be the base for our application, while it will be bolstered by our use of Javascript and other libraries.
- CSS is the industry standard for styling web pages. We will use CSS to make our application aesthetically pleasing and uniform.
- JavaScript is a language that will enable us to create dynamic and reactive web pages. This is crucial for our application, which needs client-side behavior on top of our HTML web pages. JavaScript provides the base for an interactive application.

- Node.js is an open-source, cross-platform JavaScript runtime environment. It is event-driven and non-blocking, which allows our application to stay light and efficient. Node.js is what we will use for I/O functions in regards to data and user interaction,
- GitHub is our method for version control. We will use it to collaborate and work remotely while keeping pull and push requests organized and conflict-free.



## 3.6 Conceptual Model

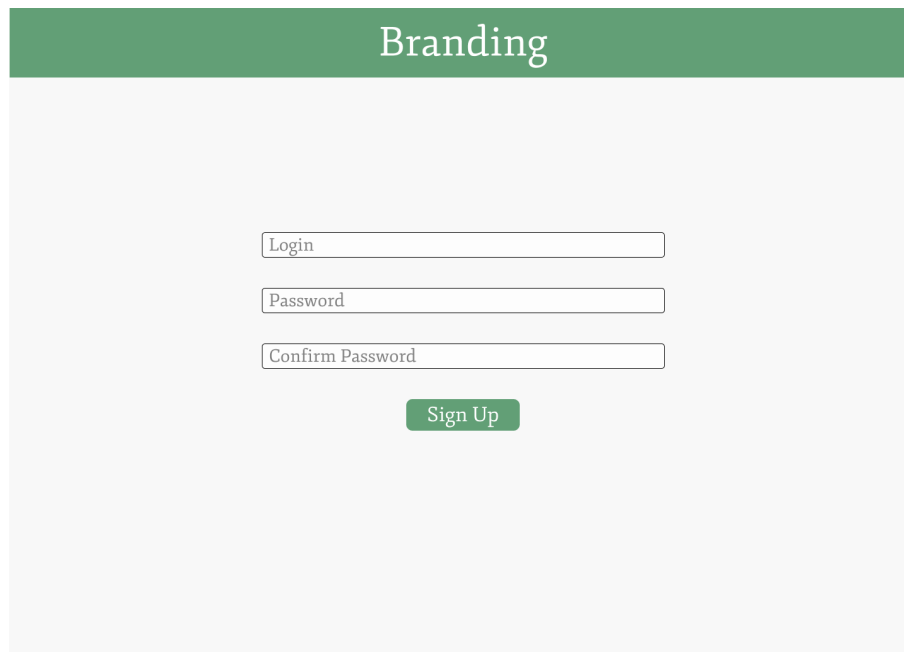
Our conceptual model is meant to show our vision of the branding and styling of our product. Core functions described in this model are open for change as the project goes further into development. The visual design of our product is meant to be clean and intuitive while offering quality interactivity to users. Two types of users will interact with our system: quiz makers and quiz takers.



Figure 3.5: Initial Login

The login page is minimalist and simple. Depending on if the user is a quiz taker or a quiz maker, they will click on their respective buttons, as shown in Figure 3.5.

### 3.6.1 Quiz Makers



The image shows a sign-up page for assessment creators. At the top, there is a green header with the word "Branding" in white. Below the header, the page has a light gray background. In the center, there are three input fields stacked vertically: "Login", "Password", and "Confirm Password". Below these fields is a green button with the text "Sign Up" in white.

Figure 3.6: Sign Up Page for Assessment Creators

On the sign up page, a user will enter their desired username and password. If the username is already taken, they will have to choose a new one. Once they choose a username and confirm their password, they can click the sign up button to register and be redirected to their home page, which is depicted in Figure 3.7.

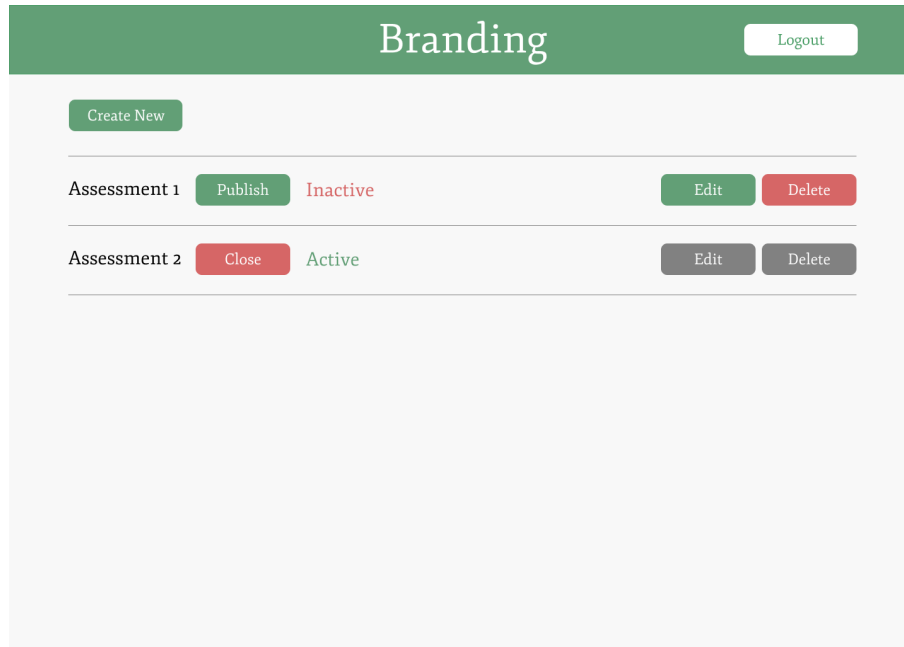


Figure 3.7: Home Page for Assessment Creators

On the home page, users will be able to see all the quiz templates they have created and their statuses. If a quiz is active, this means that it has been published and shared so that people may access it and take it. The user may not edit or delete a quiz once it has been published in this manner. To publish a quiz, the user must click the green "Publish" button. To close the quiz off from public access and to be able to edit it again or delete it, the user must click the red "Close" button.

To create a new quiz template, the user must click the green "Create New" button at the top. To delete a quiz, the user must click the red "Delete" button across from the quiz they wish to delete. When the user clicks the green "Edit" button (unclickable buttons are rendered in gray), they will be directed to the quiz editing page, depicted in Figure 3.8.

Once a user has finished, they can exit their session by clicking the white "Logout" button in the top right corner. Clicking this button will redirect them to the login page.

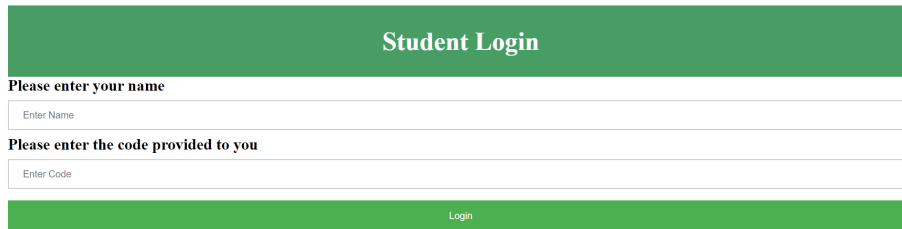
The screenshot shows a web interface for editing surveys. At the top, a green header contains the word "Branding" and a "Logout" button. Below the header, there is a "Back" button and the title "Assessment 1". The main area displays two question cards. The first card contains a question and four radio button options, with "Option 2" selected. The second card contains a question and four radio button options, with "Option 3" selected. To the right of the second question card, there are input fields for "Choices" (4), "Correct" (3), "Tags" (Lorem, Ipsum), and "Weight" (10). Below these fields are "Save" and "Delete" buttons. At the bottom of the second question card, there is a "New Question" button.

Figure 3.8: Editing Surveys

The user is able to add and edit questions on the editing page, as seen in Figure 3.8. To add a new question, the user will click the green "New Question" button underneath the bottom-most question. Once the user has created a question, they may edit the question text, the number of choices, the text of each choice, the number of the correct choice, the tags associated with the question, and the weight of the question. These inputs help our system determine the order in which the questions appear when other users take the quiz. By clicking on the green "Save" button next to the question, the user may save the question once these inputs are filled in. They may also delete question by using the red "Delete" button.

To navigate back to the home page, the user must click the green "Back" button at the top of the screen. To end their session, the user can click the white "Logout" button at the top right of the screen.

### 3.6.2 Quiz Takers



The image shows a 'Student Login' form. It features a green header with the text 'Student Login'. Below the header, there are two input fields. The first field is labeled 'Please enter your name' and contains the placeholder text 'Enter Name'. The second field is labeled 'Please enter the code provided to you' and contains the placeholder text 'Enter Code'. At the bottom of the form, there is a green button labeled 'Login'.

Figure 3.9: Student SignIn Page

A quiz taker has a different login page, where they will be asked to provide their names and codes provided to them by quiz makers. The sample login page for quiz takers can be seen in in Figure 3.9

Quiz takers see a different view of these quizzes, as they will receive access to quizzes once quiz makers publish them. A sample quiz question view can be seen in Figure 3.10.

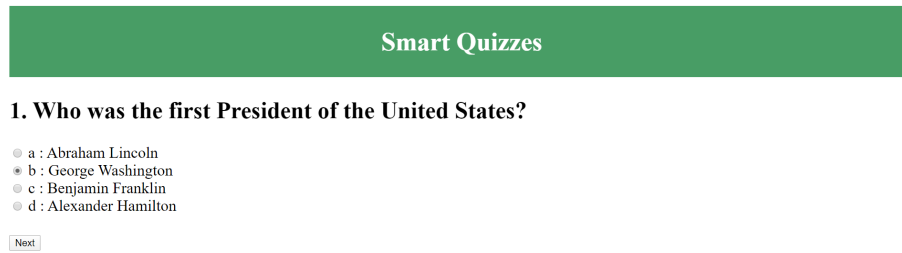


Figure 3.10: Sample Question

A quiz taker will take a quiz while encountering one question at a time. The title of the quiz and the question number are at the top, and the text of the question is seen below. In this sample, the user has four options to choose from. Once an answer has been selected, they can click the "Next" button to submit their answer and move on to the next question. Once they have answered all the questions, they will be directed to a page that indicates that they are finished. This page is depicted in Figure 3.11.

## Smart Quizzes

Final Score: 5/5

Figure 3.11: Assessment Completion Page

When they have finished, the quiz taker will see a plain page with their results correct out of the total questions.

# Chapter 4

## Development Process

### 4.1 Development Timeline

	Legend:	Andres	Cole	Team	Deadline						
	Fall Weeks 1-4	Fall Week 5	Fall Weeks 6-9	Fall Week 10	Winter Week 1	Winter Week 2	Winter Weeks 4-10	Spring Weeks 1-7	Spring Week 8	Spring Week 9	Spring Week 10
Problem Statement											
Design Document and Review											
Update Design Review											
User Interface											
Back-End Implementation											
Database Implementation											
Login Implementation											
Testing											
Final Presentation											
Final Report											
Final System											

Figure 4.1: Development Timeline

Figure 4.1 shows a rough outline of our planned schedule. Our design documents and planning were meant to be done early on in the development process. We have also established a timeline in which we plan to work on the project all throughout Winter and most of Spring quarter, and then finally testing it and fixing any errors in the second half of Spring quarter, before its presentation in the Senior Design Conference. We then plan on submitting the final report and product by the end of Spring quarter.



## 4.2 Risk Analysis

Name	Consequences	Probability	Severity	Impact	Mitigation
Technological Difficulties	Incomplete Project	0.3	9.0	3.6	Research
Bugs Found	Unintended Behavior	0.99	3.0	2.97	Extensive Testing
Time	Incomplete Project	0.2	8.0	1.6	Sticking to the Schedule
Scheduling Conflicts	Production Delay	0.2	7.0	1.4	Extensive Communication

Table 4.1: Risk Analysis

Table 4.1 shows our risk analysis. It shows our predicted probabilities of potential risks we could encounter throughout the development process, their severity, impact on the project, and what we can do to mitigate them.

# Chapter 5

## Test Plan

In this section, we list the various ways in which we tested our program

### 5.1 Unit Testing

- We tested individual modules throughout development to ensure they were functioning properly and as intended. This also helped with finding and resolving bugs in our code.

### 5.2 Client Testing

- Originally we had planned to have clients test our product as a form of black-box testing, to see how average users would feel and react when logging in and taking a test.

## **Chapter 6**

# **Social Implications**

### **6.1 Ethical**

One ethical concern about our project was that of user privacy. We address this by not storing any sensitive user data in our database. In using Google Authenticate, we don't store personal information such as email addresses or names. Another ethical concern was that of cheating. We address this issue again using Google Authenticate. Test takers are meant to take their own quizzes, and the test maker, or teacher, can take it a step further by implementing their own extra form of security such test proctoring or a third party monitor, such as Lockdown Browser.

### **6.2 Social**

Our project has a strong societal impact. It is accessible to just about anyone with a computer and an internet connection. It's simple to use, and it provides an level playing field of sorts to test takers and/or students as it minimizes stress.

### **6.3 Political**

The biggest sociopolitical impact our project can have is that it would improve the education system in the United States, and could make standardized test taking user and not as overbearing.

### **6.4 Economic**

Our target audience was meant to be for schools, and eventually school districts, so we believe that they would have the means of providing computers and an internet connection. The actual cost of development was minimal, and we intend to deploy this as a free program to use.

## **6.5 Health and Safety**

Our program would be very safe to use. In terms of mental health, we believe it would be an improvement over current testing methods and procedures as it would alleviate stress for test takers.

## **6.6 Manufacturability**

Developing the app does not require much resources. Optimizing it for web use and developing improvements for the app would only take time.

## **6.7 Sustainability**

Our program would be viable for years to come. Through further testing and improvement, we believe it could so prolonged use in test taking and in the education system in general. We have a solid foundation to build upon, and updates could be readily made to fix bugs, and implement improvements and features.

## **6.8 Environmental Impact**

Our program has no direct environmental impact. It has an indirect environmental impact because it requires a computer to use, and computers have their own carbon footprint.

## **6.9 Usability**

Our program is simple to use. There is little to no clutter and the instructions are fairly straightforward.

## **Chapter 7**

# **Difficulties Encountered**

### **7.1 System Bugs**

One of the main obstacles we encountered throughout the development process were system bugs and bugs in the code. We tried to address these bugs and resolve them to the best of our abilities throughout development and unit testing, and it was difficult to resolve some that seemingly had no explanation for their occurrence.

### **7.2 COVID-19 Pandemic**

We did not anticipate going through a pandemic that would lead to us having to go into quarantine. Not being in school and able to meet in person made it difficult to coordinate meetings and communicate. However, it did not make things impossible as we were able to adjust accordingly and continue through development. Our program was mostly able to be done using a computer, so it did not hinder our actual development progress.

## Chapter 8

# Suggested Changes

With our project subject to a designated timeline, we have some suggested changes to our program that could lead to overall improved usability and user experience. First off, improving the question weight system to be more modular would greatly improve our programs versatility and usage for different/specific tests. Currently, it only accounts for a weight system of numerical difficulty of 1 through 3. It would be worthwhile to improve this by having a more broad difficulty system, or a different weighting scheme where the questions can be labeled specifically, such as "math", "geometry", "writing", etc. Another potential improvement could be made in the User Interface could be improved. While it is not a big concern for the functionality of the program as a whole, the User Interface should be easy to look at and key functions and buttons should be easy to identify. This is of course subjective to user opinion, and would therefore require testing amongst clients.

## Chapter 9

# Lessons Learned

We learned some valuable lessons throughout our design process. The first was having a plan before actually starting development. It was quite difficult establishing our design constraints early on, and quite a bit of time was spent communicating with our advisor on what things we should consider in order to establish our design constraints. These were prone to change throughout part of the development process, and it slowed down our actual development as we were unsure of what we should be doing at first. Eventually though, we had a more concrete idea of what our constraints and requirements should be, and this helped us with what to do. Another lesson we learned was communicating effectively. With the COVID-19 pandemic and the subsequent quarantine, it was rather difficult to communicate effectively and hold each other accountable, especially when we could not meet up in person. We had to adapt and use different form of communication, which mostly included online video calls, emails, and other forms of messaging. Lastly, it was best to know and understand the coding languages we were going to use. Most of us had experience with some of the languages before, such as Javascript. We did not know about User Interface development and had to learn about HTML and CSS throughout the development process as well. It ended up working out but more time could have been dedicated to improving the User Interface to make it look more professional, especially if we had a deeper understanding of HTML and CSS.

# Chapter 10

## Installation Guide

### 10.1 Getting the Code from Github

Begin by cloning our repository using the following command:

```
git clone https://github.com/Some1Else46/smart-quiz.git
```

### 10.2 Running the Code

The code can then be run by simply clicking on the "login" HTML file in the file explorer of your computer.



# Chapter 11

## User Manual

### 11.1 Quiz Maker

#### 11.1.1 Logging In

Quiz makers will click on the "Make a Quiz Button". They will then be asked to submit their credentials when logging using the authenticator. If they don't already have an account, they will have to make one in the signup page before logging in.

#### 11.1.2 Creating a Quiz

Quiz makers will then be able to see an interface where they have the option of creating a new quiz. They will click on that and then be asked when the due date is or how long the quiz will be. They will then be asked how many people will be taking the quiz and decide that. Lastly, they will then be asked to upload a formatted JSON file containing the quiz questions.

#### 11.1.3 Looking at Completed Quizzes

The quiz makers will also have the ability to see some of their past quizzes that they uploaded, as well as the results for the past quizzes

### 11.2 Quiz Taker

#### 11.2.1 Logging In

Quiz takers will click on the "Make a Quiz Button". They will be asked to submit their credentials when logging using the authenticator.

## **11.2.2 Taking a Quiz**

Quiz takers will then be taken straight to their quiz after they entered their login and code, and will simply go through answering the quiz questions. Once they are done, they can view their results and then safely exit the quiz.