# Mini Project 2

## Overview

The purpose of my 2nd Mini Project is to apply the React.js and API skills I have learned in Modules 5-7. Expanding on Mini Project 1, I will build a similar website using React.js and Material UI (MUI), following a component-driven design approach. This project will also involve developing my own API service with Swagger documentation, implementing error handling with promise resolution and try-catch, and ensuring state management and routing for smooth navigation. Additionally, I will incorporate unit testing using Jest for both the front end and API, including snapshot testing to ensure component stability.

Setup:  
Create a new repository with two separate folder structures—one for the React Front-End and another for the Express API Back-End.

For the React Front-End folder structure:

* assets folder (inside public and src) containing images.
* components folder for all React component files.
* data folder for JSON files related to Characters and Games.
* pages folder containing React files for Characters and Games pages.
* tests folder with Jest tests written in JavaScript for the React files in both pages and components including the snapshots.
* App.css for styling.
* App.jsx as the main application component.
* main.jsx for application entry point.
* index.html for the main HTML file.
* eslint.config.js for linting configuration.
* jest.config.js for Jest testing configuration.
* package.json for project dependencies and scripts.
* package-lock.json for dependency management.
* vite.config.js for Vite configuration.
* .gitignore to specify files ignored by Git.
* README.md for documentation.

For the Express API Back-End folder structure:

* controllers folder containing JavaScript controller files for Authorisation, Characters, and Games.
* data folder for JSON files related to Characters and Games.
* routes folder with JavaScript route files for index, Authorisation, Characters, and Games.
* App.js for setting up the main application logic.
* swagger.json for API documentation.
* tests folder containing Jest tests for API files, including snapshot tests.
* .env for environment variables.
* .eslintrc.json for linting configuration.
* .gitignore to specify ignored files in version control.
* babel.config.js for Babel configuration.
* index.js for the server entry point.
* package-lock.json for dependency management.
* package.json for project dependencies and scripts.
* README.md for documentation.

A screenshot of a computer screen

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect. A screen shot of a computer

AI-generated content may be incorrect.

Acceptance Criteria (AC) / Requirements:

* Front-End needs to be built with React (At least 3 responsive working pages).
* Unit test is a must for the Front-End and the API. Include Snapshot test for at least one component where I am expected to talk about my unit tests and why is it important for my project.
* Integrate my own API service which does something.
* Integrate Swagger API for my Back-End service.
* Error handling, promise resolving an error, try catch, API call which has the .then(), .catch(), and .finally().
* Routing
* State management

## General Questions

**What is the purpose and scope of my project?**

Answered above in the overview section.

**What specific problem or challenge am I addressing with this project?**

Answered above in the overview section.

**What were my findings when comparing React to pure HTML and JavaScript?**

React offers reusable components, improved performance, and faster development. In contrast, HTML and JavaScript provide more control but can be slower and harder to scale. React is ideal for complex applications, while HTML and JavaScript work well for simpler projects.

**Can I provide an example of components that demonstrate a well-structured, component-driven approach?**

Home Page (localhost:5173), Characters Page (localhost:5173/characters), Games Page (localhost:5173/games).

**How is my API structured? What are its routes and endpoints, and what functions do they serve?**

My API is structured with a base URL: localhost:3000/api, handling game and character data through the following endpoints:

* GET /games – Fetches all available games.
* GET /games/{id} – Retrieves details of a specific game.
* GET /characters – Returns a list of all characters.
* GET /characters/{id} – Provides details of a specific character.

These endpoints deliver structured JSON data, supporting dynamic updates for your front-end components.

**How does my API integrate with my components to drive data-driven visual generation?**

My API provides data to React components via endpoints like /games and /characters. Components fetch this data, store it in state, and render dynamic visuals based on API responses. This enables flexible, real-time updates for a seamless user experience.

**What approach did I take to test both the front-end and back-end of my project?**

Jest Testing, Unit Testing, API Testing, Performance Testing

**What challenges or obstacles did I encounter during development?**

* Had trouble with Jest testing in both Front-End and Back-End as I was experiencing Reference Errors and Syntax Errors in the Git Bash Terminal.
* Had faced errors in my JavaScript code in the Back-End where my routes and controllers were not mapped properly.
* Styling and Design – The Front-End page wasn’t initially matching the designs in Figma until continuous tweaks.

**Looking back, what would I have done differently in my project?**

I would followed the development best practices a lot more to avoid differences in standards.

**If given more time, what additional features or improvements would I consider for the next phase?**

* Add the shop page which renders the shopping cart.
* Investigate further on why my JavaScript code is not integrating with my Sign Up and Login pop up components in React LocalHost.

## References

Design: <https://www.figma.com/design/zNCrUH0Aa9uWWVCDEofWkd/Institute-of-Data--IOD----Mini-Project-1?m=auto&t=DCooREBK7whNtkZf-6>

Trello: <https://trello.com/b/hjCoJU91/institute-of-data-mini-project-2>

GitHub: <https://github.com/saishk98/IoD-MiniProject2>

Batman API: <https://batmanapi.com/doc/batman-api/>