Final Project: Building A Full-Stack Website in Handyman Contracting Service

University of houston-downtown | One Main Street, Houston, TX 77002

The University of Houston-DOwntown

Duong, Thien

2020

**Project Title:** Building a Full-Stack Website in Handyman Contractor Service

**Mentor:** Dr. Kenneth Oberhoff, the University of Houston Downtown, College of Sciences and Technology

**Student Name:** Thien Duong

**Completion Date:** December 2020

**Table of Content**

(To be finished upon completion of Paper)

**Abstract**

In the age of technology, when services such as Uber and DoorDash has made a strong cases for gig workers, you can get a ride or your food delivered to the front-door at the fingertips, imagine that when your pipe busted in the middle of the night, when your air condition system shut off in the heat of Texas and you know calling the handymen services companies will be very costly since these companies operates with thousand to million dollar in overhead (administrative, building, phone system cost) . With the current trend of high school’s students flocking to colleges and university, the nation is expecting a surge in demand for handymen and there could not be a better time to be a self-employed handyman right now. With the help of technology, specifically through the web interface, client can now contact local handymen and order a specific service (plumbing, carpentry, electrical,…) with each handyman carefully considered and highly rated by other customers. Handymen, now you can be your own boss without worrying about the overhead costs, just a small percentage will be charged for each time service performed and you get paid by customers.

**Acknowledgement**

I would like to express my most sincere gratitude to my project mentor Dr. Kenneth Oberhoff for providing his guidance, suggestions, and encouragements throughout the course of this project

Also, I would like to thank all my peers in the class for their feedbacks and suggestions during and after presentations throughout the course.

**Chapter 1: Introduction and Describe Project Problem**

1. **Introduction**

With technology on the rise, our world is seeing services like Uber, Lyft, DoorDash and Airbnb… We are witnessing the success coming from enterprises that give their contractors flexibility, ownership, independence. These corporations’ success did not stem from the fact that they delivered very agile, robust, and effective technological products, but rather the first and foremost they ask themselves the very simple yet necessary question: Will my customers want this product? It is not difficult to come up with this question from a business mindset, but from the technological mindset, people tend to measure things with its “coolness” rather than its capability of delivering revenue and competitiveness. If the technology’s revenue is equivalent to the “coolness” of product, then there would not be hundreds of applications from the biggest technology companies in the world ended up in the dumpsters or be left discontinued.

Knowing how to combine these two mindsets is the secret behind the success of 2020’s technology enterprises. Demand and Sustainability are crucial.

According to Bureau of Labor and Statistic, there will be a 4% increase (60656 jobs) of general maintenance and repair workers within the next 10 years, along with 59000 jobs for electricians, 15,072 for HVAC workers,… With the current pace, trades workers will be in high demand for a foreseeable future. (Bureau of Labor Statistics, 2020)

While there are demand for trades workers, the current service delivery mode has not progressed much with limited technology usage, for example, if you want to have something fixed in your house, you have to make a call to the local trade shop’s phone number and wait. What if there is an app where you can reach directly to one of the verified contractors with experience, transparent pricing and been reviewed by our previous customers. The app, unlike many companies operates 24/7 and if the workers choose to be available 24/7, you would not be having to worry about a busted pipe at 9 PM at night. While Uber, Airbnb, and DoorDash are significantly improving our lives. An application that can make the process of delivering a service much more efficient.

1. **Project problem**

In this project, I will deliver a web application that customers can use to find local trades workers and request their services, customers can also view trades workers experience of work and the workers’ previous customers’ reviews, as well as viewing service workers’ business info and average charge price per hour. Through this application, instead of calling the laborer’s company to request service and wait, customers can reach the laborer directly and the service will be performed and billed through the application and the application will handle the laborer’s payment. Thus eliminating the overhead cost for an administrative, accounting, sales and marketing, instead more focus will be in the laborer’s performance, the customer’s satisfaction and the growth of the network of services provided. Through this medium, clients and workers as well as the project’s owner have the level of transparency when it comes to workers’ previous performance, workers’ cost of labor. This can also be a great tool for employers’ productivity management for its employees to come of with Human Resources decision (reward, praise, discipline). The system can be agile, using continuous development and deployment tools to ensure the availability of service to customers and management while the workers’ get to focus on their most importance task-providing the best services to our customers for generating revenue purpose.

In order to build a functioning web application, the following development tools will be used, also mentioned are the tools advantages as the reason why I’ll be using them for the project.

**Front-end**: Include user interfaces and user experience, frontend components makes it possible for users to see what was shown in the web pages consisting of HTML (web template), CSS (animation and styling), JavaScript(logic of programming and DOM(document object model), these three ingredients enable users to interact with websites.

For this particular project, VueJS, an open-source frontend development framework will be used to build user interfaces and single page applications within our frontend section of the project, VueJS is one of the most used frameworks in the market due to its lightweight and reactivity. Components built by VueJS can be easily reuse and recycle in different project, aside from vanilla VueJS, I will also use VueX (a state management tools for VueJS to improve code’s cleanliness and improve debugging process) and Buefy (a prebuilt CSS framework to enhance User Interfaces). VueJS’s also trusted to be used in famous fortune 500 companies (Google, Apple, Adobe, Nintendo…) (MadeWithVueJS.com, 2019)

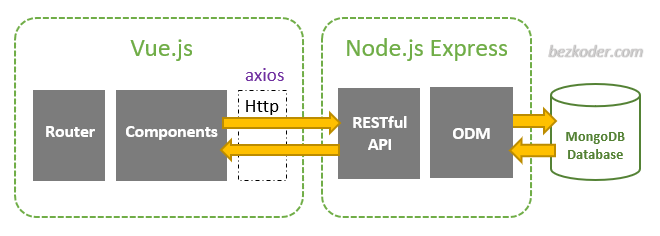
**Back-end:** Include server, data access layers of the website, in contrast to frontend, backend’s job is to ensure the end-users get data or service they requested. The codes written in the backend enabled browers to comunitcate with database and servers.

For this project, NodeJS, an open-source, cross-platform and event-driven JavaScript runtime environment that’s capable of executing JavaScript outside the web browsers, using NodeJS, users can write backend code to direct routes in for an API. Node is known for its simplicity yet effectiveness, simple to use, yet capable of handling large applications, fast to deploy and supported by a huge tech community make NodeJS the web framework of the decade. (Brewster, 2020). ExpressJS, which is a component of Node, will be responsible for building APIs routes and fetch component at particular endpoints, in the project, I will use the REST API architecture for the backend (Figure ). Methods in RESTAPI include: POST, DELETE, GET, VIEW, PUT which can help users retrieve, add, delete, and modify data from a database (Kenneth Lange, 2018). NodeJS, ExpressJS and REST APIs are reputable and lightweight modern frameworks that are used by LinkedIn, PayPal, Netflix, Uber, NASA, eBay, Walmart.

**Database:** One crucial component of a web application, database is an organized collection of data, stored and accessed electronically from a computer system. For this particular project I’ll be using NoSQL database model. Unlike traditional relational database, NoSQL follows the new model of data called non-relational document database(similar to JSON object). This approach increases the agility of the data without defining a specific schema hence data can grow and adapt in many way without complexity as its counterpart SQL.

MongoDB is one of the most popular database programs, known for its JSON like document storage and open-sourced. The platform has grown exponentially due to its scalability, availably and simplicity with many users choose MongoDB over traditional SQL for there small and medium sized projects. For this particular project, I choose MongoDB with MongoAtlas (the cloud based service for MongoDB) due to its availability and affordability.

Overall, the three components specified would be able to handle the tasks of building, showing and controlling the contents for the web application through this architecture. (Aside from what’s shown in the diagram, GitHub will also be used for version controlling purpose.)



**Version Controlling:** An optional part but extremely helpful part of the project, GitHub will be used to keep track of changes in the code as well as mapping the change in the structure of the project (change in files, deletion of file, addition of files). Git can be extremely powerful tools for developers to collaborate in a specific project as well as for management to keep track of developers’ productivity. In the future if the product is commercialized, GitHub would be transitioned to GitLab (Enterprise version of GitHub). With version control, bugs and code modifications would be much easier in both team collaboration as well as single developer team.

**Chapter 2: Details of Work**

1. **Frontend:**
2. **Structure and Overview of Files:**

The frontend part of the project is put in the folder client of the whole project, in the folder there is *package.json* file*, node\_modules* folder, and the *src* folder.

The *package.json* file specifies how to run the project on the command line interface, specifies the dependency packages needed for the frontend application to run.

The *node\_modules* folder, contains all installed packages needed as specified by the package.json file, this is a heavy folder and won’t be appearing in GitHub repository but can be installed by simply running command “npm install” while inside the *client* folder.

The *src (source)* folder contains the subfolders such as *assets, components, router,* *store, views,* and files used such as *App.vue, main.js.* Each file and folder serve a crucial part in the application.

*App.vue:* The root of the application defined in Vue Components file format; it also defines the templates of the page.

*main.js:* The JavaScript file that initialize the root component *App.vue* into an element on the page. It’s also responsible fir setting up plugins and 3rd party components such as VueX (storage and state management) and Buefy (pre-written styling library)

*assets:* The folder containing the main.scss and variables.scss stylesheets and any assets that are imported into the components

*components:* The folder containing all the Vue components that would be used in the application. Inside the *components* folder, *ActiveUser.vue* renders individual worker for *OurWorkers.vue* view of the website. *EditUser.vue* renders a form that allows admin to edit the individual worker data. *NavBar.vue contains the navigation* bar for the website. *LoginPage.vue* contains the login form for administrator authentication purpose. *UserData.vue* contains a form that allows admin to create new workers. *CustomerData.vue* contains a form that will allow customers to input data to make a request of service

*router:* The folder containing JavaScript file used to manages routes within the frontend pages of the application.

*store:* The folder containing JavaScript file used to manages the stores and the states in order to communicates between components using VueX

*views:* Containing individuals webpages in the website, also each webpage can be accessed by a route, which was specified by *router/index.js*. There’s *About.vue* page that renders the About page, *EmployeeTable.vue* renders the employees table page, *Home.vue* renders them home page and *OurWorkers.vue* renders the Our Workers page. Those are the five main routes and pages of the website which can help users (admins or customers) navigates through the website better than just a single-page web application.

1. **Details and Functions of Files**

In *components* folder:

*ActiveUser.vue:* This file renders individual worker, the rendered data in this page was received from the *OurWorkers.vue* views folder, it displays Worker’s name, expertise, phone number, email, hourly charge and an image of the worker, if user is logged in as administrator, it will display the options to edit or delete the worker, if user is not logged in (using customer mode), it will display the option to hire the chosen workers, thus rendering the *CustomerData.vue* components. The edit and delete options will directly change the data in the database.

*CustomerData.vue:* Child component of *ActiveUser.vue*, this component provides inputs for customers to enter and submit name, phone number, email, ZIP code, and the service needed, upon submission, the data entered will be sent to the server and database through *axios* as a customer request (ticket) and only administrator can view the details of the requests.

*EditUser.*vue: Child component of *ActiveUser.vue*, this component is render upon the Edit User button in *ActiveUser.vue* is clicked, it provides inputs for administrators to edit the data of the worker with input’s placeholder data from its own data, upon clicking Submit, the updated data is sent to the server through *axios* and update its current worker’s data in the database.

*NavBar.*vue: Child component for the *App.vue* file, this component is designed to be rendered in every webpage, serves as the navigation bar for the whole website where user can switch between webpages. If user is logged in (admin mode), the Log Out, Employee Table, and Customer Request option can be seen. While if user not logged in (customer mode), only the Log In option can be seen. The Navigation bar also specify if user is in Admin Mode or Customer Mode

*LoginPage.*vue: Child component of the *views/Home.vue* file, this component renders a form where users can log in to enter the Admin mode. Admin mode gives user the ability to create new worker, edit worker, delete worker, view workers table, view customer’s requests. When the login form is successfully submitted with the valid and correct username and password, an authentication token will be sent back from the API, this token will be stored to a local storage, thus unlocking the features that an administrator can do.

*UserData.*vue: Child component of the *views/OurWorkers.vue* file, this component renders a form where user(in Admin mode) can add a new worker through the inputs provided by the component, the entered data then will be sent to the server to be process and add to the database using axios. New user then added to the workers array in its parent component (*OurWorkers.com).*

In *views* folder:

*Home.vue:* Main page of the website, allows user to login to enter Admin mode. This component is the parent of *LoginPage.vue* component

*About.vue:* About page of the website, display the history and inspiration of the site and GitHub link to the website main repository

*EmployeeTable.vue:* Display the worker’s productivity data of the site, where workers’ information (ID, name, phone number, email) and productivity grade are rendered in a table (with the CSS styling coming from Buefy).

*OurWorkers.*vue: Display the list of the workers, this component is the parent of *ActiveUser.vue* and *UserData.vue* components. The file main data is an array of worker objects fetched from the backend API Http GET method and *beforeMount()* Vue lifecycle method, data pulled in JSON format is then stored to the array to be rendered by *ActiveUser.vue* component. Since the data is pulled and rendered by this view webpage, other methods in this view are: *addUser(), deleteUser(id), editUser(id), saveUser(savedObject).* Each method interacts between the client webpage and the server/database through RESTful architecture. (Diagram 2)

*CustomerRequest.vue:* Display the list of service requests made by the customers, displaying customer’s data (email, phone, ZIP code) and their entered service requests as well as the name of the worker requested, styling for this view page also used Buefy and it comes with column sort and search option.

In *store* folder:

*index.js:* This file is the main controller for VueX store of the website, the file contains *state* object, which is used to store data, *mutations* object contains methods used to change the data in the state, *actions* object contains methods that can be used to call *mutations* but with the asynchronous options. *Getters* object allows users to retrieve the data of their choice from the state without altering its original state.

In *router* folder:

*index.js:* This file defines the routes for our websites using the built-in VueRoute library , all the routes are stored in an array of route objects. Each route object in the array is specified by *path*(path to the page)*, name*(name of the route)*, component*(webpage to be rendered by the route).

**Tools Proposal**

Front-end: VueJS (HTML, CSS, JavaScript)

Architecture: Model-View-Control (MVC)

Back-end: NodeJS, GraphQL, JWT Authentication, Docker, REST API

Database: MongoDB

Version Control: Git, GitHub

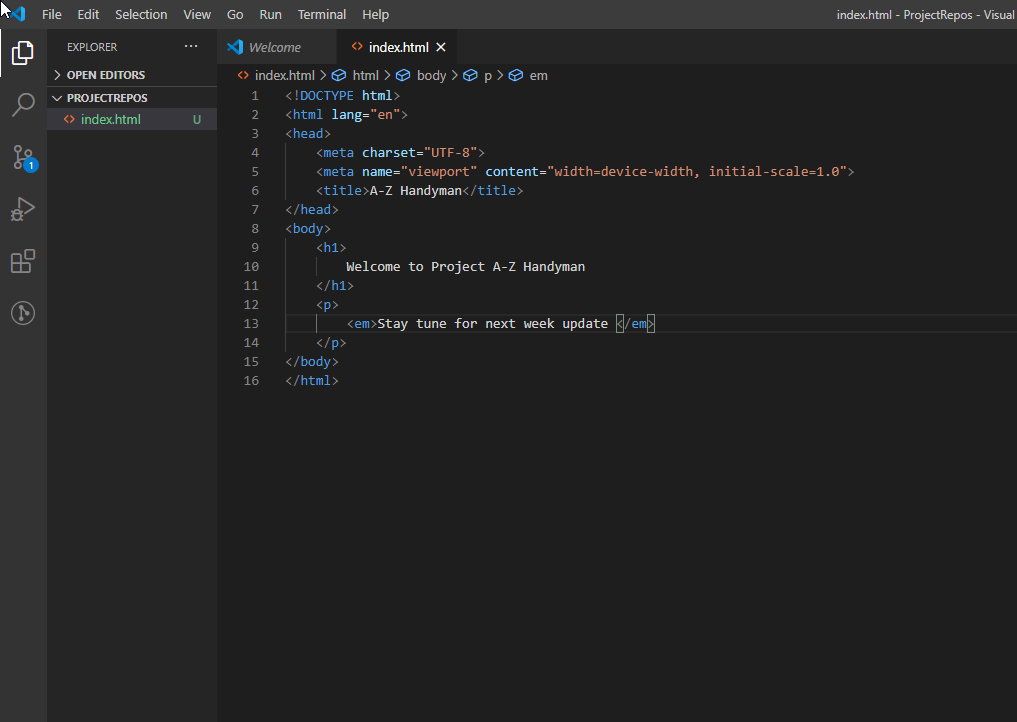
Deployment: AWS for Student

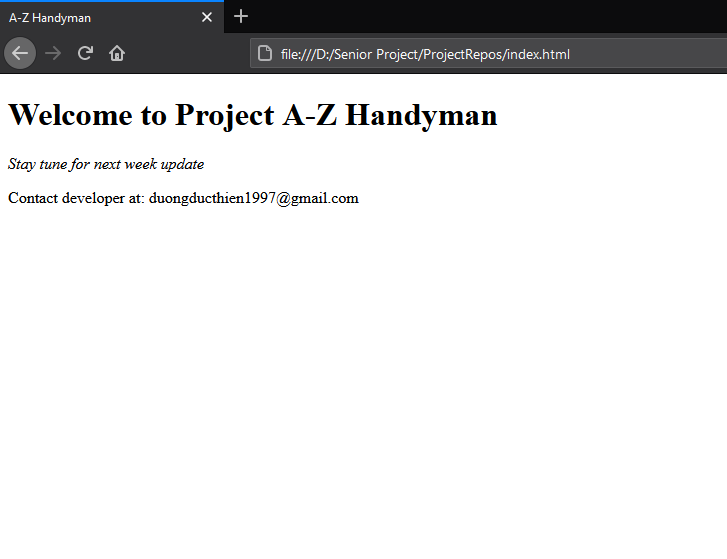
\*Tools proposed may change due to availability, developer’s knowledge or time constraint

**Week Zero-Incubating**

**Progress Made:**

* Drafted project proposal, idea forming
* Researched for the tools that fits the purpose of project
* Created Development Environment





* Sought Resources for Training Purpose (Online Courses for each tool)
  + **Training Resources:**

<https://www.udemy.com/course/vuejs-2-the-complete-guide/>

<https://www.udemy.com/course/nodejs-the-complete-guide/>

<https://learndocker.online/>

https://aws.amazon.com/training/

* Develop agenda for next week’s progress:
  + Complete front-end pages and routes architecture
  + Building Landing Page
  + Initiate Git Repository for tracking code progress