

Automated Fitness Instructor Certification Website

FINAL REPORT

Team Number: sddec19-04
Client: Dr. Mohamed Selim/AFIC
Advisor: Dr. Mohamed 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/>
AFIC Website: <https://instructor-certification-site.azurewebsites.net/>

Table of Contents

1 Introduction	4
1.1 Acknowledgement	4
1.2 Problem and Project Statement	4
1.2.1 General Problem Statement	4
1.2.2 General Solution Approach	4
2 Project Design	5
2.1 Functional Requirements	5
2.2 Non-Functional Requirements	5
2.3 Intended Users and Uses	5
2.4 Limitations	7
2.5 Interface Specifications	7
2.6 Hardware and Software	8
2.7 Operational Environment	9
2.8 Implementation Details	9
2.9 Design Analysis	10
2.10 End Product and Deliverables	10
2.11 Related Products	12
3 Testing Process	12
3.1 User Testing	12
3.2 Testing Results	13
4 Closing Material	13
4.1 Conclusion	13
4.1.1 Closing Remarks	13
4.1.2 Future Work	13
4.2 References	15
4.3 Appendices	16
4.3.1 Appendix I	16
4.3.2 Appendix II	16
4.3.3 Appendix III	19
4.3.4 Appendix IV	20

List of Figures

Figure I: Final Use Case Diagram.

Figure II: Final Component Diagram.

Figure III: Initial Client's Design.

Figure IV: Initial Dashboard Design.

Figure V: Initial Workshops Design.

Figure VI: Initial Application Design.

List of Definitions

AFIC: Automated Fitness Instructor Certification

AWS: Amazon Web Services

1 Introduction

1.1 Acknowledgement

Throughout this project, we worked with a number of different people. Dr. Mohamed Selim is our faculty advisor. His guidance was very useful to our team throughout the process. We were also in contact with 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 asked us to help them in the creation of a certification website for their instructors. Previously, AFIC kept track of all instructor information by hand in spreadsheets. Handling the information this way took 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 would 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 would 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 designed and implemented a new website for them. This website includes a database to store all of their instructor information, upcoming workshops, and anything else they need. The website also includes 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 was created using React, a well supported JavaScript library that specializes in creating beautiful and effective user interfaces. The backend of the website was 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. All of our data was organized in our database by Sequelize models and hosted on our cloud service of choice, Microsoft Azure. AFIC will now be able to instantly start using the created website to certify and add new instructors without needing to dig into multiple spreadsheets for the information.

2 Project Design

2.1 Functional Requirements

The following is a list of functional requirements of the project

- A well constructed database schema
 - Relations are designed well
- Authentication of an employee or instructor
 - Checks must be made for verification of manager or instructor
- Displaying all relevant AFIC data to users with different permissions
 - Instructor, workshop, and location lists
- Use of individual trainers data
 - Used for automating instructor certification
 - Managers have final approval of application
- Users can manage their own account data

2.2 Non-Functional Requirements

The following is a list of non-functional requirements of the project

- Only authenticated users are allowed to access the service
- Users are allowed to only see the data they are authorized to
 - Managers and instructors have different permissions
- The website does not crash and is easily maintainable
- The website will be intuitive and easy to use
- Both the web server and database should be set up in a maintainable way

2.3 Intended Users and Uses

The intended use of this product is for instructors to be able to submit and apply for certifications. This allows 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 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

- 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 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 to the users.

Since there are two types of user authorization in this program (Manager and Instructor), there are 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 cases are the registration and submitting applications use cases. The server will be in charge of authentication, sending notifications, and sending messages.

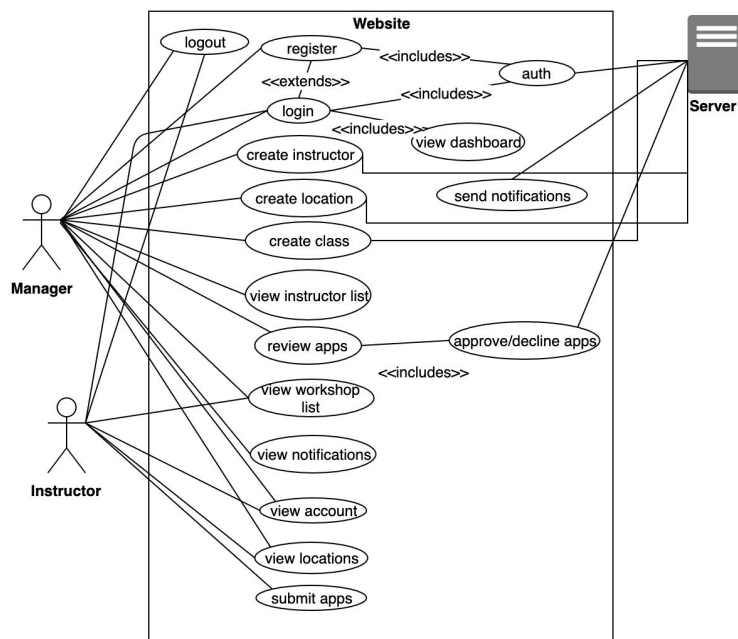


Figure I: Final Use Case Diagram.

2.4 Limitations

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

2.5 Interface Specifications

Our project does not interface with any 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.

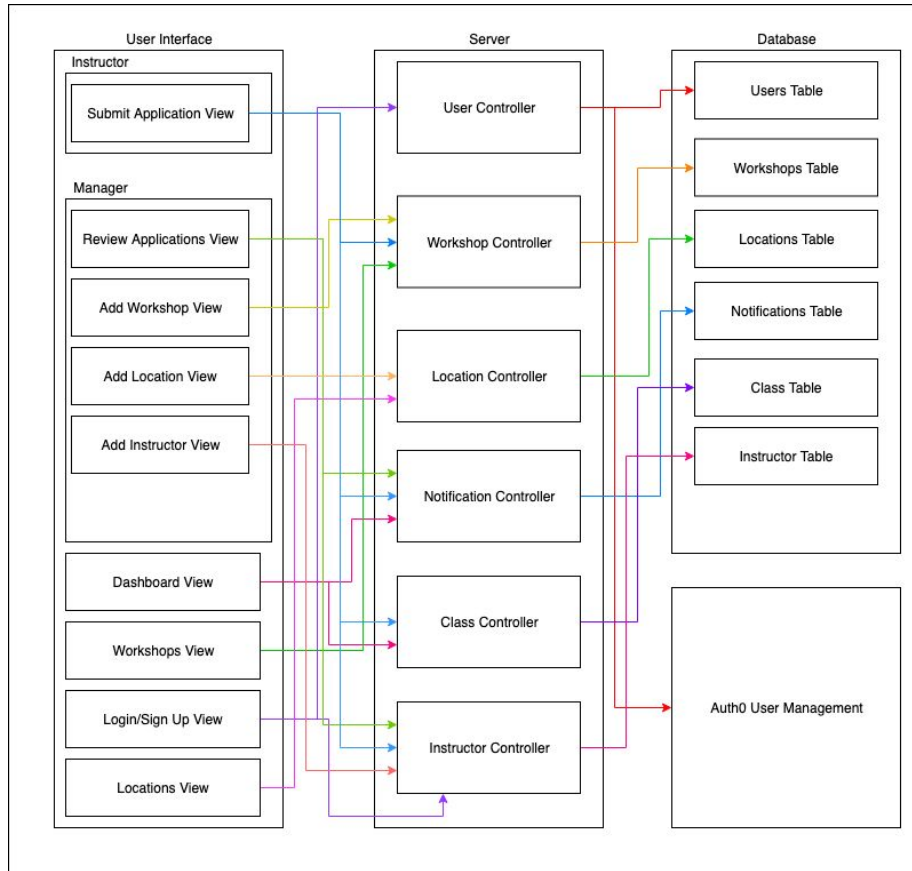


Figure II: Final Component Diagram.

2.6 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 encompass 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.

2.7 Operational Environment

The operating environment for our end product was calm and hazards were a minor concern. With our end product being a website, the project is run 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 are the servers maintained by a third party. It was the responsibility of this offsite third party to maintain and protect the servers. Therefore, any hazards pertaining to the servers were not within our team's development scope.

2.8 Implementation Details

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

For our frontend framework, we used 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 different languages that needed to communicate with each other. 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.

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

2.10 End Product and Deliverables

This project contains three primary components that will form the overall end product. The three components are our frontend, backend, and a database. This product itself is a fully functional

automated Instructor Certification Website. This website allows for the certification work currently done by hand to become a streamlined process. The three main components and final product act as our deliverables.

The first of these is the frontend, also known as the user interface. This was delivered in September and allowed for some feedback regarding usability to be delivered. The second component to be delivered was a fully functioning database, populated with up-to-date information we were provided with. This component is stored in our Azure cloud service and will be delivered in October. The last of the components is our backend Express service. This service allowed for the frontend and database to communicate and was delivered in November. The last deliverable is the final fully functioning product. All three of our components work together and reliably. No other additional materials were delivered alongside the finished website since none were requested.

- Functioning Frontend/User Interface - Delivery Date: September 6th, 2019
 - The user interface was developed with React and consists of any visual objects that are required in the website. The visuals range from images, buttons, text fields, tabs, etc. The functionality of all of these visuals may have not been fully completed but much of the basic web navigation was. The functionality that would not be completed would be anything that sends to or retrieves data from the database. This functionality began to be visible during the delivery of our backend. The frontend is designed to be intuitive and easy to navigate. Even with this design, it needed to be verified by the team at AFIC.
- Functioning Database - Delivery Date: October 23rd, 2019
 - The database delivered contains all of the up-to-date information pertaining to the instructor certification process. The information is organized in a manner that minimizes repetition and produces efficient results. The information contained within this database is able to be removed, added and modified. We host this database via Azure and have it interact with the Express backend. This has allowed for data to be retrieved for presentation or sent for modification using the user interface.
- Functioning Backend - Delivery Date: November 12th, 2019
 - The backend is the last component with the purpose of connecting the two prior components. As previously mentioned, the service used for the backend is 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 pass information through. The backend is also hosted using Azure.
- Functioning Instructor Certification Website - Delivery Date: December 5th, 2019

- The fully implemented Instructor Certification Website fulfills all of the requirements specified by AFIC. The website allows a Manager to create an account for an Instructor. From here, an Instructor is able to apply for recertification or an increase in certification. The respective Manager of the Instructor is then able to approve or disapprove. Any computer is able to access this website but only those with the proper credentials have any access. The user interface is intuitive and easy to use. The source code for this website is available via ISU's Gitlab and on the Azure Cloud Service. No additional documentation has been provided unless requested.

2.11 Related Products

Our client has a proprietary certification process and because of this, there were no other related products that we could use. There are other gym instructor certification sites on the market, but our client wanted their own website to support their certification process.

3 Testing Process

3.1 User Testing

Our development process relied on manual user testing. This process is described in detail in the next two sections.

User Interface Testing

Throughout our development of the frontend, we did frequent manual tests of the UI to make sure it functioned as expected. This included doing things wrong, such as clicking in abnormal areas or sending bad input. This ensured that our UI functioned as expected and in a desirable way. In addition, as we began to develop and integrate our database and server, we did frequent manual tests to ensure we didn't have any logic errors. As we developed, we frequently drew back on our expected specifications and compared them with what we were implementing. Doing this ensured that we wouldn't have to come back through in the future and change application logic, leading to faster development overall.

Security Testing

Our project makes use of an authentication process for instructors and managers. A user who is not an instructor or manager at AFIC should not be able to access the web application. If non-authenticated users were to have access to the web application, there would be the risk of

people other than AFIC instructors and managers accessing user information. Because of this, we did thorough testing of our authentication process, ensuring that only permitted users are allowed to access site content.

In addition, once a user is authenticated, they should only be able to view and edit content available to their permission level, either instructor or manager. To test that pages and functionality were properly protected, we signed in as different user types and attempted to access pages not allowed by our user type. Through this process, we made sure that only the correct user type can access each page or page function.

3.2 Testing Results

We tested our product throughout our development process to ensure it functioned as expected. The results of this testing process was a website free of bugs and logic errors. We also had proper permissions set up across the site, including an authentication process that requires users to login before they can access any site content.

4 Closing Material

4.1 Conclusion

4.1.1 Closing Remarks

AFIC previously had been doing all of their data management and instructor certification by hand using spreadsheets. They requested that this process be brought online in order to allow for less time spent on this task and business expansion. As a start to the project they presented us with a front-end that has been started in Angular1. We also received some simple data to help us get an understanding of their business structure and certification process. From here that initial design was redone using the React framework. AFIC's data is organized using Sequelize models that communicate with an Express API. The website's resources are hosted on Microsoft Azure allowing for access from any device. The final project accomplishes AFIC's initial request and can easily be expanded upon and maintained.

4.1.2 Future Work

In order to expand upon and improve this project a few main aspects were identified. The first of which being a bug with some page reloading where the user is directed to the login screen. The user is still logged in but has to press the login button again. The user will still be directed to the

appropriate page after but this is an inconvenience. We believe it has to deal with the compatibility of our third party permissions provider auth0 and our React state.

We would also like to revise our Express API. Our initial design and API calls were set up in a way we saw best fit before testing. It was also difficult for us to completely predict how our relations would interact with each other and what each data point would require for a successful request. Due to time constraints we had to make this work to the best of our ability but some of our API calls were messy. Revising this API would improve the overall performance of the website along with the quality of the code written.

That last point worth mentioning is improving our input validation. Not all of the possible invalid inputs were considered and some were unknown. There is also no error reporting for the user regarding what is entirely wrong with their given input. Making sure the input validation is complete would protect against illegitimate data and user confusion.

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

4.3.1 Appendix I

Running the Project Locally

1. Clone the Git repo (HTTPS)
 - 1.1. `git clone https://git.ece.iastate.edu/sd/sddec19-04.git`
2. Install node modules in each directory
 - 2.1. `npm run install-modules`
3. Create environment files
 - 3.1. Navigate to the /client directory.
 - 3.2. Look at .env.example to see what key-value pairs you need to get.
 - 3.3. Create a file called .env.development with the specified keys.
 - 3.4. Get the key values from someone that has them already.
4. To start the project from the root directory, run:
 - 4.1. `npm start`
5. Misc. general info
 - 5.1. The client runs on whatever port it is set to in .env.development.
 - 5.1.1. Use `PORT=8080` if unsure about what to set it as.
 - 5.2. The server runs on `PORT 300`

4.3.2 Appendix II

Initial Website Design

Our initial website design for some of our frontend pages are shown below. A few of our pages' designs have not changed, so we will only show those that are noticeably different in our current implementation.

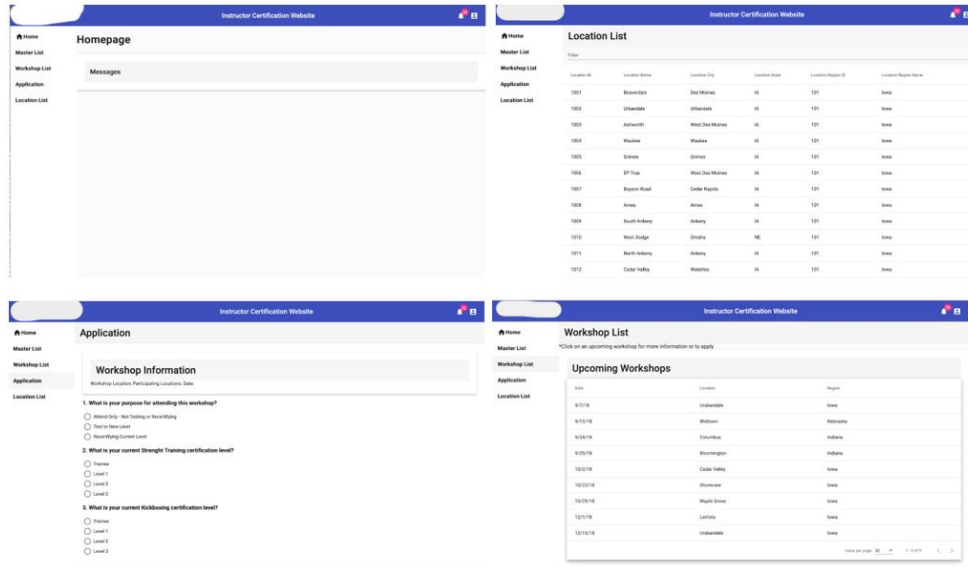


Figure III: Initial Client's Design.

We chose to redo this initial website design as the framework being used, Angular1, was deprecated when we received it. In order to create a website that would be maintainable and scalable a redesign would have to happen regardless. We chose React as it is more widely used in industry and group members had previous experience.

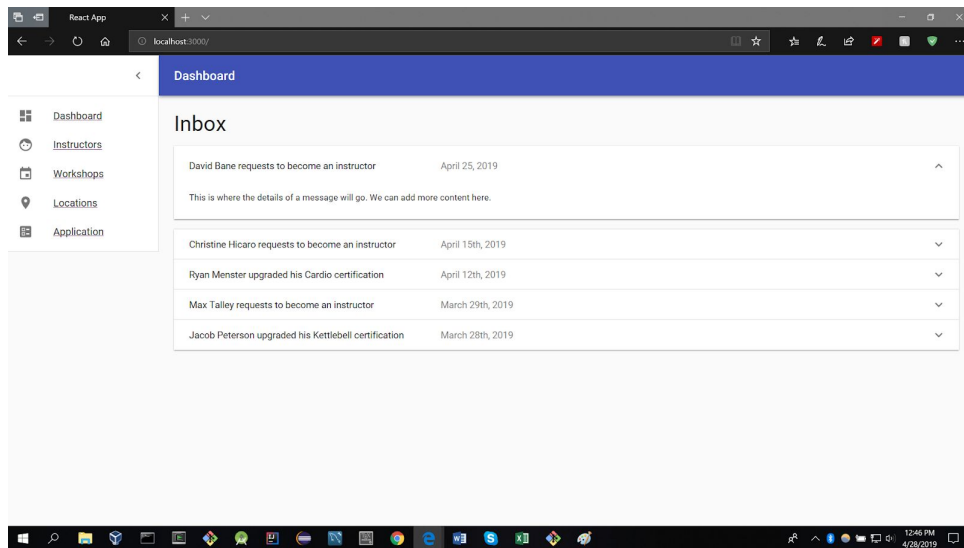
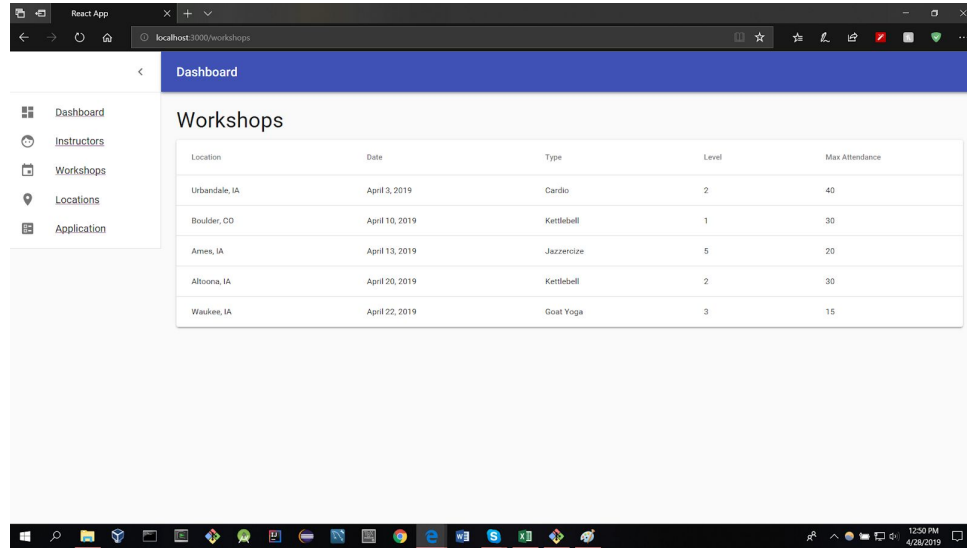


Figure IV: Initial Dashboard Design.

Our Dashboard design has changed considerably since its initial design. Our current dashboard displays more than just some notifications. We now have two different dashboards: one for Instructors and one for Managers. The Instructor dashboard shows the user's current

progress toward their next Kettlebell/Strength Training Certification level, as well as any upcoming classes that they are scheduled to teach. The Manager dashboard shows the different levels of Instructors under the currently logged in Manager.



The screenshot shows a web browser window displaying a React application. The browser's address bar shows 'localhost:3000/workshops'. The application has a dark blue header with the word 'Dashboard' in white. On the left, there is a sidebar menu with icons and labels for 'Dashboard', 'Instructors', 'Workshops', 'Locations', and 'Application'. The main content area is titled 'Workshops' and contains a table with the following data:

Location	Date	Type	Level	Max Attendance
Urbandale, IA	April 3, 2019	Cardio	2	40
Boulder, CO	April 10, 2019	Kettlebell	1	30
Ames, IA	April 13, 2019	Jazzercise	5	20
Altoona, IA	April 20, 2019	Kettlebell	2	30
Waukee, IA	April 22, 2019	Goat Yoga	3	15

Figure V: Initial Workshops Design.

Our Workshops page design has changed slightly since its initial stage. The biggest change that we made was including a modal that pops up when the user clicks on a row in the Workshops table. This modal displays additional information about the selected workshop, such as its location on an embedded Google Map, information about the location, and information about the Instructor leading the Workshop.

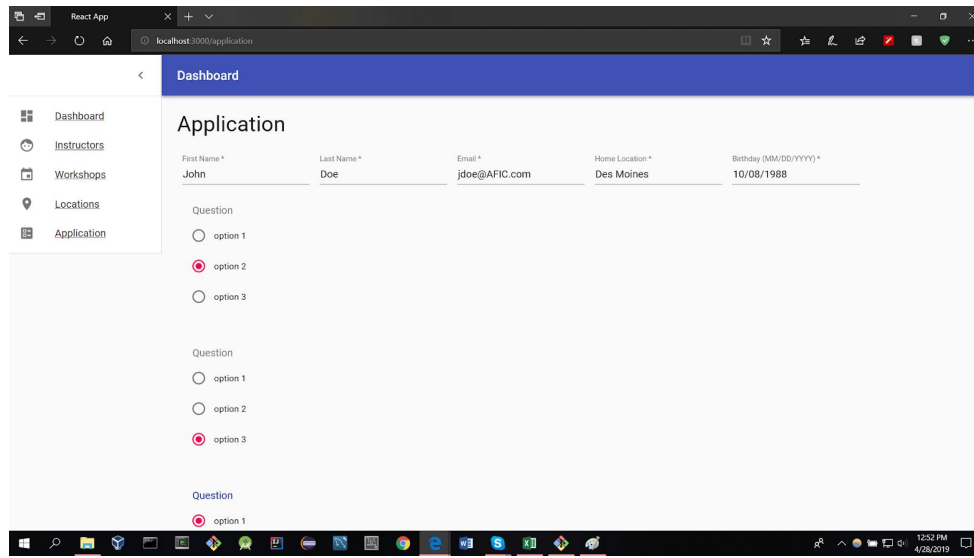


Figure VI: Initial Application Design.

Our Application page has seen a major redesign since its initial implementation. We now enable the logged in Instructor to specify which type of Application they want to submit (Kettlebell or Strength Training). We also automatically pull in the logged in Instructor's information regarding their Workshops attended and Classes taught to determine whether or not they are eligible for applying.

4.3.3 Appendix III

Lessons Learned

While working on our project we came to a realization that lead to a big lesson learned. This was about how we went about the order of our project implementation. The order we chose to go in was designing our front-end views, creating a back-end API to the front-end and then adding in authentication. We believe this is the order we chose as we were given a front-end to start. This lead us to focus on redesigning that front-end with our new framework, React. From here the line of thought we had was replacing our fake data with real data and securing it.

The order we should have chosen should have been authentication, creating the back-end API and ending with creating our front-end views. The reasoning behind this is while security was important to us, it should have been the most important thing to us. Having security set up initially would insure it was implemented properly. This would also have saved a lot of time in wrapping all of our front-end elements in our authentication system. From here being able to see our raw data be passing from the database to a front-end would have let us use it right away. All we would have had to do was make it be presented in a visually pleasing manner.

4.3.4 Appendix IV

All of the code for our project is stored in our GitLab repository.

<https://git.ece.iastate.edu/sd/sddec19-04/tree/master>