React

Baseline Training

Tim Gonzales

7/31/2019

Table of contents

[1 Introduction 3](#_Toc15375104)

[2 Goals and Benefits 3](#_Toc15375105)

[3 Prerequisites 3](#_Toc15375106)

[4 Project 3](#_Toc15375107)

[5 Course Outline 5](#_Toc15375108)

[1.1. *JSX* 5](#_Toc15375109)

[1.2. *Components & Props* 5](#_Toc15375110)

[1.3. *Hooks* 5](#_Toc15375111)

[1.4. *Redux* 6](#_Toc15375112)

[1.5. *Testing* 6](#_Toc15375113)

[1.6. *Miscellaneous* 6](#_Toc15375114)

[6 Member’s Responsibilities 7](#_Toc15375115)

[7 Reviewer’s Responsibilities 7](#_Toc15375116)

[8 Schedule Time Line 7](#_Toc15375117)

# Introduction

*Give a brief description of the training. Describe objects and give a quick overview of the material.}*

Upon completion of this course, an engineer should be able to demonstrate an intermediate level of competence in the technology.

# Goals and Benefits

The goal of this course is to provide members with an understanding of the fundamentals for React development and solution architecture. Through the completion of this course, members will develop a solid understanding of architecture and solution development basics. Completing this course and attaining endorsement from the reviewer will aid in CGI’s resource management and allocation, as management will have a more quantifiable way of measuring member’s skill levels.

The benefits of receiving the *React* course endorsement from the reviewer to the member include:

* Expansion of technical knowledge
* Recognition within the organization

The benefits of receiving the *React* course endorsement from the reviewer to CGI include:

* Fundamental architecture knowledge of across the consulting pool
* Accurate evaluation of skills across the consulting pool
* An increased culture of information sharing and peer accountability in the consulting pool

# Prerequisites

* Bootcamp
* General working knowledge of Javascript

# Project

Develop a body age calculator application. The application will present users with a series of health related questions. Once the user answers all questions successfully, the application will calculate and show the user’s body age.

* The application will contain three questions and associated responses:
  + Do you workout weekly?
    - Never, Sometimes, Always
  + Do you eat junk food?
    - Never, Sometimes, Always
  + Can you touch your toes?
    - Yes, No
* The application will store an offset value, such as -1 or 1, for each response to a question:
  + Never, offset = 1
  + Sometimes, offset = 0
  + Always, offset = -1
  + Yes, offset = 1
  + No, offset = -1
* The application will use the offset value to calculate body age from the user’s birthdate (e.g. user birthdate is 1/1/1990 and they answered all questions with a combined offset of -3. The user body age will be 3/1/2019 - 1/1/1990 = 29 years 2 months - 3 year offset = **26 years**; note the rounding of the months).
* The application will maintain the progress of questions (unanswered or answered) by the user.
* The application will contain a page that shows the results for each user, and enable users to filter the results by showing all results, recently completed (<= 7 days) results, or older results.
* The application should have some authentication mechanism to authorize users. Users will be required to login and data will be associated with the user.
* The application must store state using a Redux store.
* The application must persist the responses for each question via an API call when data is updated.
* The application must contain an implementation of a hook.
* The application must contain unit and integration tests.
* The application must comply with accessibility standards outlined in the Accessibility link in the Course Outline.

# Course Outline

* 1. *JSX*

*Understand the role of JSX in React development and know how to implement*

* [*Introducing JSX*](https://reactjs.org/docs/introducing-jsx.html)
* [*Rendering Elements*](https://reactjs.org/docs/rendering-elements.html)
* [*What is JSX?*](https://www.fullstackreact.com/30-days-of-react/day-2/)
* [*React JSX*](https://www.javatpoint.com/react-jsx)
* [*JSX In Depth*](https://reactjs.org/docs/jsx-in-depth.html)
  1. *Components & Props*

*Understand the role of Components in React development and how to implement*

* [*Components and Props*](https://reactjs.org/docs/components-and-props.html)
* [*State and Lifecycle*](https://reactjs.org/docs/state-and-lifecycle.html)
* [*Component Lifecycle*](https://www.javatpoint.com/react-component-life-cycle)
* [*Handling Events*](https://reactjs.org/docs/handling-events.html)
* [*Conditional Rendering*](https://reactjs.org/docs/conditional-rendering.html)
* [*Lifting State Up*](https://reactjs.org/docs/lifting-state-up.html)
* [*Our First Components*](https://www.fullstackreact.com/30-days-of-react/day-3/)
* [*Complex Components*](https://www.fullstackreact.com/30-days-of-react/day-4/)
* [*Data-Driven Components*](https://www.fullstackreact.com/30-days-of-react/day-5/)
* [*Composition vs Inheritance*](https://reactjs.org/docs/composition-vs-inheritance.html)
* [*Higher-Order Components*](https://reactjs.org/docs/higher-order-components.html)
* [*Uncontrolled Components*](https://reactjs.org/docs/uncontrolled-components.html)
* [*Pure Components*](https://www.fullstackreact.com/30-days-of-react/day-11/)
* [*Web Components*](https://reactjs.org/docs/web-components.html)
* [*Error Boundaries*](https://reactjs.org/docs/error-boundaries.html)
* [*Forwarding Refs*](https://reactjs.org/docs/forwarding-refs.html)
* [*Fragments*](https://reactjs.org/docs/fragments.html)
* [*Context*](https://reactjs.org/docs/context.html)
  1. *Hooks*

*Understand how to implement hooks*

* [*Introducting Hooks*](https://reactjs.org/docs/hooks-intro.html)
* [*Hooks at a Glance*](https://reactjs.org/docs/hooks-overview.html)
* [*Using the State Hook*](https://reactjs.org/docs/hooks-state.html)
* [*Using the Effect Hook*](https://reactjs.org/docs/hooks-effect.html)
* [*Rules of Hooks*](https://reactjs.org/docs/hooks-rules.html)
* [*Building Your Own Hooks*](https://reactjs.org/docs/hooks-custom.html)
* [*Hooks API Reference*](https://reactjs.org/docs/hooks-reference.html)
* [*Hooks FAQ*](https://reactjs.org/docs/hooks-faq.html)
  1. *Redux*

*Understand how to manage state with Redux*

* [*Getting Started with Redux*](https://redux.js.org/introduction/getting-started)
* [*Basic Tutorial*](https://redux.js.org/basics/basic-tutorial)
* [*Data Management with Redux*](https://www.fullstackreact.com/30-days-of-react/day-19/)
* [*Redux Actions*](https://www.fullstackreact.com/30-days-of-react/day-20/)
* [*Redux Middleware*](https://www.fullstackreact.com/30-days-of-react/day-21/)
* [*Advanced Tutorial*](https://redux.js.org/advanced/advanced-tutorial)
* [*Redux Sagas - Introduction*](https://redux-saga.js.org/docs/introduction/BeginnerTutorial.html)
* [*Redux Sagas – Basic Concepts*](https://redux-saga.js.org/docs/basics/)
* [*Redux Sagas – Advanced Concepts*](https://redux-saga.js.org/docs/advanced/)
  1. *Testing*

*Understand how to develop unit and integration tests for React*

* [*Introduction to Testing*](https://www.fullstackreact.com/30-days-of-react/day-22/)
* [*Testing Setup*](https://www.fullstackreact.com/30-days-of-react/day-23/)
* [*Testing the App*](https://www.fullstackreact.com/30-days-of-react/day-24/)
* [*Better Testing with Enzyme*](https://www.fullstackreact.com/30-days-of-react/day-25/)
* [*Integration Testing*](https://www.fullstackreact.com/30-days-of-react/day-26/)
* [*React Test Utilities*](https://reactjs.org/docs/test-utils.html)
* [*Test Renderer*](https://reactjs.org/docs/test-renderer.html)
  1. *Miscellaneous*

*Understand how to develop unit and integration tests for React, explain the differences between React and AngularJS*

* [*Accessibility*](https://reactjs.org/docs/accessibility.html)
* [*Client-side Routing*](https://www.fullstackreact.com/30-days-of-react/day-17/)
* [*Code-Splitting*](https://reactjs.org/docs/code-splitting.html)
* [*Integrating with Other Libraries*](https://reactjs.org/docs/integrating-with-other-libraries.html)
* [*Optimizing Performance*](https://reactjs.org/docs/optimizing-performance.html)
* [*Portals*](https://reactjs.org/docs/portals.html)
* [*Reconciliation*](https://reactjs.org/docs/reconciliation.html)
* [*Fetching Remote Data*](https://www.fullstackreact.com/30-days-of-react/day-14/)
* [*Introduction to Promises*](https://www.fullstackreact.com/30-days-of-react/day-15/)
* [*Deployment Introduction*](https://www.fullstackreact.com/30-days-of-react/day-27/)
* [*React vs AngularJS*](https://www.javatpoint.com/reactjs-vs-angularjs)
* [*React Bootstrap*](https://www.javatpoint.com/react-bootstrap)

# Member’s Responsibilities

A member will be required to demonstrate knowledge of all items identified in the Course Outline section of this document above and create a React solution for the Project requirements above. The intent is that the created solution does not necessarily have to contain or demonstrate all of the elements detailed in the Course Outline section. However, at demonstration time, the member must definitively demonstrate detailed knowledge of how to implement each skill.

# Reviewer’s Responsibilities

The reviewer is responsible for certifying the member as having completed this training. The reviewer must perform the following steps in the process:

* Review completed assignment with member
* Assess knowledge of all core skills identified in the Course Outline section
* Provide notification of course completion to the Member Relations Coordinator

# Schedule Time Line

The following timeline is recommended for the core competency process:

|  |  |
| --- | --- |
| Task | Timeline |
| Complete review of all items within the Course Outline section | *20 hours* |
| Complete development of assignment | *20-40 hours* |