National College of Ireland

BSc (Hons) Business Information Systems

2020/2021

Lim Hong Chun

X19134134

X19134134@student.ncirl.ie

Project Kelp

Technical Report

Contents

Executive Summary

This report act as the mid point documentation for software project “Project Kelp”. In this report, information included in previous proposal will be mentioned again, with updated progress and amendation, in sections where previous information will be brought up, these updates will be underlined for convenient reading. This include changes in requirement, steps of implementaion, further documentation of designs such as use case and graphical user interface, conclusion of the current development progress will also be included.

1.0 Introduction

1.1. Background

I am currently final year student in Business Information Systems, and plan to involve in business and technology integration field after my graduation. This leads me to thinking the demand currently in the business industry.

My family owns a traditional small retail business, and I help the company on the technology section as part time, my task includes basic IT operation task, trains staff for our just started online retail store, and communicator between frontline retail staffs and external POS system developer.

Because of my background in SME business and the demand to involves in the business tech integration industry, I came out with the idea of developing a web-based application that would help SMEs transition, hence the idea of Project Kelp produced.

1.2. Aims

The aim of this project is set to be an introduction application collection for traditional small and medium-sized enterprises (SMEs) towards complex business information systems.

Traditional SMEs commonly facing cost difficulties in the attempt of transition towards modern information systems especially the training cost of the employees.

Objective of this project is providing an easy-to-use and low-cost application to resolve that issue. The project will be web-based and contain an inventory management system and some useful small features including file sharing and timetable.

This project will be focusing on easy to use for beginner and low cost, with simple GUI and self explaning icons to avoid confusion to the user. It should also have low stress on the hardwares to lower the hardware requirement for the user.

1.3. Technology

The project will be developed as web application mainly focusing on personal computer environment. Basic web language such as HTML, JavaScript, PHP will be used and React.JS will be the framework for front-end web design.

Project is planned to be light weight for user to reduce hardware requirement, hence it will be hosted on cloud framework.

<u>Google account login and Facebook login option are provided to user to allows website account automatically created by their current logged in account. Regular sign-up form and login option is also provided to the user</u>.

MySQL will be used for the main database framework with the advice from my supervisor. MySQL have better community support and longer history for being use in business information systems development, which provides more references for this project.

<u>After further discussion with my supervisor, I decided to develop the project by not using the Data Access Layer (DAL), but just direct coding the queries, this provide the chance to showcase my personal database skillset, but more importantly it allows the flexibility on making changes in the future without needed to redo everything compared to if it was developed on DAL.</u>

After the research for implementation practicality, Google Firebase is choosen due to it's low learning difficulty and more importantly ease of implementation with the React.JS language. Firebase by Google also support MySQL which will be using for this project.

1.4. Structure

The document was seperated into two major part, first part being section one (1) to four (4), discuss about the information of this project, and second part from section five (5) onwards being related information that support the discussion in first part, including references and appendices.

The first section (1) of the report being the introduction to this project, including the background, aims and technology involved. Section two (2) primary focus on the requirements and and design, including system requirements, use cases and implementation steps, graphical user interface design and testing. Section three (3) conclude the most part of the project will some extended discussion. Section four (4) talks about the potential future development of this project.

And the following sections will be supplimentary for the previous sections, such as references and appendices.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

Feature requirements that should be implemented in the project, these are the base requirements for the web application to achieve its purpose for inventory management service provider.

2.1.1.1. Requirement 1: User Login

2.1.1.1.1. Description & Priority

The login page being on the top priority develop task, serves the purpose of allowing user login to the web app by using their email and password, or third-party login option such as Google Login. By entering the matched account information, user will be redirect to the main page of the web app which allows the user to interact with main features of the web app.

2.1.1.1.2. Use Case

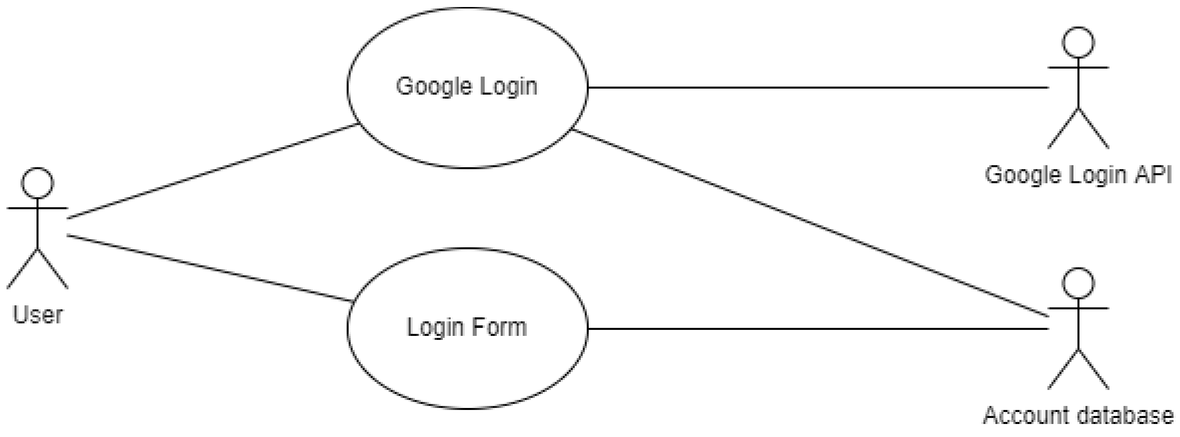
Scope

The scope of this use case is to allow user login into the website.

Description

This use case describes the interaction of user to the login page, by providing email and password registered on the website, or by third-party login option such as Google account, the user is then allowed to be logged into the website if the account provided is correct.

Use Case Diagram



Flow Description

Precondition

The user is not logged into the website.

Activation

This use case starts when a user enters the website without any user token.

Main flow

1. The system identifies the lack of user token.
2. The user enters name, email and password in login form.
3. The user choose to login with Google account by clicks on Google login button. (See A1)
4. The system fetch user information from database, compare to information entered by user.
5. The information is compared and matched.
6. The information is compared and did not match. (See A2)
7. The system assigns user token to the user, and redirect user to the homepage.

Alternate flow

A1: Google Login

1. The system shows a pop-up window, shows the current logged in google account as login option as well as the option to sign in to new google account.
2. The user login with a Google account.
3. The system retrieves account details from Google Login API and compares with information in account database.
4. If the Google account details not exist in account database, create a new account record in database and proceed, or if exist, proceed to next step.
5. The use case continues at position 7 of the main flow.

A2: Account details did not match

1. The system shows error message on top of login form, let user know their account details did not match.
2. The user enters account details again.
3. The use case continues at position 4 of the main flow.

Post condition

The system redirect user to homepage.

2.1.1.2. Requirement 2: User Registration

2.1.1.2.1. Description & Priority

The registration page being on the top priority develop task, serves the purpose of allowing user to register their account on the web app by using their email and password, or third-party login option such as Google Login. By registering the account, user will be able to enter the main page of the web app which allows the user to interact with main features of the web app.

2.1.1.2.2. Use Case

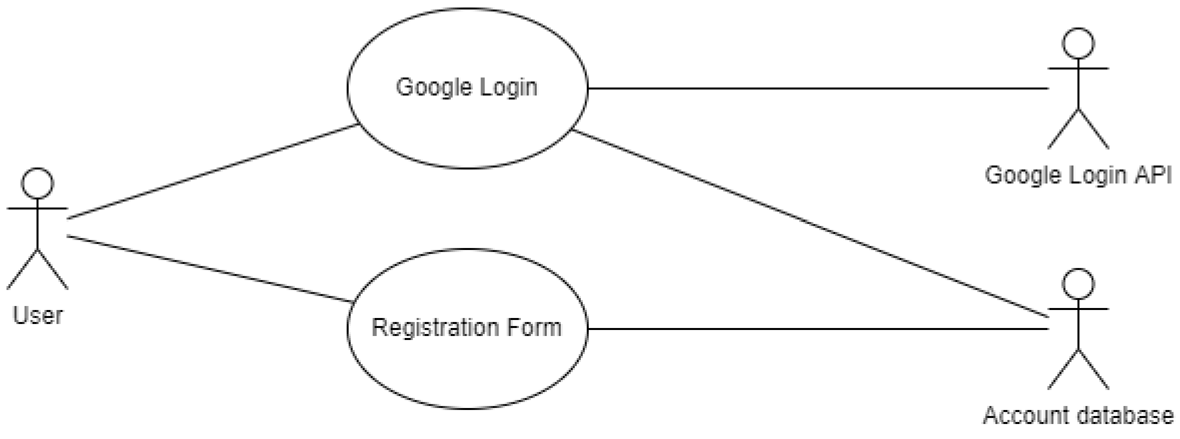
Scope

The scope of this use case is to allow user to register their account on the website.

Description

This use case describes the interaction of user to the registration page, by providing name, email, birth date and password on the website, or by third-party login option such as Google account, the user is then allowed to be logged into the website after the account registration is completed.

Use Case Diagram



Flow Description

Precondition

The user is not logged into the website.

Activation

This use case starts when a user enters the website without any user token and click on the register button on login page.

Main flow

1. The system identifies the lack of user token.
2. The user enters name, email and password in registration form.

3. The user chooses to register with Google account by clicks on Google login button. (See A1)
4. The system fetch user information from database, compare to information entered by user.
5. The user information is not existed in database, system create a new record in database.
6. The information entered are not completed or wrong format. (See A2)
7. The user information already exists in the database. (See A3)
8. The system creates a new record in account database.
9. The system assigns user token to the user, and redirect user to the login page.

Alternate flow

A1: Google Account Registration

1. The system shows a pop-up window, shows the current logged in google account as register option as well as the option to sign in to new google account.
2. The user login with a Google account.
3. The system retrieves account details from Google Login API.
4. The use case continues at position 4 of the main flow.

A2: Account details incomplete/ wrong format

1. The system shows error message on top of register form, let user know their account details did not match, including confirm password did not match and wrong email format.
2. The user enters account details again.
3. The use case continues at position 4 of the main flow.

A3: Account details already exists

1. The system shows error message on top of register form, let user know their account already registered.
2. The system redirect user back to login page after 3 seconds.

Post condition

The system redirect user to login page.

2.1.1.3. Requirement 3: User Logout

2.1.1.3.1. Description & Priority

The user logout function being on the top priority develop task, serves the purpose of allowing user to logout from the web app.

2.1.1.3.2. Use Case

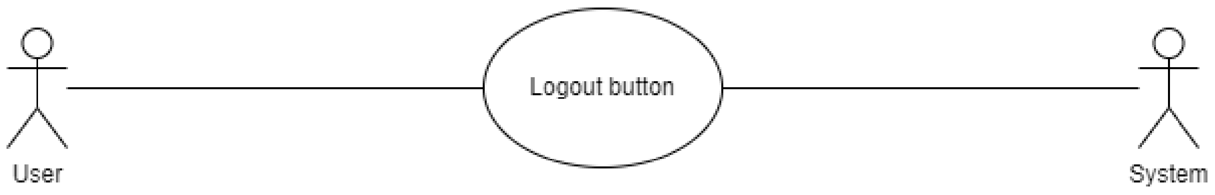
Scope

The scope of this use case is to allow user to logout from their account on the website.

Description

This use case describes the interaction of user to the logout button on the website.

Use Case Diagram



Flow Description

Precondition

The user is logged into the website.

Activation

This use case starts when a user clicks on the logout button on the website with logged in status and assigned a user token.

Main flow

1. The system identifies the user clicks on the logout button.
2. The system empy the user’s token and ends the session.
3. The system redirects the user back to the login page.

Post condition

The system redirect user to login page.

2.1.1.4. Requirement 4: View Inventory

2.1.1.4.1. Description & Priority

The user should be able to view the inventory after logged in, this feature serves the function of allowing user to browse through inventory item records in their account. This feature should be priotize after the login module.

2.1.1.4.2. Use Case

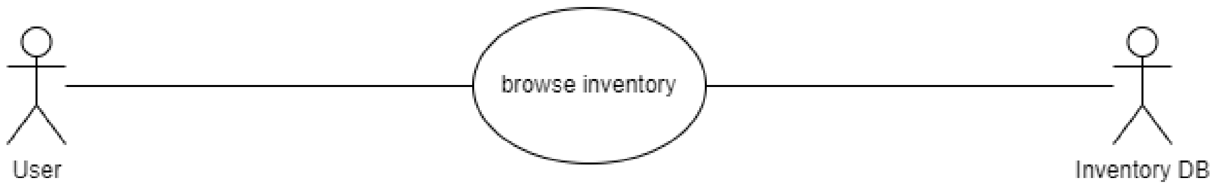
Scope

The scope of this use case is to allow user to access their inventory records in the account.

Description

This use case describes the interaction of user to browse the inventory records in the system.

Use Case Diagram



Flow Description

Precondition

The user is logged into the website.

Activation

This use case starts when a user clicks on the inventory menu on the website with logged in status and assigned a user token.

Main flow

1. The system identifies the user opens the inventory tab through the side nagivation menu.
2. The system renders the inventory graphic user interface within the same page for the user.
3. The system reaches to the inventory database looking for inventory records that matched the user details.
4. The system fetches matched records and loads into the list and display to the user.
5. The system could not find any records from the database. (See A1)
6. The system could not connect to the database. (See E1)
7. The system could not fetch records from the database. (See E2)

Alternate flow

A1: Could not find any records

1. The system shows a message in the inventory menu stated that no record is found.

Exceptional flow

E1: Could not connect to the database

1. The system shows an error message in the inventory menu stated that connection error to the database, suggest user to refresh the page.

E2: Could not fetch record from the database

1. The system shows an error message in the inventory menu stated that error fetching result from database, suggest user to refresh the page.

Post condition

The system goes into a wait state.

2.1.1.5. Requirement 5: Add Inventory Record

2.1.1.5.1. Description & Priority

The user should be able to add record to the inventory after logged in, this feature serves the function of allowing user to add inventory records to the inventory system bind to their account.

2.1.1.5.2. Use Case

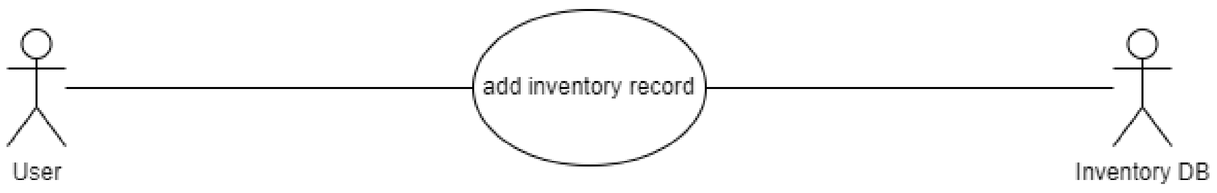
Scope

The scope of this use case is to allow user to add inventory records into the system.

Description

This use case describes the interaction of user tries to add inventory records into the system.

Use Case Diagram



Flow Description

Precondition

The user is logged into the website, being in the inventory menu.

Activation

This use case starts when a user being in the inventory menu on the website, clicks on the add item button.

Main flow

1. The system shows a pop-up window with a form that allows user enters item details.
2. The system checks the information when user submit the form.
3. The system notices the form is incomplete or wrong entries such as alphabet in currency field. (See A1)
4. The system adds a record to the inventory database when the form meets the requirement.
5. The system shows a message below the form, stating the item record is being added successfully.
6. The system could not connect to the database. (See E1)
7. The system could not add the record into the database. (See E2)

Alternate flow

A1: Form is incomplete or wrong entries

1. The system shows a message in the add item form, stated that wrong entries or form incomplete.
2. The user enters item details again.
3. The use case continues at position 2 of the main flow.

Exceptional flow

E1: Could not connect to the database

1. The system shows an error message in the add item form stated that connection error to the database, suggest user to refresh the page.

E2: Could not add record into the database

1. The system shows an error message in the add item form stated that error adding item into database, suggest user to refresh the page.

Post condition

The system refreshes inventory list after user closes the add item form and stay in wait state.

2.1.1.6. Requirement 6: Delete Inventory Record

2.1.1.6.1. Description & Priority

The user should be able to delete record to the inventory after logged in, given the item record exists, this feature serves the function of allowing user to delete inventory records from the inventory system bind to their account.

2.1.1.6.2. Use Case

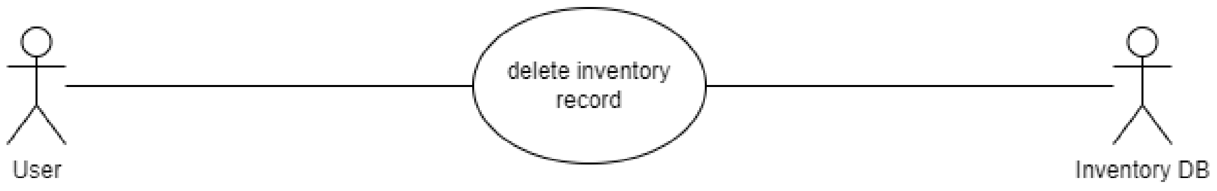
Scope

The scope of this use case is to allow user to delete inventory records from the system.

Description

This use case describes the interaction of user tries to delete inventory records from the system.

Use Case Diagram



Flow Description

Precondition

The user is logged into the website, being in the inventory menu.

Activation

This use case starts when a user being in the inventory menu on the website, clicks on the delete item button, with at least one item selected.

Main flow

1. The system shows a pop-up window with a list of selected item record to be deletes, with a confirmation message with confirm and cancel button.
2. The user clicks on the confirm button.
3. The user clicks on the cancel button. (See A1)
4. The system fetch information from the select items, create delete queries for the database.
5. The system could not connect to the database. (See E1)
6. The system could not delete the records from the database. (See E2)
7. The system successfully deleted the records.
8. The system closes the confirmation pop-up window and shows a new pop-up window stating the item records being deleted successfully, and the window close itself after 3 seconds.

Alternate flow

A1: User clicks on the cancel button

1. The system closes the delete confirmation window and refresh the inventory list.

Exceptional flow

E1: Could not connect to the database

1. The system shows an error message in the delete item window, stated that connection error to the database, suggest user to refresh the page.

E2: Could not delete records from the database

1. The system shows an error message in the delete item window, stated that error deleting items from database, suggest user to refresh the page.

Post condition

The system refreshes inventory list after the success deletion window closes and stay in wait state.

2.1.1.7. Requirement 7: Edit Inventory Record

2.1.1.7.1. Description & Priority

The user should be able to edit record from inventory after logged in, given the item record exists, this feature serves the function of allowing user to edit inventory records from the inventory system bind to their account.

2.1.1.7.2. Use Case

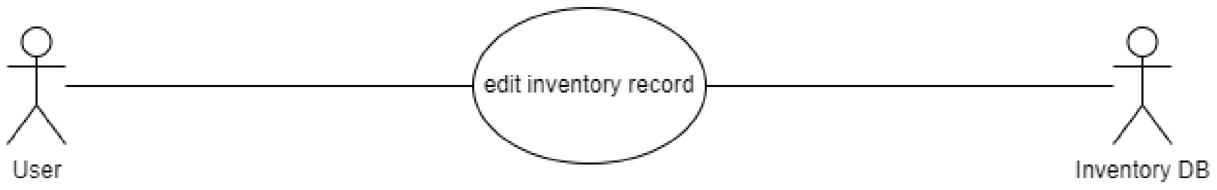
Scope

The scope of this use case is to allow user to edit inventory record from the system.

Description

This use case describes the interaction of user tries to edit inventory record from the system.

Use Case Diagram



Flow Description

Precondition

The user is logged into the website, being in the inventory menu.

Activation

This use case starts when a user being in the inventory menu on the website, clicks on the edit item button on the end of any item record.

Main flow

1. The system shows a pop-up window with a form that allows user edit item details with existing item details prefilled.
2. The system checks the information when user submit the form.
3. The system notices the form is incomplete or wrong entries such as alphabet in currency field. (See A1)
4. The system updates the item record to the inventory database when the form meets the requirement.
5. The system shows a message below the form, stating the item record is being updated successfully.
6. The system could not connect to the database. (See E1)
7. The system could not update the record into the database. (See E2)

Alternate flow

A1: Form is incomplete or wrong entries

4. The system shows a message in the add item form, stated that wrong entries or form incomplete.
5. The user enters item details again.
6. The use case continues at position 2 of the main flow.

Exceptional flow

E1: Could not connect to the database

2. The system shows an error message in the add item form stated that connection error to the database, suggest user to refresh the page.

E2: Could not add record into the database

2. The system shows an error message in the edit item form stated that error updating item into database, suggest user to refresh the page.

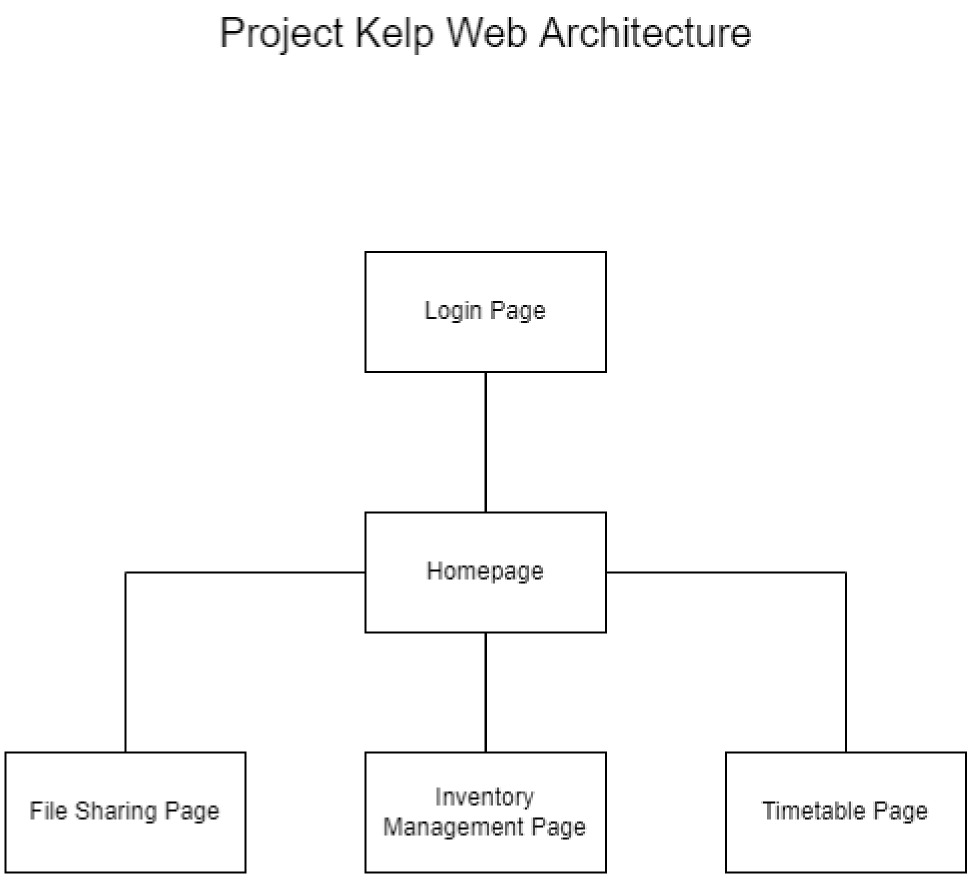
Post condition

The system refreshes inventory list after user closes the edit item form and stay in wait state.

2.1.2. Interface Requirements
This section discusses on the website interface requirements. The website interface should be written in English language, with font that not overlapping each other and stay within the divider. The interface should be able to render without refreshing the webpage as this website supposed to be a single page web application. The elements on the website should be rendering correctly without broken image or glitches.

2.1.3. Access Requirements
This section discusses about the access requirements for users to have access to the website. User should have no issue accessing the website from Ireland, and any countries that do not implement internet censorship firewall. The website should have an uptime of at least 95%. User token and session should be implemented, the website should have a fixed session expired time implemented, for example user should be able to log in for once and close the website, the next time user access the website within 5 minutes, the user should be able to not ask for login again unless manually logging out.

2.2. Design & Architecture



The project is currently still in early development phase, and because the project is develop using the SCRUM method, along with my supervisor agreed on only showcasing the login module for this mid-point submission, some parts of website components are still unclear at this stage. These information will mostly be decided on the start of each SCRUM cycle, but I will discuss on the known information in this section.

The project is aimed to implement as a single page web application after the login page, but it could be broadly separate into three webpage, which all serve its own function and feature. After the user log into the website via the login page, which screenshot is attached in the GUI section of this chapter, the user will be redirect into the homepage of the website, which is also the main layer for most activities in this web application. The homepage serves mainly the purpose of a welcome page, which nagivation bar on the left side of the webpage that allows the user to choose either the Inventory Management tab, File Sharing tab, or the Timetable tab, the new content should render within the same page without refreshing the webpage.

React js being the base of this project, it acts as the javascript library to build most of function on this web application project, with Google Firebase as the hosting and MySQL being the database to support the project.

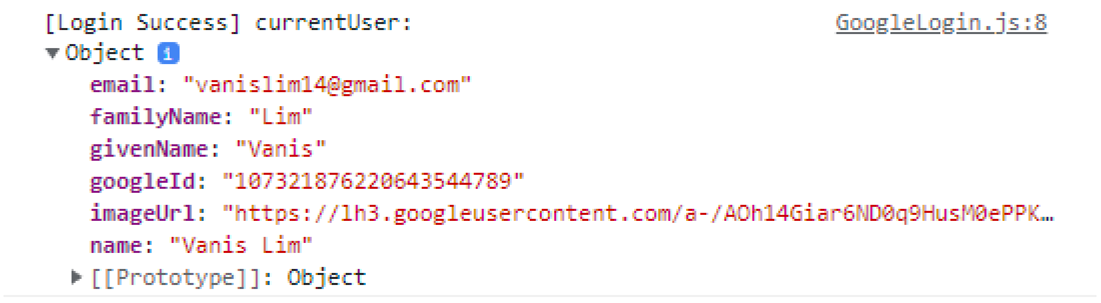
Pagination scroll will be implemented in the inventory management system instead of the more common infinite scolling that often use in popular single page application for example Facebook, the reason for this choice is that it allows user to have a better control and it puts less stress on the device when large amount of information being shown. Drag and drop feature is aimed to be implemented in the file sharing system, this enchance user experience and allow more efficient interaction.

2.3. Implementation

At this stage of development, there are no planned algorithms to be implement into the project. Most coding is done through React JS library, along with external library like react-google-login (react-google-login, 2021) and react-facebook-login (react-facebook-login, 2021) on npmjs.com, these packages brought convenient on building the web app, but it also requires time for further learning and reading through documentation.

2.3.1. Google Login Library

```
import React from 'react';

import GoogleLogin from 'react-google-login';

const clientId = 'censoredclientId.apps.googleusercontent.com';

function GLogin() {
    const onSuccess = (res) => {
        console.log('[Login Success] currentUser:', res.profileObj);
    };

    const onFailure = (res) => {
        console.log('[Login failed] res:', res);
    };

    return (
        <div>
            <GoogleLogin
                clientId={clientId}
                buttonText="Login"
                onSuccess={onSuccess}
                onFailure={onFailure}
                cookiePolicy={'single_host_origin'}
                style={{ marginTop: '100px' }}
                isSignedIn={true}
            />
        </div>
    );
}

export default GLogin;
```

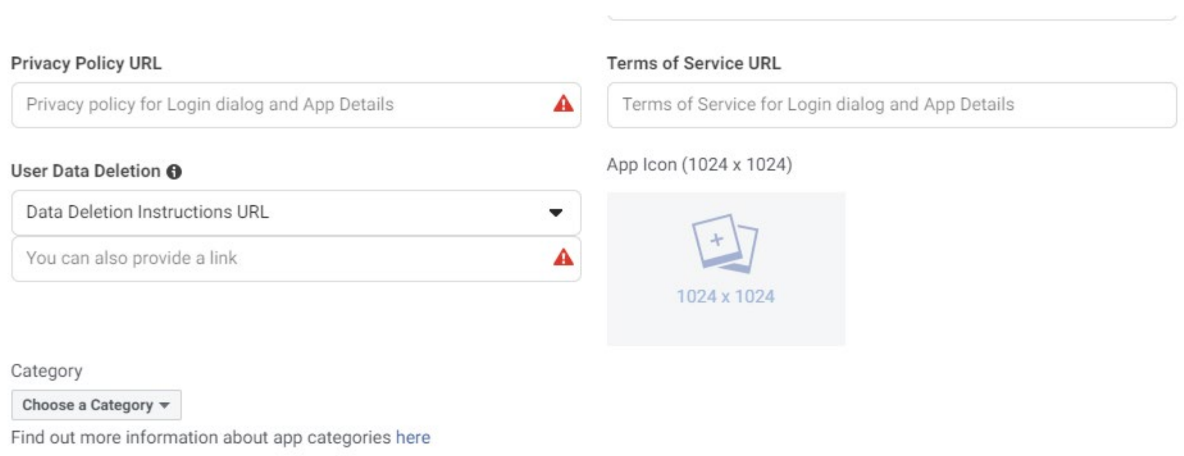


```
[Login Success] currentUser:                                GoogleLogin.js:8
▼Object i
    email: "vanislim14@gmail.com"
    familyName: "Lim"
    givenName: "Vanis"
    googleId: "107321876220643544789"
    imageUrl: "https://lh3.googleusercontent.com/a-/AOh14Giar6ND0q9HusM0ePPK…
    name: "Vanis Lim"
  ▶[[Prototype]]: Object
```

Google login library being one of the special libraries, for how simple efficient the code is. By only a few lines of code, the library allows the implementation of login via google account, and not only it works but also return many useful information like name, profile picture and email. As the system is not completed, above screenshot shows the google account's info being retrieved after logging into and showing in the browser console.

2.3.2. Facebook Login

```
/*
  const [login, setLogin] = useState(false);
  const [data, setData] = useState({});
  const [picture, setPicture] = useState('');

  const responseFacebook = (response) => {
    console.log(response);
    setData(response);
    setPicture(response.picture.data.url);
    if (response.accessToken) {
      setLogin(true);
} else {
      setLogin(false);
    }
  }

  return (
    <div class="container">
      <Card style={{ width: '600px' }}>
        <Card.Header>
          {!login &&
            <FacebookLogin
              appId="censoredappId"
              autoLoad={true}
              fields="name,email,picture"
              scope="public_profile,user_friends"
              callback={responseFacebook}
              icon="fa-facebook" />
          }
          {login &&
            <Image src={picture} roundedCircle />
          }
        </Card.Header>
        {login &&
          <Card.Body>
            <Card.Title>{data.name}</Card.Title>
            <Card.Text>
              {data.email}
            </Card.Text>
          </Card.Body>
        }
      </Card>
    </div>
  );*/
```


Privacy Policy URL

Privacy policy for Login dialog and App Details

Terms of Service URL

Terms of Service for Login dialog and App Details

User Data Deletion

Data Deletion Instructions URL

You can also provide a link

App Icon (1024 x 1024)

1024 x 1024

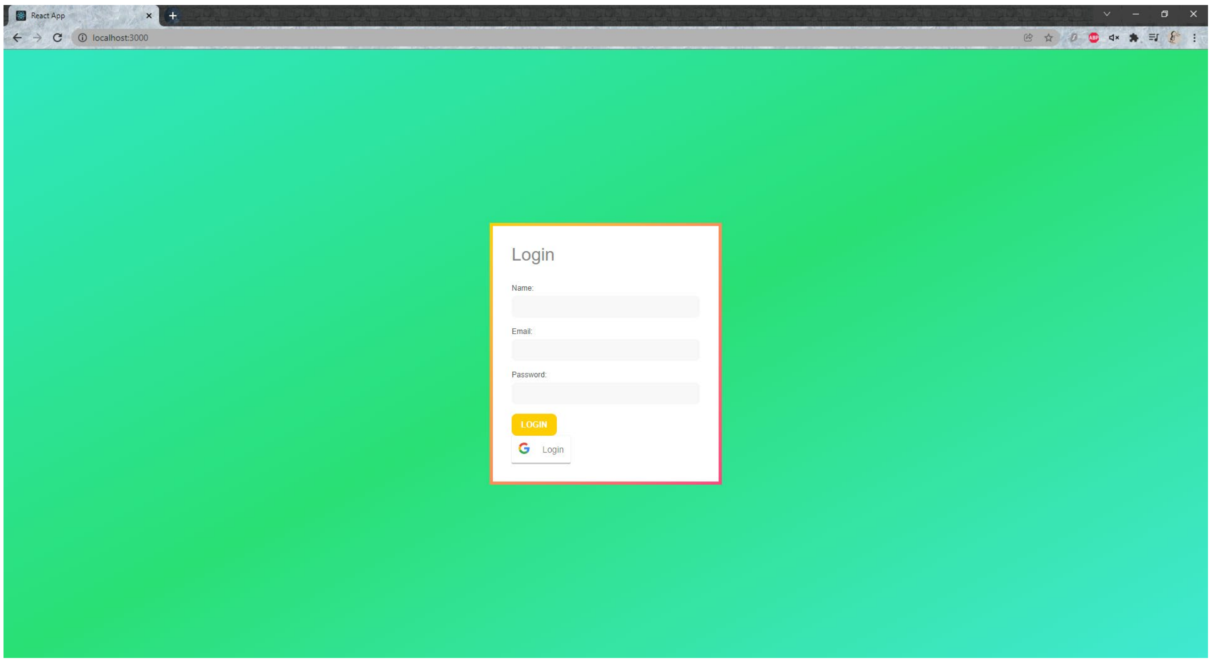
Category

Choose a Category

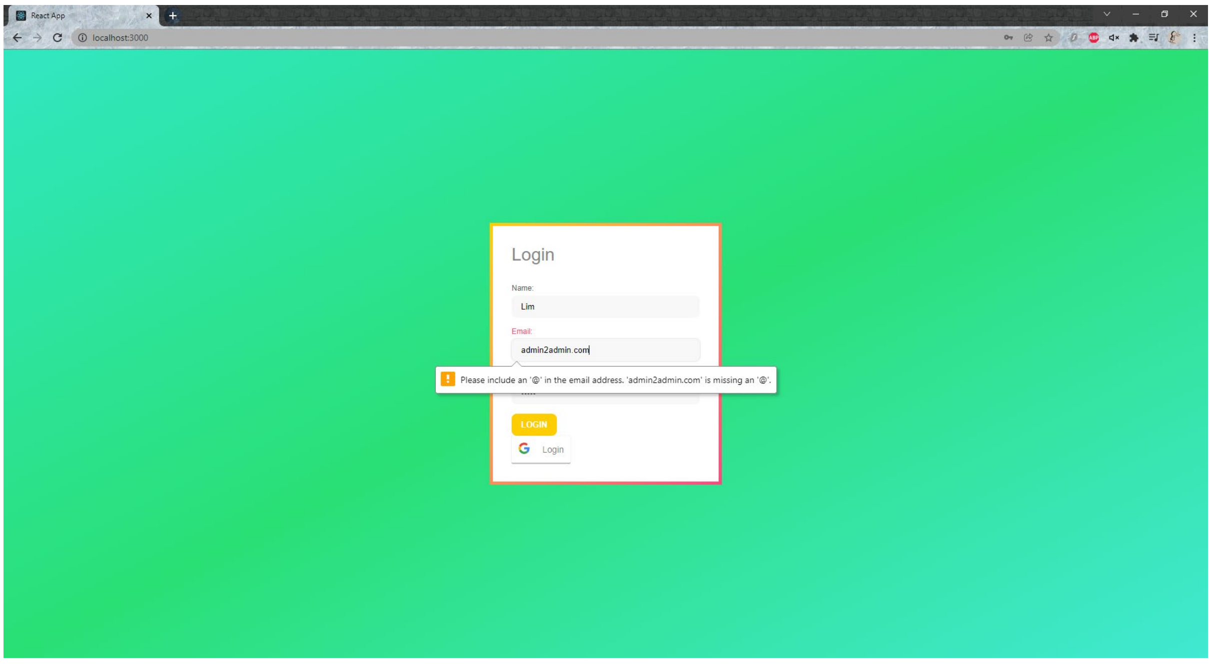
Find out more information about app categories here

Facebook login in the other hand, being a little troublesome to implement not because of its library but the privacy policy requirement, the login provider require developer to submit detailed privacy policy and user data deletion instruction before the api work in developer mode, shown in the screenshot above, which cause extended amount of work required to be done before the login option is available to be fully implemented on the website, due to the current development progress already falling behind on the initial project plan, the plan of facebook login implementation might be cancels.

2.4. Graphical User Interface (GUI)

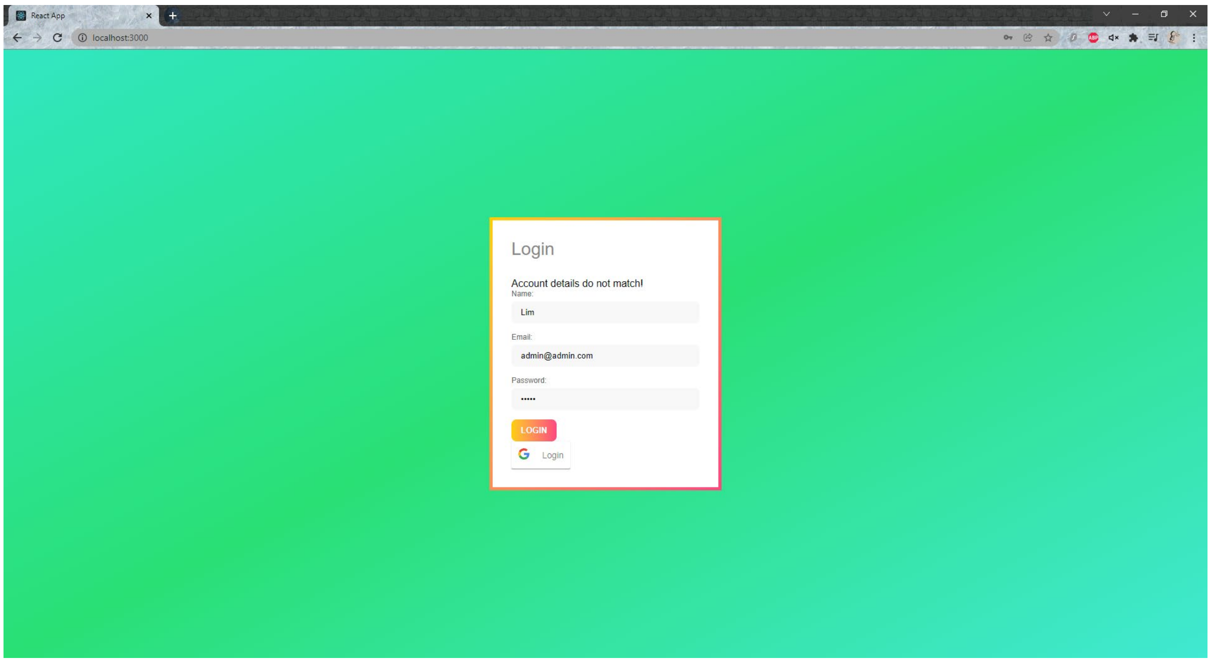


This is the login page to the website, users are required to enter their name, email and password to enter the website, google account login also being an option for them.



Please include an '@' in the email address. 'admin2admin.com' is missing an '@'.

If the user did not enter email in the correct format, a pop-up window will provide hint for the user to make correction on the email.



Login

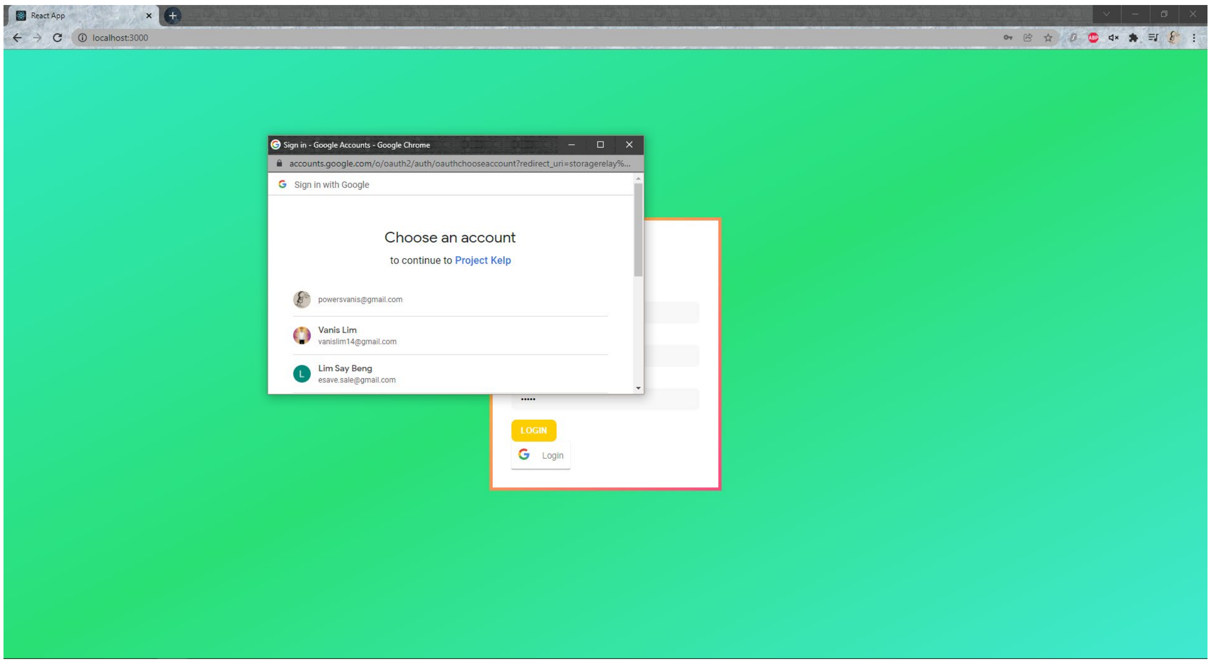
Account details do not match!

Name:
Lim

Email:
admin@admin.com

Password:
•••••

If the user enters the incorrect account details, a message will show at the top of the form to remind user the entered information is incorrect.



accounts.google.com/o/oauth2/auth/oauthchooseaccount?redirect_uri=storagerelay%...

Sign in with Google

Choose an account

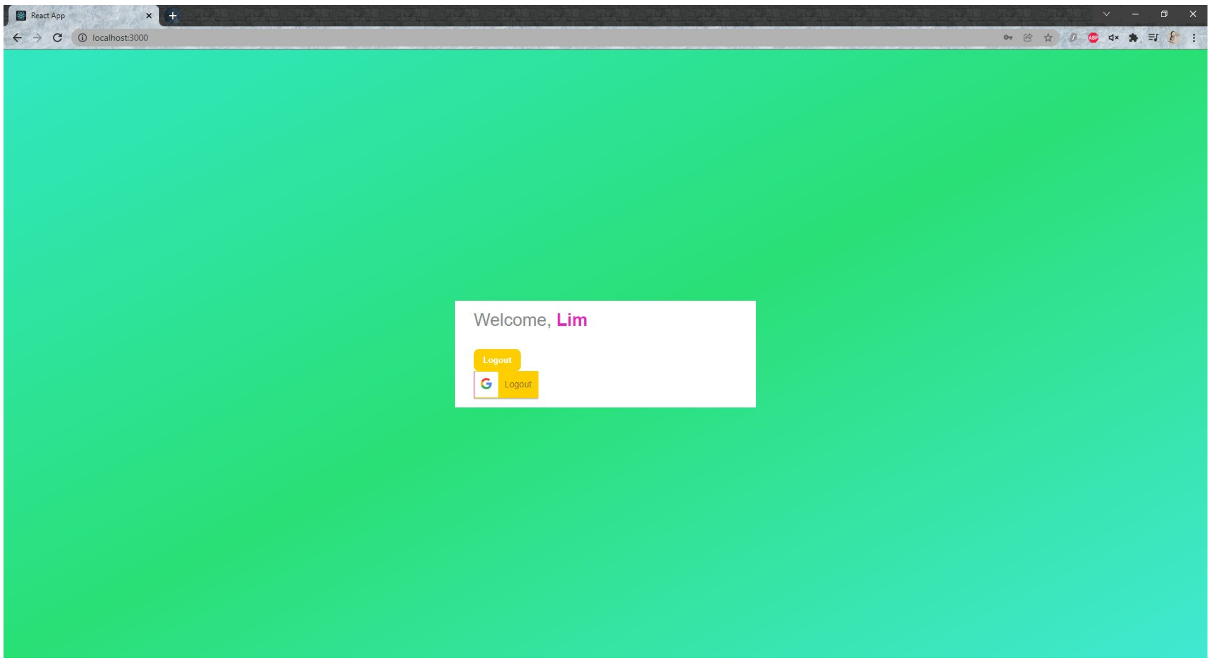
to continue to Project Kelp

powersvanis@gmail.com

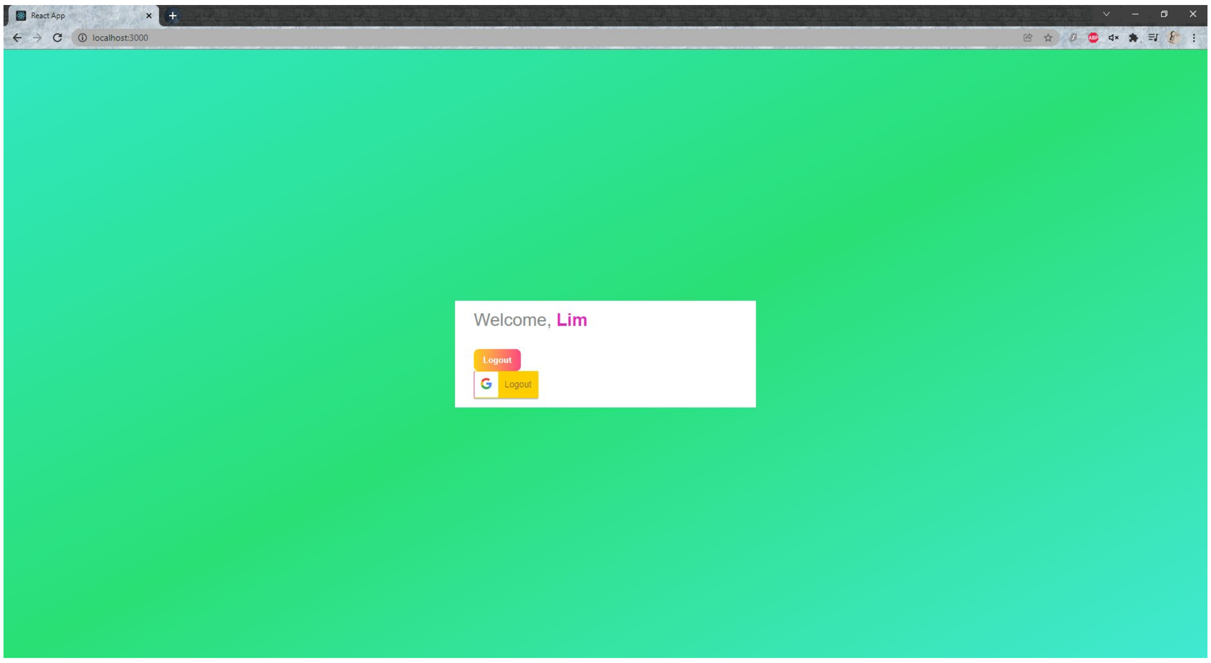
Vanis Lim
vanislim14@gmail.com

Lim Say Beng
esave.sale@gmail.com

When the user clicks on the Google login button, a new pop-up window will show up with the existing logged in google account on the browser, allowing the user to choose their selection of google account for logging into the website.



When the user enters the correct account information, they should be show will the homepage of the website, but with the website currently still in development, a temporary page will show up with the user's name and logout button.



When the user has their cursor hover on the logout button, it will show a different graphical effect and when the user clicks on the logout button it brought them back to the login page.

2.5. Testing

Application functional testing or unit testing will be performed near the end of each development sprint cycle, before the deployment of module, focusing on software bugs, function stability and accuracy. The test will be performed by using intended and not intended interaction with the application to check how the application react.

Integration testing will be near the end of whole project development by using the big bang testing method, due to modules in this project does not closely relate to each other.

Multiple system tests will be performed in between all planned project module development and after the project developement completed, including software testing (browser compatibility), hardware testing (graphical testing), and regional access testing.

Browser compatible test will perform after the deployment of each module, with extra random timed test to make sure the website providing supports to multiple browsers including Google Chrome, Microsoft Edge, Firefox and Opera.

Graphical testing will be performed near the end of project development and focus on testing how the website adjusts with resolution changes of the monitor, desktop and laptop on multiple monitors with different resolution including 1920x1080, 2560x1080, 3840x2160 (pixel) will be used to perform the test.

The project is not developed to be use on mobile phone and tablet in this phase, but spare time resources near the end of project development could be used for mobile devices graphical enhancement.

Regional access testing will be performed by using multiple Virtual Private Network (VPN) software to access the website. The test looking for result on ability to access the website from different region. Website response time will also be recorded but for reference only due to the nature of VPN. Only non-VPN connection response time will be considered and use as the standard for the test.

2.5.1. final submission update

Before the final submission, because of the limitations of Strapi CMS and limited time, many parts of the project does not able to be developed in time, and because of so, the testing will be focused on the Item List and its functions, including add item, edit item, and delete item.

The focus of the testing is to make sure the item list able to perform the intended function, for example the list should be able to show all currently existing item records, and in an easy-to-read presentation. The testing was done by opening the item list webpage and examining the shown result, for the result, the font is easy to read with a dark theme, showing the font in suitable size with light color, contrasted to the dark background color for ease of reading, and with the cursor hover over the description of each records, the longer text unfold itself for reading, and it collapse back into smaller form when the cursor moved away, this design allowed the item list to be not too crowded when there is many records.

The next is to test the edit and delete function on the item list. There are two buttons after each item record, one is edit and the other is delete button. The buttons are large and easy to recognise while not interfering with the reading of other text and having a contrast color to the background for easy to notice. When clicked on the edit button, a form will appear with the related item information already fetched in, this allow the user able to make modifications easily without needed to remember what the original information was and made mistakes. The form itself limits what type of information the user could enter, for example the name and description section allow the user to enter text and anything that could form a string, while the quantity, cost and retail price section only allows the user to enter numeric inputs. The last three inputs limit the format of numbers a user could enter for example user could enter decimal values for cost and retail price but should not be able to do that on quantity. But after testing, the user could still enter decimal value into the quantity section, but will not be able to update the form, which will lead user into confusion. In the future, error message should be showing when there is decimal value in the quantity section. User is also not able to enter any alphabet into the three numeric field, which is working as intended. And when user clicks on the submit button, the updated information is shown in the item list. But if the user clicks on the cancel button, the record for that item will remained unchanged when user was redirected back to the item list page.

The next function to test is the delete button on the item list. User can click on the delete button on each item record, and the website will promp a pop-up message to confirm if the user wants to delete the item record, when the user clicked on the confirmation button, the item record will then be deleted, while if the user clicked on the cancel button, the item record will remain in the item list. And after the user delete the item record, the next created item will not share the same item id, as the id is unique to each record whether it is deleted or not, this design avoided the possibility of causing confusion of multiple items sharing the same item id.

The add item function was tested by clicking on the add item button on the navigation bar on top of the webpage, and this leads the user to a similar form like in the edit item function, but without any prefetched information, as the purpose of this form is to add a new record into the list. Each input field perform the same as in the edit item form, the first two name and description input field allows user to enter alphabet data, while the last three inputs only allowed users to enter numeric information. And the submit button will add a new item record into the item list, while the cancel button return user to the item list webpage without adding a record to the list.

2.6. Evaluation

As the web application is develop in the form of single page webpage, search engine optimization (SEO) will not be part of the project evaluation due to the nature of single page website not working well with SEO algorithm. The evaluation of project will be done from three direction, including Usability, Content and Readability, and Asthetics, from (How to Evaluate the Quality of Your Website Design, 2021).

Usability means the performance of the website, for example how well the website performs with different devices, difference browsers. The usability will be measured by cross-browser support, website loading speed, hardware requirement to run smoothly, turn around time to server and did every feature works in their intended way.

Content and readability mean the overall graphic design for the website, how well the design perform on different monitor and screen resolution, how easy the font to be read by user, are the overall interface being easy to use.

Asthetics includes the overall theme and design of the web application, for example does the overall color matched in each section of the website, is the design being user friendly, and how efficient the layout to users.

2.6.1. final submission update

2.6.1.1. Development progress

In this section, the development progress of the project up until the final submission will be evaluated based on the the development of the website. The website was initially planned to have three main features, including the inventory management feature, the timetable feature, and the file sharing feature. During the midpoint submission, there is a prediction that it might not be enough time for the development of of the latter two features, and this is proven to be true on the final submission. The detailed reason of the development delays will be explained in the conclusion update in the next chapter. The progress of the project is the inventory management system been partially completed, working item list with edit and delete function, along with add item function. The seach function for the item list is not completed due to the underlying issue with Strapi CMS mentioned later, the same happened to the login system for the website. Even though a login system was previously developed for the project, intergration between it and the Strapi CMS failed multiple times because of the confused documentation on Strapi official documentation.

To conclude, the project only has a partially completed login system and inventory management feature, with not enough time for the development of timetable and file sharing feature.

2.6.1.2. Functionality

In this section, the usability, content, and readability and asthetics of the project will be briefly evaluated. Firstly, the usability of the website was only tested on a single device because of the project was not fully developed and the issue with Strapi CMS, the project is not hosted on cloud and only run on local hosting. The website run very smootly on the machine of Ryzen 9-3900x, Windows 10 and both Microsoft Edge and Google Chrome browser, with almost instant loaded. The completed features on the website ran as intended on multiple browsers with no hiccup.

Next is the content and readability evaluation. The website's design was changed multiple times in the development process and ended with a modified version of drop-in CSS called Water.css along with custom CSS properties. The website appeared as a dark themed background with light color font text for ease of reading, the website appearance is easy to read with great contrast between background and contents, and suitable font size. There is partially mouse hover over graphic changes that allows the user to know where the cursor is pointed at. The website is also partially responsive with tested on full hd and WQHD resolution with both having suitable sized CSS design.

For the asthetics section, most already mentioned above, the website runs a dark-themed design with dark blue colors and light color text. This design is easy to read yet contribute low stress to eye because of the low brightness in unnecessary area. The website ran the same theme across multiple webpages and that makes the website design merged as whole and each pages fit together. The buttons design was also on spot interm of size and color, being easy to notice while not blocking the content on the webpage.

3.0 Conclusions

Being a project with short development time of six months and by only one developer, many decisions need to be made around this condition, the good side of it being many parts of the project is very flexible and easy to adjust, bad side being it is hard to develop large modules and development progress could easily be delayed if anything happened to the only developer.

Small scale project like Project Kelp provides the advantage of flexible development, SCRUM development method also allows many features could be added mid-development without too much concern, this project is not funded and that also allows decision being made without external interference as there is no invester.

Disadvantage of this project being there is not enough financial support for marketing, and implementation of paid services such as paid APIs and paid server hosting. Another disadvantage being lact of refinement, with the short development window, by the time the project is close to completion, there might not be enough time for detailed testing internally by developer or by external tester.

Strengths of project Kelp being its small footprint and easy to run/learn compared to many other similar software on the market. Project Kelp provide only basic feature on inventory management which is enough for small and medium business owner which used to only work on traditional management way such as Microsoft Excel and other spreadsheet style software or even by pen and paper. Many inventory management software on the market being very feature-rich, but also have steep learning curve and often kept behind paywall, by the time of writing project Kelp is mean to be free, which is convenience for small business owners because they could try it out for free.

On the other side, project Kelp do have some unavoidable limitations mostly due to the scale of development. The inventory management system will be lack of many features, report generation, export as spreadsheet format being a few that could be named, the website itself not very refined compared to many large software on the market being another. These are the limitation of the project in the current phase but could be resolve in the future development.

3.1. final submission update

The project had faced the issue of slow development and many part of the project does not able to be completed in time. The reason of it being the only developer on the project does not have much experience in front-end web development, faced many issues during the development progress and more importantly the underlying issue with Strapi CMS. The CMS was selected during the development process around the midpoint submission period. The CMS brought many benefits to the website including the ease of development for the inventory management system, but it also brought two major issues that contributed as the main reason for the delayed development of the project. The first issue being Strapi CMS is in the process of converting from their version 3 into version 4, and the conversion brought many changes to the existing structure of code including the totally different syntax for their database, which require many testings and modification during the development. This also linked to the second issue which is the major letdown of the development progress. The CMS was not commonly used by many and there were not enough examples and usable tutorial beside the blog from Strapi itself, and the issue being Strapi decided to update all their existing example and tutorial to version 4 simultaniosly, but at a slow speed with live changes instead of a separate release for version 4, that caused almost all existing tutorial and examples to be both unusable for version 3 and version 4, same issue being on the official github project, with multiple developers complained on the problem of Strapi providing unusable examples that were not up to date. And that leads to the problem of not having any working tutorial and references on the internet for learning purpose and thus caused the slow development progress for the project.

In conclusion, the approach of development the project based on Strapi CMS is not a right decision for the time, as the CMS is currently in ongoing conversation with not enough documentation for a junior developer to develop alone. The cost of this decision caused the project to have a slow development progress and limited reference to work with, and it also caused the limited options of API integration because of Strapi V3 and V4 having a different API library with the V4 library has yet to release on the date of writing, caused many feature that available previously being missing.

4.0 Further Development or Research

The project was initially planned to have inventory management system, timetable system, staff management system, file transfer system all in one as a business tools envelope, but suggested by my supervisor, there might not be enough time to develop all these features so the primary focus as of current stage would be on the inventory management system alone. After the system development finished and deployed, the next step would be for the other systems mentioned above, given there is enough time.

And after these systems being developed, the next direction of the project with enough time and resources, would be implementing marketing elements such as advertising the web app on social platform and perform search engine optimization (SEO) for the website as single page applications are known to be not SEO friendly due to crawler algorithm. Along with developing user tier list, this allows the project to start generating financial revenue, with the lowest being free trial tier that allows new users to try out some features of the web app, followed by lite tier and business tier that target their specific audience respectively, differentiate by enabled features and number of simultaneously online user, and computing resources like cloud storage limits.

4.1. final submission update

As mentioned in previous chapters, the project was only partially completed the development on login system and inventory management system, with the timetable feature and file sharing feature not started yet. The next step of the project should be focusing on search of available resources that could implement the login system into the Strapi system, with that being done, the next step should be completing the search function of the item list in the inventory management system, along with some minor quality of life implementation for example error messages for add new project and edit project form for the numeric input field section.

With the above being completed, that conclude the basic functions for the website and the inventory management feature. The next phase of the project could be in two different directions, the first being to focus on the deployment of the website onto cloud hosting. And the second option being continue the development of the project on timetable feature and file sharing feature before deploying the project onto cloud hosting.

5.0 References

[1] npm. 2021. *react-google-login*. [online] Available at: <https://www.npmjs.com/package/react-google-login> [Accessed 15 November 2021].

[2] npm. 2021. *react-facebook-login*. [online] Available at: <https://www.npmjs.com/package/react-facebook-login> [Accessed 15 November 2021].

[3] Dynomapper.com. 2021. *How to Evaluate the Quality of Your Website Design*. [online] Available at: <https://dynomapper.com/blog/19-ux/188-how-to-evaluate-the-quality-of-your-website-design> [Accessed 10 December 2021].

[4] DEV Community. 2022. *Search Bar in React JS!*. [online] Available at: <https://dev.to/salehmubashar/search-bar-in-react-js-545l> [Accessed 10 March 2022].

[5] Strapi.io. 2022. [online] Available at: <https://strapi.io/blog/nextjs-react-hooks-strapi-auth-4> [Accessed 10 March 2022].

[6] GitHub. 2022. *GitHub - strapi/strapi-examples: List of examples using Strapi*. [online] Available at: <https://github.com/strapi/strapi-examples> [Accessed 15 February 2022].

[7] W3schools.com. 2022. *How To Create a Filter/Search Table*. [online] Available at: <https://www.w3schools.com/howto/howto_js_filter_table.asp> [Accessed 17 February 2022].

[8] React-bootstrap.github.io. 2022. *React-Bootstrap*. [online] Available at: <https://react-bootstrap.github.io/components/navbar/> [Accessed 28 March 2022].

[9] GeeksforGeeks. 2022. *Create a Responsive Navbar using ReactJS - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/create-a-responsive-navbar-using-reactjs/> [Accessed 29 March 2022].

[10] React-table.tanstack.com. 2022. *Examples: Filtering*. [online] Available at: <https://react-table.tanstack.com/docs/examples/filtering> [Accessed 25 February 2022].

[11] Watercss.kognise.dev. 2022. *Water.css*. [online] Available at: <https://watercss.kognise.dev/> [Accessed 6 April 2022].

[12] Medium. 2022. *Build a Solid and Secure Login Workflow in NextJS with Strapi*. [online] Available at: <https://javascript.plainenglish.io/build-a-solid-and-secure-login-workflow-in-next-js-with-strapi-part-1-concept-and-setup-5155ebe622bb> [Accessed 3 May 2022].

6.0 Appendices

6.1. Project Proposal

LimHongChun_Proposal BIS 2021.pdf

Double click on icon to open file after enable edit

6.2. Reflective Journals

LimHongChun_Combined_ReflectionJou