J Clarke Management Report

[Link to Hosted Website](https://dry-journey-02805.herokuapp.com/)

[Link to Website GitHub Repository](https://github.com/manopanashe/assignment)

Introduction

J Clarke is a global retail Company that has over 40 store branches across the world. Its stores are unique in many ways however the Administration for the company keeps logs for the store branches based on their numbers and locations and other various information. The logs are stored into separate CSV files and these files are passed through stores to allow them to update their data. Due to Covid-19, it has become less efficient for the company to have their store detail logs in these separate files and this has also made the data they need less accessible for them. To solve this problem, I developed a proof-of-concept Web-application that moves these processes online.

System Overview

The J Clarke web-application is a management portal for the company. It provides the administration a more efficient way to keep track of their branches during COVID-19 times. The application will contain appropriate tools that will allow them to efficiently store and retrieve information about their store branches and can be accessed by anyone.

![system diagram](/assets/System-diagram.png)

The application will be using The MVC architecture. The architecture separates the application logic into three-part Model, Controller and View. As seen in the System Architecture diagram above, the model stores and manages data from the database. The view effectively provides the user interface of the application, it contains all the functionality that directly interacts with the user. The controller connects the view and the model. it converts user input from the view into to demands for the model to retrieve or update data. The application will use a Non-relational database through mongo DB which is a document-based database that stores data in JSON-like documents.

The database uses an existing dataset from the log details which were previously stored in three CSV documents. Mongo then converts the data in the files into database objects which can then be accessed by the Model in the MVC architecture. The users should be able to access this data and interact with the Key interface of the website as shown Below.

![log-in-interface-diagram](/assets/LogIn-interface.PNG)

The users should access the portal by logging into it with their company email and password. After logging in they will be taken to the rest of web-application View pages. However, if the User is new to the portal, they will need to register to the application by entering a valid email and constructing a secure password.

![register-interface-diagram](/assets/register-interface.PNG)

Once logged into the application the users will now be able to access the rest of the application View pages which allow them to view data that is available in the database.

![view-store-diagram](/assets/view-store.PNG)

They should be able to perform Node CRUD functions on the data that is in the database like creating a new data object as seen below. They will need to enter information that matches the structure of the object as it is stored in the database.

Once the user has added a new data object to the database, they will be re-directed to the designated view page for the object they have created. The created object should be shown on the table that displays the data and the user should be able to update or delete or add more database objects to the database.

Another application interface will be to allow the user to update or delete the store logs. They will be able to update logs based on their store numbers and province. If they delete the log, they should not be able to see it on the table that displays the database data.

The users should also be able to find a specific store through the search page. They will be able to type in any word in the search bar and it will return all the store logs associated with that word

Once the user has finished performing all the website tasks that they wanted to perform they will be able to log-out of the application by clicking the log-out button in the navigation. This will re-direct them to the application Home page where they will need to log in again to re-access the data.

Key Design Choices

Before developing the application, I first designed wireframes for the view layer for the application. The first page that the users will see when they visit the webpage will be the index page.

The application will gain design inspiration from bootstrap style for its design. it will use bootstrap cover page design for the index page which will allow the users to know what the portal is about and allow then to register or log-into the portal.

The login/registration pages will contain forms that will allow the user to either log in as a returning user or to register as a new one. These forms will also tell the user what they need to enter as input into the form. The forms will also contain validation to error check the user input and also to let them know that they have entered invalid data

The data pages will visualize the database objects through bootstrap tables. The pages will also contain buttons that have descriptions of what they are used for displayed on them. The pages will also have pagination which will allow the users to switch from page to another. I choose light contrasting colours for background and the web-page content i.e., tables and buttons for these pages. This is so that the application will be accessible to users that will be colour-blind or have eyesight problems.

The application also has sub-pages that are embedded into buttons on the main data pages these will contain forms that the users can use to create/update database objects.

Database Design

The company previously stored its logs into excel sheets that were saved as CSV file. As seen in the image above this is one of the files that stored the store data logs. For this concept application I used MongoDB's model document which are constructed using flexible JSON-like document. It also provides all the capabilities needed for complex requirements.

After retrieving the three documents I then created a database that has 3 collections. The collections are then inserted with the records from the documents. The data from these documents is then converted into database objects as seen in the images above.

Security and Scalability

The application will be receiving users' passwords and their emails when they register to their site which means that it needs to be secured to make sure that in the event of a data breach their information will be secured. To accomplish this the application will use hash algorithm to store the user's passwords into a database.

As seen in the image above when the users enter their passwords into the application registration form, this will be applied to the hash algorithm that will encrypt the password and store it into a database. The application will use the bcrypt hash function which incorporates a salt protection to protect the hash functions against rainbow table and it is resistant against brute force attacks due to its slow speed.

I also made sure that the application would be able to withstand any amount of workload. This was done through using the MVC architectural pattern which separates the application into three parts and allows the application to send specific aspects to their designated component.

Conclusion and Reflection

Overall, the proposed concept will be able to meet most of the requirements that the company requires for their web application. They will be able to access their data and interact with it much more easily. However, the application is a low-level application and requires frequent management and updating to be able to store more of the data that will be accumulated by the company into the database. The application has a very simple wireframe that will allow the users to manoeuvre through the website easily and be able to understand how to interact with the application easily. However it is not user friendly to all users as users with low-eyesight may find it hard to read the content as they will not be able to adjust the font sizes on the web-application.