PCEmpire System Design

Table of Contents

Changes to PC Empire System design

- Added CRC Cards of the newly implemented Classes
- Updated existing CRC Cards to match implementation
- Updated CRC Model to Match New CRC Cards

CRC Cards

App.js

Parent Class: Index.js

Subclasses:

- Main Page Container
- Build Part Container
- Product List Container
- Nav Bar Component

Responsibilities

- Main entrance point for App

CollaboratorsReact Router

- React

Main Page Container

Parent Class: App.js

Subclasses

- Main Page Component

Responsibilities

- Container for Apps main page

Collaborators

- React

Main Page Component

Parent Class: Main Page Container

Subclasses

- None

Responsibilities

- App's main page

Collaborators

- React
- React Router
- Style.css

API.js

Parent Class: None

Subclasses

- None

Responsibilities

- Hold API calls

Collaborators

- CPU.js
- Mobo.js
- express

App.css

Parent Class: None

Subclasses

- None

Responsibilities

- Contains themes and colours of Webpage

Collaborators

- None

Index.css

Parent Class: None

Subclasses

- None

Responsibilities

- Contains themes and colours of Webpage

Collaborators

- None

Search Bar Component

Parent Class: TBA

Subclasses

- None

Responsibilities

- Provide a search capability for users

Collaborators

- TBA

Login Component

Parent Class: Login Container

Subclasses

- None

Responsibilities

- Allow user to login to their account

Collaborators

- TBA

Login Container

Parent Class: None

Subclasses

- Login Component

Responsibilities

- Container for login

Collaborators

- TBA

Register Component

Parent Class: Register Container

Subclasses

- None

Responsibilities

- Allow user to register their account

Collaborators

- TBA

Register Container

Parent Class: None

Subclasses

- Register Component

Responsibilities

- Container for register

Collaborators

- TBA

Build Parts Container

Parent Class: App.js

Subclasses

- Build Parts Component

Responsibilities

- Container for Build Parts component

Collaborators

- React

Build Parts Component

Parent Class: Build Parts Container

Subclasses

- Product Button Component

Responsibilities

- Holds the product types during a build

Collaborators

- React
- PropTypes

NavBar Component

Parent Class: App.js

Subclasses

- None

Responsibilities

- Creates a Navigation bar and buttons with which we can navigate the UI

Collaborators

- React
- Bootstrap

Product Button Component

Parent Class: Build Parts Component

Subclasses

- None

Responsibilities

- Button that sends the user to a specified product selection page

Collaborators

- React
- Bootstrap
- React Router

Product List Container

Parent Class: App.js

Subclasses

- Product List Component

Responsibilities

- Renders Product List component

Collaborators

- React

Product List Component

Parent Class: Product List Container

Subclasses

- Slider Component

Responsibilities

- Populates a list of Parts on the webpage

Collaborators

- React
- PropTypes
- Bootstrap

Slider Component

Parent Class: Product List Component

Subclasses

- Slider Handle Component
- Slider Track Component

Responsibilities

- Build a slider using slider handle and track

Collaborators

- React

Slider Handle Component

Parent Class: Slider Component

Subclasses

- None

Responsibilities

- Builds the moveable dot for the slider.

Collaborators

- React

Slider Track Component

Parent Class: Slider Component

Subclasses

- None

Responsibilities

- Builds the Rail for the slider.

Collaborators

- React

Action Type

Parent Class: None

Subclasses

- None

Responsibilities

- Defines possible actions.

Collaborators: None

API Request

Parent Class: None

Subclasses

- None

Responsibilities

- Defines API requests.

Collaborators:

- Axios
- Action types

Product Types

Parent Class: None

Subclasses

- None

Responsibilities

- Defines Product Types.

Collaborators: None

Table Headers

Parent Class: None

Subclasses

- None

Responsibilities

- Defines Table Headers.

Collaborators: None

Brands Reducer

Parent Class: Reducers Index

Subclasses

- None

Responsibilities

- Holds available brands in the state.

Collaborators:

- Action Types

Filter Reducer

Parent Class: Reducers Index

Subclasses

- None

Responsibilities

- Holds available filters (Brands, Prices, etc..) in the state.

Collaborators:

- Action Types

Reducer Index

Parent Class: None.

Subclasses

- Brands Reducer
- CPU Reducer
- Filter Reducer
- Motherboard Reducer
- Prices Reducer
- Search Reducer

Responsibilities

- Combines all reducers into one.

Collaborators:

- Action Types

CPU Reducer

Parent Class: Reducers Index

Subclasses:

- None

Responsibilities

- Holds available CPUs in the state.

Collaborators:

- Action Types

Motherboard Reducer

Parent Class: Reducers Index

Subclasses

- None

Responsibilities

- Holds available Motherboard in the state.

Collaborators:

- Action Types

Prices Reducer

Parent Class: Reducers Index

Subclasses

- None

Responsibilities

- Holds available Prices in the state.

Collaborators:

- Action Types

Search Reducer

Parent Class: Reducers Index

Subclasses

- None

Responsibilities

- Holds Current Search Term in the state.

Collaborators:

- Action Types

Index.js

Parent Class: None

Subclasses

- App.js

Responsibilities

- Creates the Store
- Renders the App
- Facilitates Async actions
- Dispatch logging

Collaborators:

- React
- React Dom
- Styles.css
- Service Worker
- Thunk
- React Router
- Logger

Service Worker.js

Parent Class: None

Subclasses

- None

Responsibilities

- Allows webpage to be Run in the background

Collaborators:

- None

CPU.js

Parent Class: None

Subclasses

- None

Responsibilities

- Defines a CPUSchema

Collaborators:

- Mongoose

Mobo.js

Parent Class: None

Subclasses

- None

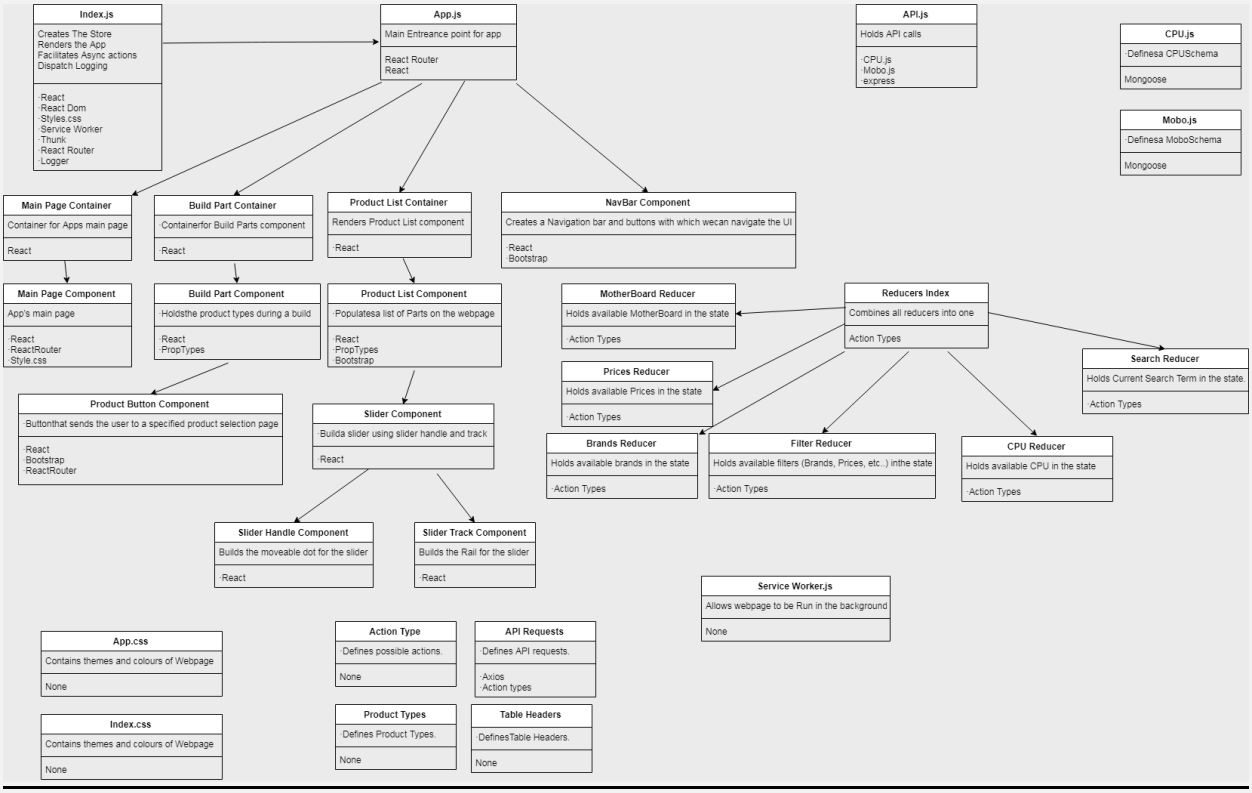
Responsibilities

- Defines a MoboSchema

Collaborators:

- Mongoose

CRC Model



System Interaction

PCEmpire will require the following assets for creation:

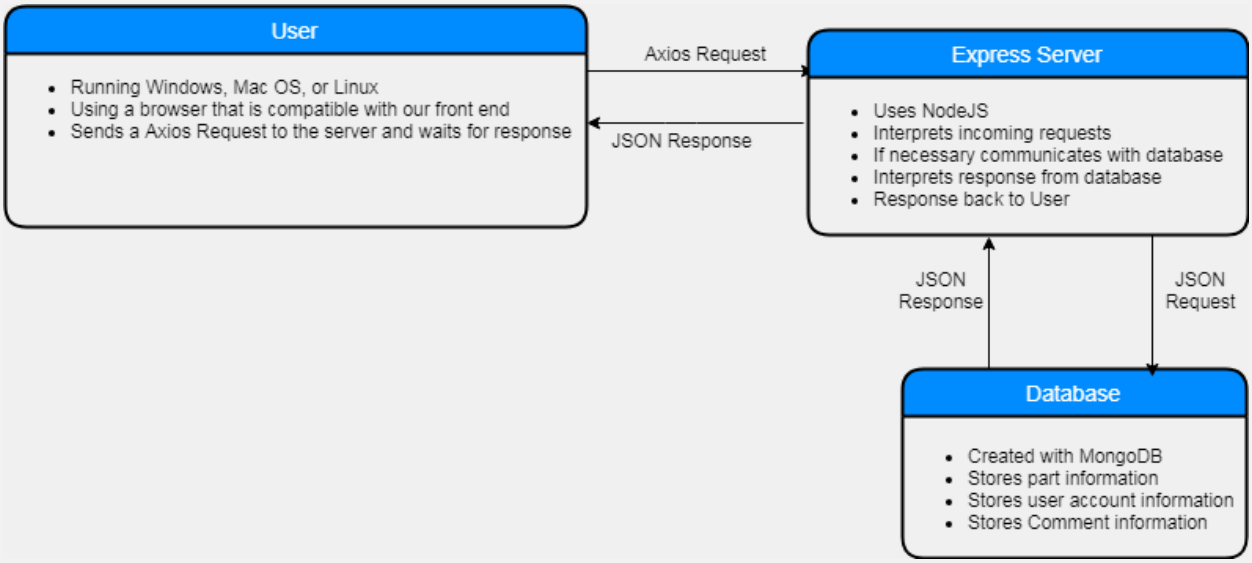
Server: NodeJS, Express, Mlab

Database: MongoDB

Front-End: HTML

Majority of users have computers that run on either Windows, Mac OS or Linux and as such should be able to run a browser compatible with the front end of PCEmpire. The database will be created in MongoDB and then hosted on a server provided by Mlab which can easily respond to request made by these users.

System Architecture



System Decomposition

PCEmpire can be broken down into three main components the user-side component, the server-side component and the Database-side component.

User-side: The user-side classes are those that directly impact the user viewing experience these ones are the classes that require user input and communication with the database via our server. These include the following: App.js. This Class is the one that require the user to make inputs and will be error checked by making sure that all inputs are of the proper type and size. We will also attempt to portray as clearly as possible the intended format for user input in order to minimize the chance of user error.

Server-Side: The server-side classes are those that communicate between the Database and the user. These classes include: Main Page Container, Login Container, Register Container and API.js. These functions are the ones that hold all the calls between the user and database as well as how to interpret them in an efficient way. To avoid errors, we will be checking the format of the inputs and outputs and in the case of an exception we will send a specific error code that can be read in the user-side and output a relevant error message.

Database-side: The Database-side Classes are the ones that pull information from the Database and send them upstream to be interpreted by the server. These Classes include: Main Page Component, Search Bar Component, Menu Bar Component, Login Component and Register Component. These are the functions that parse information from the database and sent it to the server. If there is some error in pulling from the database, we will send a specific error code upstream so that the user can receive an informative error description.

For network and server errors we plan to have a client-side page that will direct users in ways to help them hopefully reconnect with PCEmpire. Things such as refreshing the page or restarting their internet connection. These network errors are difficult to solve and are often due to user connection problems and cannot be solved by our team as such all we can do is to attempt to guide the user to solve the problem with our guidance