Overview libraries using in the project 🐮

This uses some supporting plugins:

- React Libraries (Main Platform): 'react', 'react-dom'.
- React Router V4 (React Plugin): 'react-router', 'react-router-dom'.
- Redux (Management State Library): 'redux', 'redux-react', '@reduxjs/toolkit'.
- Middleware Saga(handle side effect: asynchronous, timer...base on generators in JavaScript (-from es6): 'redux-saga'.
- Webpack (Bundling Module support to build project): 'webpack'
- SASS - Pre-Processor: 'sass', 'node-sass'
- Library UI: 'react-bootstrap',
- react-select: a component build for ReactJS with multi-select, autocomplete and ajax support.
- react-date-picker.
- react-toastify.

Guideline for focusing and developing to project

1. Using command line (CLI) in project

*Note: You can 'yarn' or 'npm' to work with this project.

Current time, we just use 'start' & 'build' to develop and pack modules in the project:

- Build project for the production environment:

```
npm build
yarn build
```

- Start project at dev environment:

```
npm start
yarn start
```

2. Structure of project

```
Corporate-account-ui
├──nginx
└──src
    │   index.html
    │   index.tsx
    ├──app
    │   │   app.scss
    │   │   App.tsx
    │   └──shared (shared components)
    │   ├──corporate-account-tool
    │   │   ├──header-bar
    │   │   ├──nav-bar
    │   │   └──search-panel
    │   └──corporate-account-report
    ├──assets
    ├──common
    │   ├──constants
    │   ├──enums
    │   └──types
    ├──services
    └──store
        │   store.ts
        ├──middleware
        │       root.saga.ts
        └──slice
                root.reducer.ts
```

- 2.1. **tsconfig.json**:

  File configures for typescript project such as compile decorator.

- 2.2. **package.json**

  File contains all configurations of project (libs-dependencies, script-task, plugins...)

- 2.3. **nginx/** folder:

  It stores some configurations for running project with nginx.

- 2.4. **build/** folder:

  It stores sources of project after building.

- 2.5. **public/** folder:

  It stores sources (css, data-resources, fonts, images, locales) of project after building at the dev environment.

- 2.6. **script/** folder:

  It includes files using to build and start project.

- 2.7. **src/**

  This is the main folder in project. You can develop anything in here. It separates to 5 sub-folders: **api/ , common/ , component/ , static/, middleware/, slice/, store app.tsx and index.tsx file**

  - 2.7.1. **apis/** folder

    This includes all of apis.

  - 2.7.2. **common/** folder

    This includes common files, logic, component, constant... which can re-use more than one time in project.

  - 2.7.3. **app/** folder

    This includes ts and scss files of component group by every feature.

  - 2.7.4. **static/** folder

    Folder stores build svn version.

  - 2.7.5. **middleware/** folder

    This includes middleware as redux-saga, redux thunk (this project apply **redux-saga**).

  - 2.7.6. **slice/** folder

    This includes actions and reducers of redux. 'Redux Toolkit' has recommended by the Redux base on 4 criteria: Simple, Opinionated, Powerful, Effective.

    view more: https://redux-toolkit.js.org

  - 2.7.7. **store/** folder

    This folder includes file store to create a store to storage data to share between components.

  - 2.7.8. **app.tsx/** file

    App.tsx is a start-point to any process, and imported out of **index.tsx** to run project.

  - 2.7.9 **index.tsx file**

    This is the first file called from server after running project. All threads of project will begin from here.

3. Basic knowledge and how to apply to this project - Redux + Saga-middleware

3.1: Create a store - configureStore with redux-toolkit

- To define a store and declare reducers

```
import createSagaMiddleware from 'redux-saga';
import { configureStore } from '@reduxjs/toolkit';
import {RootReducer} from "./slice/root.reducer";
import {RootSaga} from "./middleware/root.saga";

const sagaMiddleware = createSagaMiddleware();

const Store = configureStore({
reducer: RootReducer,
middleware: [sagaMiddleware]
});

sagaMiddleware.run(RootSaga);

export default Store;
```

3.2: To use this store:

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Provider } from 'react-redux';
import { BrowserRouter } from 'react-router-dom';
import App from './app/App';
import store from './store/store';

const rootNode = document.getElementById('root');

ReactDOM.render(
<Provider store={store}>
  <BrowserRouter basename='corporate-account-ui'>
  <App />
  </BrowserRouter>
</Provider>,

  rootNode
);
```

3.3: Create file saga to watch action (in here is :'accounts' which exporting from Slice).

3.3.1: Create a CommunicationTabsSaga file.

```
    import {all, call, put, takeLatest} from 'redux-saga/effects';
    import { getAccountListingService} from
'../../services/account.service';
    import {loadAccounts, loadAccountsSuccess, loadAccountsError} from
'../slice/account.slice';

    function* loadingAccountsAsync(param: any) {
        try {
            const data = yield call(getAccountListingService,
param.payload);
            yield put(loadAccountsSuccess(data));
        } catch (err) {
            yield put(loadAccountsError());
        }
    }

    export function* AccountSaga() {
        yield all([
            yield takeLatest(loadAccounts, loadingAccountsAsync)
        ]);
    }
```

3.3.2: import CommunicationTabsSaga in initSaga to register sagaMiddleware.

```
    import {all} from 'redux-saga/effects';
    import {AccountSaga} from './account.saga';
    import {DirectorSaga} from './director.saga';

    export function* RootSaga() {
        yield all([
            AccountSaga(),
            DirectorSaga()
        ]);
    }
```

3.4: Create a file slice and public a reducer and actions.

```
import {createSlice, PayloadAction} from "@reduxjs/toolkit";
   import { toast } from "react-toastify";

   const initialState = {
    accounts: {},
    loading: false,
    error: ''
   };

   export const AccountSlice = createSlice({
```

```
    name: 'tabAll',
    initialState,
    reducers: {
        loadAccount: (state, action: PayloadAction<any>) => {
            state.loading = true;
            state.error = '';
        },
        loadAccountSuccess: (state, action: PayloadAction<any>) => {
            state.loading = false;
            state.error = '';
            state.commentsTab = action.payload;
        },
        loadAccountError: (state) => {
            state.error = 'failed';
            toast.error("Fetch data accounts failed. Please contact
admin!");
        },
    }
  });

  export const {
    loadAccount,
    loadAccountSuccess,
    loadAccountError,
  } = AccountSlice.actions;

  export default AccountSlice.reducer;
```

3.5: Use in Component: dispatch actions to saga and reducer, after that use useSelector hook trigger to get data from store once data has updated.

```
    import { useDispatch, useSelector } from "react-redux";

    function nameComponent() {
        const dispatch = useDispatch();
        dispatch(loadAccounts(paramObject));
        const account = useSelector((state: any) => state.accounts);
    }
```

4. Define API services in Project.

```
    export const accountsApi = async (userId: number) => {
    const response = await axios.get(`/getAccounts`);
    return parseItem(response, 200);
    };
```

View more:

Using Axios with React: https://alligator.io/react/axios-react/

Using redux: https://redux.js.org/

Applying redux-saga middleware: https://redux-saga.js.org/

Redux-toolkit: https://redux-toolkit.js.org

6. Run project

1. npm run start (port 4400)
2. Start Nodejs server (default port 4400, Can change port at the webpack devServer port).
3. Open web browser with url: http://your_ip and login

7. Note commit in project

Don't commit these paths folder and file in the project. Because, they will auto generate when build

\corporate-account-tool\target

\corporate-account-tool\build

\corporate-account-tool\package-lock.json

\corporate-account-tool\yarn-lock.json

\corporate-account-tool\yarn.lock

\corporate-account-tool\yarn-error.log

\corporate-account-tool\debug.log