P&G Coding Assessment Technical Documentation

Version 1.00

Revision History

Date	Version	Description	Author
2022-02-13	1.0	Final	Jihyun Kim

Table of Contents

1. Introduction

1.1 Purpose of this document

The purpose of this document is to give technical information about P&G coding assessment.

1.2 Intended Audience

The intended audiences are:

- Supervisors, to analyze the design and implementation.
- Web application team members.
- Future developers to extend or use some ideas of web application project.

1.3 Scope

This document will describe the design and some technical guidance of the project.

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions
HTML/CSS/Javascript	Generate GUI with styling and animation
Node JS	Open source development platform to execute JavaScript code server-side
React JS	Open source Javascript library building user interfaces specifically for single-page applications
Axios	React module to get data from JSON data
Redux	React module to manage the state of the application in a single place and keep changes in the app more predictable and traceable
Fuse	React module to fuse multiple JavaScript or HTML files into one

1.4.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
GUI	Graphic User Interface
JSON	JavaScript Object Notation
Localhost	Refers to the local machine currently making the request
NPM	Node Package Manager

2. General overview

2.1 Technologies used

P&G Fake Store Shop is a web based application using HTML, CSS and Javascript. I (ie. Team) used React JS Library with various required modules which help the application more manageable, traceable and interactable. All technologies and components are open source which includes Fake Store API providing product information.

2.2 General functioning

The following are the major system functionalities:

- Build a React app
- Utilize the Fake Store API
- Display all products
- Allow user to search through products
- Allow users to filter on product results
  - Price
  - Category
  - Allow user to sort on product
- Detailed product page
- Checkout page
- Checkout confirmation

The sorting and checkout functionalities are not yet integrated to the application and partially implemented.

3. Technical requirement

3.1 Client requirement

As React web application is an interpreted language, it could be run on most of custom operating systems via a web browser. I (ie. Team) tested the application in Localhost with Chrome Version 97.0 and Microsoft Edge Version 98.0 and found no problem.

3.2 Server Requirement

The following technologies are required for local server.

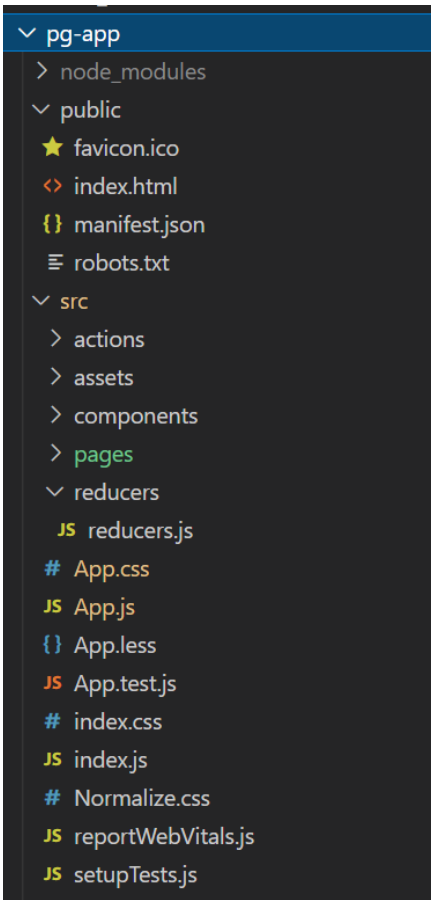
- Windows 10
- Node JS v16.14.0
- NPM v8.4.1
  - Dependencies
    - React v17.0.2
    - React-dom v17.0.2
    - Webpack v5.68.0
    - Webpack-cli v4.9.2
    - Webpack-dev-server v4.7.4
  - Dev Dependencies
    - Babel-core v6.26.3
    - Babel-loader v8.2.3
    - Babel-preset-env v1.7.0
    - Babel-preset-react v6.24.1
    - Html-webpack-plugin v5.5.0

3.3 React JS Library

React is a JavaScript library for building user interfaces. It makes the application painless to create interactive UIs and designs simple views for each state in web application. React JS efficiently updates and renders the right components when the data changes. To use React JS in this fake store app, it requires some various modules to build encapsulated components that manage their own state, then compose them to make complex UIs. There are some documents and samples of using React JS that could be found in https://reactjs.org/. Some special very close related document for this project that are available for React and Fake Store API could be reach at https://fakestoreapi.com/.

4. App Source structure

There are two main folders that represent the architectural tiers of the project. Figure 1 shows the files I (ie. Team) had for implementation. The presentation layer as Interface Adapter contains 4 files. “Components” contains 5 files and finally “Reducer” updates the application state corresponding to dispatched action. The key operators in this applications are “Index.js” which provides a context for React to render the application and “App.js” which acts as a container for all other components.



```
pg-app
  node_modules
  public
    favicon.ico
    index.html
    manifest.json
    robots.txt
  src
    actions
    assets
    components
    pages
    reducers
      reducers.js
    App.css
    App.js
    App.less
    App.test.js
    index.css
    index.js
    Normalize.css
    reportWebVitals.js
    setupTests.js
```

Figure 1

5. Architecture

Architectural design consisted of high level architecture.

5.1 High Level Architecture

The App component is a container with React Router. It has navbar that links to routes paths. – Five components that dispatch actions to Redux Trunk Middleware which makes Fake Store API call. Product component gets and displays Product Items. Cart component has form for storing product's order details. Store component has form for searching product items. “axios” is to make HTTP requests and receive responses. Figure 2 shows a high level architecture of Fake Store application.

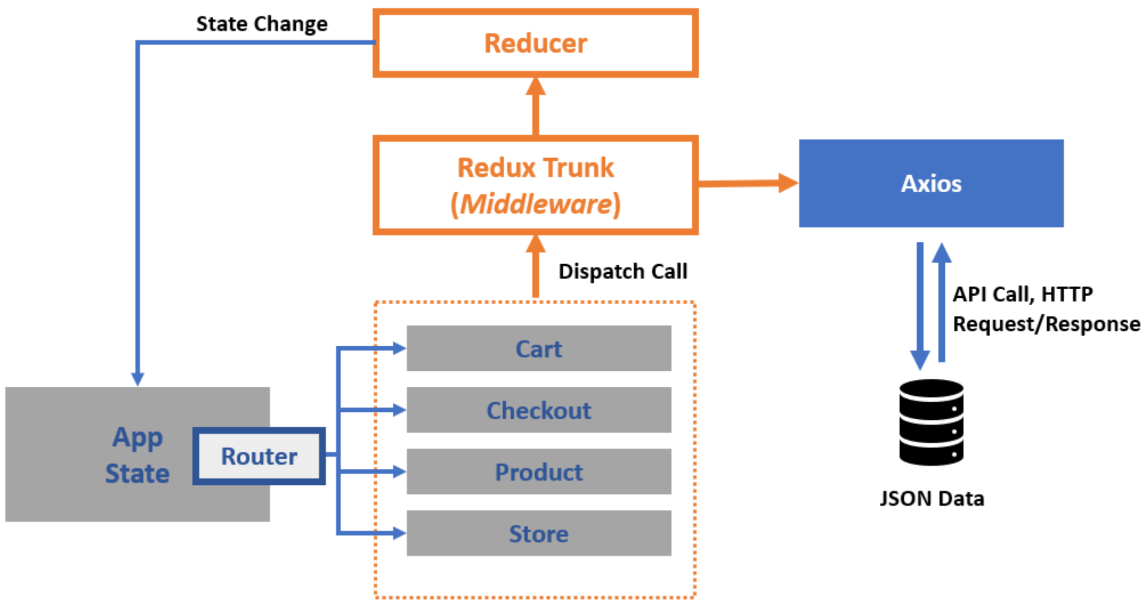


Figure 2: High level architecture of React Fake Store App

6. References

- React JS Official website, https://reactjs.org/
- React, Fake Store API,
    - https://fakestoreapi.com/
    - https://github.com/keikaavousi/fake-store-api
- https://github.com/alvillaraza/fake-store
- https://fx-cart.vercel.app/