Automated Fitness Instructor Certification Website

DESIGN DOCUMENT

Team Number: sddec19-04
Client: AFIC
Advisor: Dr. Mohammed Selim

Team:
David Bane / Group Facilitator
Ryan Menster / DB Admin
Christine Hicaro / Group Admin
Maxwell Talley / Scribe

Team Email: sddec19-04@iastate.edu
Team Website: http://sddec19-04.sd.ece.iastate.edu/

Table of Contents

1 Introduction

1.1 Acknowledgement

Throughout this project, we will work with a number of different people. Dr. Mohamed Selim is our faculty advisor. His guidance will be very useful to our team throughout the process. We will also be in contact with is Nathan Schaffer, who started the project last summer and helped us get the project off the ground.

1.2 Problem and Project Statement

1.2.1 General Problem Statement

Our client, AFIC has asked us to help them in the creation of a certification website for their instructors. Right now, AFIC keeps track of all instructor information by hand in spreadsheets. Handling the information this way takes time away from people who could otherwise be doing more meaningful work. As the company grows and more instructors are hired on, the amount of work needed to manually certify instructors will grow exponentially, which can lead to a major delay in the certification process as a whole. Bringing this process online would both increase the accuracy of the information as well as greatly decreasing the amount of bookkeeping work they have to do. This website will allow AFIC a quick, streamlined process for instructor certification.

1.2.2 General Solution Approach

To bring AFIC's certification process online, we will need to design and implement a new website for them. This website will include a database to store all of their instructor information, upcoming workshops, and anything else they need. The website will also include a fully functional user interface so instructors and managers can access all the data they need as well as automating as much of the certification process as possible. The frontend of the website will be created using React, a well supported JavaScript library that specializes in creating beautiful and effective user interfaces. The backend of the website will be created using Express, which utilizes Node.js to create a functioning component of the website that seamlessly communicates with both the frontend and a database.

By the end of the project, our team hopes to complete an Instructor Certification Website that meets AFIC's expectations and allows them to instantly start using the created website to certify and add new instructors without needing to dig into multiple spreadsheets for the information.

1.3 Operational Environment

The operating environment for our end product will be calm and hazards will be a minor concern. With our end product being a website, this will be ran on a computer. The website is for a company so it is expected that this computer will be kept within an enclosed office. Apart from the usual minor hazards of having an electrical appliance plugged in, there are no more to be identified. Another aspect of our project will be servers maintained by a third party. It will be the responsibility of this offsite third party to maintain and protect the servers. Therefore, any hazards pertaining to the servers are not within our team's development scope.

1.4 Intended Users and Uses

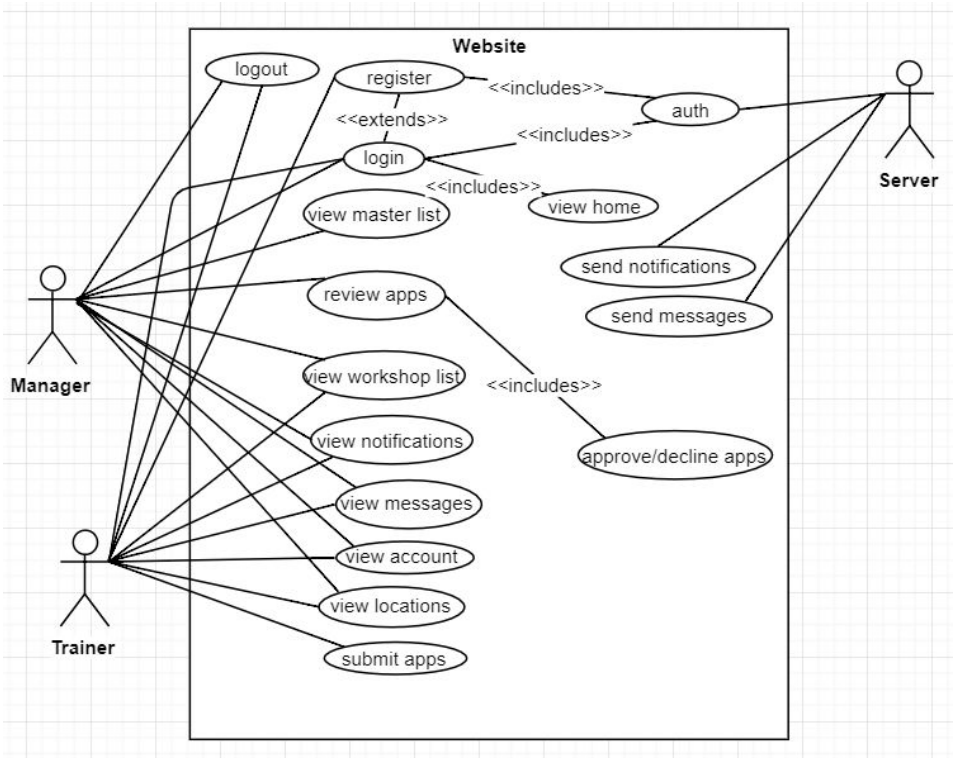
The intended use of this product is for instructors to be able to submit and apply for certifications. This will allow for much of the work to be automated instead of the current system being by hand. There will be three types of users for the system:

1. Manager
    a. The higher level of the two primary users of the website. These individuals can do everything that an Instructor/Trainer can do, other than apply for certification. Along with those abilities, Managers can view the master list of instructors in the company, as well as review certification applications, where they can then choose to approve or decline the application. Managers do not have to register their account. It will be done so automatically.
2. Instructor/Trainer
    a. The most basic user level. These individuals can login to the website, where they can see any workshops, messages, or notifications that are currently available to view, as well as modify their account and view locations. Finally, Instructors/Trainers can submit an application for certification.
3. Server
    a. The final user of the website. The system is responsible for keeping the website running. This includes performing login/register verification and sending out notifications and messages to the users.

Since there are two types of user authorization in this program (Manager and Instructor/Trainer), then there must also be two sets of permissions towards the data available to use.

Many of the use cases are shared between the types of users. The shared use cases are viewing workshops, locations, notifications and routine login behaviors. The use cases unique to managers are viewing a master instructor list, reviewing applications, and deciding to approve or

decline those applications. For instructors, their unique use case are the registration and submitting applications use cases. The server will be in charge of authentication, sending notifications, and sending messages.



1.5 Assumptions and Limitations

- Assumptions
  - An existing style has been set and can be used if we wish
    - A previous intern had created an initial frontend of the website
    - We can base our design off of his already existing design
  - We are free to create the backend with whatever technology/programming language that we wish
    - Currently, we plan to use Express, a Node.js framework
  - We are free to create the frontend with whatever technology/programming language that we wish
    - We will be using React, a JavaScript library that specializes in creating user interfaces
    - We will be using material-ui, a React framework, to assist in development
  - We are free to use any hosting platform that we desire, provided that its costs are manageable (i.e. around the same cost as Amazon Web Services)
    - Currently, we plan to use Microsoft Azure

    - A database will be provided via the hosting platform
    - Up-to-date instructor certification information will be provided at the start of implementation
    - The website will not need to be too scalable
        - User count will remain relatively consistent throughout the website's lifecycle
    - The product will be used only on computers
        - Mobile devices will not need to be accounted for
    - Only Managers can create accounts for Instructors
    - There are only two types of users: Managers/Owners and Instructors

- Limitations
    - Current limitation of expected requirements
        - Will require future meetings with client to verify that our current direction is correct
    - Budget will be practically nonexistent for our team
    - Test cases can be difficult to create, due to number of components involved
    - Extensive database information is currently unknown
    - No current structure for the backend is known
    - Original client is no longer with the project
        - Will need to decide which components that the client initially wanted will be kept and which will be changed
        - Will need to determine if a new "faux client" will be implemented
    - Group members' schedules do not line up well with each other
    - Website must be intuitive and easy to navigate through
    - Group members are not familiar with the programming languages (React and Express) that will be used for the project.

1.6 Expected End Product and Deliverables

This project contains three primary components that will form the overall end product. The three components will be a frontend, a backend and a database. This product itself will be a fully functional automated Instructor Certification Website. This website will allow for the certification work currently done by hand to become a streamlined process. The three main components and final product will act as our deliverables.

The first of these will be the frontend, also known as the user interface. This will be delivered in

September and allow for some feedback regarding usability to be delivered. The second component to be delivered will be a fully functioning database that will be populated with up-to-date information we are provided with. This component will be stored in our Azure cloud service and will be delivered in October. The last of the components to be delivered will be our backend Express service. This service will allow for the frontend and database to communicate and will be delivered in November. The last deliverable is the final fully functioning product. All three of our components will work together and reliably. This will be AFIC's new certification website and will be delivered in December. No other additional materials will be delivered alongside the finished website unless requested.

- Functioning Frontend/User Interface - Delivery Date: September 13th, 2019
    - The user interface will be developed with React and consist of any visual objects that are required in the website. The visuals will range from images, buttons, text fields, tabs, etc. The functionality of all of these visuals may not be fully completed but much of the basic web navigation will be. The functionality that would not be completed would be anything that sends to or retrieves data from the database. This functionality will be visible during the delivery of our backend. The frontend will be designed to be intuitive and easy to navigate. Even with this design it will need to be verified by the team at AFIC.
- Functioning Database - Delivery Date: October 18th, 2019
    - The database delivered will contain all of the up-to-date information pertaining to the instructor certification process. The information will be organized in a manner that minimizes repetition and produces efficient results. The information contained within this database will be able to be removed, added and modified. This database we plan to host via Azure and will interact with the Express backend. This will allow for data to be retrieved for presentation or sent for modification using the user interface.
- Functioning Backend - Delivery Date: November 22nd, 2019
    - The backend will be the last component with the purpose of connecting the two prior components. As previously mentioned the service used for the backend will be Express. Express is a backend that interfaces well with our React frontend and is written in Javascript. Express is what any send or receive calls will pass information through. The backend is also planned to be hosted using Azure.
- Functioning Instructor Certification Website - Delivery Date: December 13th, 2019
    - The fully implemented Instructor Certification Website will fulfill all of the requirements specified by AFIC. The website will allow a Manager to create an account for an Instructor with a default level of "Trainee". From here an Instructor will be able to apply for recertification or an increase in certification. The respective Manager of the Instructor will then be able to approve or

disapprove. Any computer will be able to access this website but only those with the proper credentials will have any access. The user interface will be intuitive and easy to use. The source code for this website will be available via ISU's Gitlab and on the Azure Cloud Service. No additional documentation will be provided unless requested.

2 Specifications and Analysis

2.1 Proposed Design

To solve the problem we have been given, we will use three frameworks to help us manage the frontend, backend, and database. These frameworks all run in Javascript and will allow us to quickly build a secure and robust platform for our client.

For our frontend framework, we will be using React. React is component-based and will allow us to pass data to our app without keeping any of that data in the DOM. React is considered the industry standard for frontend Javascript frameworks and is well supported with a very active community [10]. Before we chose React, we debated using one of the other major frontend frameworks, Vue. Vue is very similar to React, but we chose to go with React because it is considered the industry standard [10], [13]. Functionally, either of these frameworks would have worked. We decided that the experience we would gain using React would aid us in the future, and since they are functionally equivalent, we chose to go with React.

The next framework we chose to use is called Express, a backend framework written in Javascript [5]. The other frameworks we debated using were: Laravel, a PHP framework; Django, a Python framework; and Spring, a Java framework [3], [9], [12]. We chose to go with Express for a few reasons, the first being that all of us had experience using Javascript and it would be easy to use the language throughout our project rather than have multiple. Express is also very easy to set up and has a robust set of tools for managing the backend of a web application. Express was also chosen partially so that we could use a really nice database ORM, Sequelize.

Sequelize is a Javascript framework that is used to manage a database. It is a very robust tool that allows for easy setup and teardown of a database. It is easy to set up and manage relations as

well, making it a simple choice for us [11]. One of our team members has used this framework with Express in the past and though we didn't do any research for this project regarding database ORMs, Sequelize was deemed the best option for a past project with very similar needs. For all these reasons, we decided this framework would suit our project as well.

2.2 Design Analysis

Our design analysis was of the work done with the project before it landed in the hands of our team. While looking over this prior work we noticed a few flaws. The first of these flaws was that the **framework** used was a **deprecated** version of **Angular** [2]. To continue using this framework in our development of the project would result in bugs and difficulty for future maintenance. We also felt the decision to use **AWS** as a **cloud service** was lacking in comparison to other services like **Azure**.

Apart from the flaws we did like the choice to use **Express** as the **backend service**. We were warned about failed attempts using Express as the backend but still felt it was the right choice. We also knew that the current **design** had been appreciated by our client. Although the framework and libraries would be different, the current design would still be very useful in creating a new but similar design.

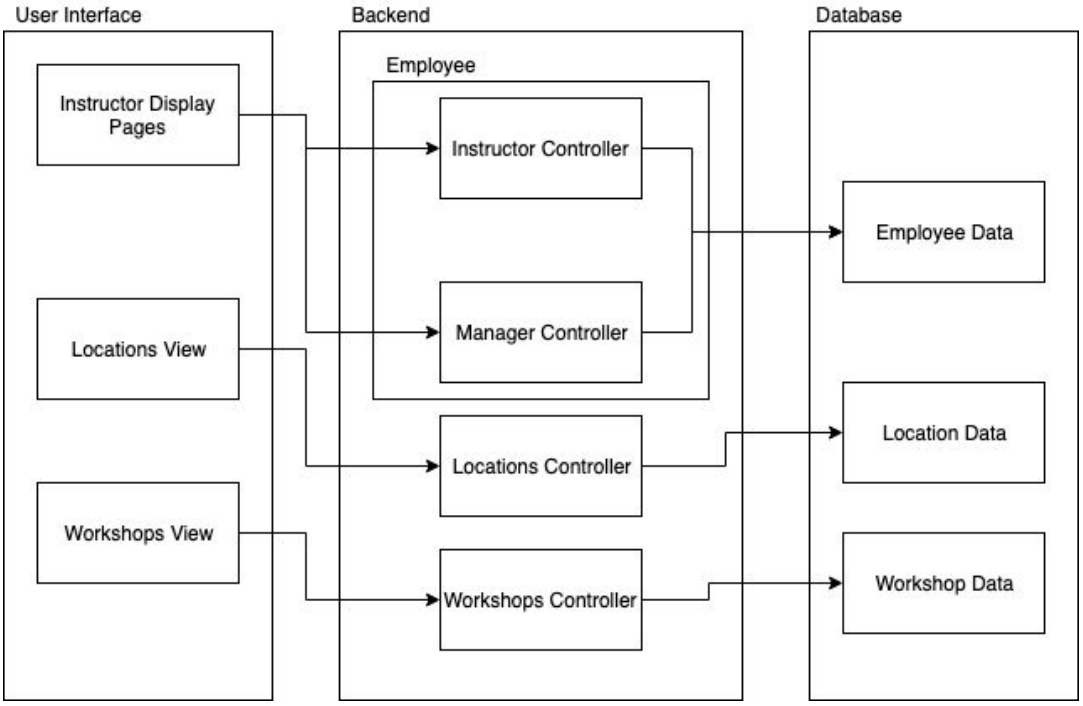
As we planned to keep the positive aspects of the current project development, we had to look at what would be changed. To counter the deprecated framework, a decision was made to use **React**. Apart from being reliable react would also work with one of the positive traits. The **React framework** and **Express service** both work very well together. As referred to earlier, our team had opposing thoughts towards the use of **AWS**. Instead, using **Azure** as the **cloud service** for this project was presented. It was deemed that Azure could be cheaper, more intuitive and more reliable [8]. With these changes we were confident that the project is headed in the right direction.

All of these components will allow us to create an application AFIC can use to automate their instructor certification process. All of the data will be stored in a backend database that is routed to the frontend via the **Express service**. The frontend, being the user interface is created through the **React** framework. All of this technology is stored within our **Azure** cloud service acting as our server. In order to create the automation their will be a backend algorithm that will parse the required data. After determining whether or not a certification is approved the appropriate action will be routed. This way all of the requested requirements will be completed for our client.

3 Testing and Implementation

3.1 Interface Specifications

Our project does not interface with hardware. In terms of software interfacing, our database interfaces with our backend, which interfaces with our server. Moreover, the backend defines how to check the database and interact with it; it consists of controllers that process user input accordingly. The frontend retrieves data from our server, and the user interface displays the data.



3.2 Hardware and Software

Our project can be broken into 3 main components: the user interface, backend, and database. The user interface is composed of the instructor display pages, which encompasses all pages the instructors will view their information on. This includes pages such as the certification application page, account page, and login/logout page. The frontend also includes a locations view, which allows managers and instructors to see the different gym locations as well as a workshops page, which allows instructors to see the different workshops they can attend to upgrade their certifications. The backend is composed of controllers that correspond to each of the pages, as well as specific instructor and manager controllers to serve different content based on what type of user is logged in. A manager has admin privileges and can add locations and

workshops, approve new instructor addition requests, and manage existing instructors. The database is composed of models that correspond to each of the pages, including employee information, locations, and workshops.

For the testing phase, we will be using Jest and Enzyme to test our project. These two testing frameworks will be used to integration test our project. When done right, integration testing is the only form of testing that is strictly necessary. We will be able to cover some unit testing bases with our integration tests. Integration testing is basically ensuring that our different components/modules work as they should together. Some components may work fine when tested in isolation, but combined with other pieces of the project, problems are bound to occur.

We will use Jest to test parts of the project such as our API calls. Jest has a really nice mocking tool, which will allow us to fake any data that might need to be sent with those API calls [7]. Enzyme will be used to test things such as if a button does what it's supposed to when it's pushed [4].

3.3 Functional Testing

There are three types of testing that we will utilize: unit testing, integration testing, and acceptance testing. The following goes into detail about each stage of testing:

Unit Testing

We will be using unit testing to catch bugs early and often in the development of our project. A good practice to approach unit testing would be to automate a click-through of the entire application and make sure everything returns what is expected. This stage of testing only tests individual components, meaning there will be no interaction between different components when testing them out. Thus, there will be no direct interaction with the database and tests that require information from the database can use fake data that represents the information from the database instead.

Integration Testing

Integration testing will primarily be for testing interactions with our database. For instance, if a manager wants to add an instructor, then an integration test must be made to see that the instructor has been added to the database. This is similar to a unit test, however, the value being returned is directly from the database as opposed to being mocked or fake data.

Acceptance Testing

Acceptance testing is the final stage of testing to be completed after unit and integration testing. The entire application should be fully developed and function as expected, upon completion of unit and integration testing, before writing acceptance tests. When writing acceptance tests, we must make sure that assumptions and constraints are considered ahead of time. For instance, execution of a certain use case of the application can be affected by different operating systems and web browsers; the acceptance tests should be performed on each operating system and web browsers that are specified ahead of time. An example of an acceptance test for testing if a manager can approve a certification application would be to have a manager follow the following steps:

1. Visit the web application with the given URL
2. Log in with their given username and password
3. Navigate to and click on the “Certification Applications” tab
4. Click on one certification application that is requesting to be approved
5. Assuming the application can be approved, click on the “approve” button

The expected result would be:

1. The application shows up as “approved”
2. A pop-up shows confirming that the application has been approved and has a button to return to the home page or certification applications tab

3.4 Non-Functional Testing

For non-functional testing, we will be test for performance, security, usability, and compatibility of the web application.

Performance

In terms of performance, there is not much that needs to be tested. The only thing that would need to be confirmed is that there is little to no wait time to load pages. This should not be an issue unless there are thousands of entries in the database that need to be queried at once, which is not expected. If there were thousands of entries in the database to filter through for one query, then the load time for pages could be slowed down depending on if pagination were implemented or not.

Security

Our project will make use of authentication for instructors and managers. An outsider who is not an instructor or manager at AFIC should not be able to access the web application. If outsiders were to have access to the web application, there would be the risk of people other than AFIC instructors and managers accessing the instructor and/or manager information. Thus, we will need to test that the authentication works properly. Moreover, we will need to test that users can

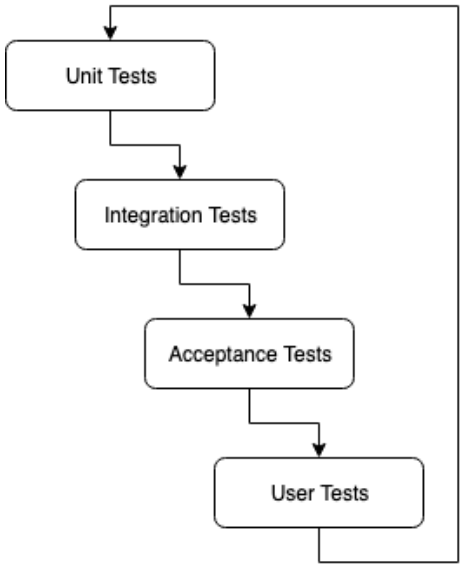
log in with only their username and password.

Usability
Since the web application will be developed specifically for AFIC instructors and managers, it should be easy to follow and intuitive for someone who is most likely not highly technical. To test the ease and overall flow of the application, we will have the actual instructors and managers use the application and provide us with feedback on how we could make the application easier to use.

Compatibility
Although our project has been requested to be mainly a web application, it can be extended to other platforms if time permits. If so, then the application would be able to be accessed on other smart devices, such as iPads, and on mobile. On each platform, the web application should function the same way as the web application on a regular browser. In addition, the formatting should be appropriately modified on each platform, and the overall user interface should look clean.

3.5 Process

Unit to integration to acceptance to user with loops back to each prior step for the diagram.
As stated in section 3.2, we will test the entire project with integration testing.



3.6 Results

This project is still currently underway so the amount of results for our final project we can conclude are limited. We were although able to make some conclusions from the initial website

AFIC presented us with. The website they were developing was using Angular as their frontend framework. We found that this version of AngularJS was now deprecated [2]. In order to update this framework to the current version of Angular it would need to be completely rewritten. This gave us the opportunity to pick a different framework, leading to us switching to React. We also took note that AFIC planned to use AWS as their cloud service. After completing research into other options we came across Microsoft Azure. This cloud service was more intuitive, provided extensive customer support and on average was a cheaper service than AWS [1], [8]. We look to take these current design decisions to improve AFIC's Instructor Certification Website. The result we plan to see is a robust, user friendly website that automates the instructor certification process.

4 Closing Material

4.1 Conclusion

So far, our team has researched different libraries for frontend and backend implementation. Out of this research, we have chosen to use React for the frontend, as it provides a well-supported, easy to learn environment for creating a functional, good-looking user interface. For the backend, we have chosen to use Express, which is a JavaScript framework for Node.js. Express is easy to set up and works well with React, so we felt that it was the best choice to use for the Instructor Certification Website.

We will also be using Azure, a Microsoft cloud service, to host the website itself. Initially, we planned on using Amazon Web Services to host the website, but after researching, we believe that Azure is the best choice, as it is generally less expensive and has better options for support. Azure will also be where we implement our database for the website [8].

For testing, we will be implementing a variety of different testing techniques. For functional testing, we will use Unit Testing to catch bugs early in the development, Integration Testing to verify that the communication between our backend and database is working correctly, and Acceptance Testing to verify that the final product works as specified by the client. When all three of these tests pass, the project will be considered satisfactory for completion.

Along with functional tests, we will also perform non-functional tests. These include Performance Testing to that the website is responsive and fast, Security Testing to make sure that malicious users cannot do harm to the website or its data, Usability Testing to make sure the website is easy to understand and navigate, and Compatibility Testing to make sure that the website itself is able to be viewed on devices of any size, such as smartphones and tablets.

Our main goal for this project is to create a website that allows AFIC to streamline their instructor certification process to make it faster and more efficient than it currently is. Right now, employees at AFIC manually enter instructor certification data into spreadsheets, which is a tedious and timely process. Moreover, the current instructor certification information process is also costly considering employees spend their time entering in the instructor certification information when they could be doing other work that has equal or higher priority. Automating the instructor certification process within a web application that both AFIC managers and instructors can utilize will help keep the instructor certification data be more consistent and accurate while providing an organized approach to how the data is collected and managed.

4.2 References

[1] "Amazon Web Services (AWS) - Cloud Computing Services", *Amazon Web Services, Inc.*, 2019. [Online]. Available: https://aws.amazon.com/. [Accessed: 26- Mar- 2019].

[2] "Angular", *Angular.io*, 2019. [Online]. Available: https://angular.io/. [Accessed: 26- Mar- 2019].

[3] "Django", *Djangoproject.com*, 2019. [Online]. Available: https://www.djangoproject.com/. [Accessed: 26- Mar- 2019].

[4] "Enzyme", *Airbnb.io*, 2019. [Online]. Available: https://airbnb.io/enzyme/. [Accessed: 26- Mar- 2019].

[5] "Express - Node.js web application framework", *Expressjs.com*, 2019. [Online]. Available: https://expressjs.com/. [Accessed: 26- Mar- 2019].

[6] "Farrell's eXtreme Bodyshaping", *Extremebodyshaping.com*, 2019. [Online]. Available: https://extremebodyshaping.com/. [Accessed: 26- Mar- 2019].

[7] "Jest · Delightful JavaScript Testing", *Jestjs.io*, 2019. [Online]. Available: https://jestjs.io/. [Accessed: 26- Mar- 2019].

[8] ”Microsoft Azure", *Azure.microsoft.com*, 2019. [Online]. Available: https://azure.microsoft.com/. [Accessed: 26- Mar- 2019].

[9] T. Otwell, "Laravel - The PHP Framework For Web Artisans", *Laravel.com*, 2019. [Online]. Available: https://laravel.com/. [Accessed: 26- Mar- 2019].

[10] "React – A JavaScript library for building user interfaces", *Reactjs.org*, 2019. [Online]. Available: https://reactjs.org/. [Accessed: 26- Mar- 2019].

[11] "Sequelize", *Docs.sequelizejs.com*, 2019. [Online]. Available: http://docs.sequelizejs.com/. [Accessed: 26- Mar- 2019].

[12] "spring.io", *Spring.io*, 2019. [Online]. Available: https://spring.io/. [Accessed: 26- Mar- 2019].

[13] "Vue.js", *Vuejs.org*, 2019. [Online]. Available: https://vuejs.org/. [Accessed: 26- Mar- 2019].

4.3 Appendices

No hardware was used for this project, so there are no data sheets or related materials available. Common programming libraries will be used within the web application, so there are no appendices to list for this project.