The focus is on Client-side routing.

Which means we are gonna render a single page and when user made a request like ‘/about’, then we are just gonna swap out the components.

contentBase: path.join(\_\_dirname, 'public') // here u mention those files which you wanna serve and are static

‘react-router’ is for all android, ios, web apps. But we need it for only web app, so we r gonna install ‘react-router-dom’.

<Routes>

      <Route path="/" element={<ExpenseDashboardPage />} exact={true} />

      <Route path="/create" element={<AddExpensePage />} />

      <Route path="/edit" element={<EditExpensePage />} />

      <Route path="/help" element={<HelpPage />} />

</Routes>

Here according to new version of react, now we have to use ‘element’ instead of ‘component’, and also ‘ExpenseDashboardPage’ is written inside ‘< />’.

‘Link’ module is used for client-side routing. When we made a request, page does not get refreshed, it just render new component.

Using ‘NavLink’ we can give styles on active state

‘npm cross-env’ will set up the environment variable(i.e. whether we are in testing, development or production)

‘process.env.NODE\_ENV’ is an environment variable you’re currently in. This gets automatically set for us on Heroku. Heroku set this value equal to the string ‘production’.

In ‘test’ script in package.json you can not have a single script that work under all environments. So to fix this we use ‘cross-env’ module. We don’t need to do it for production because Heroku do it, and similarly no need for development because absence of all environment variables will tell us that it is development environment.

The information in firebase(i.e. set of key-value pairs) is kind of secret(not to be visible to public), so we make two other files(i.e. for development and testing, but not for production cause that we will do by Heroku command line interface) which we are not including in Github. Then we will set ‘process.env.NODE\_ENV’ equal to ‘production’/‘test’.

* ‘setupFiles’ is by default provided by enzyme for react v16 version.
* ‘thunk’ help to combine firebase and redux. Allows to have asynchronous actions which communicate with firebase and dispatch a regular action which changes redux store.
* ‘thunk’ just add support for dispatching functions

‘Thunked’ actions or async actions gets called with dispatch, but it also get called with getState to get current state.

**Environment variables are setup when we run the scripts npm start, npm test, npm run build as development, test and production respectively.**