Danone Display Only Project

The University of Auckland Micro-Internship

Mentor:

Evgeniia Golovina

Team 9

Mazyar Zarepour

Jason Puk

Jabin Chen

Jerry Nguyen

Wenjie Li

Mengzhe Zhao

July 2023

Table of Contents

1- Executive Summary

Danone Nutricia Specialized Nutrition is a company dedicated to producing specialized foods for infants and young children, driven by their mission of bringing health through food to as many people as possible. With manufacturing, distribution, and development facilities across three sites in New Zealand, including Auckland and Balclutha, they are known for their market-leading brands of infant milks and toddler supplements. As part of their commitment to nature and digital manufacturing, the company is actively pursuing the digitalization and digitization of their processes to improve efficiency and sustainability.

Danone Nutricia utilizes a microservice-based plant management system. This system provides numerous benefits, such as scalability, flexibility, and maintainability. It allows for the independent development and deployment of various microservices, ensuring that each component can be updated or added without disrupting the entire system.

1-1 Current Situation and Challenges

Currently, Danone faces challenges related to paper printing in their operational processes. The company relies on manual printing and distribution of paperwork, leading to inefficiencies and environmental concerns. The need for physical documents poses obstacles such as delays in accessing up-to-date information, the risk of errors in handling and distributing printed files, and the generation of paper waste. Recognizing the need for improvement, Danone is actively addressing these challenges by pursuing digitalization and digitization initiatives to streamline their processes, reduce reliance on paper, and enhance overall operational efficiency and sustainability.

1-2 Objective(s)

The objective of this project is to introduce a "Display Only" microsystem within Danone’s plant management system. This microsystem consists of a back-end and front-end component and aims to digitalize the process of viewing PDF files when the latest orders on the shop floor are requested. This eliminates the manual process of printing and distributing PDF files. Instead, the selected PDF files will be displayed on a user-friendly website accessible through various devices such as iPads, mobile phones, and laptops

1-3 Main Outcomes

The three main outcomes of the project are:

1- Figma Design with Prototype:
   - Gathered requirements for the organization.
   - Created visual designs.
   - Revised designs based on team feedback.
   - Developed a functional prototype.

2- Backend GitHub Repository with Description:
   - Initialized the GitHub repositories with the appropriate project structure.
   - Designed the backend program with a functional modular architecture in Python/ Flask, considering structural testability and credibility.
   - Developed the backend MVP.
   - Provided clear descriptions.

3- Frontend GitHub Repository with Description:
   - Initialized repositories for the frontend with the appropriate project structure.
   - Designed the frontend architecture using React.
   - Developed the frontend MVP.
   - Provided clear descriptions.

2- Methodology

During the preparation of the project plan, the team engaged in active discussions to determine the necessary steps and timelines for project completion. The identified steps were then followed accordingly. The Figma design process was initiated, while at the same time, an initial back-end architecture was devised and coding started. The front-end development in React was initiated in parallel with the Figma designs. As the Figma designs were approved by the team, necessary revisions were made to the front-end to align with the finalized designs. Meanwhile, the back-end was continuously developed and debugged. In the final phase, both the front-end and back-end were thoroughly debugged and connected. Throughout the entire process, the project documentation, descriptions, and guides were consistently updated to reflect the progress and changes made. The following subsections explain each main part of the project in more details.

2-1 Design and Prototyping in Figma

Figma, a tool for user interface and user experience design, is used in our first stage of project, which serves as a visualisation tool for the final product of the web application,

and sets the framework for both backend and frontend development. Also, the prototyping features in Figma can create interactive flows that explore how a user may interact with the designs.

Our client, Danone, provides a Figma template for our design, which contains the specific fonts (in this project, “Brush Up” and “Bariol” are used which had to be downloaded and installed in our PCs) and colours our client requires, as well as raw components that make up the web application. For the “Display Only” project, four additional components are created, namely the landing page, “Display-Only” pages for Production Order (PO), Packing Spec (PS) and Shopping Layout (SL).

The landing page is the main page that lists all the current file names of PO, PS and SL. As confirmed by the client, there would be two production lines, each of which contain a PO number (9-digit) for PO and a SKU number (6-digit) for PS and SL. These numbers are contained in the .dat file under the folder of each production line. The file names are targeted to be retrieved from the frontend. Two green buttons are also included next to the names of production lines, which are used to fetch the file names of the latest order. If PO, PS and SL cannot be found, instead of the file names, a message “No PDF is retrieved!” will be displayed.

There are also three “Display-Only” pages for PO, PS and SL. Each page contains a title specifying the production line and the document type, followed by the file name. The next line contains three hyperlinks, the first of which (“PDF MENU”) will be directed to the landing page, the rest of which will be directed to the other “Display-Only” pages. Finally, the pdf file is displayed below these hyperlinks.

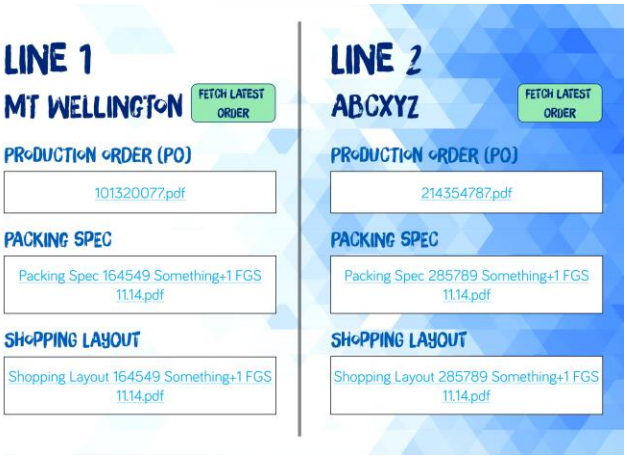

Figure 1a: Landing Page

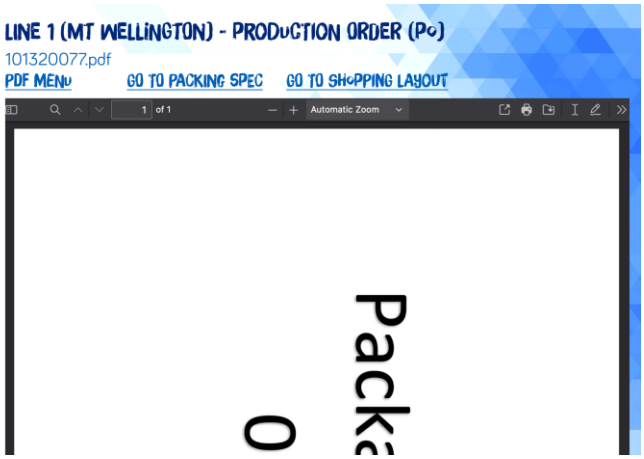

Figure 1b: “Display-Only” page for PO under production line 1. The “display-Only” pages for both PS and SL are similar to this.

After that, the components are grouped together to form the desired webpages. There are three sets of designs, as the client requires to display pdf in Laptop (with dimensions 1280x800), iPad (768x1024) and Mobile (568x320). The webpage contains a header (nothing changed from Danone's template), a lateral bar (added "Display Only"), a small arrow button in the bottom (from Danone's template), and the new components as described above.

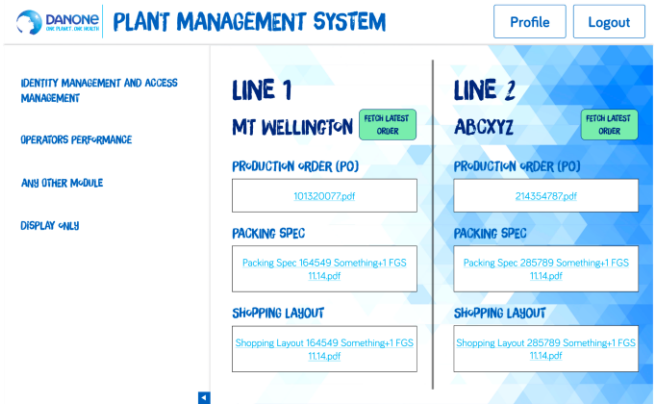

Figure 2a: Full design of landing page, with the lateral bar

Figure 2b: Full design of landing page, without the lateral bar

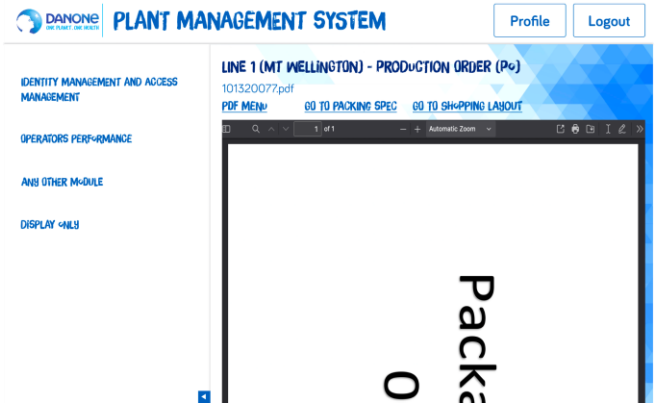

Figure 2c: Full design of a "Display-Only" page (PO under line 1), with the lateral bar

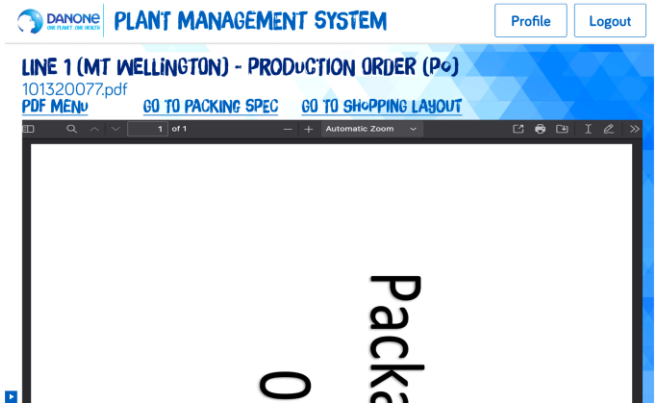

Figure 2d: Full design of a "Display-Only" page (PO under line 1), without the lateral bar

Finally, the prototyping allows a simulation of the flows when users click on the buttons. In Figma, the prototypes are shown as an arrow from the source to the destination. Here we have to consider all possible choices of (source, destination) pairs. For example, clicking the "Display Only" button in the lateral bar will always be directed to the landing page (Figure 2a), while clicking the small arrow button in the lateral bar in the landing page (Figure 2a) will effectively close the lateral bar (Figure 2b). In our Figma prototyping, we

only show the flows using Production Line 1 as basically it will be the same for Production Line 2.

These designs and prototypes provide the roadmaps for both the frontend and backend development, which will be detailed in later sections.

2-2 The backend MVP

Based on the initial project brief, the following steps were taken:

1- the back-end code structure is devised which significantly would help with the coding process. The modular flow-chart is shown in figure 3.

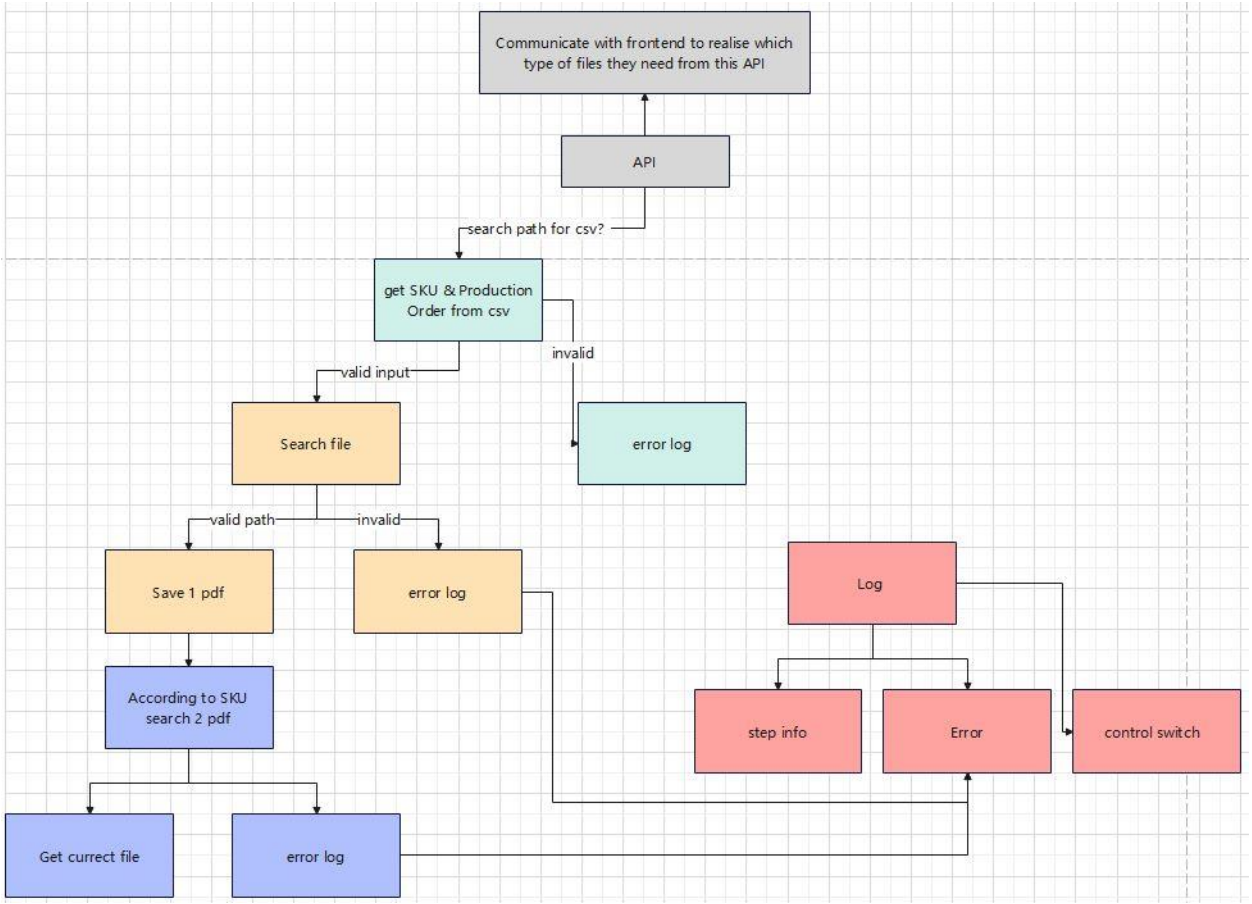

Figure 3: Initial Architectural Design of the Backend

2- Modular coding

- Import necessary modules and libraries.
- Create a Flask application object.
- Set up Cross-Origin Resource Sharing (CORS) to allow requests from a specific source.
- Define the get_po_pdf_path function to retrieve the path of a Purchase Order (PO) file.
- Define the get_sku_pdfs_path function to retrieve the paths of Stock Keeping Unit (SKU) files.
- Define the create_filepath_dict function to construct a dictionary containing the file paths.
- Define the build_filepath function, which builds the file paths based on the given PO and SKU and returns them as a JSON response.
- Define the get_homepage function as an API that returns a simple string as a response.
- Define the get_file_paths function as an API that retrieves production data from the Orders.dat file and calls the build_filepath function to construct the file paths and return the response.

Installation Guide:

- Open a terminal or command prompt.
- Navigate to the file path of the "flaskapp" directory in the terminal using the cd command.
- For Windows users:
  Set the FLASK_APP environment variable by running the following command:

```
set FLASK_APP=run.py
```

Start the Flask application by running the following command:

```
falsk run
```

-For Mac users:

Set the FLASK_APP environment variable by running the following command:

```
export FLASK_APP=run.py
```

Start the Flask application by running the following command:

```
flask run
```

The program will start running on a local development server. You can access it by opening a web browser and navigating to http://localhost:5000.

2-3 The frontend MVP

In line with the Figma designs, we have used the React to create the main page, landing page and created a responsive web. The purpose is to design a plant management system interface that is able to show the reuired PDF files.

Here's a general summary of the steps we took:

- We imported the necessary images for the background and logo.
- We created the basic structure of the page using HTML and inline styles in JSX.
- We created different components of the designed pages in React.
- Finally, we made the page responsive by using percentage values and vw and vh units for font sizes and dimensions, ensuring the elements adjust according to the viewport size.
- The purpose of these steps is to create a visually appealing and responsive interface for the Display Only micro-system. A sample of the created page based on the Figma designs is shown below.
- Finally, we debugged the backend and frontend and connected them to each other.

In the following we explain the created platform using some screenshots. The platform is developed based on the approved Figma designs, Danone font and color requirements.

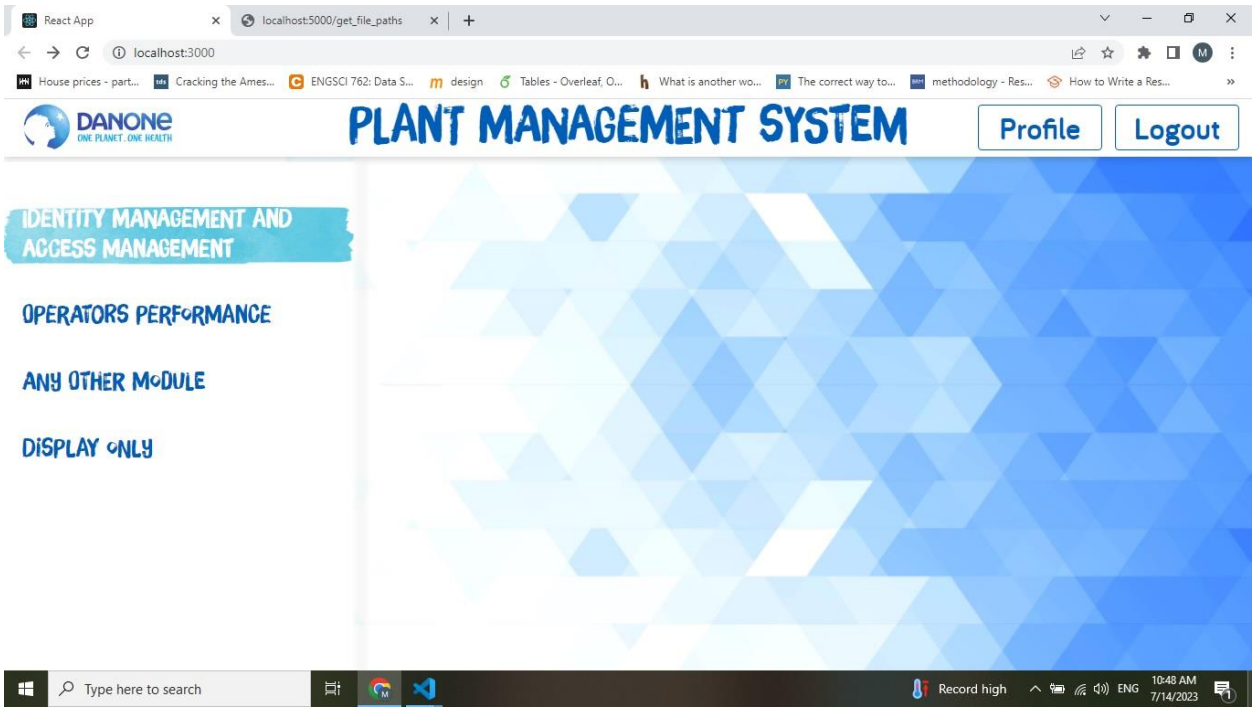

Fig 4: The navigation bar and main page

Figure 4 shows the navigation bar in the main page based on Danone plant management system format.

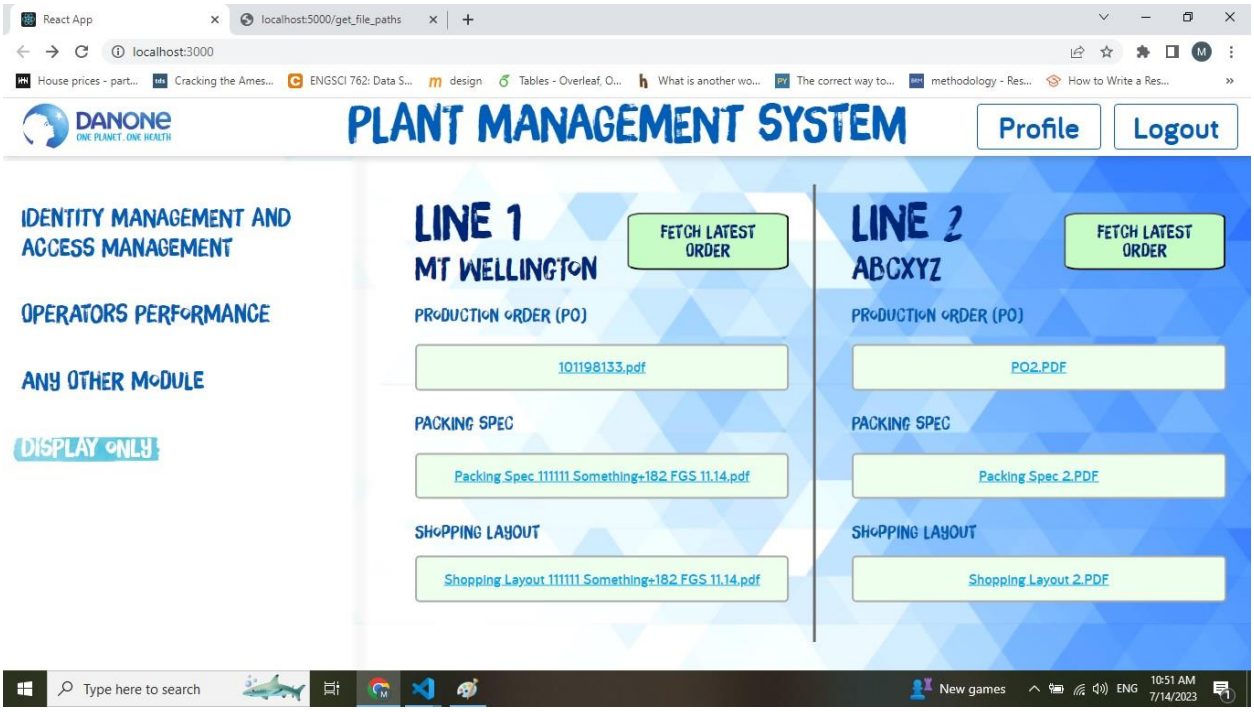

Fig 5: DISPLAY ONLY

Figure 5 shows the DISPLAY ONLY details. Once the DISPLAY ONLY is clicked the list of latest PDFs are shown for each production line. The name of each button is dynamic and is sent via a JSON file when the API is called. So, when the order is updated the name of the buttons would change. Also, the “Fetch Latest Order” button refreshes the pages to fetch the latest order details from the back-end.

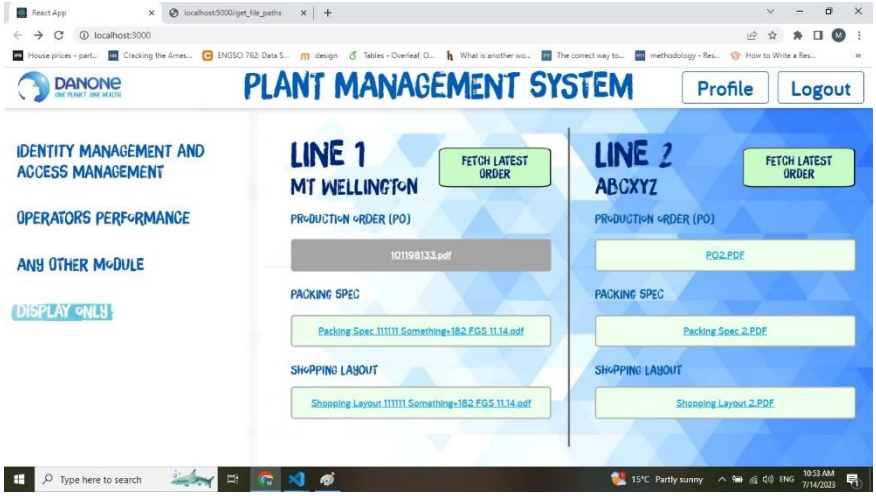

Fig 6: Button style

Figure 6 shows the button style change when you hover over the button.

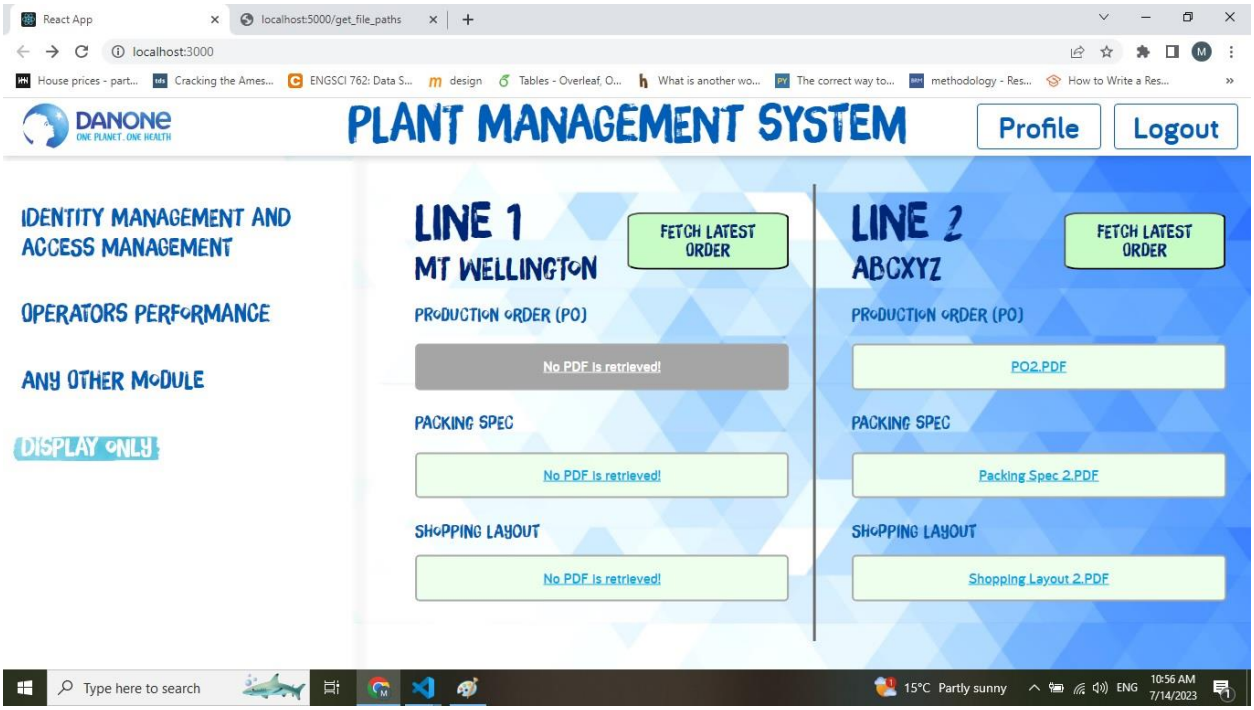

Fig 7: Button style

Figure 7 shows what happens if the right JSON file is not received from the backend. Instead of popping up an error box, the buttons simply would show “No PDF is retrieved!” and the buttons would become unclickable.

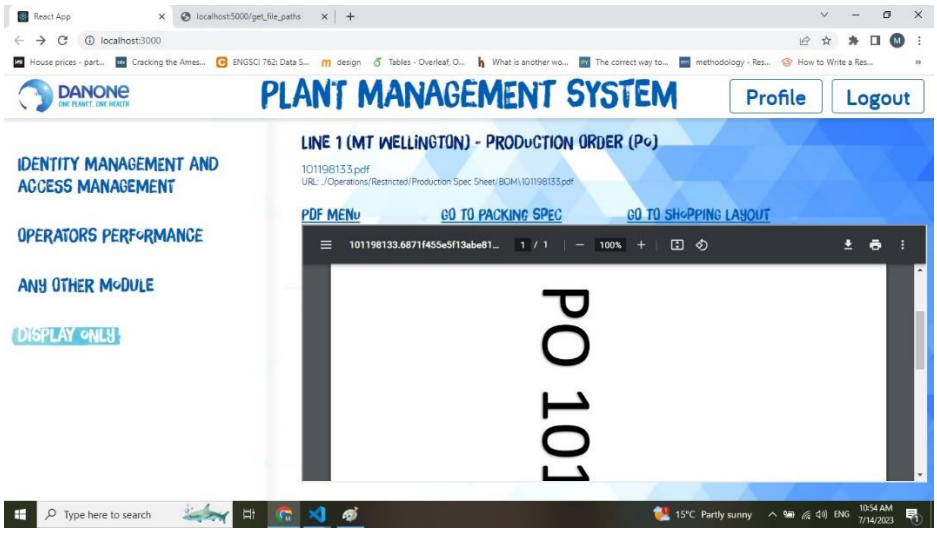

Fig 8: The PDF file

Figure 8 shows page when a PDF button is clicked. The name of the pdf file is showed on the top of the page and the name is dynamic. The URL address below the PDF name is temporary and will be removed from the final version. It shows the PDF URL that we receive from the back-end. And finally, the PDF viewer shows the PDF file. The style of PDF viewer is subjective and can be changed based on Danone's request.

3- Areas of Improvement

Throughout the course of this project, significant progress has been made in the development of the Display Only microsystem, utilizing Figma for design, React for front-end implementation and Python/Flask for the backend. However, there are still areas for further improvements and suggestions to enhance the system's functionality and user experience. One area of improvement is the logging mechanism. While the current code incorporates logging, additional enhancements can be made, such as implementing different log levels, formatting, and directing the output to a specified location. These improvements would aid in better tracking and debugging of the application. Furthermore, input validation can be strengthened to ensure the robustness of the system. Although the code includes basic length validation for PO and SKU inputs, additional validation logic, such as format validation, can be implemented. This would enhance the system's reliability and prevent potential errors caused by incorrect input data. In terms of file path configuration, it is recommended to make the file paths configurable and read them from a separate configuration file. This flexibility would allow for easier management and adaptation of file paths in different environments, enhancing the system's adaptability and maintainability. Additionally, conducting comprehensive unit testing is vital to ensure the code's quality and stability. By writing unit tests for each function, including boundary condition testing, potential issues can be identified and addressed early on, leading to a more reliable and robust system. To further improve the platform's functionality, it is suggested to implement a delete button that allows operators on the shop floor to remove completed orders from the system. This addition would help avoid conflicts and provide better order management capabilities.

4- Conclusion

In conclusion, the implementation of the Display Only microsystem has shown promising results in digitalizing the process of viewing PDF files on the production shop floor. The utilization of Figma for design and React for front-end development and Pyhton/Flask have facilitated the creation of visually appealing and responsive interfaces. We also suggested some areas of improvement for the project. By addressing these areas, the system can achieve higher levels of efficiency, reliability, and user satisfaction, aligning with Danone's commitment to

digitalization and continuous improvement in their plant management processes. In what follows we briefly summarize the whole process.

The project involved the development of a Display Only microsystem for Danone Nutricia's plant management processes. The aim was to digitize the process of viewing PDF files on the production shop floor, replacing the manual printing and distribution of paperwork. The backend part of the project played a crucial role in enabling the seamless functioning of the Display Only microsystem. It leveraged Python Flask, a microframework for web development, to build the back-end functionality and API endpoints. The backend incorporated various modules and libraries, ensuring the smooth integration of different components. The project also utilized Figma, a user interface and user experience design tool, to create visual designs and prototypes that served as a blueprint for the development process. React, a JavaScript library, was employed for the front-end development, ensuring a responsive and interactive user interface. The system's main components included a landing page, display-only pages for production orders, packing specifications, and shopping layouts. The pages were designed to dynamically retrieve the latest order details and display them in a user-friendly manner. The integration of the front-end and back-end components facilitated seamless communication and data flow.