Introduction

This documentation concerns the Tetractys assignment for the developers Front End position. The purpose of the document is to demonstrate the idea and the implementation structure based on the solution. The requested assignment was to implement 2 main pages, one to represent a table on with a set of data and an other one that implements a form in order to apply new entry on the set.

The repository that you are able to view and download the source code from git :
https://github.com/JohnValadakis/SingleSPAExprences
and view the application hosted at:
http://johnvaladakis.github.io/SingleSPAExprences

Build

Current application has been built using **webpack**, **babel** and **esLint**. As source code dependencies we have imported **material-ui** components to apply a more friendly UI with widgets. Some extra dependencies are **moment** to handle the date entities and **clsx** in order to parse and the styling of classnames of elements. The hole styling of the project is parsed through js properties.

Building scripts

- Download the repository on a local folder using the github public profile.
- Navigate to the source folder containing the **package.json**
- Run **"npm install"**
- Run **"npm start"** in order to run the application on a local environment
- Run **"npm build"** in order to build the project on the folder **"./dist"**
- Run **"npm run predeploy"** and **"npm run deploy"** if you want to build and run the application on git hub pages.

You can configure the predeploy and deploy commands in order to deploy the project on your own preferred server.

Hosting the site on IIS you have to clone the files from **"./dist"** folder to a configured website.

The application

Current Structure on Source Code is implemented that way in order to become full generic and configurable. Under /src directory exists all the implementation files used to create the application. The main application renders on an element with id "root".

The main Component s is App and includes page containers
The 2 basic Pages are :

- History
- Forms

Those components contain react components that use react hooks **useState** and **useEffect,**

The folder structure of the src contains 4 basic folders:

- components
  - common
    - elements
    - header
  - display
    - generic-content
- containers
  - app
  - pages
    - forms
    - history
- files
- utils
  - constants
  - helpers

Components

Folder **components** implements reusable react components that can be used across all pages

Common

- Elements are designed to apply formatted elements such as Currency etc.
- **Header** is the Application header containing the Links of the Router entity

Display

- **ContentDisplay** that is designed to switch the visible style or the results between Table and Grid using the below Elements
- **GenericTableContainer** implementing a material ui table with headers as data and functions handling paging and sorting on data.
- **GridContainer** “not yet implemented”

Containers

Folder **containers** implements react components that contains logic. There are 2 main pages of History and Forms.

History

History component contains:

- an **Initial Data Button** an initial data **Button** that loads a file of data from the utils
- a **ContentDisplay** component that gets the data and a set of actions

Implements an on useEffect hooks that then the data is changed stores and fetches the data from browser’s localstorage on key **“expenses”.**

Actions that trigger the hook are :

- Load initial data Button
- Action Function Parsed as a property on **<ContentDisplay />** that deletes the selected row
- Page Refresh as the **<History />** element will re-render and fetches the data from storage

On any case the user is able to load the initial data by performing a click event on the Load Button.

Forms

Forms is a container implemented with react hooks in order to give the user the ability to store new data on the collection. Used to parse an object with properties

- **Description** as an input element that prevents the empty data application
- **Amount** as a Numeric input applying a fixed display format with thousands separator "," and Float using "."
- **Type** as a Dropdown element with values income outcome
- **Date** as a **DateTimePicker** parsing the date to an IsoStringFormat with appended GMT
- **Clear** Button that apply default values into properties of the object that is handling
- **Submit** button that sets the state of the item and using hooks append that item to the collection storing back to local storage

Files

Contain json for multilingual purposes and the data collection

Utils

Created as a common share functionality file in order to give the ability to the whole application to import and use:

- Constants
- and Helpers

Constants contains:

- Languages
- Menu references
- and Expense Types

Helpers contains:

- Translators
- DataHandlers
- BrowserStorageHandlers

The helpers have the idea to create Wrappers on functionalities and are implemented like pure functions with content agnostic entities. For example a possible HTTP calls handler could be

implemented there in order to avoid boilerplates or other plugins that serve common functionalities.

Wrapping up

- The idea of the project is to be fully generic and allow **containers** keep a simple logic on pages.
- There are no implemented tests due to the lack of time.
- There are no styling files to avoid browsers mismatch due to the lack of a template.
- Styling properties are implemented using the flex box approach and through the classes defined as objects.
- prop-types are currently disabled to be decided on a final project based on business dependences or is typescript will be used.

The project was implemented Keeping the idea that the purpose of the assignment was to note some React skills and build skills and an architectural approach the supervisor may find the solution not strictly implemented with the rule of simplicity.

Known Issues

React router on deployment repo is lacking on refresh after navigation

No custom validation on form is kept only on widgets

No form item has been stored on local storage as a result item will be set to default values after navigation to history and back

No tests have been implemented just a set of renders