Software Design Document

2/14/2020

DigiTool Inc.

Project Sponsor: Xi Zhou

Team Faculty Mentor: Fabio Santos

Team Members

Traybourne North

Spencer Clark

Langqing Dai

Miguel Quinones

Version Number: 1.1

Table of Contents

1.0 Introduction

1.1 Project Background

The Digital Logic course here at NAU is one of the most important courses an engineering student can take, whether they're Mechanical Engineers, Electrical Engineers, etc. The course teaches students about several important logical reasoning concepts such as Karnaugh Maps, truth tables, and numeric conversions. Naturally, students might encounter issues when dealing with these subjects, such as trying to understand them or feeling like they don't have enough time to practice them. Last semester, there were about 75 students enrolled in the course, so if we assume that at least a third of those students had trouble understanding the topics, they would then have trouble participating in the course. Therefore, it is important that they are able to understand these topics immediately.

1.2 Project Sponsor

Our project sponsor is Dr. Xi Zhou who is an Assistant Professor of Practice here at NAU. He teaches the Intro to Digital Logic course on campus and is very invested in the issues that students have with the course. He has explained that the course is structured so that the topics described above are introduced and worked through one at a time. In other words, students begin with Karnaugh Maps and truth tables and from there, they move onto numeric conversions, such as decimal to hexadecimal, and so on. For each topic, students practice working on examples and homework in order to better understand the topic at hand before moving on. This workflow follows the same general structure as most courses.

1.3 Problem Statement

Students may find themselves facing difficulty learning the topics of the Intro to Digital Logic course. Often times these students are told that their only solution is to just keep looking over their problems again and again or talk with a professor during their office hours. However, these might not be viable options for students as they have other tasks they have to worry about as well, such as work for other classes, a job, and more. Due to potentially having tight schedules, these students don't have the time to continually try to figure out problems on their own or to meet with their professor to talk. This is why students need an option that they can access on their own time, whenever they need to.

1.4 Planned Solution

Our solution to the problems described above is a web application that will allow students to practice the content of the course, anytime they need to. Our web application will be designed in Node.js, as this will make front-end and back-end development easier through the use of libraries and frameworks. We are using the React framework within Node.js to create components of the class topics and because they are being implemented as components, they will be reusable. For our back-end, we will be using a database on MySQL to keep track of user login information and their current progress within the application. Node.js will also allow us to communicate with the database. Through the use of Node.js and the various libraries and frameworks, we will be able to deliver the web application that we have been tasked with creating.

1.5 Key User Level Requirements

The key user level requirements we obtained were:

- Students need to be able to save their progress within our modules
- Each module within the application will award stars to students based on how well they answer questions and these stars will allow students to progress through the modules

These requirements are used to shape our application, from its intended use to how we implement our functional requirements.

1.6 Functional Requirements

We gathered our functional requirements based off of the key user level requirements which are:

- The system will have functions that allow students to practice module questions
- The system will have a function that gives stars to the user based on the number of missed attempts they used for each question
- The system will save a user's progress

1.7 Non-Functional Requirements

Non-Functional requirements that should be noted while developing this web application are:

- System must support 150 users at the same time without affecting the web application's performance in a large way
- Response time must be between 0.1 - 2.5 seconds
- System must be operational year-round

1.8 Environmental Constraints

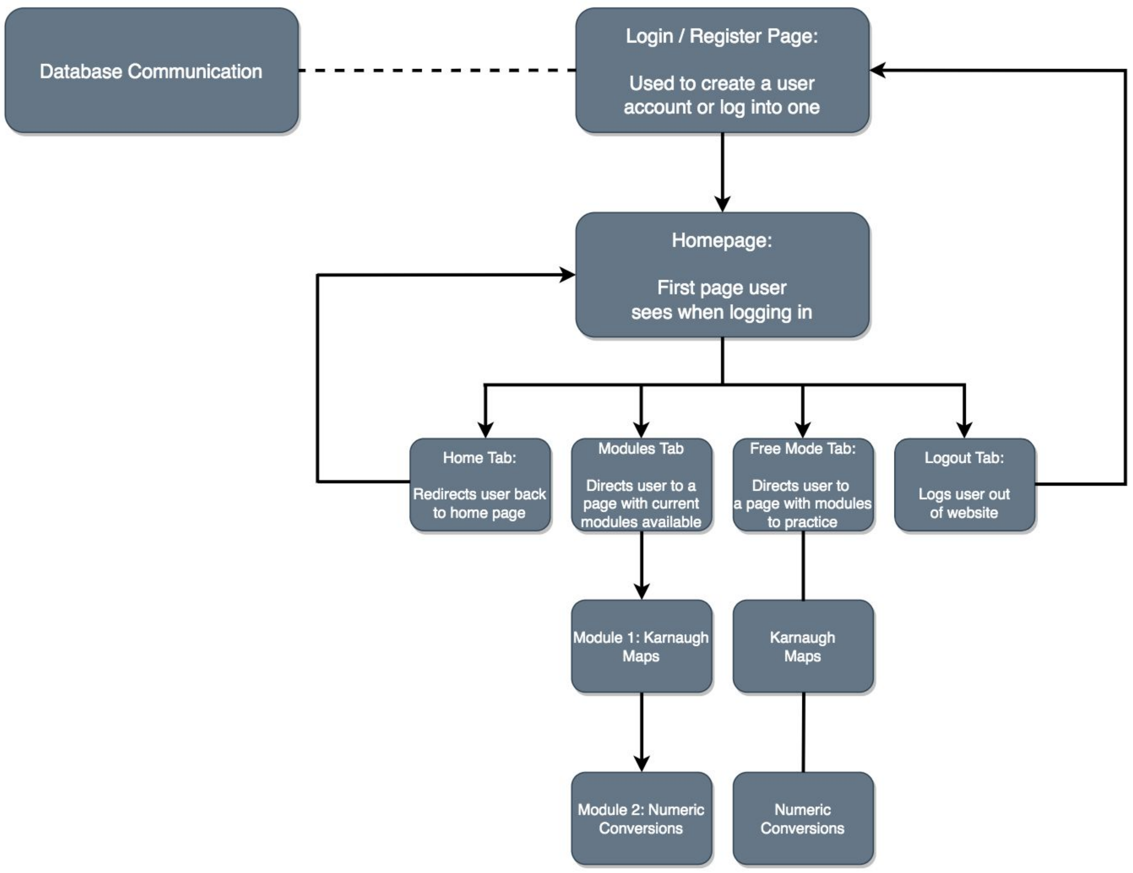
The environmental constraints that have to be followed for this project are:

- Our web application uses HTML5, meaning that browsers supporting HTML5 must be used to access the application. This includes:
    - Internet Explorer 6 & above
    - FireFox 3.5 & above
    - Google Chrome 8 & above
    - Opera 11 & above
    - Safari 5 & above
- Computers need at least 100 - 500MB of free memory to install one of the browsers mentioned above
- An internet connection is required to access our web application

2.0 Implementation Overview

The general idea is that we will have a series of interactive content modules and each one has a set of questions to help a student practice aspects of digital logic. They will initially be prompted with a login that redirects them to a main page. From the main page, they can navigate to a module or go into our practice mode to practice a module. All other navigation will be done automatically by submitting answers to questions given. Figure 1 shows an idea of how everything within our website will be implemented.

Figure 1


Database Communication
Login / Register Page: Used to create a user account or log into one
Homepage: First page user sees when logging in
Home Tab: Redirects user back to home page
Modules Tab Directs user to a page with current modules available
Free Mode Tab: Directs user to a page with modules to practice
Logout Tab: Logs user out of website
Module 1: Karnaugh Maps
Karnaugh Maps
Module 2: Numeric Conversions
Numeric Conversions

User navigation through the system

Our web application needs to be portable and work across a variety of systems. To do this, we will be creating a web application that uses JavaScript since it will run in an internet browser. An internet browser is an incredibly low barrier to entry and available on many systems. Inside the browser we will be using Node.js, React.js, and an npm (node package manager) library for communicating to our database. On the development side, we will also be using TypeScript and WebPack.

2.1 Node.js

Node.js is a runtime environment for JavaScript and it allows us to use things not standard to running JavaScript directly in the browser. This includes non-php based database communication and more dynamic page content.

2.2 React.js

React.js is a web framework designed for user interfaces and gives us a way to break our web application in sections that can be more easily customized and integrated with many different web pages. This is helpful when creating our page content and it takes advantage of Node.js's ability to be more dynamic.

2.3 MySQL

MySQL is a small library that contains a few JavaScript based commands that will allow us to communicate and modify our database from within Node.js. We won't have to use external php files which cleans up project files and removes complexity.

2.4 TypeScript

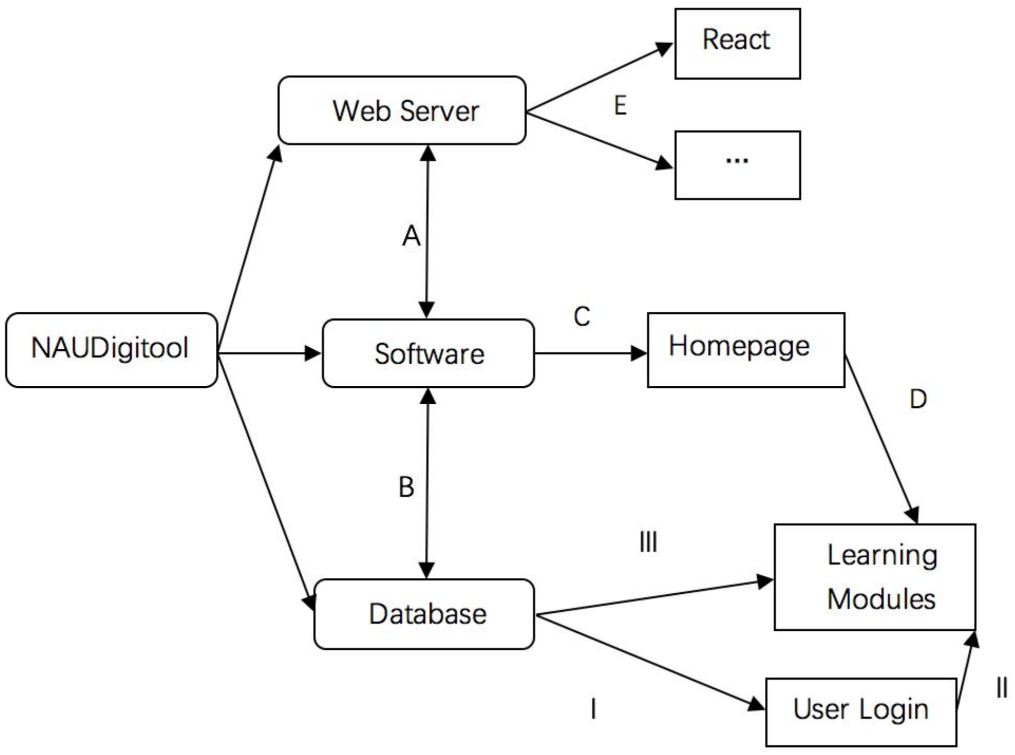
TypeScript is a superset language of JavaScript that forces strict typing. It forces the assignment of types to basics and objects as well as the passing of parameters to functions. This removes the guessing of variables that is standard to JavaScript and prevents runtime errors that the compiler can't catch, ensuring that our app will be more stable and easier to develop. The reason this is on the purely developmental side is that when you compile TypeScript it turns directly into JavaScript.

2.5 Webpack

Webpack is a new addition since the design is purely for ease of development and does not contribute much to the end goal. It is a way to bundle all of our scripts and their dependencies into one file. This removes the need for a large amount of linked scripts or fetching dependencies that need to be loaded for a page. It speeds up our ability to create pages and ensures that everything the page needs to execute is available.

3.0 Architectural Overview

Figure 2



Control Flow of Architecture

3.1 Key Responsibilities & Features of Each Component

Digitool has 3 key responsibilities, which include the database, software and web server. The database is used to collect user information. This database needs a higher level of security to ensure that the user's information is protected and secure. It also needs some stability and scalability because this database needs to be used continuously and data could possibly be added or changed constantly. In addition, portability may also be needed because it may be necessary to redeploy their servers on other systems when the user changes.

The software is mainly written in JavaScript and will be presented in the form of web pages. The software is targeted at students and professors with different servers. The main body of the software are learning modules that can either give Karnaugh Map practice or Numeric Conversion practice.

Website services can facilitate users to log in, register, and update a series of operations and website services facilitate our various tests during the design stage.

3.2 Communication Mechanisms & Control Flows of Architecture

Based from Figure 2, we have 8 main communication mechanisms and flows:

1) A: Website server that helps users login and register. The software can also be tested by the website server.
2) B: The account information of both students and teachers is necessary, as well as the internal information of the system, so we need a database to support.
3) C: When you start using DigiTool, you will be taken to a home page and then you can choose different modules.
4) D: It has many learning modules, and the modules are arranged in order from simple to difficult.

5) E: The web server will do the react and some other jobs to help the system running well.
6) I: When the user does log in, it needs the user information from the database.
7) II: If a user wants to select a module to study, they must log in first.
8) III: In each module which is chosen, it will have question information and answer information that's either from the database or stored in our HTML / JavaScript files..

3.3 Architectural Style Influences

Our architectural style is a layered pattern and with this architectural pattern, we only need to focus on the necessary places during the development process. From there, we can easily replace the old implementation. The dependency between the various parts is low and joint problems are not easy to occur. When a problem occurs in one layer, we don't need to worry about the other parts. Because of this layered design, we can focus more on the standardization of procedures. The most important point is that when we want to call a certain part of the logic, it doesn't take much time to pass one of the architectures.

4.0 Module and Interface Descriptions

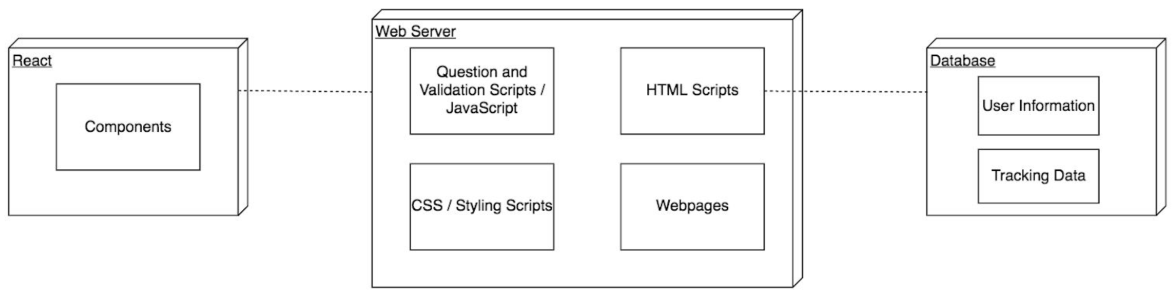
This section will provide descriptions about our modules and interfaces that we plan to use when creating our Digital Logic Self-Study Toolkit. Each description will explain the responsibilities of the components used and how it gels with other parts of our web application. A UML class diagram of the classes involved in this component will also be provided, since our project involves using an object-oriented design. Lastly, each description will conclude with public methods along with the return types and parameters for the classes that each component consists of.

4.1 Webpages

Components within our webpage is something that is heavily needed within our web application since it can provide strong encapsulation for reusable components such as a reusable user interface. Since we are using React to develop our web application, React will automatically keep the DOM in sync with our data which makes things beneficial towards us. The main responsibility of this structure is to allow the user to navigate smoothly through the web application.

Once a user logs into the web application they will be able to access a Home Tab, Modules Tab, Free Mode / Practice Tab, and a Logout Tab. The Home Tab will allow the user to navigate back to the homepage, while the Modules Tab will navigate the user to a list of modules to choose from (Karnaugh Maps, Numeric Conversions, etc.). Our goal is to unlock modules as users complete the previous module so that it acts as a learning platform similar to some of the ones here at Northern Arizona University (BbLearn, LearnSmart, WebWork, etc.). Figure 3 shows the overall components of our overall software design.

Figure 3



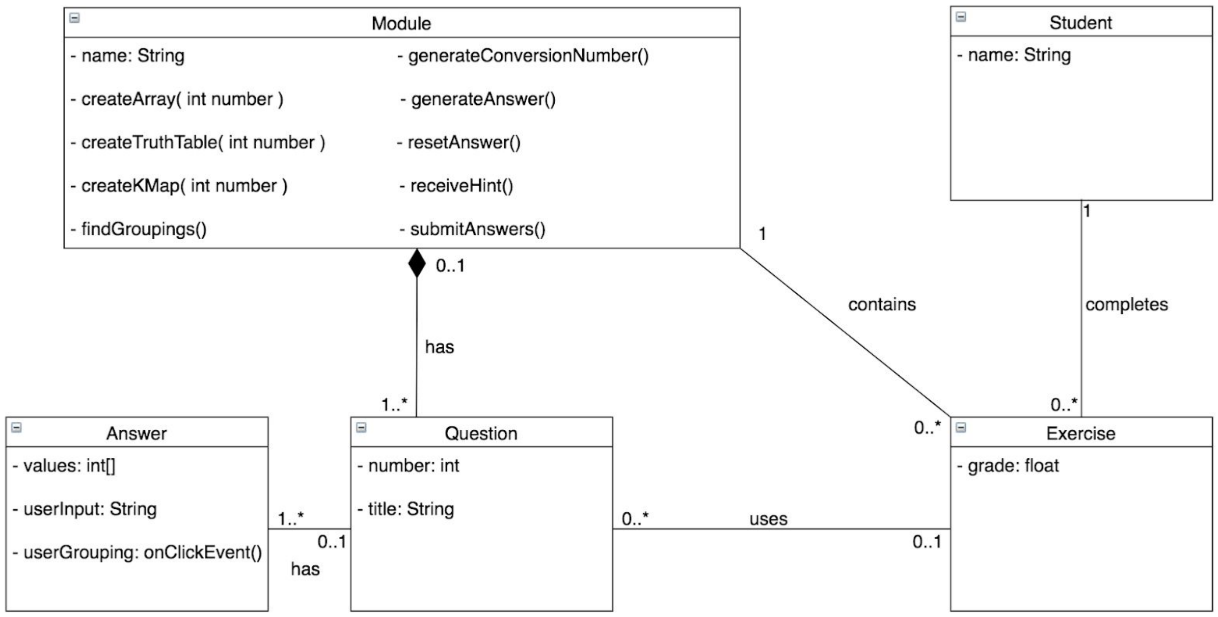
Components of the System

Since this component primarily involves navigating from webpage to webpage, this interface doesn't involve any public methods along with return types and parameters.

4.2 Karnaugh Maps

The Karnaugh Maps module is the first module that users can access before advancing to the next module, which is Numeric Conversions. This component acts as a large part of a users interaction with our web application. This component also acts as a stepping stone towards advancing to the next module by receiving a mastery score per module. This is all judged with stars. If a user receives a mastery score of 80% or higher, the next module will unlock, but if they do not reach that goal, they will have to attempt the Karnaugh Maps module again. Not only will users be able to type in answers within our web application, but they'll also be able to create groupings using our sketch color tool. Figure 4 displays a UML of what Module 1 and 2 entail. The user will be able to submit answers, clear answers, and receive one hint per question. This is just a rough idea of how the interface will look when accessing the Karnaugh Maps module. Changes may be added for modularity and reusability if necessary.

Figure 4



Karnaugh Maps and Numeric Conversions: UML

Karnaugh Maps revolve around an array class, which contains a Karnaugh Map array, grouping array, boolean flag, and a string that generates the correct equation. The Karnaugh Map array contains the array that's shown in the truth table, while the grouping array is used to add matching ones. The matching ones are grouped based on 2 to the nth power (2^2, 2^3, etc). Once created, the grouping array will be compared to how the user groups the ones together (sketch tool). The boolean flag would then be used to check whether there's a valid group in the grouping array.

An array class will be created that takes in an integer parameter (3) that will represent the number of variables that will be provided in the Karnaugh Map. Once passed, 2 will be raised to the integer parameter to represent the number of cells contained in the array. A for loop will then run and every time the array enters a new index, a random number will be generated (0 or 1) and added into the array. Once finished, the class will return the generated Karnaugh Map array.

Within that class, public methods are created to check if octet, quad, or pair groups can be formed within the Karnaugh Map array generated. A for loop runs that begins at index 0 and ends at the length of the array minus one. At every index reached, helper methods are called that check to see if ones can be formed. Each helper method passes in anywhere between 2 to 8 index parameters that get quickly checked for grouping. Nothing gets returned but certain grouping array indexes do get changed if there are groups of ones formed from the Karnaugh Map array. As groups are being formed, another method gets called that passes in indexes and generates a string equation. Once again it doesn't return anything, but it does add to the equation string. The sketch tool will consist of colors that the user can pick from to trace circles around the 1s grouped together. Since this is a mouse click event, a method will have to be created that checks the grouping array as well as traced group of ones to make sure they match with each other.

4.3 Numeric Conversions

Once the user has received an 80% or higher on Module 1 (Karnaugh Maps), Module 2 (Numeric Conversions) will unlock. Numeric Conversion activities specifically focus on converting binary numbers to hexadecimal numbers, decimal numbers to binary numbers, etc. The same grading system and user interface will be provided, the only difference will be the questions within the module. Once again this component acts as a stepping stone towards advancing to the next module by receiving a mastery score per module.

What ends up being used within this component depends on the question given. If the question is a decimal or hexadecimal number to binary number conversion, then a string array class will be needed since the binary number will consist of 0s and 1s. If the question is also a binary or decimal number to hexadecimal number conversion a string array class will be needed since the result could consist of the letters A through F. On the other hand, if the question were to consist of a binary or hexadecimal number to a

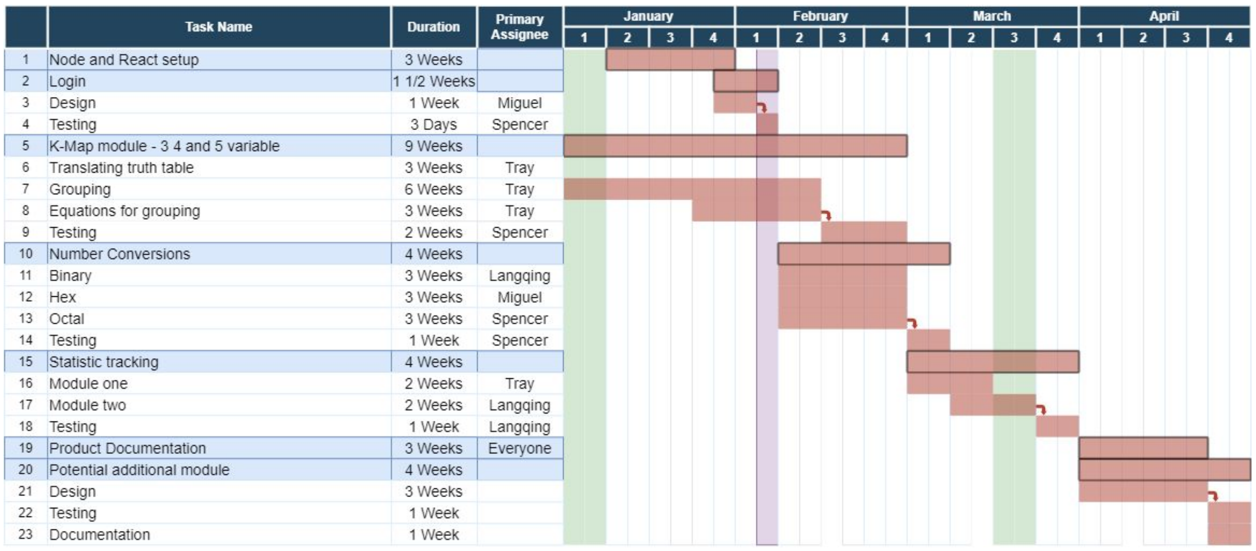
decimal number conversion, an integer array class would be needed because the answer would always result in a whole integer number.
The types of methods needed would have to be an array generator based on the questions given, an algorithm to solve the question, and an array resulting in the answer to the question. The result array would then need to be compared to the answer that the user enters to validate their answer. Since we are using React as a framework / library, various parts of a web page can be reused as components over and over again. Although we have a rough prototype, changes will be made within our modules to ensure that further development will be modular and reusable.

5.0 Implementation Plan

Due to the nature of our project, most of our time will be spent on creating content. The initial steps were laid out during last semester and put in place at the beginning of this semester creating the groundwork that we will build our content on. The login has been primarily done by Miguel. The most complex content module, our K-Maps, was worked on heavily by Tray before this semester and will be finished and ready for testing soon. Now that we have an example that the team is comfortable with, we will be splitting the next module into each of its functions so that each team member can work on an aspect over the next few weeks. With the completion of our second module we will have to start closing up the project by implementing the system for tracking usage statistics and subsequently creating the documentation that will be delivered alongside it. The work on the tracking system will be split into 2 teams; one for each content module with the second starting a little later due to a team still testing content from module two.

If there is any shift in our schedule it will primarily affect our ability to make a third content module. Delays will reduce our available time as the semester is limited, but this should not stop us from completing our primary goal of two content modules. There is also a possibility that the client is unsure about using a tracking system, as it was not stressed much in our meetings and would prefer a third content module. In this event, we would swap the timings of the two if data tracking becomes the extra. Figure 5 shows our planned scheduling for this semester.

Figure 5



	Task Name	Duration	Primary Assignee
1	Node and React setup	3 Weeks	
2	Login	1 1/2 Weeks	
3	Design	1 Week	Miguel
4	Testing	3 Days	Spencer
5	K-Map module - 3 4 and 5 variable	9 Weeks	
6	Translating truth table	3 Weeks	Tray
7	Grouping	6 Weeks	Tray
8	Equations for grouping	3 Weeks	Tray
9	Testing	2 Weeks	Spencer
10	Number Conversions	4 Weeks	
11	Binary	3 Weeks	Langqing
12	Hex	3 Weeks	Miguel
13	Octal	3 Weeks	Spencer
14	Testing	1 Week	Spencer
15	Statistic tracking	4 Weeks	
16	Module one	2 Weeks	Tray
17	Module two	2 Weeks	Langqing
18	Testing	1 Week	Langqing
19	Product Documentation	3 Weeks	Everyone
20	Potential additional module	4 Weeks	
21	Design	3 Weeks	
22	Testing	1 Week	
23	Documentation	1 Week	

Implementation schedule for the semester

6.0 Conclusion

To conclude, many students are still struggling with concepts discussed in the Introduction to Digital Logic course taught at NAU. Although students have access to tools that can help them with learning, these tools only give out answers rather than having students practice it on their own. That is why it is our goal to provide students with a Digital Logic Self-Study Toolkit to help them practice and understand content discussed in lectures.

From here on out, we believe that our prototype can advance and look much better than the previous one. Our goal is to get the login and modules done by spring break and after we receive feedback, we could add other modules within our web application. Although we could have minor obstacles in the months to come, such as creating a tracking system and saving and loading a student's progress, we are confident that we can get the job done in the months to come. We are positive that we can meet the goals and requirements given to us by our client and we foresee a fully functional web application in the future.