Final Report

5/7/2020

DigiTool Inc.

Project Sponsor: Xi Zhou
Team Faculty Mentor: Fabio Santos

Team Members

Traybourne North
Spencer Clark
Langqing Dai
Miguel Quinones

Version Number: 2.0

Table of Contents

1.0 Introduction

The Intro to Digital Logic course here at NAU is one of the most important courses an engineering student can take, whether they're a Mechanical Engineering major, Electrical Engineering major, or even a Computer Science major. A large number of students enroll in the course each year and we estimate this number to be around 75 students per semester. The course teaches several foundational topics such as translating Karnaugh maps into truth tables, creating groupings for those tables, and then finding the equations for those groupings. Another important topic of the course is performing numeric conversions between different bases, such as binary, octal, decimal, and hexadecimal. Due to the complexity of these topics, students might have trouble learning them and so having a tool that lets them practice these topics would be very beneficial for them.

Students only have a limited amount of time in class to work and get help and outside of class, they might not have the time to meet with their professor. This means when they need help, their only solutions are to either hire a tutor, which costs the student money, or try to solve the problems on their own, which can be very frustrating at times. Similarly, a busy professor might hire teaching assistants to help them, which costs the professor money. This is why our client wants a free tool that allows students to work on problems that cover topics taught in the course.

Our solution is to create a free web application that will allow students to practice problems based on the topics taught in the course anytime they want to. This solves our client's problems of needing a tool for his students to practice with, having that tool be free, and having it be accessible at any time. Additionally, it solves the problem of not having a full understanding of the topics that students struggle with. Our web application solves the problems of our client and those that students might have and in the following sections, we will go into more detail over this.

2.0 Process Overview

Throughout the course of the project our team used Agile principles to direct the flow of the project. This allowed us to be flexible with how we implemented the aspects of our project while still retaining a form of cohesion throughout it. Team members took on different roles as well, with each member focusing on one core aspect of the project to contribute to while also providing assistance to other members when any problems were encountered. In order to keep everyone on track and ensure that each member did their contributions, we would hold two weekly meetings outside of our mentor meetings. These team meetings allowed everyone to come together and share their progress, as well as any problems they might have had or solutions to problems other members were having. Regarding how we managed the project on a technical level, we used Git along with GitHub for version control and hosting the source code where everyone on the team could access it and contribute. This made it very easy for each member of the team to access the project and this overall process is what worked for us.

3.0 Requirements

We obtained all of our system requirements through constant contact with our client. Our initial requirements came from the project introduction back at the beginning of last semester and we were able to continue gathering requirements throughout the year by maintaining constant contact with our client via in-person meetings and email correspondence. As a result of this process, the main functional and non-functional requirements we obtained were that:

- Users need to be able to create accounts they can log in to and save their progress within the modules
- Users need to be able to load their progress when returning to the system
- The system will have two learning modules that cover Karnaugh maps and numeric conversions, respectively

- The system will have practice modules that allow students to practice questions of their choosing
- System will be hosted online for students to access
- System must be accessible throughout the school year
- System must be able to support at least 150 users at the same time without affecting performance

These requirements were the core requirements of the project as designated by our client, and so these were the ones we focused on the most. Our project allows users to create accounts, log into them to work on the learning modules of the system, save their progress on each question, and load their progress back when they return to the system. Additionally, the project is hosted online allowing students to access it any time they need to while also allowing the designated number of students onto it, fulfilling our client's requirements.

4.0 Architecture and Implementation

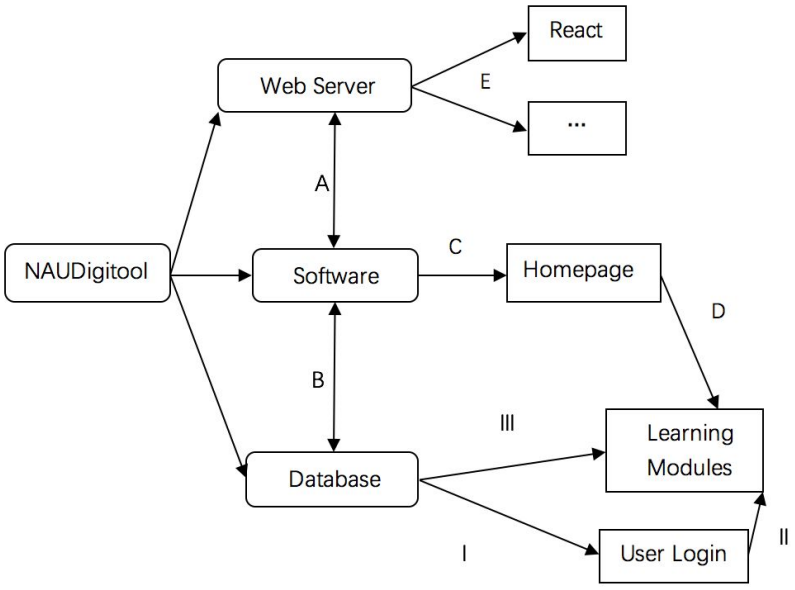

Figure 0: Control Flow of Architecture

4.1 The key responsibilities and features of each component

Digitool has 3 key responsibilities: the database itself, the software, and the browser. The database is used to collect user information. This database needs a higher level of security to ensure that the user's information is completely protected. It also needs stability and scalability because this database has to be continuously used and the data within the database is constantly being added or changed. In addition, this database is also portable, in case of redeployment to another server / host in the future.

The software is mainly written in JavaScript code and it is presented within a majority of the web pages throughout our web application. The software is targeted towards students and professors and the software within our web application consists of a learning module that can be divided into four parts: truth tables, K-Map, equations, and numeric conversions.

Web site services can facilitate users to log in, register, and update a series of operations and web site services also facilitate our tests during design.

4.2 Communication mechanisms & control flows of the architecture

Our main communication mechanisms and flows are as follows (as shown at Figure 0):

A) It helps users to login and register and the software can be tested .
B) The account information of both students and teachers is necessary, as well as the internal information of the system, so we need database support.
C) When you start using Digitool, you will be taken to a homepage and then you can choose different modules.
D) It has multiple learning modules and the modules are arranged in order from simple to difficult.
E) It will do the React rendering and other internal jobs to help the system run well.

I) When the user does log in, it needs the user information from the database.

II) If the user wants to select a module to study, he / she must log in at first.

III) Question and answer information will be communicated to the database.

4.3 Influences from architectural styles embodied by this architecture

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 and 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 often don't need to worry too much about other parts. Since it's a layered design, we can focus more on the standardization of the 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 through one of the architectures.

5.0 Testing

This section will provide the procedures for how we tested our product. These tests primarily revolved around Module 1, since Module 2 used JavaScript methods from already built in libraries. Once the procedures have been explained, we will then talk about the results from our testing and talk about any changes we made to the source code / web application thereafter in response to our testing results.

When it came to unit testing for Module 1, we needed to test every function to make sure that they functioned 100%. We also needed to make sure that each function was accurate and matched what it was supposed to do. Since some functions involved using other functions in that same function, we had to create individual tests for every function and that was done through a Visual Studio Code extension called Jest. Jest is used specifically for JavaScript unit tests and once this tool was downloaded, it was now time to run unit tests. We did encounter problems on functions that returned void as an answer so we were unable to test a couple of functions, but besides that, every other function was able

to be tested. After gathering the results shown in Figure 1, it was now time to analyze our findings.

File		Statements		Branches		Functions		Lines	
addToEquationArray.js		100%	7/7	100%	0/0	100%	1/1	100%	7/7
arraysEqual.js		100%	2/2	100%	0/0	100%	1/1	100%	2/2
checkForDuplicates.js		100%	21/21	83.33%	10/12	100%	1/1	100%	19/19
checkIfArraysMatch.js		76.47%	13/17	50%	6/12	100%	1/1	73.33%	11/15
countElementsInEquationArray.js		100%	8/8	100%	0/0	100%	1/1	100%	8/8
createArray.js		70.37%	19/27	25%	5/20	100%	1/1	69.23%	18/26
createArraySpace.js		100%	5/5	100%	0/0	100%	1/1	100%	5/5
getLengthOfArray.js		100%	6/6	100%	0/0	100%	1/1	100%	5/5
getTimeTaken.js		100%	13/13	100%	2/2	100%	1/1	100%	13/13
removeDuplicates.js		100%	9/9	83.33%	5/6	100%	1/1	100%	7/7
swapArrayIndices.js		100%	5/5	100%	0/0	100%	1/1	100%	5/5

Figure 1: Unit Test Results

A majority of our functions passed, while some were yellow and the reason behind this was because of the fact that multiple if statements were within one function. This meant that every test would eventually lead to one if statement, so no matter how many tests we ran, the result would always remain yellow. Based on these results, we came to a conclusion that the functionality of any of the functions didn't need to be changed. Once that was done, it was now time to run usability tests.

Not only were we able to run usability tests on Module 1, but our client was also able to share problems and concerns that he had while using the web application. The procedure was pretty simple: we had to go through both practice mode and regular mode for Module 1 & 2 to make sure that every button worked and functioned properly. We were able to click every button and everything was able to navigate to where it needed to go, but on our client's computer, he noticed a user interface problem that appeared in the practice mode for Module 1. As you can see in Figure 2, our client was unable to see the radio buttons which the user can click in order to generate a random practice question.

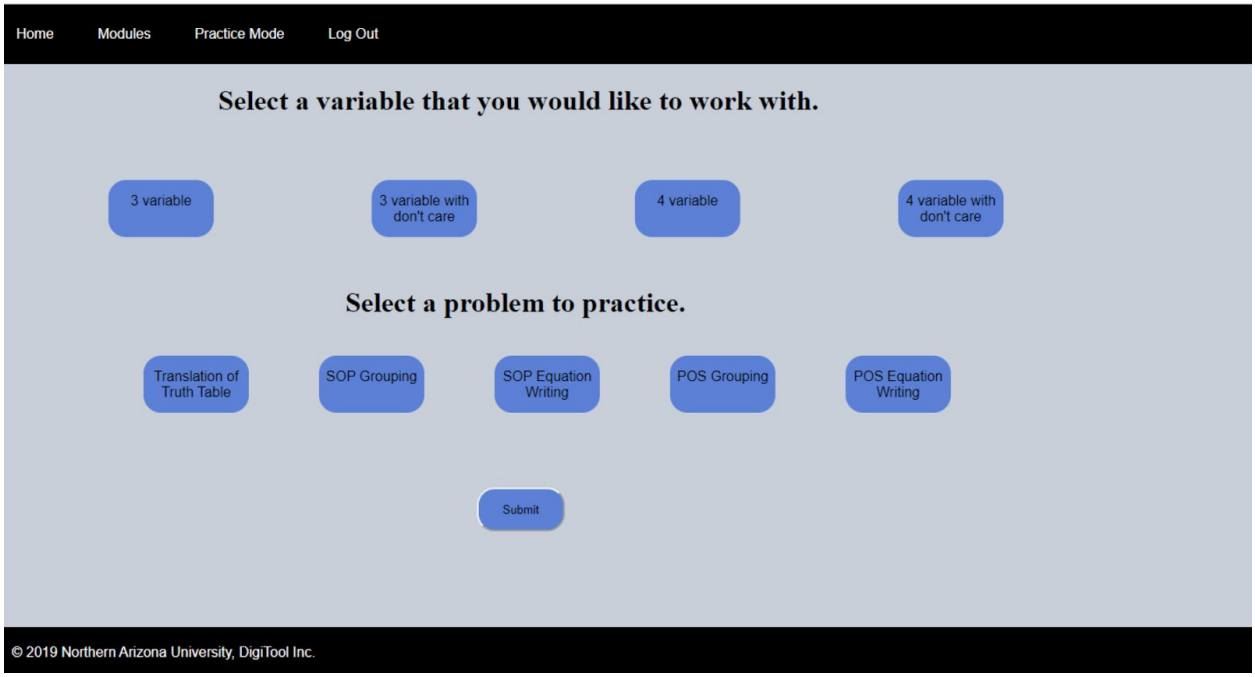

Figure 2: Module 1 - Practice Mode UI

To solve this, we had to reformat the UI to where the radio button could be seen on both his computer and any other computer. As shown in Figure 3, the buttons were able to be shifted and visible for any computer.

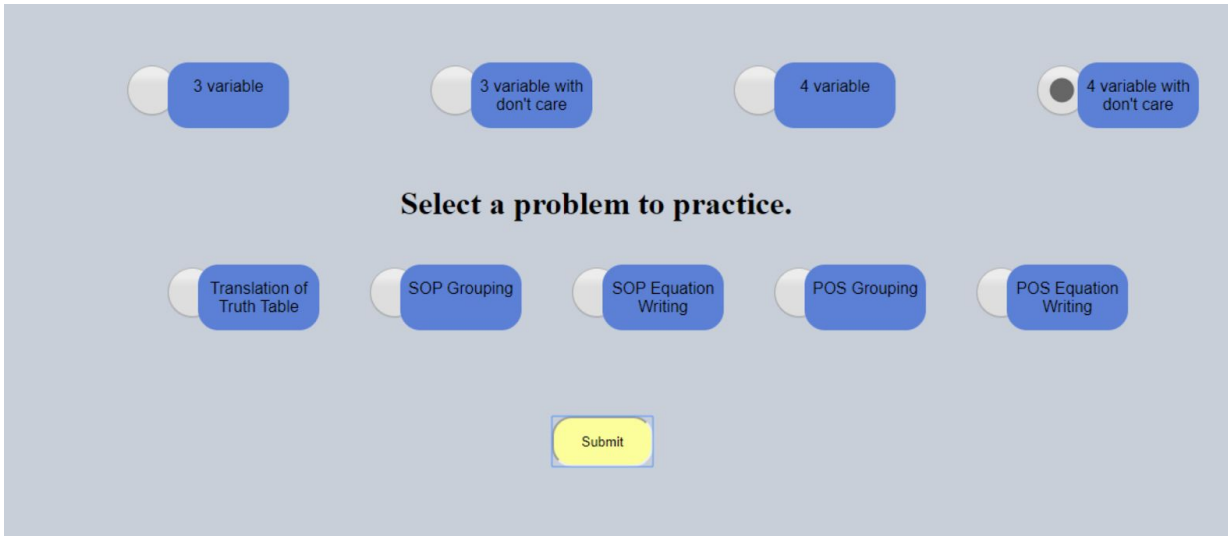

Figure 3: Module 1 - Reformatted Practice Mode UI

Figure 3 is just one example of how it would look on an older computer, but on a newer computer, it should look something like this as shown in Figure 4.

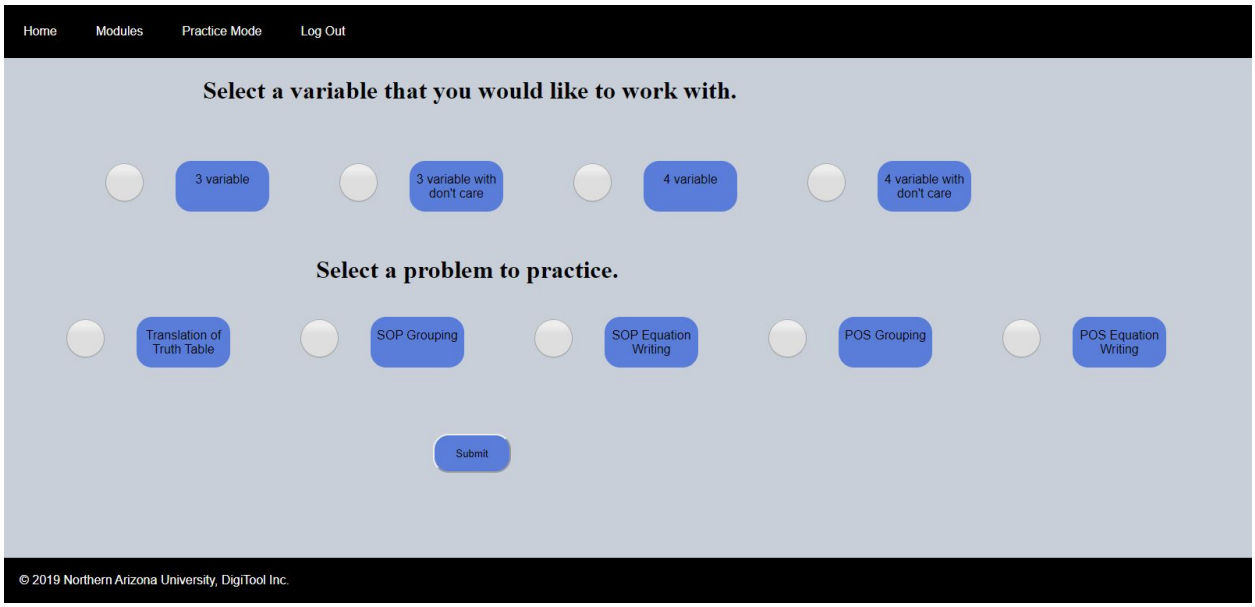

Figure 4: Module 1 - Other Example - Reformatted Practice Mode UI

We did come to the conclusion that if you use an older computer, the UI of our website could look slightly different, but everything should be present. We were also able to detect a bug to where the website would crash if the user rapidly clicked the “Submit” button before everything within the page was loaded (Ex. timer, K-Map, truth table, other buttons). This was fixed by disabling the “Submit” button until the whole page was loaded.

6.0 Project Timeline

	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 and 4 variable (possibly 5)	11 Weeks	
6	Translating truth table	6 Weeks	Tray
7	Grouping	6 Weeks	Tray
8	Equations for grouping	6 Weeks	Tray
9	Testing	1 1/2 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	Design	3 Weeks	Everyone
21	Additional Testing	3 Weeks	Everyone
22	Deployment	1 Week	Tray & Spencer
23	Final Documentation	1 Week	Everyone

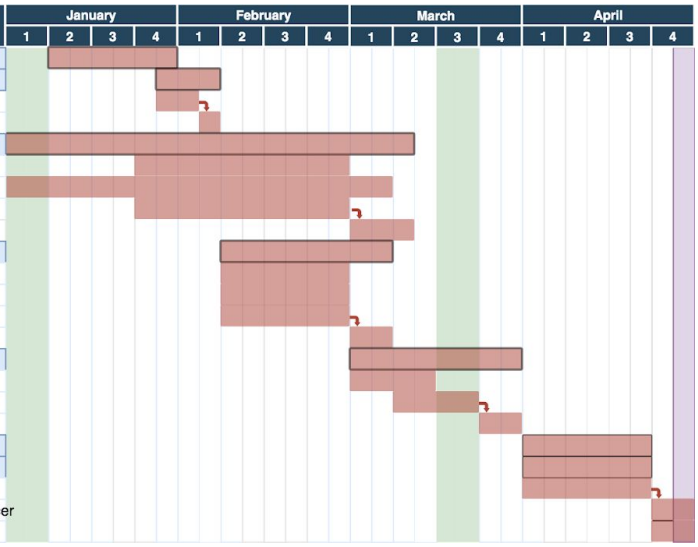

Figure 5: Spring Schedule

The main phases of our project, as shown in Figure 5, consisted of creating Modules 1 and 2 as well as the practice mode for each of those Modules. Database communication was also set up to our web application to save / load a user's work as they logged in and out of our web application. Once we were given a task or assignment, our team would conduct a meeting and decide which person wanted to do which part. This was normally done after every mentor meeting and once that was done, everyone would do their individual parts and the day before it was due, our team lead would combine the parts and re-read any documents to fix any mistakes that needed to be fixed before submitting it. This process was done for every assignment. In terms of programming and coding, we met on Thursdays and Sundays to program together and solve issues that each person may have encountered during their individual programming parts.

As we approach finals week, Tray and Spencer are currently communicating with NAU ITS to get our web application and database up and running on the NAU servers. We are experiencing deployment delays and as well as communication delays between us and NAU ITS. If we were all in Flagstaff as a team, it would be easy to communicate and

show NAU ITS what we have, but due to unfortunate events (COVID - 19), we are faced with a very difficult challenge. We have thought of an alternative, which is to temporarily host our web site and database on a free hosting web site. Besides that, we are finishing up all documents that need to get finished and turned in before 3pm on Thursday, May 7th.

7.0 Future Work

7.1 Additional Modules

If a 2.0 version of our web application were to be created, we really think that adding additional modules would be an excellent feature. Since time was limited and we had to plan / build our web application from scratch, we were only able to create 2 modules. If another team worked on this project in the years to come, they would not only be provided with an already set up database and web application, but they would also have a template to how everything should look. This would allow them to add on to the project right away, rather than worry about how to integrate everything and make it functional because all of that stuff would be done already. Adding more modules can also allow students to practice / work on even more topics that are discussed in class, which could increase their proficiency rate.

7.2 Secondary Application

A secondary application can also be created that automatically analyzes statistics from our database. This would make it easier for our client to gather data and he wouldn't have to touch and modify the database constantly. This secondary application could contain things such as a spreadsheet of all the results, graphs to show student proficiency rates, etc. This application would simply be created to make data easier to get a hold of and analyze.

8.0 Conclusion

To conclude, our main goal was to design an easy to use and reliable web application for introductory level electrical engineering students. Our client felt that it was important to create this self-study toolkit to give students an alternative towards practicing topics taught in the course such as Karnaugh Maps and Numeric Conversions. Our envisioned solution is to give students access to this web application and practice modules whenever they need to. By doing this, students should be able to have a better understanding about the topics discussed in class and the statistics tracked from our web application will give our client an idea of what topics and concepts should be lectured and practiced more. This solution will not only save Northern Arizona University money, but it will also give students a virtual learning tool that previous students never had. We wanted to make sure that our database could communicate with our website efficiently and we wanted to make sure that all of our modules were correct. After testing our product and meeting the requirements necessary, our client was very satisfied with our end product. Although deployment issues arose due to unfortunate circumstances , alternatives were made to get our website up and running.

The team as a whole is proud of our accomplishments and are amazed by everything that we learned in the Capstone class. Not only did we learn to work and communicate in a professional environment, but we also learned how to do our individual parts on a large team. Although it was a little rough at first when it came to getting roles fulfilled and carried out, the end result left us all amazed at the work we were able to accomplish. We could've had better documentation and bug tracking when it came to our project's process, but all in all, we are proud of the work we were able to achieve and we couldn't have done it without the guidance from our mentor Fabio Santos and the patience from our client Xi Zhou.

9.0 Appendix A: Development Environment and Toolchain

9.1 Hardware

Potentially as a side effect of creating our application with minimal requirements on the user's end, the process of development is not resource intensive either. The difference between a strong PC and an old laptop is saving you about 10 minutes a day at most and (if it's *really* old) some frustration. For reference, I could reasonably work on my 10 year old laptop with the database and localhost running. The laptop is 32Bit Windows 7 with 2GB of ram and a Celeron(R) T3000 processor. Most modern phones can easily beat those specs.

Since the project has well supported web development and scripting, it can be developed on nearly any platform with a text editor. This is no need to worry about whether you are using a Mac, Linux, or Windows system. As stated before, you could probably even work from your phone, although it's not recommended for productivity.

9.2 Toolchain

The team mainly used Visual Studio Code and Brackets as IDE's. These two are convenient for live preview (Brackets natively, VS code with a plugin). VS Code also has a built in command line which is useful for working with 1 of our tools. We also used localhost via XAMPP to test PHP files and database interactions.

Most of the tools integrated into the project were for development conveniences. To restrict JavaScripts dynamic typing, we started the project with TypeScript. We added lots of versatility to our content via React components. For deployment and HTML page building, we added Webpack. All of these packages can be installed with NPM (node package manager).

9.2.1 TypeScript

Typescript is a superset of the Javascript language, meaning it adds onto it. All of Javascript's functionality is usable in Typescript along with some nice additions and enforced types. We originally started with Typescript in our React components (can be seen as the .tsx files), but as the project went on, development was faster in regular Javascript. This isn't a problem since all Typescript is eventually compiled into Javascript and there is no incompatibility in mixing them. It is however a shame that the development environment is not entirely in Typescript, and converting Javascript files into Typescript would be quite beneficial. In our project setup, the compilation from Typescript to Javascript is managed by Webpack.

9.2.2 React

React is a way to extract functionality into components. These components can be reused and interacted within web pages, making it really easy to add reused content to a new page. For example, our Navbar and footer; they are used on every page we have, but we only wrote it once and then added the div tag that React is told to render the component on to each page. You can tell React which component to render on which tags in the index.tsx file. React is in the same situation Typescript is for our project, it was used in the beginning and kind of fell out of the loop. It is highly recommended for anyone revisiting the project to convert the plain Javascript files for the Kmaps and Conversions to React components in Typescript and pass in props for what needs to be generated. Doing that would remove the need for all of the "module1QuestionX.js" files that contain the info for which problem is being created on the page.

9.2.3 Webpack

Webpack is a “bundler.” It takes all of our scripts and adds them to a singular main.js file so that when we create a new page, the only import line should be main.js. It is also configured to compile our Typescript and .jsx files to normal Javascript. To use webpack, make sure you have the node module installed. This can be done by typing “npm i webpack” into the VScode command line. Make sure you are in the project directory when doing this. To run webpack, type “np**x** webpack” into the command line (notice that it is X, not M).

9.3 Setup

To get ready working on the project, all you need to do is download the repo from GitHub. Then install Typescript, React, and Webpack through npm to the project directory. All project specific configuration can be found in package.json, tsconfig.json, and webpack.config. These settings should not be modified on a per person basis, only when changing aspects of the project in its entirety.

9.4 Production Cycle

The cycle of producing and updating our web application relies on a good understanding of Javascript and a basic understanding of file transfering. The first thing you would have to do is take the code files given to our client and use a code editing tool to open / edit the code. This can be done using Brackets.io or Visual Studio Code, but we personally recommend Brackets.io because it’s very easy to use and you’re constantly able to see web application changes as you make them. When you preview the current web page you’re working on (using live preview), you’re also able to see errors or warnings that pop up by inspecting a page and then navigating to the console section. This can be useful for identifying errors to ensure the web page completely compiles before finalizing a page. Changing logic within the web application relies on adding additional functions and changing the layout of our web application relies on modifying the HTML / CSS

pages within our web application. Database changes would have to be made by using SQL commands that work asynchronously with PHP. Once changes are finalized, a person would then have to download / use an File Transfer Protocol client in order to update the current version of our web application. Cross - Platform clients consist of FileZilla, Cyberduck, or CrossFTP. Once logged in using the credentials given to our client, the changed files would have to be overwritten into the current folder that the previous version was written to (htdocs folder). The website link, product manual, and additional tutorial videos will also be given to our client that can give additional clarification and guidance when editing, compiling, and deploying future versions of our web application.