Proof of Concept Web Application for K and J Medical Supply

Michael Castro

2022 Laurea

Laurea University of Applied Sciences

Proof of Concept Web Application for K and J Medical Supply

Michael Castro
Business Information Technology
Bachelor’s Thesis
December, 2022

Laurea University of Applied Sciences
Business Information Technology
Bachelor's Thesis

Abstract

Michael Castro

Proof of Concept Web Application for K and J Medical Supply

Year 2022 Number of pages 36

Advancements in technologies have paved the way in digitizing various services. Companies that utilize digital services gain a positive impact in their businesses, and their employees can perform more effectively with the help of technology.

This thesis covers the whole process of developing web applications POC (proof-of-concept) to account for the inventory of a medical equipment store. K and J Medical Supply is a one-stop medical supply shop located in Manila, Philippines.

The theoretical part of the thesis report various methodologies: application development methodologies including software development life cycle (SDLC), project management methodology and multiple software frameworks.

Primary data for this thesis were collected by interviewing the representative of the company. Literature review from print and online sources are the main sources of secondary data and are used to support the thesis.

The project was able to successfully create a database, a frontend application, and a backend application, that manage the inventory of the company. The last part of the thesis also discussed challenges encountered and future development ideas.

Keywords: POC, HTML, CSS, JavaScript, React, Node JS, NestJS, PostgreSQL, GitHub

List of Abbreviations

POC	Proof-of-concent
HTML	Hypertext Markup Language
CSS	Cascasding Style Sheet
JS	JavaScript
API	Application Programming Interface
URL	Uniform Resource Locator
W3C	World Wide Consortium
ECMA	European Computer Manufacturers Association
Amazon EC2	Amazon Elastic Cloud
CLI	Command Line Interface
CRUD	Create, Read, Update and Delete
MVP	Minimum Viable Product
SEO	Search Engine Optimization

Contents

1 Introduction

We are witnessing advancement in technologies for decades now. Different industries are benefiting from technological gains and software development has its fair share of significant growth. Companies are designing their enterprise resource planning (ERP) to adapt new technologies and shift from traditional practices to a much modern electronic processes (Flanding, Grabman and Cox 2018, 1-2).

The case company for this thesis is K and J Medical Supply. The assignment was to explore and create a POC (proof-of-concept) web application for the case company. POC provides justification that a particular concept can address certain issues thru evidence. Determination if concept can work on the determined setting relies on the presented evidence (Marcolongo 2017, 75).

1.1 Objectives

The project was to make a proof-of-concept web application that can be used by K and J medical supply to manage their inventory. The web application aims to replace the current inventory system that the company employs. There are two main objectives of the thesis:

- Having a database that can securely manage and keep track of the inventory items.
- Adding a backend application that has direct connection with the database and has different APIs for managing each function.
- Creating a web page where company employees can access the database and perform operations in the items, which includes, new item creation, item update and item removal.

The web application is protected with authentication, only registered users with admin privilege can access the page and perform operations. When the users visit the page, they will be taken to a login page where they can input their credentials. All requests are sent to the backend application.

1.2 Project Scope

A frontend and backend applications will be built in the project. Web application is the responsibility of the frontend. On the other hand, the backend will be responsible for API creation, database connection and all the services required.

The project, being a proof-of-concept application, does not have a full-working application, but a minimum viable product (MVP). The MVP will be used to validate the feasibility of the idea. It will also be used by the client company to get an idea of how the new inventory system works and can then provide some feedback and suggestions.

2 Theoretical Framework

2.1 Application Development Methodologies

2.1.1 Requirement Analysis

Defining and identifying the needs of the business is an essential part of software development and is done at the early phase of the project. This stage includes gathering important details about the business requirements, analyzing the gathered data, and collecting supporting details. Further analysis is also taken on how it would affect the end-users and the organization (Mohapatra and Rath 2020, 40).

2.1.2 Design

Design stage identifies different details of the application. In this stage, important specifics are documented, like what programs are needed, how features will interact with each other, how the user interface will look like and how are data be presented (Mohapatra and Rath 2020, 40).

2.1.3 Development

The development phase, also called iteration phase in agile development process, is when the requirements gathered from the earlier stage are put into action. In this phase, several modifications are done to improve the overall project outlook. The company and client should maintain open communication during the development phase, as each software releases are evaluated by both stakeholders (Nehra 2022).

2.1.4 Testing

The codes that were written are subjected to testing. During this phase, the program is checked against different criteria and assessed if it is working according to intended behavior. Codes are written in parts; these parts are tested separately. Also, the application is tested in its entirety, to evaluate if the parts are working together as intended in the whole application (Mohapatra and Rath 2020, 40-41).

2.1.5 Maintenance

Some maintenance or updates may be needed when the application is running. This can be performed by the original developers of the application or outsourced from another firm. Different categories of maintenance are listed below:

- Adaptive maintenance is performed to ensure that the application is fit to run in the changing environment.
- Corrective maintenance is done if problem is discovered after the deployment.
- Emergency maintenance is an urgent maintenance that is not anticipated and carried out to keep the application running, while waiting for corrective maintenance to take place.
- Preventive maintenance are modifications that is needed to prevent issues found that might cause operational failures (Hughes 2016, 425).

2.2 Project Management Methodology

Software engineering is comparable to other engineering projects, both undertaking the needs to achieve specified tasks. At the beginning of software development, a plan must be made to guide the software engineer along with the project. It is also important that software engineers are aware of the potential problems that may arise and be prepared to deal with it (Stephens 2015, 4). It is common for software projects to select the most appropriate methodologies that fit their needs.

2.2.1 Agile Software Development

For this project, agile software development has been chosen. Agile approach means that there should be continuous communication between the client and the IT firm, and that both parties must actively participate in every step of the software development (Cooke 2012, 23-24). In addition, Kanban, an agile method, is utilized in this project. Kanban is the Japanese process for continuous improvement, and stories in Kanban refers to tasks. There are stages in Kanban development, and these can be periodically released (Highsmith 2010, 197).

Agile approach also introduced incremental planning. Traditional IT projects relies on heavy planning, even before development phase starts. This became an issue because even the most competent IT professionals cannot anticipate everything that may happen during the project implementation. On the other hand, incremental planning allows client and the development team to adjust to issues that arises during the project. The team can shift focus, or they can adapt solutions that they think is best for the project, as the development moves forward (Cooke 2012, 25).

After discussing how agile methodology works, it seems that it will work well with the project. The client company and the researcher need to work in close cooperation throughout the project. The researcher can leverage incremental planning, as it is inevitable that issues may arise as project progresses. Multiple iterations will be made and, in each iteration, different phases of application development methodologies may be utilized, depending on the situation.

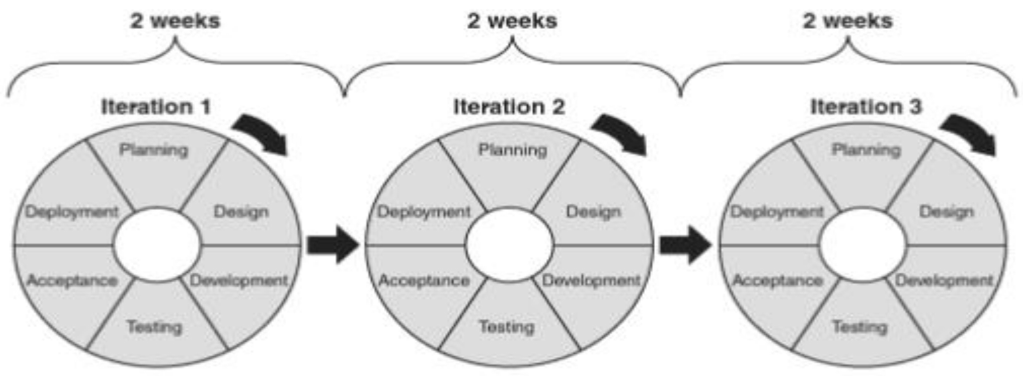


Figure 1.1 Agile Software Development (Hughes 2016, 13)

3 Backend Development

3.1 Node JS

Node.js is first introduced in 2009 by its creator, Ryan Dahl. It is running on top of JavaScript's V8, which is a high-performance engine especially at that time. The main feature of Node.js is that it can handle multiple requests and perform multiple operations at the same time (Robbins 2012, 2-3).

Node JS uses module to build applications and can be set up by requiring the module in question (see figure 1.2). Modules must be installed at the beginning, and it can be used by giving module identifier. Once required, modules' functions and methods can be used in the Node JS application (Herron 2020, 73-74).

```
const cookieSession = require('cookie-session');
```

Figure 1.2 Requiring module cookie-sessions and identifying it as cookieSession

Node JS is chosen in this project because it is an open-source cross-platform environment and ease of usage. Setting up Node JS application is straightforward and applications running on Node JS runs fast. Security and support are also attended since Node JS is updated regularly. Alternatives are PHP, Java and Python.

3.2 NestJS

According to NestJS' official documentation, NestJS is a framework under Node.js and is use for building server-side application. It has extensive support that allows various libraries and framework to be integrated in the NestJS application. It is well updated and use the latest JavaScript features.

The project make use of NestJS as it is lightweight, and it makes backend development simpler. It has several built-in features that the developers can use conveniently. Furthermore, starting an application using NestJS is easy, as it has its own powerful CLI. Alternatives for Node JS framework are Express JS, Koa.js, Meteor-js and Socket.IO.

3.3 PostgreSQL

PostgreSQL is a server that can handle various types of data and to some extent, non-data server processes. It has extensive support for different languages, including Perl, JavaScript, C and Python. Performing queries in PostgreSQL is done by using single or multiple function calls (Krosing, Mlodgenski & Roybal 2013, 26).

PostgreSQL is relational database, and it works well with NestJS and that is the main reason why it was the database of choice for this project. Also, it has an excellent development platform called "pgAdmin", which is easy to use and comes with the PostgreSQL package. Alternatives includes MySQL, Microsoft SQL, SQLite and MongoDB.

4 Frontend Development

A web page has frontend, and sometimes, a backend development process in it. Frontend development is also referred as client-side of the application and it deals with the visual aspects of a web page. A user visiting a website and interacts with the web application is made possible because of the frontend technologies. Some of user interaction elements includes forms, textbox, and buttons (Sharma 2022, 176).

Frontend development can be further categorized to browser-based or mobile. It is important to understand that codes that are written by developers are rendered to the users thru the help of browsers. Popular browsers include Google Chrome and Mac Safari. Stacks, or technologies used in web development, that are used in browser-based development are HTML, CSS and JavaScript (Sharma 2022, 176-177). Figure 2.1 provides summary of web page stacks.

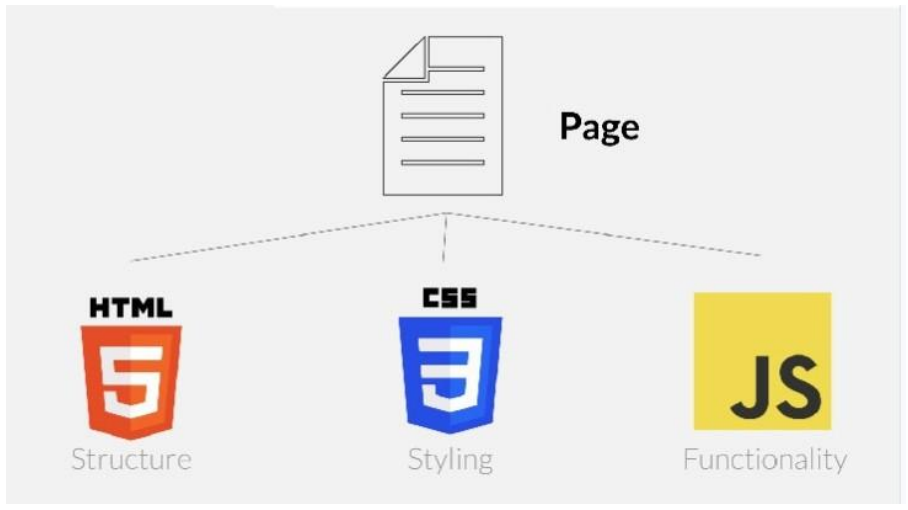


Figure 2.1 HTML, CSS, JavaScript (Kumar 2017)

Mobile development, on the other hand, can be further divided into two types, the native and hybrid apps. Native apps are specifically developed for a specific platform like Mac's IOS and Google's Android. Hybrid mobile applications are usually cross-platform, which means it is non-specific and can be accessed by different platforms. For native apps, Java and Objective C are two of the most popular stacks. For hybrid apps, common programming languages are HTML, CSS, JavaScript, React Native and Ionic (Sharma 2022, 177-178).

4.1 HTML

HyperText Markup Language or HTML is a building block of any web application. HTML is not a programming language, but rather a markup language, and is responsible in web page's description (Northwood 2018, 104).

HTML elements are at the core of HTML pages. Elements or tags are designated with arrow brackets, see figure 2.2. Opening and closing tags are used when elements contain texts. Some common HTML elements are div(division), h1(heading) and p(paragraph) tags (Northwood 2018, 104).

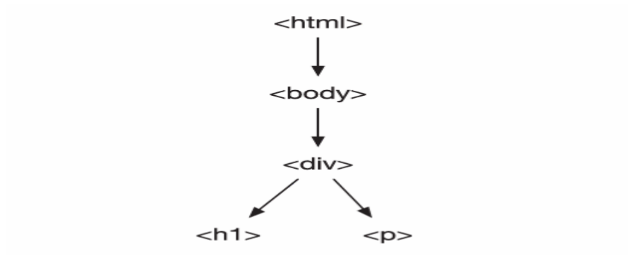


Figure 2.2 HTML structured as a tree (Northwood 2018, 107)

Web browsers are responsible for displaying HTML in a way user can visually relates. Aside from expressing structure of the documents, HTML tags can also be utilized in other manner like search engine optimization (SEO) and accessibility tools (Northwood 2018, 105).

HTML is the hands down, markup language of choice for modern web applications. It is lightweight, easy to understand, open source and works well with other stacks. Alternatives to HTML includes Haml, Markdown and Slim.

4.2 CSS

Cascading Style Sheets or CSS works in conjunction with HTML and is responsible in controlling the presentation of the HTML elements within the document. CSS was developed by Worldwide Consortium (W3C) with the aim of manipulating the document markup. Before CSS was introduced, HTML have style attributes and style-specific element tags were used to control presentation of documents. W3C was able to come up with a solution that allowed separation of structure (HTML) and presentation (CSS) facet of web page (McGrath 2020, 236-237).

CSS allows developers and users to style their HTML or XML application by including rules (Lawson and Sharp 2011, 10). There are two parts of CSS rule, the selector, and the declaration, see figure 2.3. CSS selector are HTML element/s selected for styling, while CSS declaration defines how the selected elements should be styled (Duckett and Duckett 2011, 244).

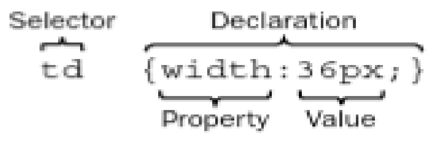
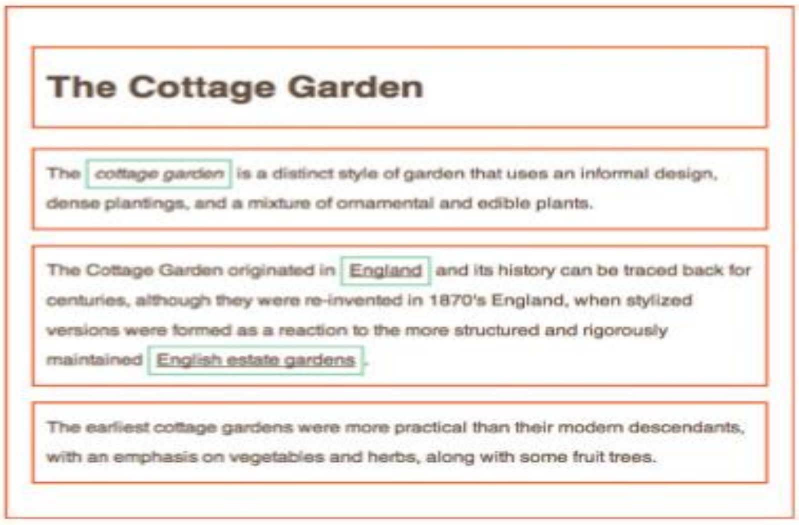


Figure 2.3 CSS selector and declaration (Ducket 2011, 244)

Another important CSS concept is the CSS box model, which means there is an invisible box around each HTML elements. CSS rules can be set on the HTML invisible box, and this rules control how elements are presented in the document (Ducket 2011, 229).


The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

Figure 2.4 An example of HTML elements in their own box (Ducket 2011, 230)

CSS is the clear choice for styling web applications. It is by far the most used language for controlling how web pages look and it can also add animations. It works very well with HTML and JavaScript, which are also utilized in this project. There are very few alternatives for CSS, like eXtensible Stylesheet Language (XSL) but there are pre-processors and extensions that can extend CSS, like Leaner Style Sheets (LESS) and Syntactically Awesome Stylesheet (SASS). These preprocessors and CSS extensions have very handy features that are then compiled to CSS.

4.3 JavaScript

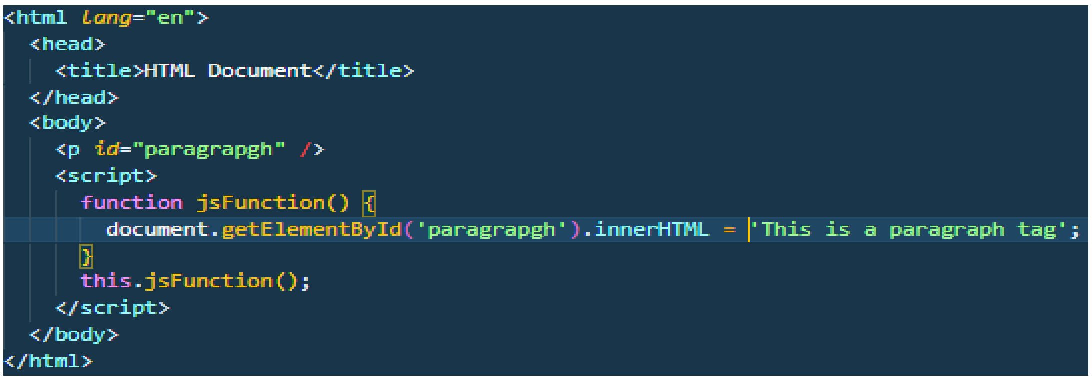
In the early days of the Web, it can only display static contents with the help of HTML. Hyperlinks allows HTML documents to be linked with other web pages, but aside from that, web’s capabilities are very limited. Webmasters, driven by their desire to add more features like form validation, came up with different solutions to improve web experience and one of it is JavaScript. Because of its simplicity and how it can be easily incorporated to HTML, JavaScript was embraced by many developers. Ultimately, ECMA (European Computer Manufacturers Association) developed the foundation of JavaScript programming language touted ECMA-262 (Stefanov and Sharma 2013, 26-27).

JavaScript is a computer language, and it allows developers to give sequence of instructions that computer understands and is task to do. These instructions are called code and it runs synchronously, which means that it runs one operation at a time. JavaScript operations include asking inputs from the user, ability to display and change text content in the document, and even manipulating or moving an image (Wilton and McPeak 2009, 1-2).

According to Mulraj (2021, 27) web development is divided into three technologies, HTML as the foundation, CSS as the style and JavaScript, which is optional but highly compelling. Without JavaScript, web page will have no user interaction. In addition, JavaScript has some awesome features:

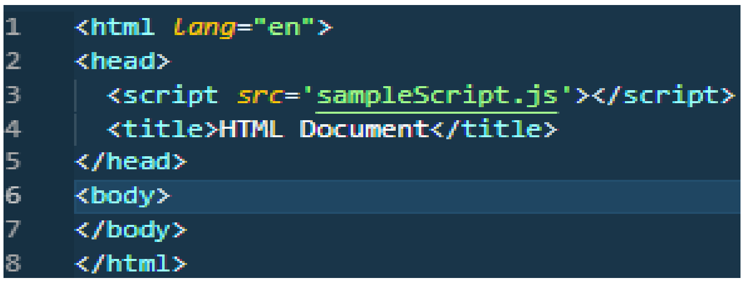
- Responding to event handlers, like click and keyboard press.
- Included in majority of web browsers, including Google Chrome, Mozilla Firefox and Mac's Safari.
- Ability to fetch data from servers.

JavaScript can be added to HTML document by either using <script> tag inside the head or body of the HTML (see figure 2.5) or adding the source of an external script file (see figure 2.6) (Ranjan 2020, 88).



```
<html lang="en">
  <head>
    <title>HTML Document</title>
  </head>
  <body>
    <p id="paragrapgh" />
    <script>
      function jsFunction() {
        document.getElementById('paragrapgh').innerHTML = 'This is a paragraph tag';
      }
      this.jsFunction();
    </script>
  </body>
</html>
```

Figure 2.5 JavaScript embedded in the body of an HTML document



```
<html lang="en">
<head>
  <script src='sampleScript.js'></script>
  <title>HTML Document</title>
</head>
<body>
</body>
</html>
```

Figure 2.6 JavaScript from external file (Ranjan 2020, 88).

JavaScript is chosen in this project mainly because it is recommended for smaller applications, against the more powerful but more code demanding TypeScript. It is also faster to develop web applications using JavaScript, which fits the project's timeline. Nevertheless, it is recommended to consider using TypeScript in the future if the application gets more complex.

4.4 React JS

React is first released as an open-source library in 2013 by Facebook. According to React's official web site, it is "A JavaScript library for building user interfaces". JavaScript libraries can be leveraged by developers as starting from scratch is not always necessary. These libraries are codes that are written by other developers and can be used through script (Wandschneider 2013, 497).

React is not a large framework, neither a full-stack solution, it rather deals with user interfaces and is considered as the view section of an application (see figure 2.7). In a web page, data can be generated and passed to React Component, which then display it to the page. The advantage of React with other JavaScript frameworks is that it is simple and easy to learn (Boduch, Derks and Sakhniuk 2022, 4-5).

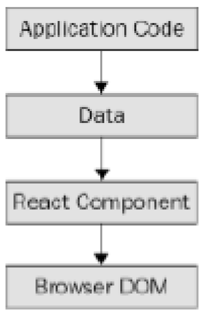
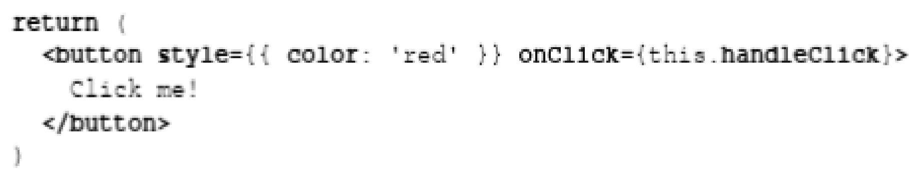


Figure 2.7 Layers of React application (Boduch, Derks and Sakhniuk 2022, 4)

Combining HTML and JavaScript have been one of React's main features and it was against a long-standing practice that HTML and JavaScript should be created separately. Since HTML and JavaScript work closely together, React is able to effectively integrate both in the same process. Rendering an HTML button with a JavaScript function can have few lines of code, as illustrated in figure 2.8. using React components (Roldan 2021, 14-15).



```
return (
  <button style={{ color: 'red' }} onClick={this.handleClick}>
    Click me!
  </button>
)
```

Figure 2.8 React component rendering an HTML button with JavaScript function (Roldan 2021, 15).

React is the library of choice for this project for several reasons. It makes use of the virtual DOM, which makes updating the content of the page extremely fast and efficient. It is well maintained and enjoys a broad community support which means there are many other developers that can help if issues occur. There is also no sign that React is slowing down, especially since it is backed by Facebook. Alternatives are Svelte, Solid.js, Angular JS and Vue JS.

5 Other Development Tools

Aside from the technological requirements, the project employs other tools that are needed for a successful web application implementation.

5.1 GitHub

GitHub is by far the most popular platform for repository management. Repositories in GitHub contains all the files, and it also keeps the different versions of code of the application. It delivers an ideal environment for web applications and web development team with multiple members can leverage the platform as well (Chandrasekara and Herath 2021, 1).

GitHub is chosen as it works well with the deployment platform, Heroku. It is also easy to use and has an excellent user interface. There are large community of developers working with GitHub that can be of great help when issues arise. Alternatives for GitHub are Bitbucket, TaraVault and SourceForge.

5.2 Heroku

Heroku is an application delivery platform that is built on Amazon EC2 (Amazon Elastic Cloud). Heroku offers variety of options that includes managing its own stacks. Teams can focus on the developing the web application more and let Heroku perform deployment related tasks, like security and availability of the application online. Heroku is also a good choice for agile methodologies, as it allows continuous web application deployment and integrates well with teams that has multiple members (Kemp and Gyger 2013, 3).

There are many options for cloud services, but Heroku was preferred because of how the project's other technologies adopt to Heroku. The project's use case does not need other much powerful cloud service, rather it only needs platform for building and running both the frontend and backend application, which Heroku can provide. It also has an excellent user-interface, and it is not overwhelming. Other options are Google Cloud, Microsoft Azure and Amazon Web Service.

5.3 Docker

Docker was started by Solomon Hykes to take advantage of containerization. Containerization allows developers who are working on a project, runs its dependencies uniformly. Performing tasks like container creation, container management and container administration is the feature of Docker (Ghosh 2020, 18, 22).

Docker is chosen for this project mainly because of how easy it can integrate PostgreSQL in Next JS. Also, it can help developers to have the same environment running in their local machines. Lastly, running services using Docker is getting more common. Substitute for Docker are Buildah, BuildKit and Podman.

6 Project Implementation

This section describes the whole process of developing the proof-of-concept web application using the application and project development methodologies.

6.1 Project Requirement

6.1.1 Requirements from the Client

The communication with the K and J Medical Supply was done online since the company office is in the Philippines. During the first discussion with the client, the specifics of the project were tackled. The researcher presented the proof-of-concept web application to manage their inventory. High level technology was explained to the client and the rationale behind it. The client was able to come up with the wish list for the proposed inventory system and are summarized below:

- The inventory items are safely stored. Only authorized users can access the items.
- Users must be able to manage the inventory, like adding, editing, and updating items.
- If possible, multiple users can access the inventory at the same time, as current company setup does not allow it.
- Ease of usage, as not all potential users are tech savvy.

- Potential for future development. Opportunities for more features, like showing inventory to customers, are to be considered.

Other basic information were also collected during the course of the project implementation, these includes the properties that should be included in the items, what are the preferred layouts and colors of the web page, and what functionalities they would like to be present on the pages.

6.1.2 Application Requirement

Taking into consideration the requirements identified by the client, the researcher determined that two applications should be created, one on the backend and one on the frontend.

For the backend application, the requirements are:

- Users' entity that will manage the users of the web application.
- Items' entity that will manage the items of the web application.
- Web APIs for CRUD operation for both users and items entities.
- Database that will save both entities.

For the frontend application, the requirements are:

- A login page where registered users can input their credentials
- Items page where registered users can perform

6.2 Project Design

Components of a web application can be illustrated using the web application architecture diagram. The diagram can also give a visual overview of how frameworks interact with each other (Prakash, 2021).

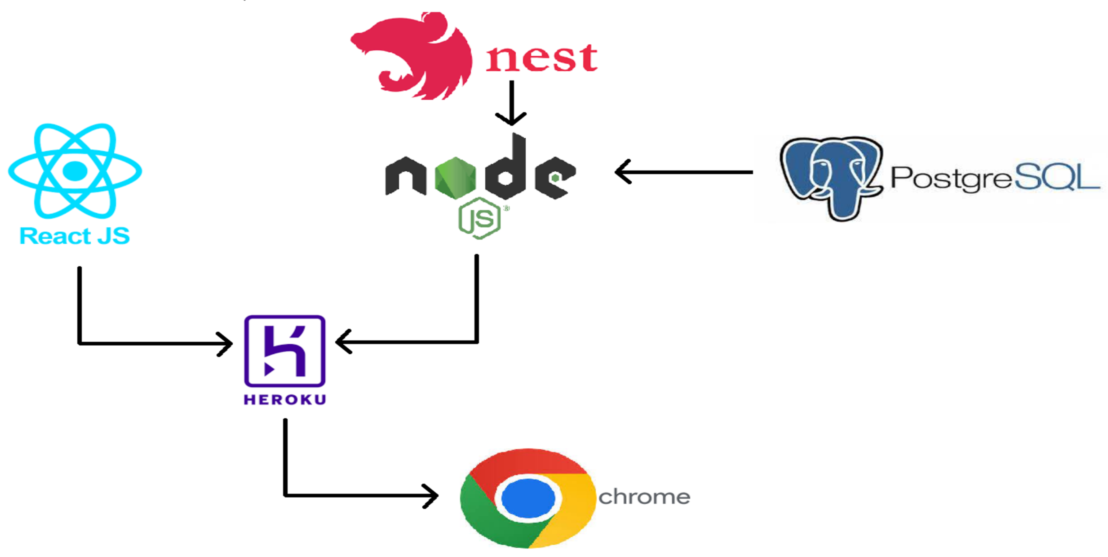
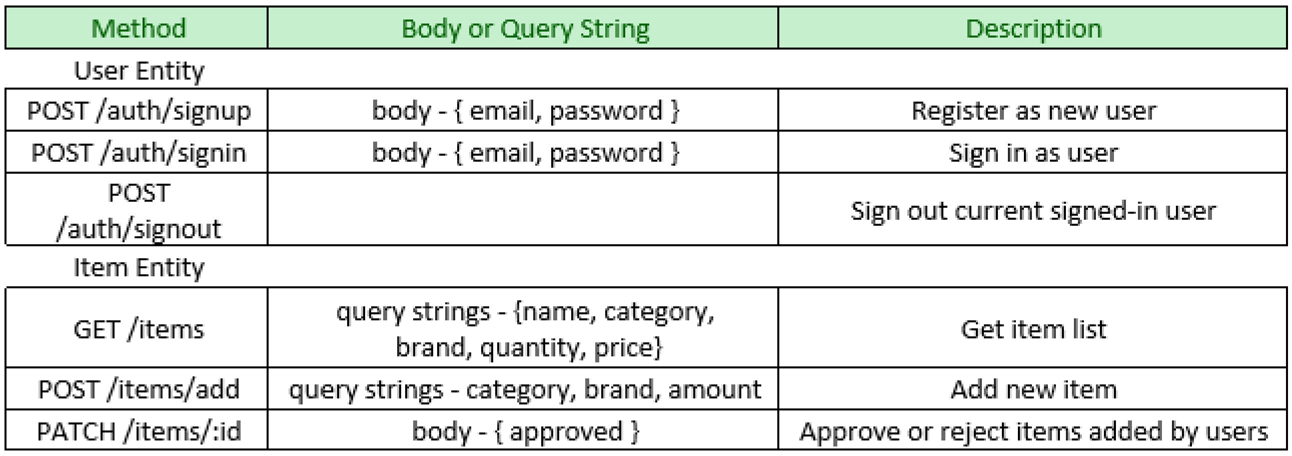


Figure 3.1 Web application architecture of the project

Figure 3.1 shows the high-level technological overview of the proof-of-concept web application. NestJS, a NodeJS framework, will be the main technology in the backend. It is responsible for implementing the CRUD operations and their respective APIs. Figure 3.2 shows the API design of the project and includes both the users and items entity.



Method	Body or Query String	Description
User Entity		
POST /auth/signup	body - { email, password }	Register as new user
POST /auth/signin	body - { email, password }	Sign in as user
POST /auth/signout		Sign out current signed-in user
Item Entity		
GET /items	query strings - {name, category, brand, quantity, price}	Get item list
POST /items/add	query strings - category, brand, amount	Add new item
PATCH /items/:id	body - { approved }	Approve or reject items added by users

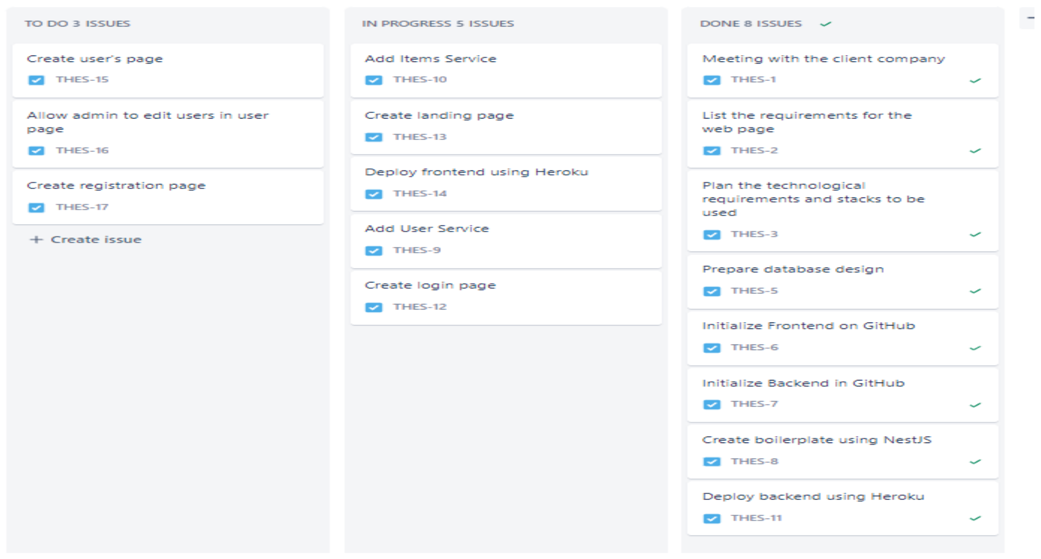
Figure 3.2 API design

In the frontend, React will be utilized extensively. Data from the database will be accessed by the frontend and will be shown to the users. Items can then be added, edited, or removed by sending request to the created function in the backend.

Both backend and frontend will be hosted using Heroku. And users will be able to access the site by visiting the URL with the help of their browsers, like Google Chrome.

6.3 Project Kanban Board

The project development started with creating issues in the Jira's Kanban board. It has good task visualization and gives better option to track and record project's progress. A sample of the actual Kanban board is illustrated on figure 3.3.



TO DO 3 ISSUES

- Create user's page — THES-15
- Allow admin to edit users in user page — THES-16
- Create registration page — THES-17
- + Create issue

IN PROGRESS 5 ISSUES

- Add Items Service — THES-10
- Create landing page — THES-13
- Deploy frontend using Heroku — THES-14
- Add User Service — THES-9
- Create login page — THES-12

DONE 8 ISSUES

- Meeting with the client company — THES-1
- List the requirements for the web page — THES-2
- Plan the technological requirements and stacks to be used — THES-3
- Prepare database design — THES-5
- Initialize Frontend on GitHub — THES-6
- Initialize Backend in GitHub — THES-7
- Create boilerplate using NestJS — THES-8
- Deploy backend using Heroku — THES-11

Figure 3.3 Project's Kanban Board

6.4 Backend Implementation

The initial task before creating backend functionalities was to get the requirements of the web application. The data that will be saved in the database are divided into two entities: item and user. Each entity will have different properties as illustrated in figure 3.4. The relationship that exists between user to item is one-to-many, meaning, a user can have many items related to them, as they are responsible for adding new items.

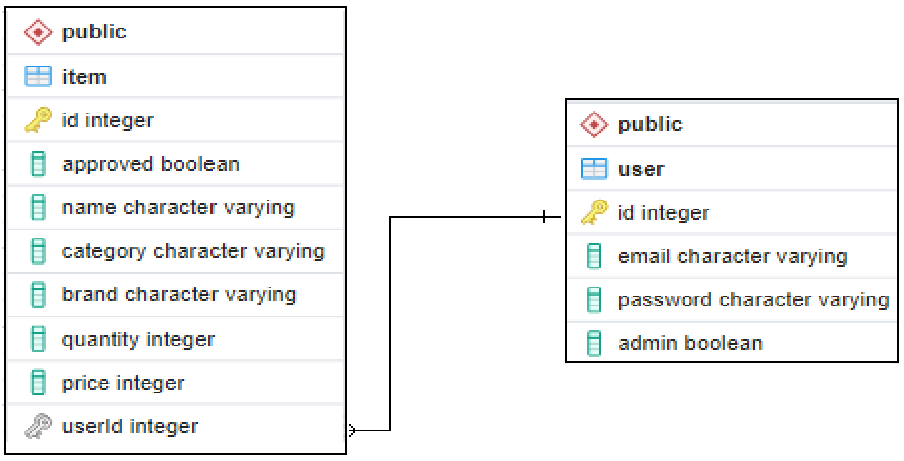


Figure 3.4 Entity relationship diagram

6.4.1 Setting up project using Node JS and NestJS

NestJS requires that Node JS is already installed in the local machine. The first step is to download Node JS in their official download page at https://nodejs.org/en/download/ then execute the downloaded file. NestJS provides an easy way to start a new project using NestJS CLI or command-line interface. The first command is to install NestJS globally and creating a new project:

```
$ npm i -g @nestjs/cli
$ nest new db-med
```

Furthermore, some Node JS packages need to be installed to help in the project. The first package is TypeORM, which gives different features to manage the database. The second is PostgreSQL, which will be the relational database.

```
$ npm install @nestjs/typeorm typeorm pg
```

6.4.2 Running Project on Docker

Like Node, Docker should be installed on the local machine before running. Postgres container is added to the docker and ormconfig.js file needs to be updated to use the created Postgres container.

6.4.3 Initializing GitHub Repository

GitHub repository for the web project is important to keep track of the progress and to efficiently back-up the codes. GitHub should be installed in the local machine to access its features. Once installed, the project repository can be created using the command line and GitHub page (see figure 3.5). After adding the repository, changes can now be committed to GitHub and it is advisable to commit as frequent as possible.

```
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ mikel-js/db-medSupply.git
git push -u origin main
```

Figure 3.5 Code to initialize project’s Git repository

6.4.4 API Design

The backend of the web application will have different routes for accessing data. Each entity will have their own parameters that corresponds to different function. Creating User Entity, Controller, Service and Repository.

Creating user entity meant that it would have a corresponding user table in the database. User entity will have columns, relations, and a primary column. Primary column is important in any entity as it is the unique identifier of a row in the table. In the web application, user primary column is required in the item entity, to track who added the item in the database. NestJS uses decorators to change the behavior of a property or to add additional data to a class. In the application, user entity is added by marking a class with @Entity decorator, as illustrated in figure 3.6.

```
@Entity()
export class User {
  @PrimaryGeneratedColumn()
  id: number;
  @Column()
  email: string;
}
```

Figure 3.6 User entity creation

Next is to create a user controller, which will be responsible for receiving request and route it to the respective function. Creating user controller is like creating the user entity, only difference is marking the class with the @Controller decorator, instead of @Entity. Multiple routes are available in the user controller like auth/signin and auth/signout. The type of request is also marked at the beginning of the route and NestJS provides an easy way to define it using route decorators (see figure 3.7).

```
@Controller('auth')
@Serialize(UserDto)
```

```
export class UsersController {
  constructor(
    private usersService: UsersService,
    private authService: AuthService,
  ) {}
  @Post('/signin')
  async signin(@Body() body: CreateUserDto, @Session() session: any) {
    const user = await this.authService.signin(body.email, body.password);
    session.userId = user.id;
    return user;
  }
  @Post('/signout')
  signOut(@Session() session: any) {
    session.userId = null;
  }
```

Figure 3.7 Defining post request route using @Post decorator

Next, user service and repository are added. Both classes are managed in the same file. The user service will be responsible for logical functions, and it will fetch data from the user repository. User repository interacts directly with the database, and use TypeORM, as it has built-in functions that make it easy connecting with the database. Declaring a class as service is straight-forward as the entity and controller. An @Injectable decorator should be placed, and repository is added by injecting it to the constructor function of the user service, as illustrated in figure 3.8.

```
@Injectable()
export class UsersService {
  constructor(@InjectRepository(User) private repo: Repository<User>) {}

  create(email: string, password: string) {
    const user = this.repo.create({ email, password });

    return this.repo.save(user);
  }
```

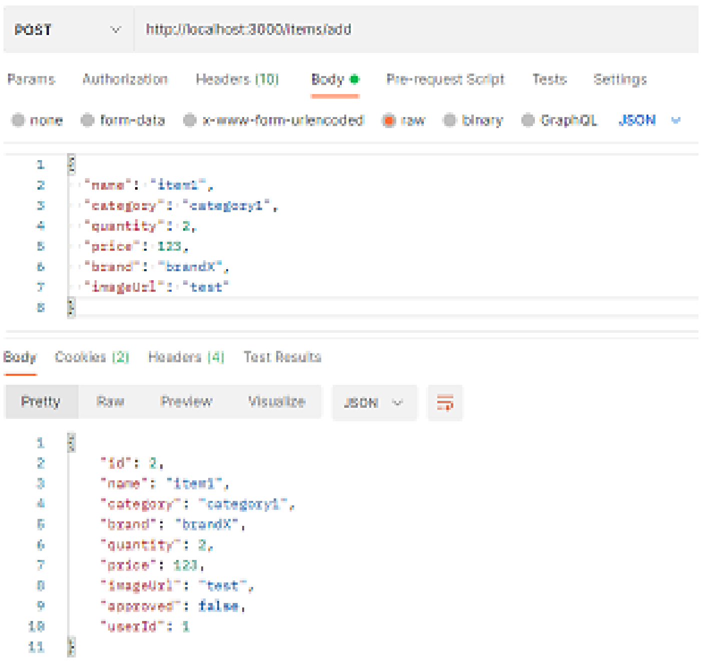
Figure 3.8 Implementing user service and user repository

6.4.5 Creating Item Entity, Controller, Service and Repository

Item entity is created following the same procedure as the user entity. Class is mark with decorators to define them as entity or controller. @Injectable is use in items service and add constructor function to connect to the database using the items repository.

6.4.6 Testing Routes

Routes can be tested to confirm it is working properly by using Postman. Requests can be sent to the API routes created. The example in figure 3.9 shows a post request to add an item using the route /items/add. The request submitted successfully and created a new item.



```
http://localhost:3000/items/add
```

```
{
  "name": "item1",
  "category": "category1",
  "quantity": 2,
  "price": 123,
  "brand": "brandX",
  "imageUrl": "test"
}
```

```
{
    "id": 2,
    "name": "item1",
    "category": "category1",
    "brand": "brandX",
    "quantity": 2,
    "price": 123,
    "imageUrl": "test",
    "approved": false,
    "userId": 1
}
```

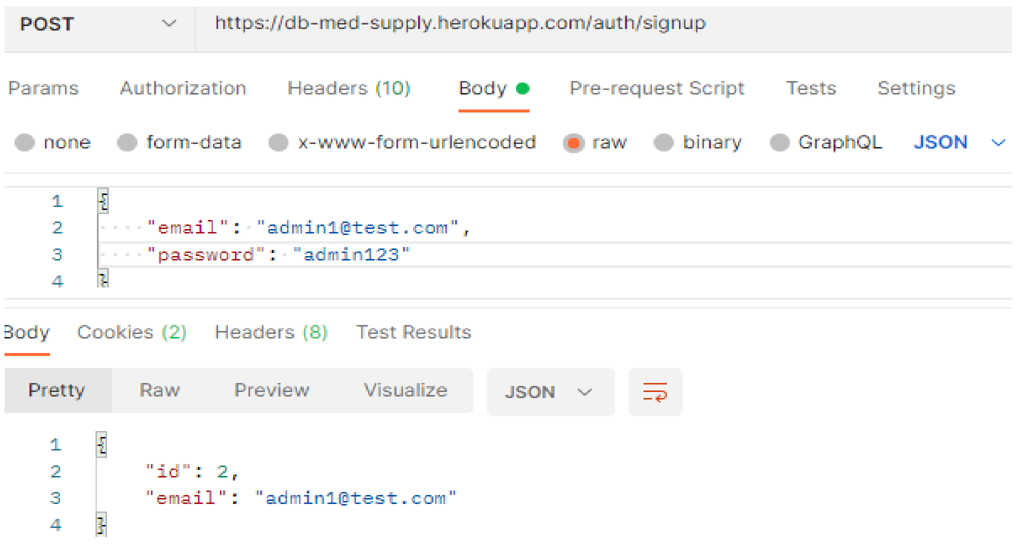
Figure 3.9 POST request to add new item

6.4.7 Deploying Database In Heroku

Finally, web application can be deployed in Heroku so that it can be accessed outside local server. First step is to create a new app in Heroku site, https://dashboard.heroku.com/apps. After creating the new app, run the command illustrated in figure 4.1 in the terminal to connect the GitHub repository to Heroku app that was created. A new user can now be created using Postman, but now sending it to the newly created app in Heroku, instead of the local server (figure 4.2).

```
heroku git:remote -a test-db-med-supply
git add .
git commit -m "web deployment"
git push heroku main
```

Figure 4.1 Web application deployment in Heroku



```
https://db-med-supply.herokuapp.com/auth/signup
```

```
{
    "email": "admin1@test.com",
    "password": "admin123"
}
```

```
{
    "id": 2,
    "email": "admin1@test.com"
}
```

Figure 4.2 Creating a new user using the deployed database in Heroku

6.5 Frontend Implementation

After setting up the backend, the frontend can connect to the APIs created. The frontend will be responsible for displaying the items that are saved in the database. Before accessing these items, users must be able to log in first using a registered email and password.

6.5.1 Starting the Project Using React

React is by far the main frontend technology used in the project. React has create-react-app, which is used to set up new environment and it includes packages that are necessary to start the project immediately. Some other packages that are needed are styled-components, axios and react router dom. Packages are easily added by running the codes illustrated in figure 5.1.

```
npx create-react-app med-frontend
npm install axios styled-component react-router-dom
```

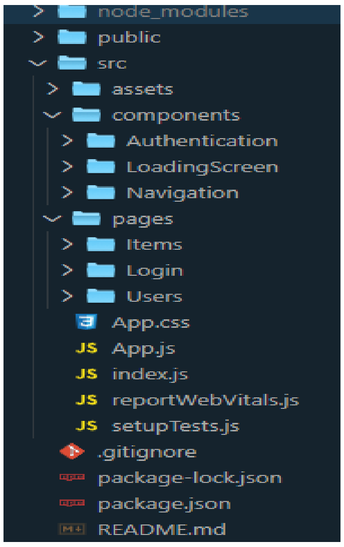
Figure 5.1 Starting React application and installing other packages

6.5.2 Creating GitHub Repository

Same with the backend, the frontend needs to have its own GitHub repository. For this web project, the name of the repository is med-supply.

6.5.3 Folder Structure

The web application will have different folders for pages and components. All the relevant folders and files will be inside the source folder (src), these include the assets, components, pages and the main App.js and index.js (see figure 5.2). Components folders have JavaScript files which are re-usable and are mostly used inside pages. Almost all the components have actual CSS inside, which is possible using the styled-components package.



```
> node_modules
> public
v src
  > assets
  v components
    > Authentication
    > LoadingScreen
    > Navigation
  v pages
    > Items
    > Login
    > Users
    App.css
    App.js
    index.js
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

Figure 5.2 Web application's folder structure

6.5.4 Managing Routing Inside the Main JavaScript File

In the main JavaScript file, where the App.js is, routing is managed. In the project, there would be two routes, the login, and the items page. The two routes are divided into unprotected or protected route. Unprotected route is accessible to anyone and in the project, it is the login page. The protected route is only for the admins of the site and in the project, it is the items page. The web application made use of the react-router-dom which is a package for routing pages.

6.5.5 Adding Login Page

The landing page for the web application is the login page and it has an input field for email and password, as illustrated in figure 5.3. Anyone can visit the page and its only purpose is for the user to enter their credentials to access other protected pages. Once the users entered a valid email and password, a request will be sent to the API that was created in the backend implementation. Axios, a package for making http requests, is used to send requests to check if the user credentials are valid. Once the request succeeded, it can either redirect to the items page, if credentials are verified, or stays in login page, if credentials are not valid.

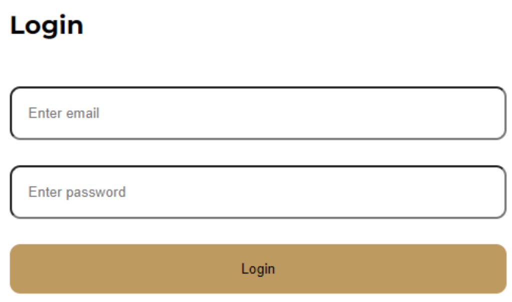


Figure 5.3 Actual login page

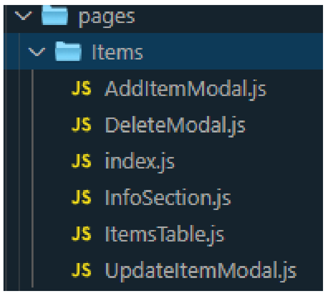
```
const onLoginSubmit = async (e) => {
   if (!email || !password) {
     alert('Email and password are required!');
     return;
   }
   const reqBody = {
     email,
     password,
   };
   try {
     const res = await axios({
       method: 'post',
       url: loginUrl,
       data: reqBody,
     });
     if (res) {
       if (!res.data.admin) {
         alert('You do not have permission!');
         return;
       }
       localStorage.setItem('userId', res.data.id);
       navigate('/items');
```

```
        }
    } catch (e) {
        alert('Something went wrong');
    }
  };
```

Figure 5.4 Code for sending request to the backend to check user's credential.

6.5.6 Adding Items Page

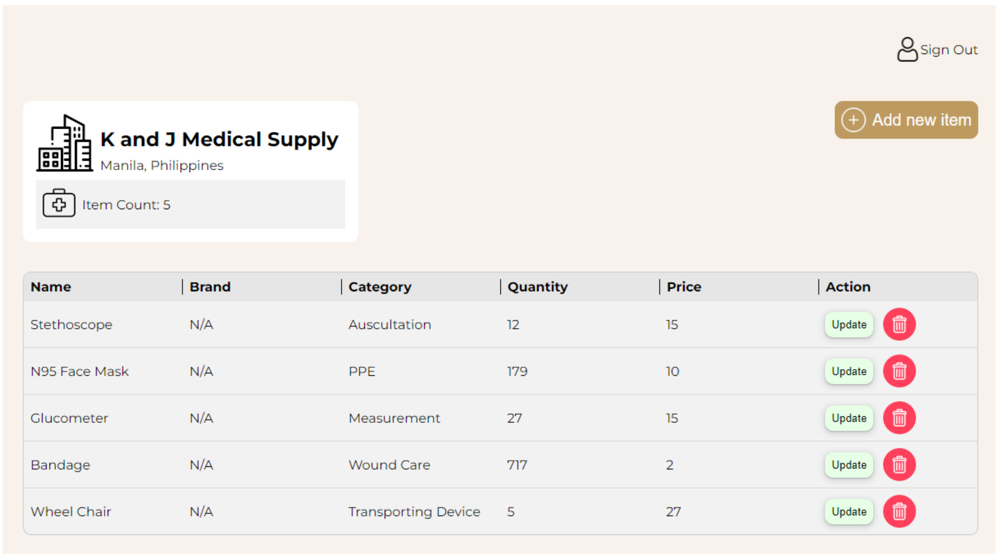
Items page will be responsible for displaying the items in a table format. It will also have functionalities to add, update and delete items by making full use of the APIs created in the backend. The components are divided into different JavaScript files for ease of access and better maintenance.



```
pages
  Items
    AddItemModal.js
    DeleteModal.js
    index.js
    InfoSection.js
    ItemsTable.js
    UpdateItemModal.js
```

Figure 5.5 Items folder with all the components used in the page

User can interact with the UI by clicking on the buttons that corresponds to the action they want. Requests will then be requested to the API in the backend and will respond with the requested data. The page will update if new item is added, an item has been updated or if item is deleted. Figure 26 shows the actual item page.



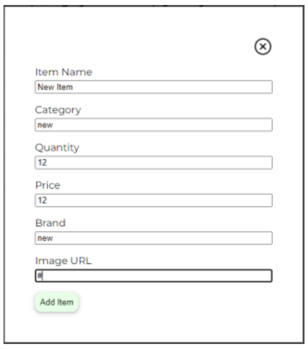
K and J Medical Supply
Manila, Philippines
Item Count: 5

Name	Brand	Category	Quantity	Price	Action
Stethoscope	N/A	Auscultation	12	15	Update
N95 Face Mask	N/A	PPE	179	10	Update
Glucometer	N/A	Measurement	27	15	Update
Bandage	N/A	Wound Care	717	2	Update
Wheel Chair	N/A	Transporting Device	5	27	Update

Figure 5.6 Items page

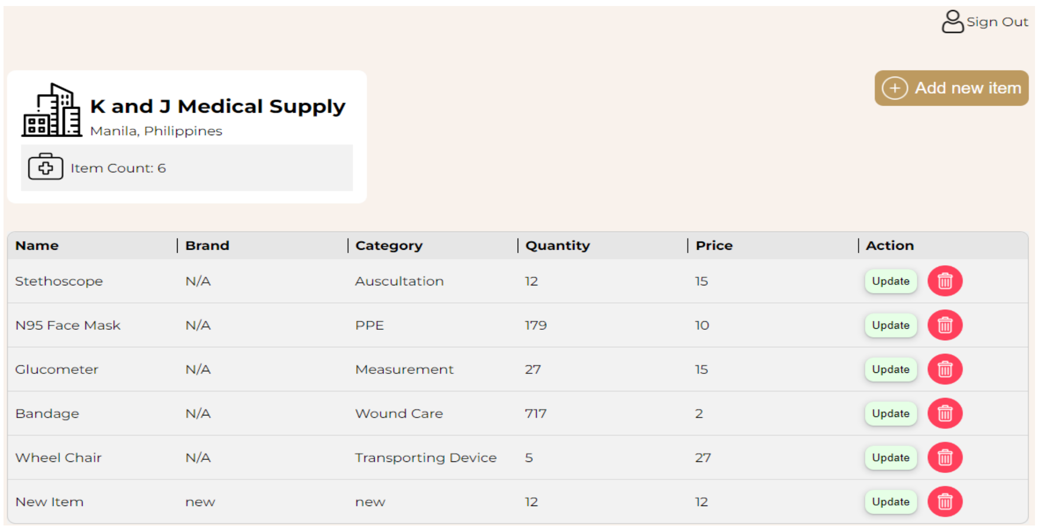
6.5.7 Testing Functionalities

Testing if the web application is successfully sending requests to the backend can be confirmed by performing the CRUD operations in the items page. New items can be created by clicking the new item button and a create new item modal is displayed (see figure 5.7), where users can input the item's details. When the new item is submitted, the table is updated and the newly created items is listed on the items table, as illustrated in figure 18.



Item Name: New Item
Category: new
Quantity: 12
Price: 12
Brand: new
Image URL: #

Figure 5.7 Create new item modal


Sign Out

K and J Medical Supply
Manila, Philippines

Item Count: 6

Add new item

Name	Brand	Category	Quantity	Price	Action
Stethoscope	N/A	Auscultation	12	15	Update
N95 Face Mask	N/A	PPE	179	10	Update
Glucometer	N/A	Measurement	27	15	Update
Bandage	N/A	Wound Care	717	2	Update
Wheel Chair	N/A	Transporting Device	5	27	Update
New Item	new	new	12	12	Update

Figure 5.8 New item listed on the table

6.5.8 Deploying Web App in Heroku

The frontend web application is then deployed through Heroku to be accessible in the web. The first step is to create a new application in the Heroku website, then filling-up what would be the app’s name. Heroku will give instructions on how to proceed. When all steps are made, the page should be visible in the web. Since the name of this app is med-supply-frontend, it automatically has the URL https://med-supply-frontend.herokuapp.com/.

6.6 Project Maintenance

The project is created with maintainability in mind. Folder structures are organized properly, to help developers in the future find the file easily. In-line documentations are written, as illustrated in figure 5.9, to assist developers about the code’s rationale.

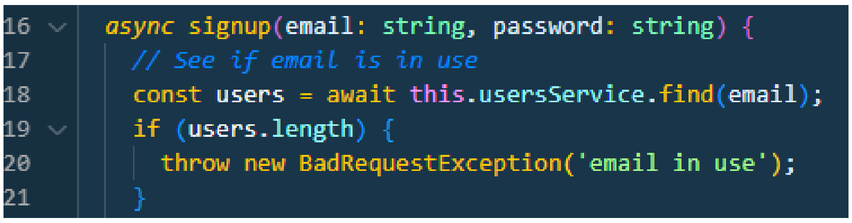

```
async signup(email: string, password: string) {
  // See if email is in use
  const users = await this.usersService.find(email);
  if (users.length) {
    throw new BadRequestException('email in use');
  }
```

Figure 5.9 Sample function with in-line documentation

Another practice that was considered is the reusability of the components and functions. It is ideal to always employ reusability whenever possible, this will help when future features need a similar function, like the users table, which can make use of the existing items table.

Lastly, the project used as less dependencies to other packages as possible. Some dependencies can be problematic, especially if future support is not maintained. Less dependencies meant less risk for these kinds of issues.

7 Conclusion

Using the application development and project management methodologies, in combination with different web development stacks, the project validated the feasibility of using modern web technologies in inventory management of K and J Medical Supply.

There are two web applications built, in accordance with the objectives. The first application, which was referred as the backend part, was built and it ensures that the web application performs properly by having different functions to manage users and items in the inventory. The second application, which is the frontend part, works simultaneously with the backend and allows users to visualize the items in the database, in a coherent and systematic manner.

7.1 Technology Evaluation

The project was implemented within few weeks and the technologies used were put into test. Usually, full-stack application takes months, if not years to complete, but since the product is straight-forward and the requirements are clearly stated, the application development was made in much lesser duration.

It can also be attributed to the choice of technologies. The main requirement during planning is how well the technologies can be used together, and the quick and effective results proved that these technologies could provide efficient and operational application.

7.2 Challenges Encountered

The main challenge faced during the project is the scope. Originally, the scope includes a company web page, and a client web page. The client web page was supposed to provide the current stocks of medical items to potential customers. This can improve the daily operation of the company by first, reducing the number of phone inquiries by customers and second, by lessening the amount of visiting customer to inquire if products are available. This issue was resolved by re-evaluating the scope and adding the client web page in the future development.

The second challenge is time constraints. Development of a web application takes a lot of time. To address the issue, a minimum viable product was defined. It is not a full application, but rather an earlier version of the product, which can be tested by the client.

Time difference, availability of both stakeholders, and location between the researcher and the client company also became an issue but were solved by using online channels of communication and thorough planning.

The last challenge is the backend application. The researcher has less experience with backend development, and it took some time to finish the backend application. Applying the frameworks also helped with time management. Having knowledge on the phases of application and project management development and using it first-hand allows the researcher to work a lot more efficiently.

7.3 Future Development

The project currently constructed has a lot of future development opportunities. The admin web page can have a user's page, like items page, which will list all the registered users of the company. The admin of the company can perform tasks like updating or deleting registered users.

The client web page can also leverage the benefits of having an SEO, or search engine optimization. SEO is promoting products thru search engines, like Google Search. The company can have an effective web presence and reach targeted people thru SEO and can increase their customers significantly, at an affordable pricing, since they are usually reasonably priced.

Further improvement can be done to the existing items page. A sorter can be added based on different criteria, like sort by name, sort by quantity or sort by brand. Search functionalities will also be beneficial. It will work as an input field and users can easily find items based on the search term. It is especially convenient when the need to update a particular item is needed and it is expected to have hundreds of product types on the database, and each product could have different brand as well.

In addition, an image repository can be added. Images would be helpful in some cases, as some medical instruments are not common, and can be confusing. Currently, the project has no repository to save the images. Another platform can also be considered to host the image.

Finally, the database can also be used to build the client web page that will be accessible to customers. The client web page can be used to allow customers to view the inventory status of the shop. This can assist the company as it will significantly lower the amount of phone calls from customers inquiring if items are available and limit the number of customers visiting the physical store to inquire personally.

References

Printed Sources

Highsmith, J. 2010. Agile Project Management Creating Innovative Products, Second Edition. Pearson Education, Inc.

Lawson, B & Sharp, R. 2011. Introducing HTML5. New Riders

Robbins, J, 2012. Learning Web Design, Fourth Edition. O'Reilly Media, Inc.

Wandschneider, M. 2013. Learning Node.js: a hands-on guide to building web applications in JavaScript. Addison-Wesley

Electronic Sources

Boduch, A., Derks, R. & Sakhniuk, M. 2022. React and React Native: Build Cross-Platform JavaScript Applications with Native Power for the Web, Desktop, and Mobile. 4th Edition. Packt Publishing, Limited. Accessed 15.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=7007141

Chandrasekara, C and Herath, P. 2021. Hands-On GitHub Actions : Implement CI/CD with GitHub Action Workflows for Your Applications. Apress L. P. Accessed 13.11.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=6508456

Cooke, J. 2012. Everything You Want to Know about Agile: How to Get Agile Results in a Less-Than-agile Organization. IT Governance Ltd. Accessed 15.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=918799

Duckett, J. & Duckett, J. 2011. Beginning HTML, XHTML, CSS, and JavaScript. Wiley. Accessed 15.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=479886

Duckett, J. 2011. HTML and CSS: Design and Build Websites. John Wiley & Sons Incorporated. Accessed 15.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=817871

Flanding, J., Grabman, G. & Cox, S. 2018. The Technology Takers: Leading Change in the Digital Era. Emerald Publishing Limited. Accessed 08.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=5589432

Ghosh, S. 2020. Docker Demystified. BPB Publications. Accessed 20.11.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=6891909

Herron, D. 2020. Node.js Web Development : Server-Side Web Development Made Easy with Node 14 Using Practical Examples. Packt Publishing, Limited. Accessed 11.11.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=6276150

Hughes, T. 2016. SAS Data Analytic Development : Dimensions of Software Quality. John Wiley & Sons, Incorporated. Accessed 18.11.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=4658596

Kemp, C. and Gyger, B. 2013. Professional Heroku Programming. John Wiley & Sons, Incorporated. Accessed 13.11.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=1119184

Krosing, H., Mlodgenski, J. and Herath, P. 2013. PostgreSQL Server Programming. Packt Publishing, Limited. Accessed 12.11.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=1220931

Marcolongo, M. 2017. Academic Entrepreneurship: How to Bring Your Scientific Discovery to a Successful Commercial Product. John Wiley & Sons Incorporated. Accessed 08.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=5015545

McGrath, M. 2020. HTML, CSS, and JavaScript in Easy Steps. In Easy Steps Limited. Accessed 10.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=7075502

Mirtalebi, M. 2017. Embedded Systems Architecture for Agile Development: A Layers-Based Model. Apress L.P. Accessed 08.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=5112690

Mohapatra, H. and Rath, K. 2020. Fundamentals of Software Engineering. BPB Publications. Accessed 17.11.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=6891773

Nance, C. 2014. Typescript Essentials. Packt Publishing, Limited. Accessed 15.11.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=1822800

Nehra, M. 2022. 6 Stages of the Agile Development Lifecycle. Accessed 18.11.2022. https://www.decipherzone.com/blog-detail/agile-development-lifecycle

Northwood, C. 2018. The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer. Apress L.P. Accessed 09.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=5601350

Prakash, L. 2021. Web Application Architecture. Accessed 10.10.2022 https://medium.com/geekculture/web-application-architecture-800d3ecd8019

Ranjan, A., Sinha, A. & Battewad, R. 2020. JavaScript for Modern Web Development. BPB Publications. Accessed 14.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=6891907

Roldan, C. 2021. React 17 Design Patterns and Best Practices: Design, Build, and Deploy Production-Ready Web Applications Using Industry-standard Practices. Packt Publishing Limited. Accessed 15.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=6627606

Shah, R. 2021. Decoding JavaScript. BPB Publications. Accessed 14.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=6891890

Sharma, M. 2022. Full Stack Development with MongoDB. BPB Publications. Accessed 09.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=29283631

Stephens, R. 2015. Beginning Software Engineering. John Wiley & Sons Incorporated. Accessed 15.10.2022. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=1895174

React. A JavaScript Library for Building User Interfaces. Accessed 11.10.2022 https://reactjs.org/

The Difference Between HTML, CSS, and JavaScript. Accessed 10.10.2022 https://www.web-development-institute.com/the-difference-between-html-css-and-javascript/