|  |
| --- |
| School of Software Design and Data Science |
| **Full stack JS App** |
| WEB422 - Project |
|  |
| **Daniel Park, Angela Marie Tangudin Chua, Amany Omar**  **Submission Date** |
| **4/15/2025** |

|  |
| --- |
| This document explains how to build a frontEnd dashboard that consume backend API …………………………. |

Table of Contents

[Task 1: Pick Your Store and Product/Service Inventory 2](#_Toc174091863)

[Task 2: Project Planning, Wireframing, Layout, Dynamic Content 3](#_Toc174091864)

[Task 3: Authentication/Authorization 4](#_Toc174091865)

[Task 4: "Favourites" functionality 5](#_Toc174091866)

[Task 5: "Route Guard" functionality 6](#_Toc174091867)

[Task 6: deployment 7](#_Toc174091868)

[Bonus questions 8](#_Toc174091869)

[Project planning/sharing tasks 9](#_Toc174091870)

[Project performance check 10](#_Toc174091871)

[Summary 11](#_Toc174091872)

# Task 1: Pick Your Store and Product/Service Inventory

(Describe the major steps for identifying product/service/API)

For this project, we chose a marketplace model that offers a variety of Pokemon cards. The major steps for identifying our services, products, and API were:

1. **Initiation:** We brainstormed multiple different ideas and finalized on creating a marketplace for Pokemon cards as it was a shared interest and we felt that it was a niche product to work with.
2. **Research:** We discussed and researched through a variety of API’s and products that we could use for our Pokemon marketplace but ended up finalizing and choosing the pokemon TCG API. This API provided all of the information we needed along with data sets of the products (cards) we were going to list.
3. **Service Inventory:** We defined our product inventory based on the available data and information from the API, making sure that our inventory covers necessary details such as name, image, description, prices, and categories for each card.

# Task 2: Project Planning, Wireframing, Layout, Dynamic Content

(Describe the project planning (Gantt Chart) and major steps for designing the layout (Figma) and visualizing data using Card design)

1. **Project Planning:** We made a Jira Kanban board to keep track of tasks to do and completed tasks, assigning tasks to specific members of our team.
2. **Wireframe:** Using Figma we created wireframes for our website to create a basis of how our UI should look.
3. **Dynamic Content:** 
   1. Homepage: We followed the wireframe layout for our homepage and other pages, displaying featured cards from our inventory and showing them in a responsive card component layout.
   2. Card Page: Displayed all of our cards in the dataset using pagination, cards can be searched by category.
   3. Individual Card Detail: Made using next.js dynamic routing card/[ID], these pages show full details of the cards and allows for favouriting.
   4. Pagination was done to limit the number of card items per page

# Task 3: Authentication/Authorization

(Describe the major steps on how to make JWT user authentication in the app)

**User Registration:** In register.js and user.js a new user submits their username and password into the form, meanwhile in the back end we hash the password using bcryptjs and then we save it to our MongoDB database. In the front end register.js will send the registration request with registerUser() in authenticate.js.

**User Login and JWT Token:** The front end will send a POST request with the username and password to our api/user/login using the authenticateUser() function.

If this is valid the backend make a JWT token, and the token is returned and stored in local storage.

**Session State and Atoms:** Once a user is logged in successfully the app sets the atoms states loggedInAtom, favouritesAtom, searchHistoryAtom by using getFavourites() and getHistory().

**Authorization:** For our protected routes like getting the users favourites and search history the client sends the token in the request header.

**Logout:** The token is removed from local storage.

# Task 4: "Favourites" functionality

(Describe the major steps on how to make “favorite products” in the app)

**Favourites Data Model:** In user.js each user document has favorites: [{ cardId: { type: String } }], which will hold the list of the favourite cards by ID per user in MongoDB

**FrontEnd:** In favoritesAtom.js we use jotai to track favourite cards client side, export const favoritesAtom = atom([]); This global state allows components updates and reads favourite cards.

**Displaying Favourites:** If the user is not logged in they will be redirected to the login page, if they are logged we call /api/user/get/favorites with the JWT token, for every cardID that is returned we call {cardId} to get the full card data, and display results using CardList.

**Toggle Favourites:** If a user is not logged in the favourites toggle button will be disabled, however when a logged in user clicks the toggle button we get the token from local storage, if the card is already in the favourites list it sends a delete request, if not it sends a put request.

**Task 5: "Route Guard" functionality**

(Describe how to implement protected routes)

To ensure only registered users could access features like searching and favoriting Pokemon cards, we implemented a Route Guard in our app using Next.js and Jotai to restrict the favorite page and search page to authenticated users only, by redirecting the user to login page when attempting to search or favorite.  
  
Steps:

1. Created RouteGuard.js Component inside components/ folder. It wraps around the main layout and checks user login status using loggedInAtom from jotai.
2. Used PUBLIC\_PATHS array including to only protect the /favorites and /search pages. This made the homepage and public content accessible without foricing the user to log in immediately.
3. Used useEffect to check login status
4. Reused the updateAtoms() function from our login page to re-sync user data (favorites and history) to ensure favorites didn’t disappear after refreshing the page.

# Task 6: deployment

(Describe the major steps for deployment)

Once the Poke Mart project was initialized, added and pushed to github repository. Created initial homepage with “Website coming soon!” and deployed to Vercel.

Steps:

1. Created initial homepage with initial message.
2. Pushed our code to GitHub under a private repository
3. Signed into vercel.com using GitHub authentication
4. Imported our GitHub repo to Vercel and selected the correct project directory to deploy.
5. Clicked “Deploy”
6. Vercel generated live URL that automatically updates evertime we push to main.

\*Learned how to read, understand, and debug deployment errors

# Bonus questions

(Describe the major steps for designing the bonus question)

# Project planning/sharing tasks

(Describe how did you divided the work and did project planning. Have you changed the day-1-planning/milstones/deliverables that we did at the beginning of project in the class?)

We did not deviate too much from the original tasks. We delivered milestone deliverables on time. My original tasks were to implement routeguards but instead we decided that I would implement most of the design for the pages and components and Amany took over that role instead.

# Project performance check

(check your project performance using google lighthouse and add the screenshot of your current app status here )

A screenshot of a website

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Summary

(Describe your experience in this project, share your feedback on how this project was align with the course objective and how do you think the project can be changed to provide a better experience, …)

Daniel- This was a really great experience for me as it helped me learn more about working together as a team on a full stack web application. I would have liked to work more with the backend side of the project but I also enjoyed implementing the front end and making things look nice.