# WEB701 Project Part Two

Carlos Amos

## Contents

# Introduction

For this report, I will be discussing the purpose of web frameworks, select and review two potential web frameworks and select a framework out of the two reviewed frameworks.

# Web Frameworks

A web framework is a tool that is used by developers to help with the creation web applications. The use of a framework can help reduce the development time on applications as a framework will supply a library of code. The common features of a web framework include templates like navigation bars or buttons, URL routing, Input form managing and validation, database connections, web security and session repository and retrieval. (GoodFirms.)

In the development of a web application the templates can be used to design the UI of the application. A common template that are in frameworks is a navigation bar that works on a desktop but also can be resized to fit onto mobile. If the developer were not using a framework, they would have to write the code for each part of the navigation bar individually but if a framework were used, the developer could just generate the desired template into the web application by inserting the code into the app.

Many frameworks can interact with databases using a package and also have the ability to handle URL requests.

# Proposed Frameworks for Project

## First Framework

Vue.js

The first proposed web framework is Vue.js. Vue.js is a progressive open-source framework that can be used to develop web applications for both desktop and mobile apps. This framework is used for the front-end development for applications and features reactive components and is easy to use as the templates can be inserted into the app.

Using Vue, the developer can start simple and add more tools and features as the app develops. It can be used to build components which can be connected to a html template and the data within the components is reactive so if a value within the data is changed, the component will automatically update. The framework also supports a plug-in system where the developer can drop in anything they want.

## Second Framework

Angular.io

<table>
<tr><th>Vue.js</th><th>Angular.io</th></tr>
<tr><td colspan="2">Installation</td></tr>
<tr><td>Installing Vue.js for the first time was a simple installation process and was setup and ready to begin designing within four minutes. As Vue has a reactive system, I was able to make changes in real-time without having to restart the application every change. Vue also has a network automatically setup so I can view the changes on a second device. The UI of the website was made using Vue Materials and BootstrapVue.</td><td></td></tr>
<tr><td colspan="2">Vue better. Angular bigger file size. Automatic network.</td></tr>
<tr><td colspan="2">1. Register/ Login/ Edit Account</td></tr>
<tr><td>For the register and login element of the website, I implemented firebase authentication and store the account information on a firebase database. The register component works by when the register button is pressed and the form is filled out, the email address is first checked in firebase authentication and if it does not exist, the account will be added to both the firebase authentication and database. The login system is very similar to the register system which uses the firebase authentication. The login system will first check if the user in the firebase auth exists and if the user does exist, the user will be taken to the profile page.<br>Currently there is no way to edit the accounts information without an admin changing the details within the database itself.</td><td>For the register and login element of the website, I used Auth0 of which was able to successfully create an account and log in. To test the system, I created a simple button on the home page and pressed the button to login. I was logged in successfully as the button is supposed to say “log out’ when logged in as a user. I was also able to create and use a email address called “admin@admin.com.” Figures 1.0.0 to 1.0.2</td></tr>
<tr><td colspan="2">For the register and login element of both frameworks, I first implemented Auth0 of which I was able to implement with any problems with Angular, but I did run into many errors while trying to implement it into Vue. At first the button for the login would not appear and when that was fixed, the app would not get redirected to auth0. I did manage to get it working but decided firebase to be a better option as firebase could be used for both authentication and storage. I was able to implement firebase into my Vue app successfully. Creating this component in Vue, it wasn’t too complex as components can be called from other files, but also from within the same file. A Vue file can contain the template of the app, the scripts and the style.</td></tr>
<tr><td colspan="2">2. Register Products</td></tr>
<tr><td>For the register products or in my case listings, there is a listing component that has all the code for presenting the listings in the form of cards. The listing component first calls the API for all the listings, is put into a list and then each item in the list is displayed as a card using the ID. The card will display the name of the listing, the donator, the description of the listing</td><td>Have a register page but isn’t able to call listings from the database</td></tr>
</table>

| | |
|---|---|
| and the user will be able to view an individual listing using the button. It is not yet implemented but when the user presses the view listing button, it will call the listing API with the id of the listing to retrieve all the data on the specific listing and display it on its own page. | |
| For the register product component of my app, I did not find it too difficult creating it in Vue because I used code from a recent React app to help with the display component of my app which displays all the listings with their own individual cards. I also found examples from other projects online to be very helpful. For creating it in angular, creating the visual element of the register page was not too difficult but trying to call the API from the angular app didn't work for me. | |
| **3. Interactive Elements** | |
| The listings of products are displayed in cards of which people will be able to view the details of the listing by clicking on an individual card. Doing so will take the user to a new page showing the details of the selected listing. | No interactive elements |
| For the interactive elements of my Vue app, I used cards in the listing section of which users can view each individual listing. The complexity of the elements in my Vue app a relatively easy to understand as pressing on the view listing button on a listing will take the user to the individual listing. | |
| **4. Token System** | |
| Created automatic token generator. Plan is to link token with listing when purchasing item.<br>For my token system, I have implemented an automatic token generator which will generate a token when the user wants to purchase the item. The generator successfully generates a random sequence of letters and numbers and displays it to the user. It is not yet implemented but the token will be tied to the product by pairing it with the product in database and will be removed from the store | No token system |
| For the token system, creating the system was not too difficult because I researched how to make a password generator and then repurposed it to generate a random sequence of letters and numbers which could be used as a token. I found adding the token system to the app was quite simple, but the complex part was trying to stop the generator from generating a code every time the page was refreshed. I wanted it to generate the code and then send the code to the database and link it to the listing. | |
| **5. Server-side Database** | |
| | |
| My Vue.js app can successfully store and retrieve data from my server-side database. I am current using firebase database as my database and can communicate with my express backend. Using the listings as an example, I | Server-side database is setup but struggled to call the API on my Angular website. Was unsuccessful at calling my backend. |

| | |
|---|---|
| am able to create a new listing and then see that listing appear in the store along with other listings. | |

For adding the server-side database to my angular app, I found it confusing and was not able to get it working. I tried following many tutorials but found them too confusing. Whereas for my Vue app, I was able to get it working without and trouble. I didn't find it too complex because I found tutorials that were very helpful, and I was able to reuse the code for other parts of my app including the user system.

**Pros and Cons**

**Vue**

**Pros**

Only one file component contains the template, the scripts, and the styles.
Can create components and call the components.
Easy to learn.
Size of the framework and app is very small.
Reactive App. Can make real-time changes.
Community support.
Setups the app up for localhost and in a network so app can be viewed on a second device.

**Cons**

Not a lot of resources that can be used. Creating the UI of the app can be difficult without a external library like Bootstrap. Plugins.
Not considered the best for large projects.

**Angular**

**Pros**

Reactive App. Can make real-time changes.
Can inject components.
The support from the community.

**Cons**

Size of the app is much larger than Vue.
Component is made up of 4 files.
Can be very difficult to learn.
Performance.

## Selected Framework

The framework that I am going to be using for my final app is Vue.js.

One of the main reasons I did not select angular as my framework was the steep learning curve of the framework. As I created the two prototypes, I went back and forth between each project making progress on each but every time I would work on the angular project, it felt like I was learning it from scratch again. Whereas with the Vue project, I found it easy to understand. Another reason is the way the components are setup, Angular has the components setup with four different files, the CSS, HTML, Spec.ts and the Typescript folder whereas Vue has all the components within a single file. Having all the components in one file helps with creating the app by designing the template of the app and applying the style to it in the same file instead of having to open another file to make style changes. I also prefer Vue over Angular because I can import the whole Vue materials library with two lines of code but for angular, I must import each individual component from the Angular library.

My final reason for choosing Vue is because I enjoyed creating the Vue app as creating the Angular app was very stressful.

# Conclusion

Overall, my first selection of framework I was happy with because I enjoyed learning the app mainly because it was not too difficult. Whereas for my second selection, I did not enjoy creating the Angular app as it was a very stressful experience and wanted to choose a different framework but as I didn’t have enough time, I decided to stick with Angular.