Victor Tarus

ORDER-NOW SYSTEM

CENTRIA UNIVERSITY OF APPLIED SCIENCES
IT Engineering
December 2020

ABSTRACT

Centria University of Applied Sciences	**Date** December 2020	**Author** Victor Tarus
Degree program IT Engineering		
Name of the thesis Order-Now		
Instructor		**Pages** **23**
Supervisor Jari Isohanni		

Computers have become an essential part of the day to day operations in most organizations. Most cleaning companies have digitized their businesses by investing heavily in developing systems such as systems recording work hours for employees. However, when it comes to ordering supplies, most companies still use pen and paper. Orders are written down on paper and left at specific collection points, which require someone to pick them up. The whole process can be cumbersome and time-consuming. Furthermore, reporting occupational hazards such as broken doorknobs, leaking pipes, or exposed electric wires can be daunting.

The Order-Now application is a web or mobile-based application that seeks to provide workers in cleaning companies with an efficient way of ordering supplies and reporting occupational hazards. This thesis aims to explore system design and build a web-based ordering system using React.js. Firebase Realtime database will be incorporated into the system to provide data storage. Agile methodology is, however, implemented in the development so that functionality can be delivered quickly to the market.

Keywords
Components, Firebase Realtime Database, React Props

ABSTRACT
CONCEPT DEFINITIONS
CONTENTS

1 INTRODUCTION

Good data provides a basis for making the right decisions. Without an efficient system of managing data, it can be hard to monitor the performance of critical systems in any organization. The analog way of managing data can be useful, but it has its limitations. As the size of an organization grows, it can be more error-prone compared to digital systems. With a digital system in place backed up by databases, it is easier to track data and set performance goals that positively contribute to an organizations' overall performance.

The analog way of placing cleaning supplies orders is typical in cleaning companies in Finland. However, the advent of new technologies such as WhatsApp has contributed to modernizing this system; unfortunately, this ordering system is still inefficient; the same data must be handwritten or typed repeatedly, which is tedious. Consequently, upon receiving the data, the person in charge of placing the orders must also organize the data. This process can be inconvenient, unreliable, and time-consuming instead of having a centralized database system that is easier to coordinate and as accurate as possible.

The logic behind the development is that data passed on; that is, the types of supplies needed will always remain constant for a considerable amount of time; hence, this application implements a menu restaurant setting's analogy. The menu remains constant, but the orders might differ in terms of quantity and time of order.

The order-now application aims to tackle a common business problem experienced across cleaning companies: ordering supplies. The application will be a full-stack web application whereby the front end will implement the react library coupled with a firebase realtime database on the back-end. Data is both stored and retrieved in JSON format. Subsequently, data retrieved is transformed from json format to human readable form and then passed as arguments to create views that make up the user interface.

2 PRODUCT VISION

A product vision is a statement that defines the essence of the product under development. The product vision creates a basis for developing more detailed attributes and features. Furthermore, it explains how the product differs from other competing products. (Sommerville 2019,7-8).

This thesis's product vision is to develop an application that serves small and mid-sized cleaning companies' supplies departments that seek an efficient way of managing their supplies inventory and further enable employees to report occupational hazards. The Order-Now application is a web-based service that provides a service whereby users can place orders on their phones with minimal need to type in or write the data. Furthermore, users can report occupational hazards at workplaces using the same application. Unlike other means of placing orders, the Order-now application allows for better tracking and storage of data, making future decisions like predicting order frequency.

Cleaning companies are the target customers for Order-now. For a long time, systems like Duunissa focus more on recording time used by cleaners, optimizing profits (Korttilinna, 2020). Still, a further step in improving inventory management will significantly improve company performance. It will collect user analytics, such as ordering patterns for a specific cleaning location, thereby enabling the ordering department to predict when to make the next orders. Subsequently, bulk ordering will be feasible and cheaper than ordering small quantities of supplies in the freight industry.

3 FEATURES, SCENARIOS, AND STORIES

Some software products are inspired, and an example of one of them being Facebook, which has been a marvel of the twenty-first century. Such products do not follow the generic way of developing software (Sommerville 2019,1-7.). Order-now being such a product solely is inspired by business knowledge in the cleaning industry, user problems, and user interaction. This chapter presents a run-down of features, scenarios, and stories.

3.1 Personas

A persona in software terms refers to an 'imagined user,' with a character that portrays the user type that will use the product developed. In this project, personas working in the cleaning industry help imagine what they want to do with this software. Furthermore, they help to envisage difficulties users might have in understanding and using some product features. (Sommerville 2019, 53-58.).The personas cited below provide an insight into the 'imagined' potential users of the order now application.

To begin with, a persona for a cleaner. Timo, age 47, is a cleaner in Kokkola, a small town in Northern Finland. He works in two different places under the same company, a bank in his morning shift and a fish market in his evening shift. Timo did not get proper access to education while growing up, and that means he is not confident in all matters of reading and writing. However, he has a smartphone, which he uses to communicate and access information through social media; hence he has the know-how for everyday application operations.

Secondly, a persona for a cleaning company supervisor named Kia, age 29, is a supervisor working with a local cleaning company in Kokkola. Her primary role is to ensure the smooth running of the company's cleaning functions and, most importantly, ensure that deliveries of cleaning supplies ordered by employees are on time to their respective cleaning sites. Kia has experience in web development, which gives her a head start in matters of IT. To cut the cost of visiting all the cleaning supplies lists, she created a WhatsApp group whereby employees would put forth orders for cleaning supplies and improve on the paper-based ordering system. Kia oversees 78 cleaners who work in 118 different client sites, and part of her job is to compile all the supply orders and pass them to her co-worker Tico who is in charge of inventory and making orders. Kia envisions the use of modern technology to make the system more efficient.

Finally, a persona for an inventory manager of a cleaning company named Patrick, age 25, works as an inventory manager at a cleaning company. His main job is to procure orders made by cleaners. Furthermore, making sure the cost of supplies is at an optimum level is part of his job. Consequently, he has to make sure the transport costs are at a bare minimum too. Procuring goods in bulk has proven effective in cutting transportation costs; however, the main challenge he faces is spontaneous supply orders made by cleaners.He is particularly interested in using the order-now application that would enable him to get a clear picture of the trends in the cleaning supplies' ordering, and it would be simpler to predict what type of supply will run out and should be pre-ordered.

3.2 Scenarios

Scenarios describe situations in which users use certain product features to achieve a particular goal. The Personas identified above provide a basis for writing scenarios; thus, the targeted users can relate to the various scenarios in real life. (Sommerville 2019,59). The Scenarios explored below help select and design features by imagining how users could interact with the product.

First and foremost, a scenario of the Timo using the Order-now system to report occupational hazards. To improve work safety, the head of the safety department has requested all employees to make safety observations. On a particular workday, Timo notices water leaking from the roof in one of the offices he cleans. He then uses the report hazard feature, whereby it uses his mobile work phone's camera resource. Coupled with that, he adds a short comment on the exact location of the safety observation; however, the application also uses the GPS location of the phone and tags it on the photo hence providing enough information for the situation to be remedied.

Similarly, Timo uses the application when there is a delay in the delivery of supplies. Due to unavoidable circumstances, there has been a delay in the delivery of supplies ordered by Timo. Timo contacts Patrick, the inventory manager, and fills him in on the situation. From home, Patrick logs into his Order-Now application and uses the inventory manager feature to check which other places among the company's cleaning sites still stock the same supplies Timo needs. Having found the most likely cleaning location that still has a good stock of the supplies needed, Patrick requests one of the supervisors to pick up the supplies and deliver them to Timo. Consequently, the system stores information on the transaction and will provide analytics for predicting future orders.

3.3 User Stories

User stories are narratives that are set out in a more detailed and structured way to present what the user requires from a software system (Sommerville, 2019, p. 66). The cleaner represents a user with a basic account, and the supervisor represents an administrator who has unlimited access to all the system's data and settings.

3.3.1 Cleaner

The following user stories present what the cleaner will need from the product for it to be viable.

US-1. The cleaner needs a way to order supplies using a few clicks on their work phone to not write down every time supplies on a piece of paper or a text.

US-2. The cleaner needs a way to be able to log in to the Order-Now account anywhere using login credentials provided by the employer that they shall use in all applications and hence not having to remember a new set of login credentials

US-3. The cleaner needs a way to check if a co-worker has ordered the cleaning supplies to a specific site.

US-4. The cleaner needs a way to be able to indicate if an order is urgent or not.

US-5. The cleaner needs a way need to report occupational hazards accompanied by pictures.

3.3.2 Supervisor

Subsequently, the system requires an administrator whose job is to do the setup, like adding employees to the system. For this purpose, the supervisor is the user. The following are the user stories.

US-1. The supervisor needs a way to add new cleaning sites and their respective cleaning supplies into the system.

US-2. The supervisor needs to check the inventory list if there is a delay in the new orders made sot that no place misses their cleaning supplies for more than 24 hours.

US-3. The supervisor needs not to have to provide login credentials every time accessing the Order-Now application

3.4 Features

A feature is a unit of the functionality of a software system that satisfies a requirement. In other words, it is a fragment of functionality that implements some user or system need.
Significant features are as follows:

- **FE-1:** Order cleaning supplies to be delivered.
- **FE-2:** Create, view, modify and delete cleaning sites from the list of cleaning locations
- **FE-3:** Suggest to users to make orders by gathering analytics from previous orders and predicting the average time between orders.
- **FE-4:** Give a rating to different cleaning products.
- **FE-5:** Indicate priority of order
- **FE-6:** Check inventory
- **FE-7:** Report occupational hazards using pictures attached with comments such as broken door handle
- **FE-8:** Assign cleaners to sites

3.4.1 Scope of Initial and Subsequent Releases

Table 1 below details the scope of initial and subsequent releases of the order now application. Due to time limitations and the need to quickly release the product to the market, the final product is in three release phases.

Table 1. Scope of initial and subsequent releases

Feature	Release 1	Release 2	Release 3
FE-1	Implemented for a few cleaning sites	Fully implemented	
FE-2	Implemented partly	Fully implemented	
FE-3	Not implemented	Implemented for the first trial sites	Fully implemented
FE-4	Not implemented	Fully implemented	
FE-5	Fully implemented		
FE-6	Not implemented	Fully implemented	
FE-7	Not implemented		Fully implemented
FE-8	Implemented partly	Fully implemented	

4 APPLICATION DESIGN

In this chapter, the order-now application design is explored, coupled with a run-down through the functional and non-functional requirements. The application design describes the various modules developed along with their functionalities.

4.1 Requirements

Software requirements are a core part of software development and requirements engineering, which involves developing services the customer requires from a system and the constraints it operates. (BUEDE & MILLER, 2016, pp. 145-149.)

4.1.1 Functional requirements

Functional requirements are the system's behavior, how it should react to inputs, and its services (Sommerville, 2011) . Functional requirements tell developers what the system should do for the system to accomplish its purpose.

FR101. The system shall let a user who is logged into the system place an order of one or more cleaning items

FR102. The system shall allow the user to specify whether the order is urgent or a pre-order.

FR103. The system shall display a list of cleaning sites assigned to the cleaner after successful authentication

FR104. The system shall allow the user to order multiple identical cleaning items

FR105. The system shall prompt the user to confirm an order

FR106. The user shall either confirm the order or request to edit the order

4.1.2 Non-functional requirements

Non-functional requirements refer to how well the system performs rather than what it does (Bennet, Farmer & Mcrobb 2010, 167). Non-functional requirements define system behavior, features, and general characteristic that affect the user experience.

NFR101. The system shall accommodate 100 users during the peak usage time window that is between 4:00 am to 11:00 am

NFR102. Confirmation messages shall be displayed to the users within 4 seconds after the user submits information to the system

NFR103. All web pages shall take a maximum of 6 seconds over a 2g network, that is over a 40KBps connection

NFR104. Responses to queries shall take a maximum of 5 seconds to load after the user submits a query.

NFR105. The system shall make use of database encryption to encrypt personal information

4.2 Use cases

Use case diagrams are behavioral diagrams used to capture, specify, and visualize required system behavior: Actors, use-cases, and the relationships connecting them used as the main elements. Actors are entities used to model users or other systems that interact with the system, such as operators using the system, sensors providing information, and a client computer in a client-server system. (Software engineering design 2012,55-57.)

4.2.1 Use Case diagram

Figure 3 below captures users, functions, and their relationships using use case diagrams. The stereotype <<include>> represents relationships between use cases and further provides a way to extract common parts of the behavior of two or more use cases.

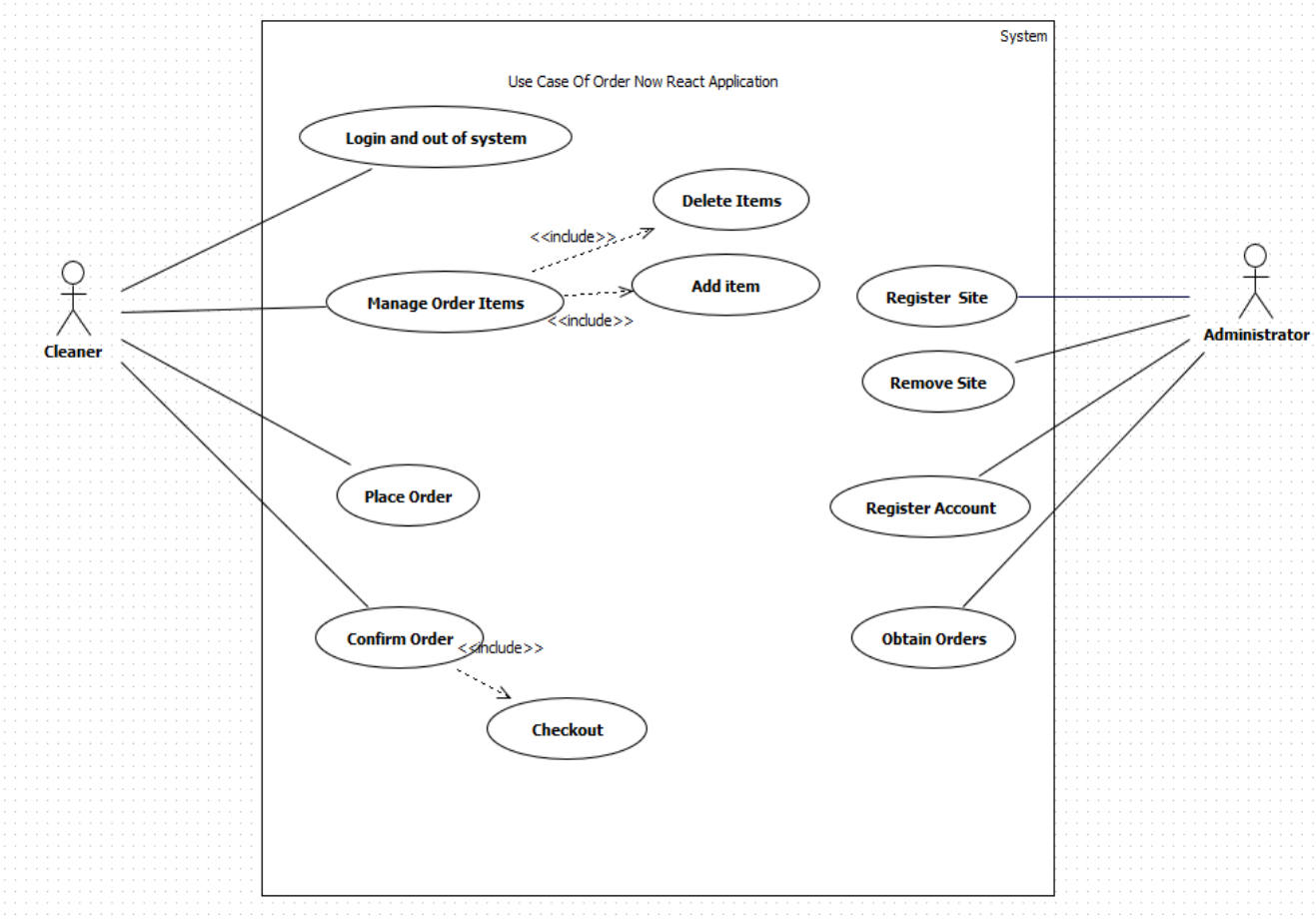

Figure 1. Use Case Diagram

4.2.2 Description of Use Case diagrams

Tables 2 to 9 below provide detailed descriptions of the use cases shown in Figure 3 above. Each use case has a unique id, a use case name, and a description. The other vital parts of a use case description captured on the tables are pre-conditions, post-conditions, normal-flow and alternative flows. The pre-conditions refer to conditions that must hold for the use-case to begin, while post-conditions refer to conditions that must hold once the use case is complete. Furthermore the normal-flow refers to the most

frequent use case scenario, while alternative flows refer to less frequent scenarios. Alternative flows represent flows that not directly in support of the goals of the system.

Table 2. Login Use Case

Use Case ID:	001
Use Case Name:	Login
Actors:	Cleaner
Description:	User can log in with their credentials to use the system
Trigger:	User presses login button
Pre-conditions:	The admin has registered the user to access the system
Normal Flow:	The user is already registered and can access services provided by the system.
Alternative Flows:	The user does not have the correct details and is prompted to use the forgot password pop up window
Exceptions:	
Post-conditions:	
Includes:	
Assumptions:	Customer will edit or delete their information once they are beyond what the system entails
Notes and Issues:	

Table 3. Manage Order Items Use Case

Use Case ID:	002
Use Case Name:	Manage order Items
Actors:	Cleaner
Description:	The user selects a site out of the sites automatically enlisted, and another window pops up with a list of cleaning items for that site coupled with controls.
Trigger:	Login
Pre-conditions:	The user has been assigned cleaning places by the administrator and has logged in successfully
Normal Flow:	User logs in, and all the cleaning sites are loaded; the user picks one cleaning site and proceeds to manage the orders
Alternative Flows:	The user does not have the correct details and is prompted to use the forgot password pop up window
Exceptions:	
Post-conditions:	
Includes:	Delete Items, Add Items
Frequency of Use:	
Notes and Issues:	

Table 4. Place Order Use Case

Use Case ID:	003
Use Case Name:	Place Order
Actors:	Cleaner

Description:	The user adds at least one item to the order list and can proceed with placing an order
Trigger:	
Pre-conditions:	At least a single item is present in the cart
Normal Flow:	
Alternative Flows:	
Exceptions:	Another user places an order with the same attributes
Post-conditions:	
Includes:	
Frequency of Use:	
Assumptions:	

Table 5. Confirm Order

Use Case ID:	004
Use Case Name:	Confirm Order
Actors:	Cleaner
Description:	
Trigger:	User presses the confirm order button
Pre-conditions:	
Normal Flow:	The user confirms the order after going through the list on the modal and proceeds to checkout
Alternative Flows:	
Exceptions:	
Post-conditions:	
Includes:	Checkout
Frequency of Use:	
Special Requirements:	
Assumptions:	

Table 6. Register site Use Case

Use Case ID:	006
Use Case Name:	Register Site
Actors:	Administrator
Description:	User registers a new cleaning site using the register button
Trigger:	
Pre-conditions:	All site details are provided, including the address and cleaning products used
Normal Flow:	The new site does not exist in the system and is successfully registered
Alternative Flows:	A new site already exists in the system and can only be edited
Exceptions:	
Post-conditions:	
Includes:	

Frequency of Use:	
Special Require-ments:	
Assumptions:	

Table 7. Remove Site Use Case

Use Case ID:	007
Use Case Name:	Remove Site
Actors:	Administrator
Description:	User can remove a site from the system, for example, when a termination of a cleaning contract occurs
Trigger:	
Pre-conditions:	The site exists in the system
Normal Flow:	The administrator removes the site
Alternative Flows:	
Exceptions:	
Post-conditions:	
Includes:	
Frequency of Use:	
Assumptions:	

Table 8. Register Account Use Case

Use Case ID:	008
Use Case Name:	Register Account
Actors:	Administrator
Description:	Admin registers all the users who can use the system and in turn provides them with login credentials
Trigger:	
Pre-conditions:	
Normal Flow:	User is successfully registered
Alternative Flows:	
Exceptions:	
Post-conditions:	Users can log in using the credentials handed to them by the admin
Includes:	
Frequency of Use:	
Special Require-ments:	
Assumptions:	
Notes and Issues:	

Table 9. Obtain Orders Use Case

Use Case ID:	009
Use Case Name:	Obtain Orders

Actors:	Administrator
Description:	The user obtains all orders made for further action
Trigger:	
Pre-conditions:	
Normal Flow:	Orders made by the other actors that are cleaners, successfully, the user gets a list of all orders made.
Alternative Flows:	
Exceptions:	No pending orders
Post-conditions:	
Includes:	
Frequency of Use:	
Assumptions:	

5 APPLICATION DEVELOPMENT AND IMPLEMENTATION

This chapter provides the various steps taken in the development and implementation of the order now application. It provides an in-depth look at the architecture used in the development and the various technologies used. Furthermore, it provides an overview of how the user will interact with the user interface by providing a graphical representation of the user interface.

5.1 Software Architecture

According to IEEE, Software Architecture is a broad term that describes a software system's entire organization embodied in its components, relationships to each other and the environment, and the principles guiding its design and evolution. (Sommerville, 2019, pp. 81-84.) This project's architecture focuses solely on presenting an interactive user interface even though the process flow and components might defer slightly depending on the users. The model view controller architecture was chosen for this project and shall be discussed later after highlighting the user interface for better understanding.

Three main parts make up Order-Now application in its architecture. First and foremost, the database holds all the essential data from the users and, more importantly, data used to create components that make up the user interface.

Secondly, the User interface created using components. The user interface is what the user interacts with within the system. Coupled with that, users can input data into the system using the components in the user interface.

Furthermore, another vital component of the system is the server. It serves as a communication channel between the database and the UI.

5.2 User Interface Design

As briefly mentioned earlier, the user interface is a collection of components, which begs the question, what is a component? A component is an element that implements a coherent set of functionality or

features. The order-now application uses a component-based approach, with the first step being the de-composition of software into several components.

Furthermore, the Order-Now application relies heavily on the react library in its implementation. Subsequently, react is component-based. React is a JavaScript library for building user interfaces that allow for; passing data through the application and keeping the state out of the Document Object Model. React allows the rendering of the right components whenever the data changes, thereby creating an interactive feel.

Figure 5 below shows the user interface in the various stages of interaction by the user. However, this is the basic version of the first release; more features are underway on the subsequent releases.

Step 1: The user enters their login details and presses the green button to log in

Step 2: The user selects a cleaning site and is lead to the next page

Step 3: The user can choose either one of two buttons, the timer, or place orders. If the user chooses to place orders, they are lead to the next page.

Step 4: The application generates a unique list of supplies to the UI, depending on the cleaning site. Coupled with that are buttons to add or remove the quantity of the supplies to the user.

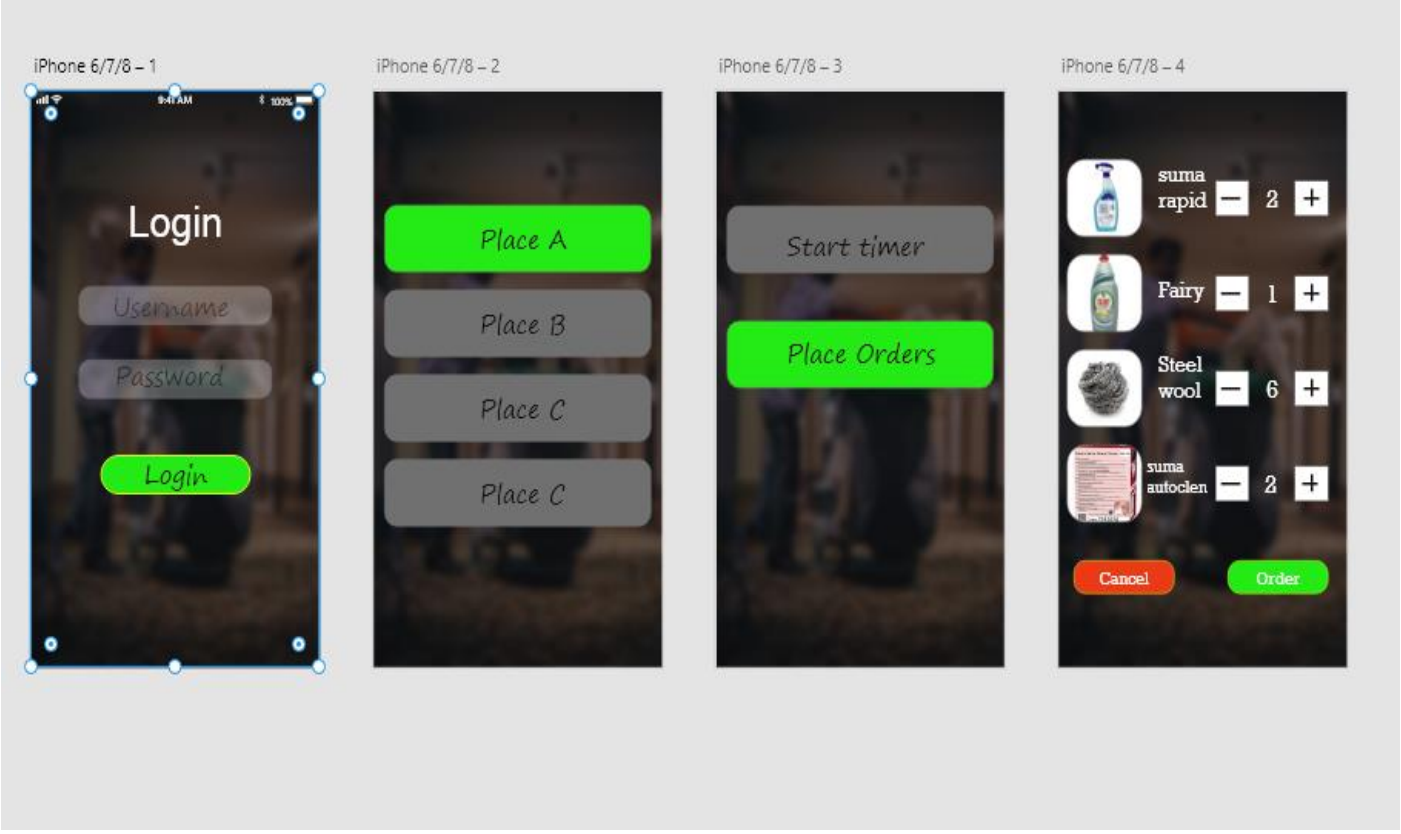

Figure 2. User interface design

5.3 React Library

Before taking a deep dive into the react library, an overview of the document object model, commonly known as DOM, is discussed. DOM is an API for HTML, XML, and SVG. The DOM offers a webpage as a tree of elements and allows scripting languages such as javascript to access them. Besides, DOM is used to build documents, navigate their structure, and add, modify or delete elements and contents. In javascript, which is the primary language of choice in this project, contains a required method called the DOM interface acts as a gateway to the DOM structure. (Wilton & McPeak, 2009.)

5.3.1 Virtual Dom

Javascript operations are relatively fast compared to the DOM manipulation, which is at the heart of the modern interactive web. The fact that most Javascript frameworks update the DOM much more than they have to has not made the situation any better. To circumnavigate the slow DOM manipulation, developers use a virtual DOM.

The virtual Dom is a JavaScript object that is a representation of the browser DOM. It is fast compared to the browser Dom. It can produce 200,000 virtual DOM nodes per second; hence it is significant in the application's high performance by cutting down on load time; this comes in handy whereby every time there is a change in the application, creation of a new virtual DOM occurs. (React, 2020).

5.3.2 Model View Controller Architecture

React follows the Model-View-Controller architectural pattern, commonly known as MVC architecture, which comprises three general components: the Model, the View, and the Controller. The model manages the data and rules of the application. The view refers to what the user interacts with on an application. On the other hand, the Controller takes user input and converts it into commands for the Model or View layers. Figure 6 below shows the implementation of an MVC architecture.

To better understand how the MVC architecture works, a user in the client computer requests a list of cleaning sites from the database. The client sends a request to the server; the controller component re-

ceives the request in the server. In turn, the Controller passes the request to the Controller, which eventually communicates with the database. The data is retrieved from the database and through the model component and forwarded to the Controller. Instead of the retrieved data being sent directly to the client, it is sent to the view component because the database's data may not always be in a human-readable format.

Therefore, the view's main task is to modify the data into a human-readable format, and style is applied to the data to make it more presentable to the user. As illustrated, the model oversees interactions with the database and executing business logic, while the view's task is what the user sees on the screen and generating the user interface. Coupled with that, the Controller takes user input and interacts with both the view and the model. Hence rubber-stamping the idea of having different components with coherent functionality.

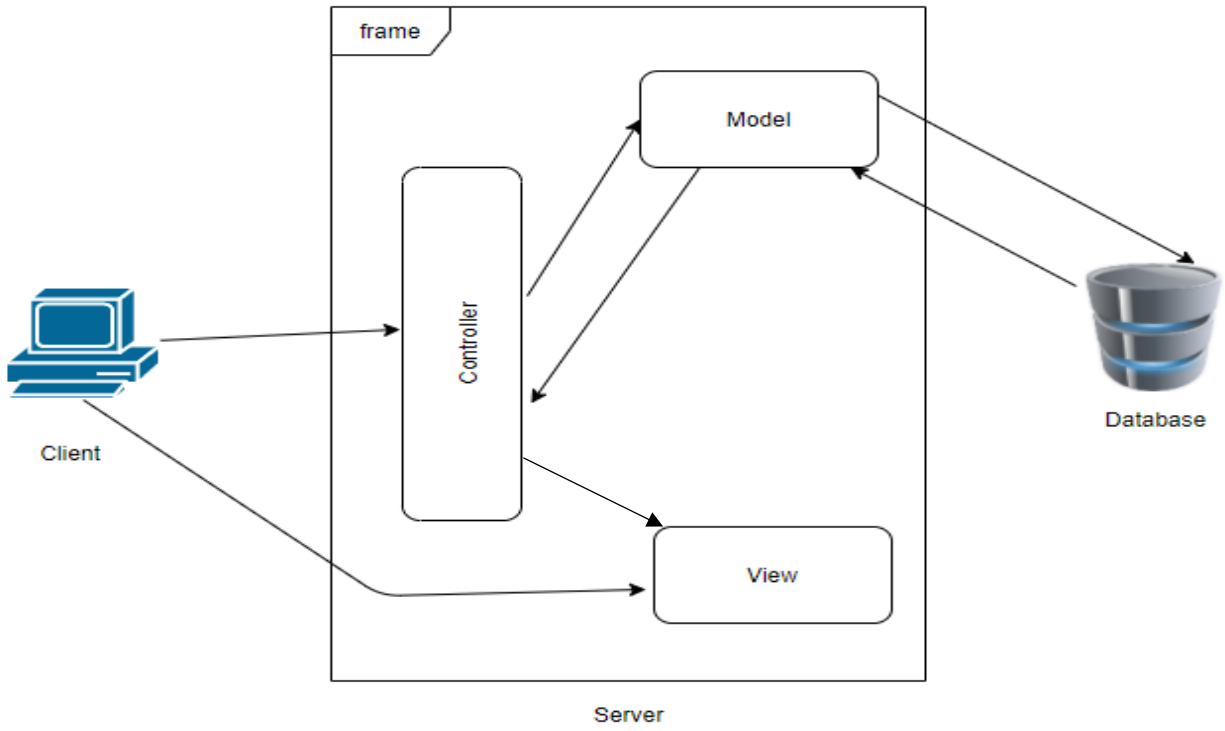

Figure 3. MVC Architecture

5.4 Development

The development phase involves laying out a component tree that provides a blueprint of how the components will be implemented and enhance reusability, one of the react library's critical foundation. Each component's implementation is as a coherent fragment of functionality that provides a particular service. When planning components according to the React library, there are two types of components to be considered, that is stateless and stateful components. Stateful components keep track of changing data, while stateless components are presentational components in that they print out data given to them via props.

5.4.1 Layout Component tree

As mentioned earlier, components are vital in creating react applications; the first step is to identify components and create a component tree. In this subtopic, a component tree for the order now application shall be explored. Figure 7 below outlines the key components of the application.

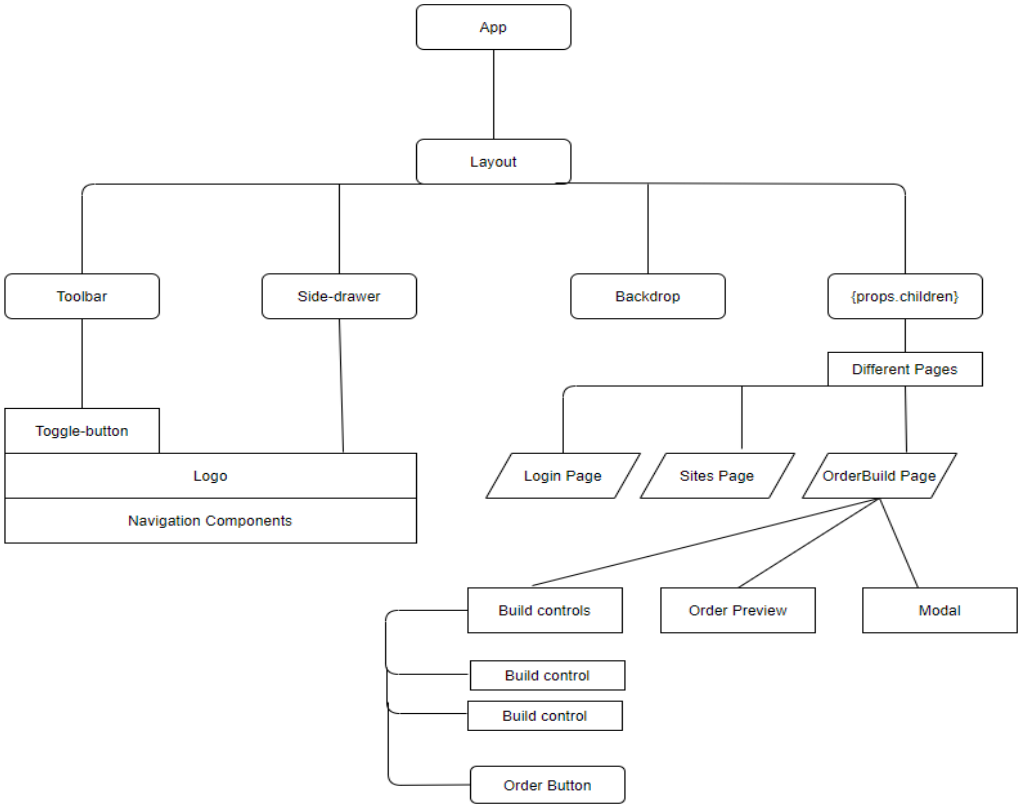

Figure 4. Component tree for order-Now application

5.4.2 Planning the state

Planning the state gives room for deciding which of the components should be stateful and which ones should be stateless. Stateless components are merely functional components that do not have a local state; however, with React hooks' advent, adding a local state to a functional component is possible. On the other hand, stateful components, otherwise known as containers, keep track of changing data, unlike stateless components that print data received via props. Props are arguments passed into react components, and this application utilizes them as a cornerstone in building scalable views using data from the back-end. Data from the back-end transforms and is passed as arguments to create react components, thereby giving an interactive feel to the application. The state manages the different locations' cleaning locations obtained from the database, the cleaning products to be ordered, and the order priority.

5.5 Server

A server can refer to either hardware or software or both working harmoniously. A web server is a computer that stores web server software and a website's component files, in this case, the JSON file. JSON (JavaScript Object Notation) is a data exchange format. It is text-based and lightweight for data exchange between clients and servers. Figure 8 displays a raw HTTP request from the client. In this case, it would be a cleaner's mobile phone to the server. The server serves the requests and responds as expected. Correspondingly, JSON used as the data format in the two way communication is a serialized string with a combination of key values enveloped in parentheses. (D'mello & Sriparasa 2018, 8.)

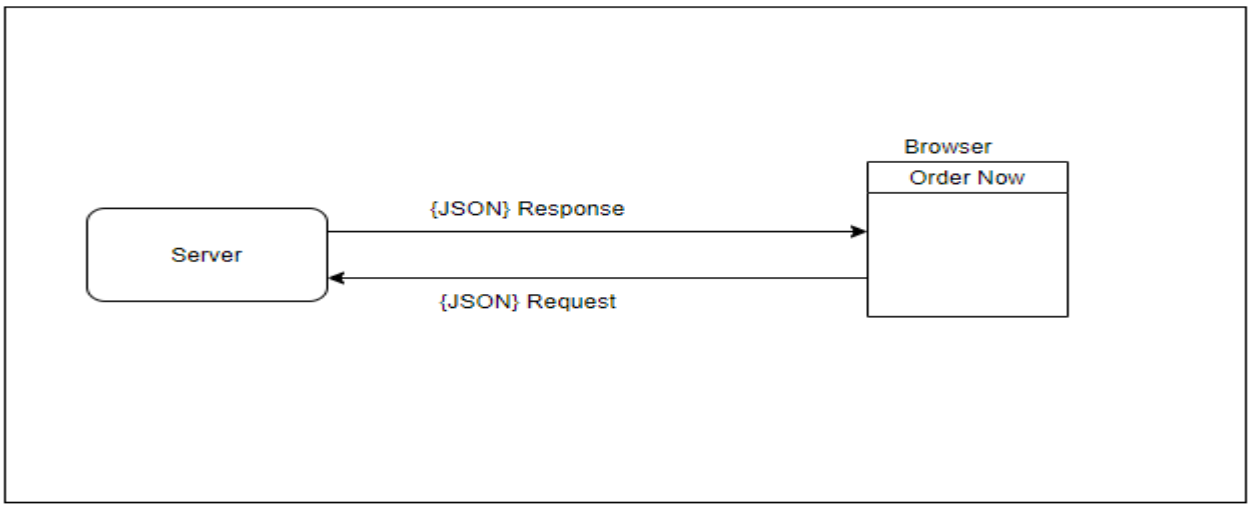

Figure 5. Client-Server Communication

5.6 Firebase Realtime Database

For this project, the Firebase Realtime database is the designated database, as it is easy to implement and is secure. Firebase Realtime Database is a cloud-hosted database that stores data as JSON and synchronized Realtime to every connected client. (Google, 2020.)

The data synchronization of firebase allows for any connected device to receive updates within milliseconds every time data changes, which is critical in making the application responsive. Coupled with that, the Firebase Realtime Database can be accessed directly from a mobile device or a web browser because there is no need for an application server.

However, implementing other back-end solutions is in the pipeline because of a few shortfalls of the Firebase Realtime Database. One of the drawbacks is limited querying capabilities since the whole database is a huge JSON file, making it difficult to make complex queries. Furthermore, because of its data modeling," data as a single file" structure, relations between data items cannot be implemented. (Google, 2020.)

Firebase Realtime database is a NoSQL database that stores data as JSON objects. Adding data to the database leads to the automatic creation of nodes in the existing JSON structure with an associated key. FIGURE 9 below provides a visual representation of the database structure. The three small circles represent objects with multiple instances, while the rectangles with rounded edges represent nodes in the hierarchy. The first node on the far left represents the parent node as the hierarchy moves from left to right.

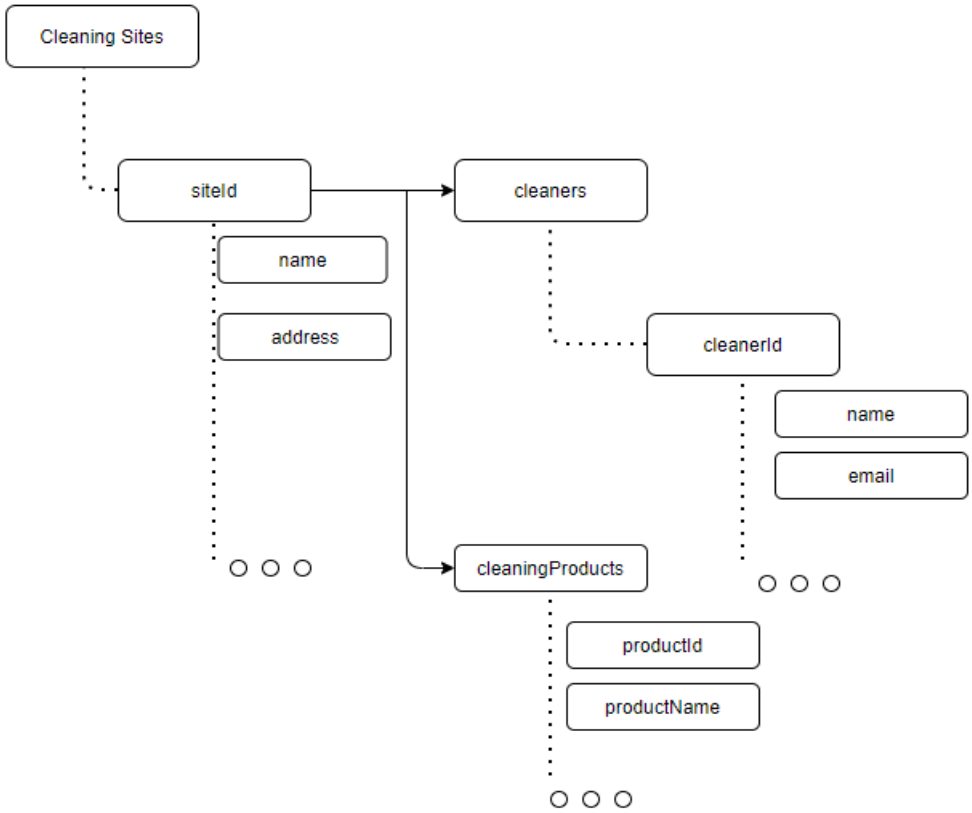

Figure 6. Database Structure

5.7 Authentication

Authentication is a process of ensuring anyone who accesses the system is who they claim to be and is the backbone of any services system which seeks to provide services to the right users. Furthermore, it ensures that the information providers are the only ones mandated to manipulate the information. Generally, authentication bases itself on three approaches: User knowledge, user possession, and user attributes.

User-knowledge-based authentication relies on the user providing private information when registering to use the system. Subsequently, every time the user wants to use the system, part of the information is

requested. On the other hand,possession-based authentication relies on the user having a physical device connected to the authenticating system. For instance, a user can receive a text message with a code on their phone and input it to confirm that they possess the device. The user inputs this code and is in turn compared with a code generated by the authenticating system. Subsequently, Attribute-based authentication bases itself on unique biometric features of the user, such as fingerprints.

Each of the authentication approaches mentioned above has its advantages and disadvantages. Attribute-based authentication takes precedence in terms of being more secure while user-knowledge-based coming last. However, new authentication systems use hybrid authentication systems that combine two or more approaches. Service providers such as Google offer a two-stage authentication method that uses a password coupled with a confirmation code sent on the user's phone. Furthermore, when choosing an authentication method, the product is put into consideration. In the Order-now application, confidential user information such as financial information is not a requirement from the user. Therefore, knowledge-based authentication is all that is required. (Sommerville 2019,195-197.)

6 CONCLUSION

The product of this work, order now, is a user-friendly web-based service that provides ordering services to cleaning companies. The purpose of this thesis was to present an opportunity and create a viable commercial product to be developed and eventually availed to the market. Moreover, the idea came about through personal experience in the cleaning industry, which exposed the gaps in existing systems that could be improved using the software.

As more and more companies automate their business, custom software has become less favored considering its associated costs. It has become clear that most companies face common business problems. Cleaning companies face the same problem when it comes to ordering supplies. Therefore, the approach taken for this software was product based engineering as opposed to project-based engineering. It provided an opportunity to develop features and build on existing systems' weaknesses, hoping to be a game-changer in the cleaning industry and potentially have a broad market appeal.

The various phases of this project's development have been an incredible learning experience and, coupled with the various challenges experienced in solving different problems, brought an insight into the software development industry. Creation of personas, scenarios, and user stories was the highlight of this project as it finally made sense of how important they are in feature identification. Some takeaways are that software development is evolving and should be fully used to solve common business problems.

Furthermore, with the ever-growing desirability towards artificial intelligence, plans are on course to invest time in finding ways to better this application and the system by utilizing the power of artificial intelligence. Artificial intelligence, combined with the power of machine learning, have been the two main components in enabling better predictability in various business entities.

REFERENCES

Bennet, S., Farmer, R., & McRobb, S. 2010. Object-oriented Systems Analysis and Design: Using UML. New York: McGraw-Hill Education.

Buede, D. M., & Miller, W. D. 2016. the engineering design of systems.3rd edition. Hoboken, NJ: John Wiley & Sons, Inc.

D'mello, B. J., & Sriparasa, S. S. 2018. JavaScript and json essentials.2nd edition. Birmingham - Mumbai: Packt Publishing Ltd.

Google. 2020, 11, 10. Firebase. Available: https://firebase.google.com .Accessed 11 November 2020

Korttilinna, O. 2020, 11, 2020. Duunissa. Available: https://www.duunissa.fi/ .Accessed 23 November 2020.

React.2020.React.Available:https://reactjs.org/docs/faq-internals.html#gatsby-focus-wrapper .Accessed 15 October 2020.

Sommerville, I. 2011. Software engineering. In S. Ian. Hoboken, NJ: Pearson.

Sommerville, I. 2019. Engineering software products (1st ed.). Hoboken, NJ: Pearson.

Wilton, P., & McPeak, J. 2009. Begining Javascript. Indianapolis, Indiana: John Wiley & Sons ,Incorporated.