Bootcamp 2020 Class 6: React Expense Tracker

Part I - Create React Project using Create React App

Step 1: Create a new local repository
First, I navigate to my local GitHub folder where all my local repositories are kept.

```
Adil:~ adil$ cd /Users/adil/Documents/GitHub_
```

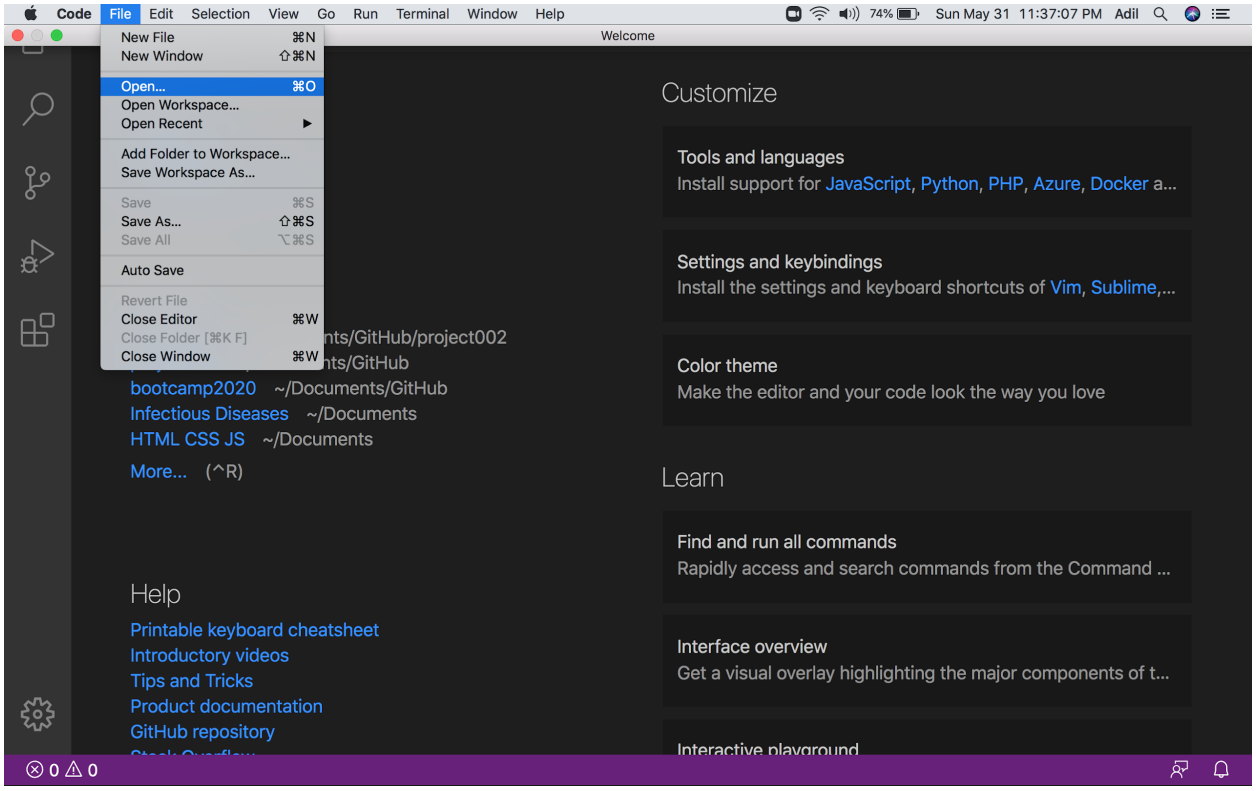
Now I create my local repository by creating my basic react application with:

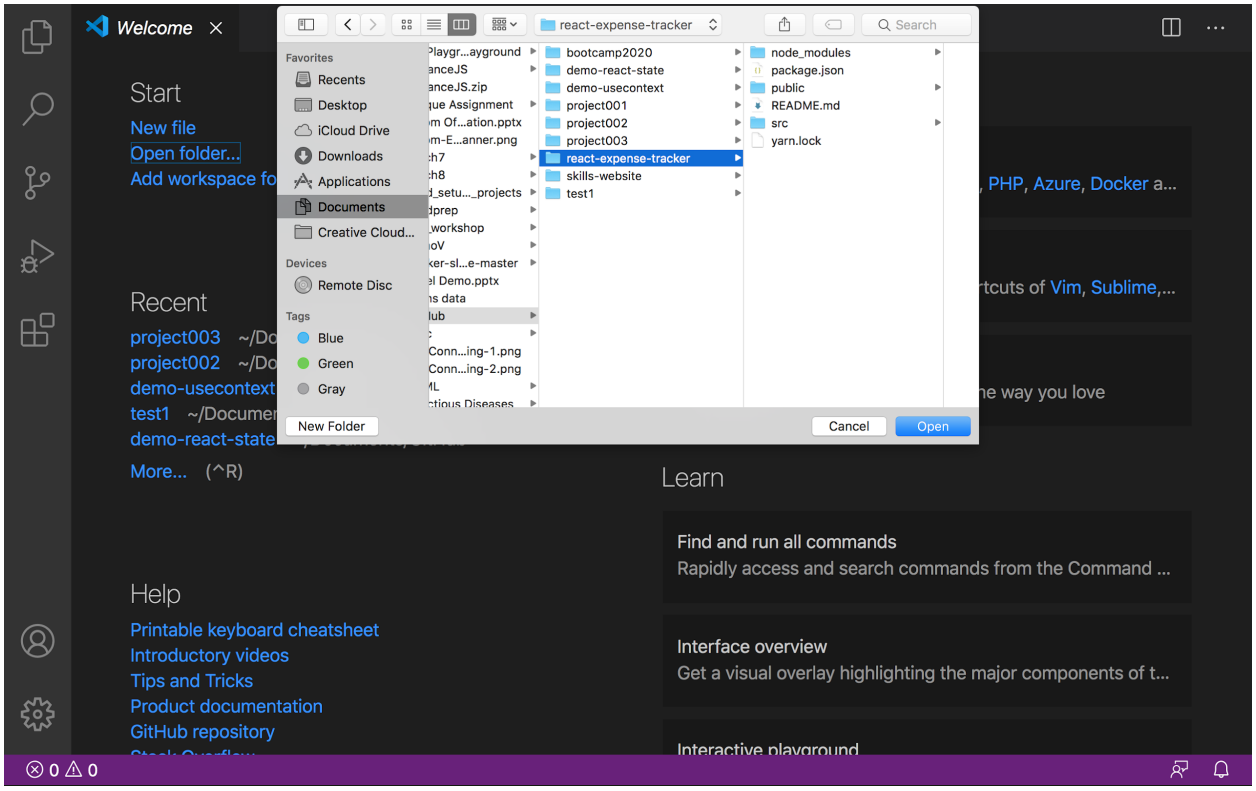
npx create-react-app <app-name>

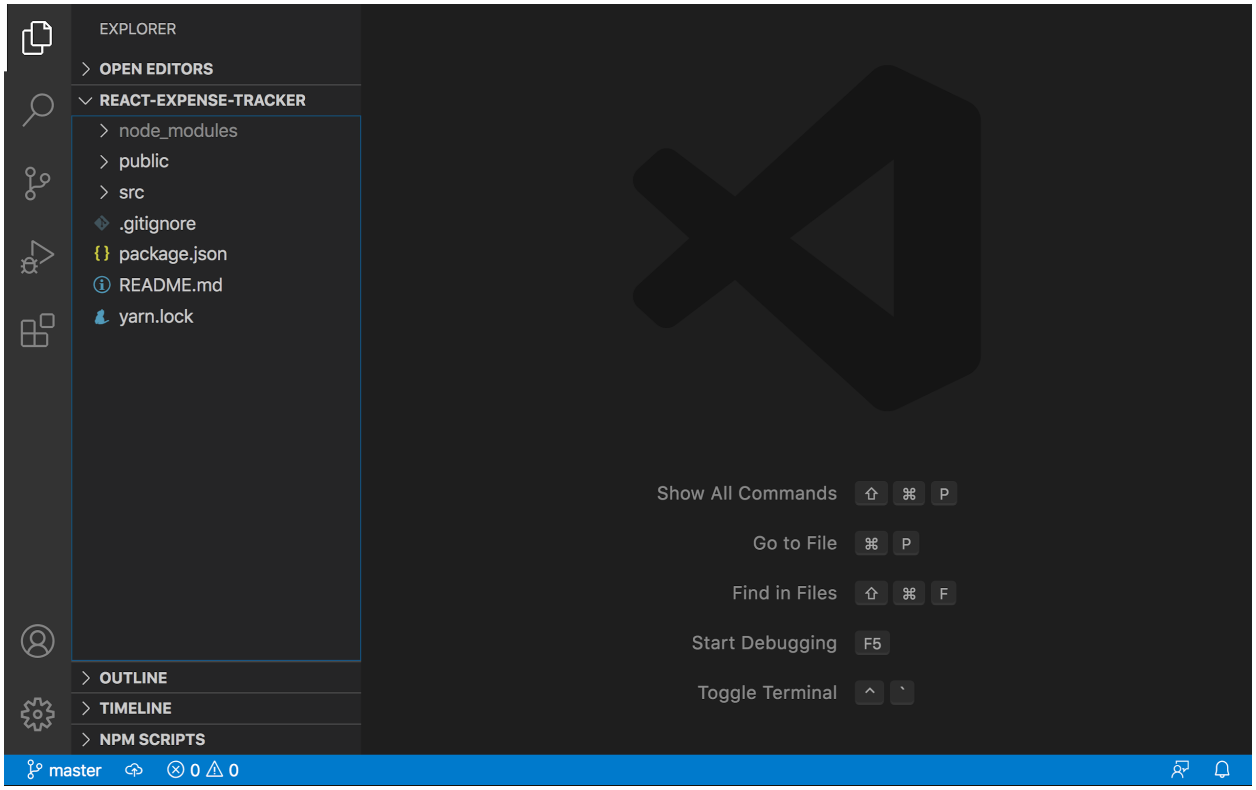
```
Adil:GitHub adil$ npx create-react-app react-expense-tracker
npx: installed 98 in 10.594s

Creating a new React app in /Users/adil/Documents/GitHub/react-expense-tracker.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

yarn add v1.19.1
[1/4] 🔍  Resolving packages...
[2/4] 🚚  Fetching packages...
[3/4] 🔗  Linking dependencies...
warning "react-scripts > @typescript-eslint/eslint-plugin > tsutils@3.17.1" has unmet peer depend
ency "typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0-dev || >= 3.4.0-dev || >= 3.5.0-dev || >= 3.
6.0-dev || >= 3.6.0-beta || >= 3.7.0-dev || >= 3.7.0-beta".
[4/4] 🔨  Building fresh packages...
success Saved lockfile.
warning Your current version of Yarn is out of date. The latest version is "1.22.4", while you're
 on "1.19.1".
info To upgrade, run the following command:
$ brew upgrade yarn
success Saved 14 new dependencies.
info Direct dependencies
├─ cra-template@1.0.3
├─ react-dom@16.13.1
├─ react-scripts@3.4.1
└─ react@16.13.1
info All dependencies
├─ @babel/plugin-syntax-typescript@7.10.1
├─ @babel/plugin-transform-flow-strip-types@7.9.0
├─ @babel/plugin-transform-runtime@7.9.0
```

```
Success! Created react-expense-tracker at /Users/adil/Documents/GitHub/react-expense-tracker
Inside that directory, you can run several commands:

  yarn start
    Starts the development server.

  yarn build
    Bundles the app into static files for production.

  yarn test
    Starts the test runner.

  yarn eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd react-expense-tracker
  yarn start

Happy hacking!
Adil:GitHub adil$ _
```

Part II - Open the Project with VS Code

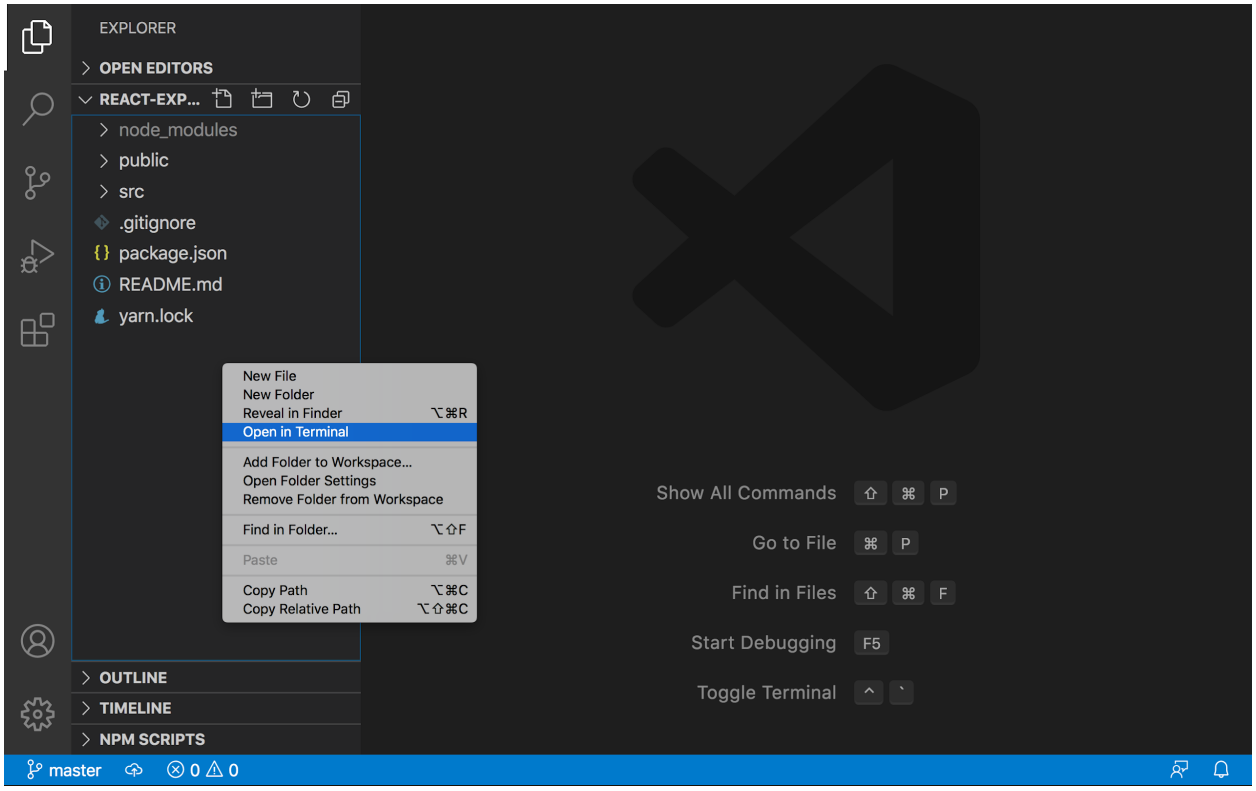


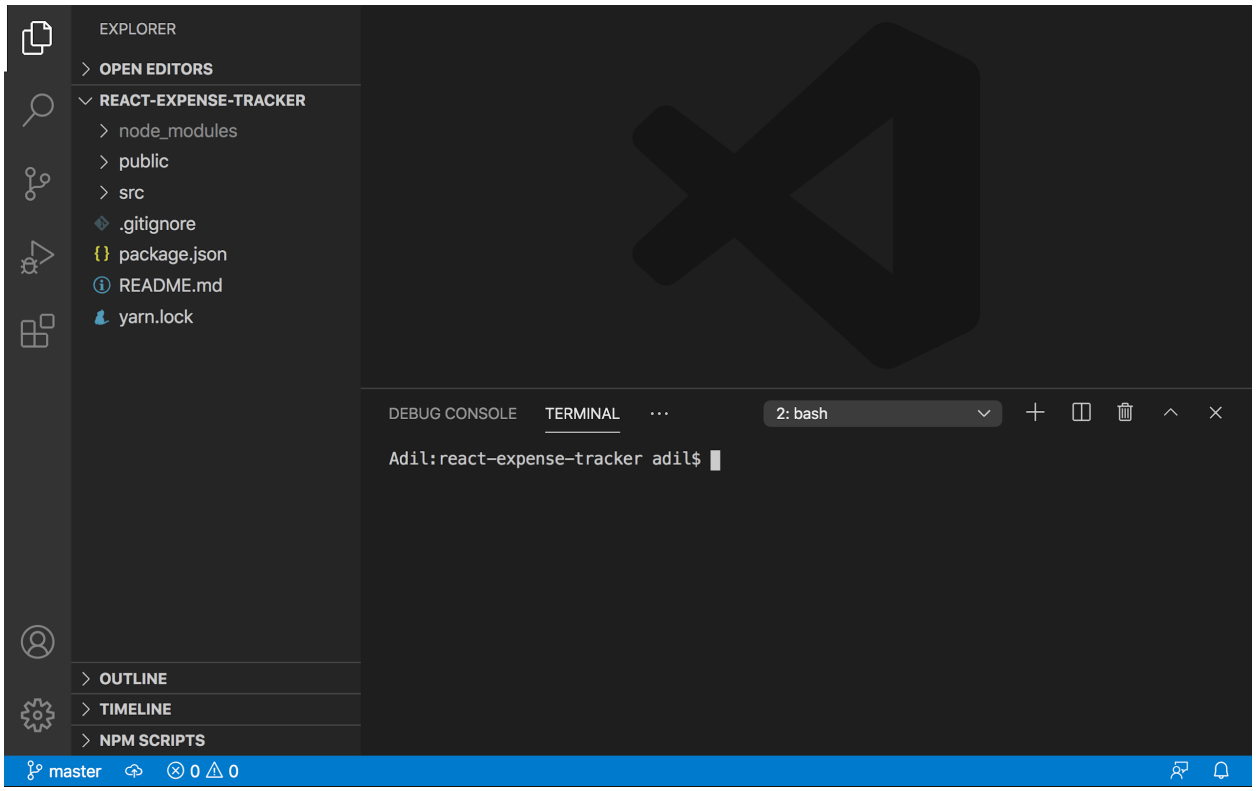


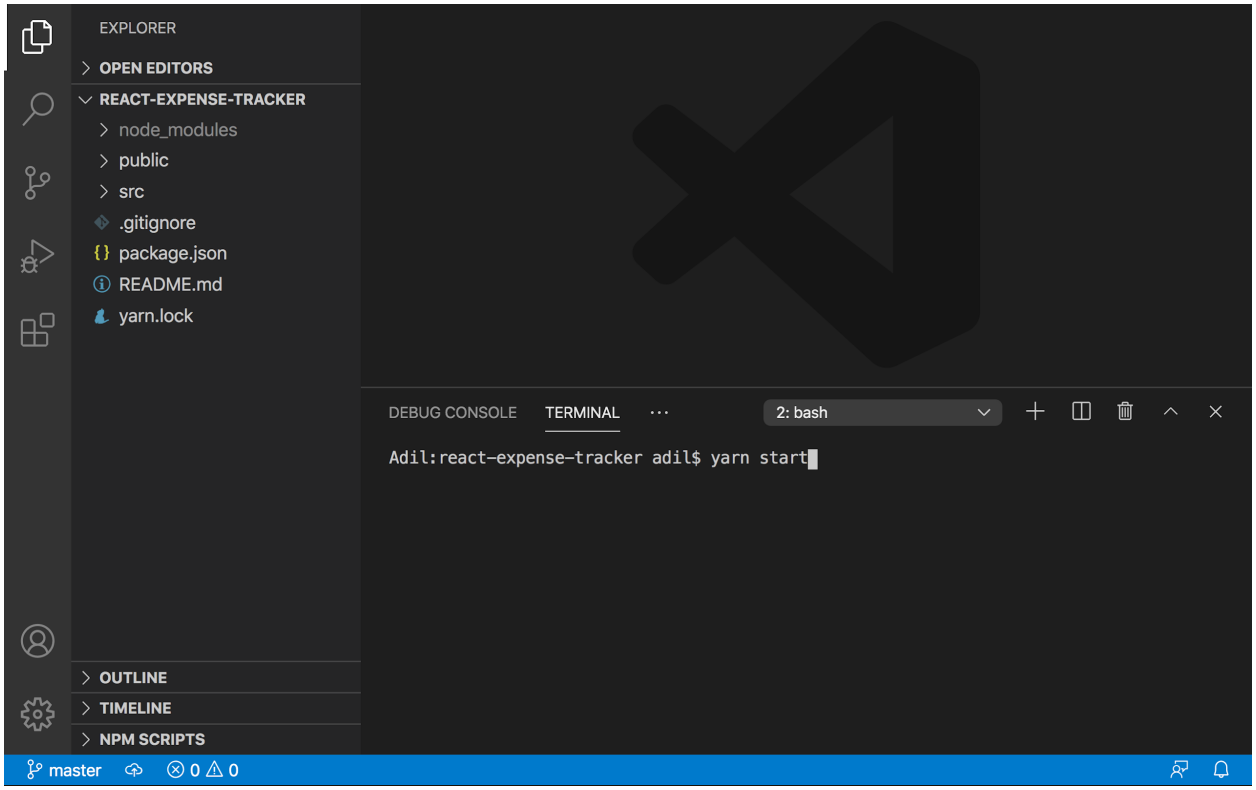

EXPLORER
OPEN EDITORS
REACT-EXPENSE-TRACKER
node_modules
public
src
.gitignore
package.json
README.md
yarn.lock
OUTLINE
TIMELINE
NPM SCRIPTS
Show All Commands
Go to File
Find in Files
Start Debugging
F5
Toggle Terminal
master
0
0

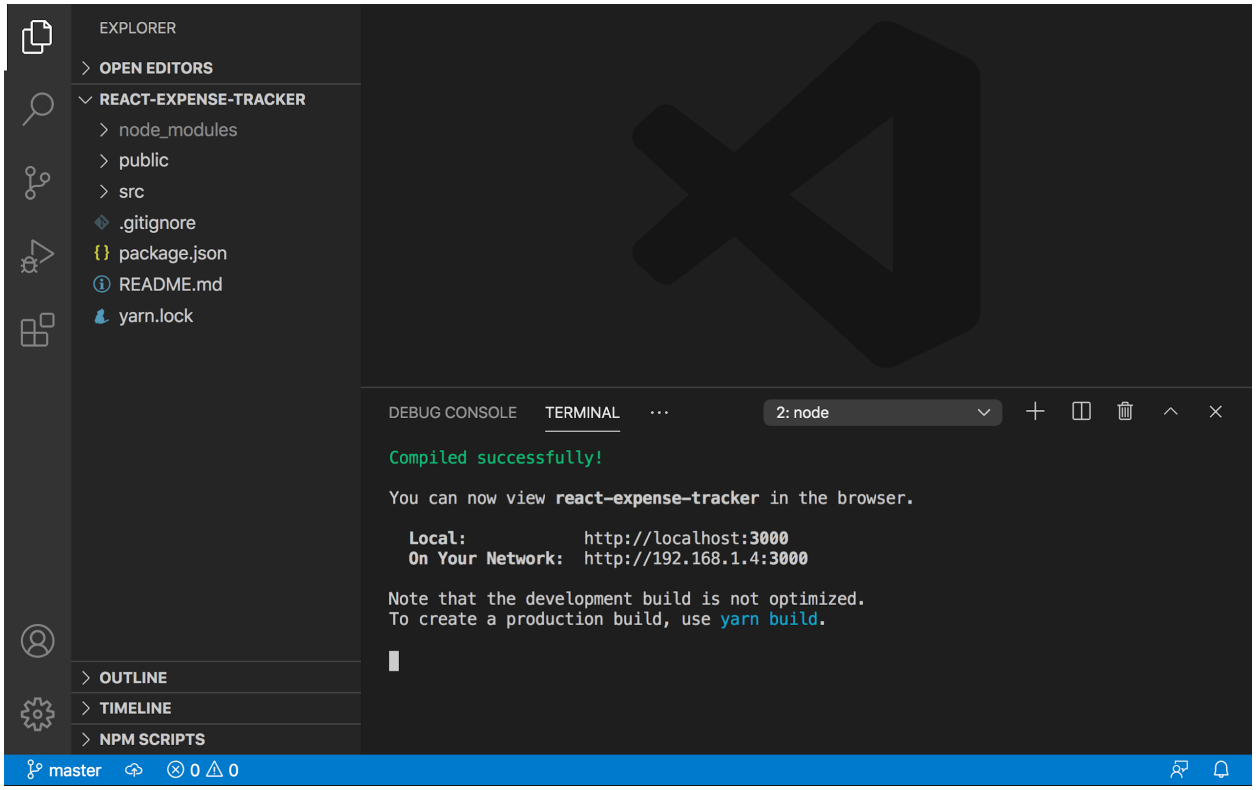
Part III - View the Project in the Browser

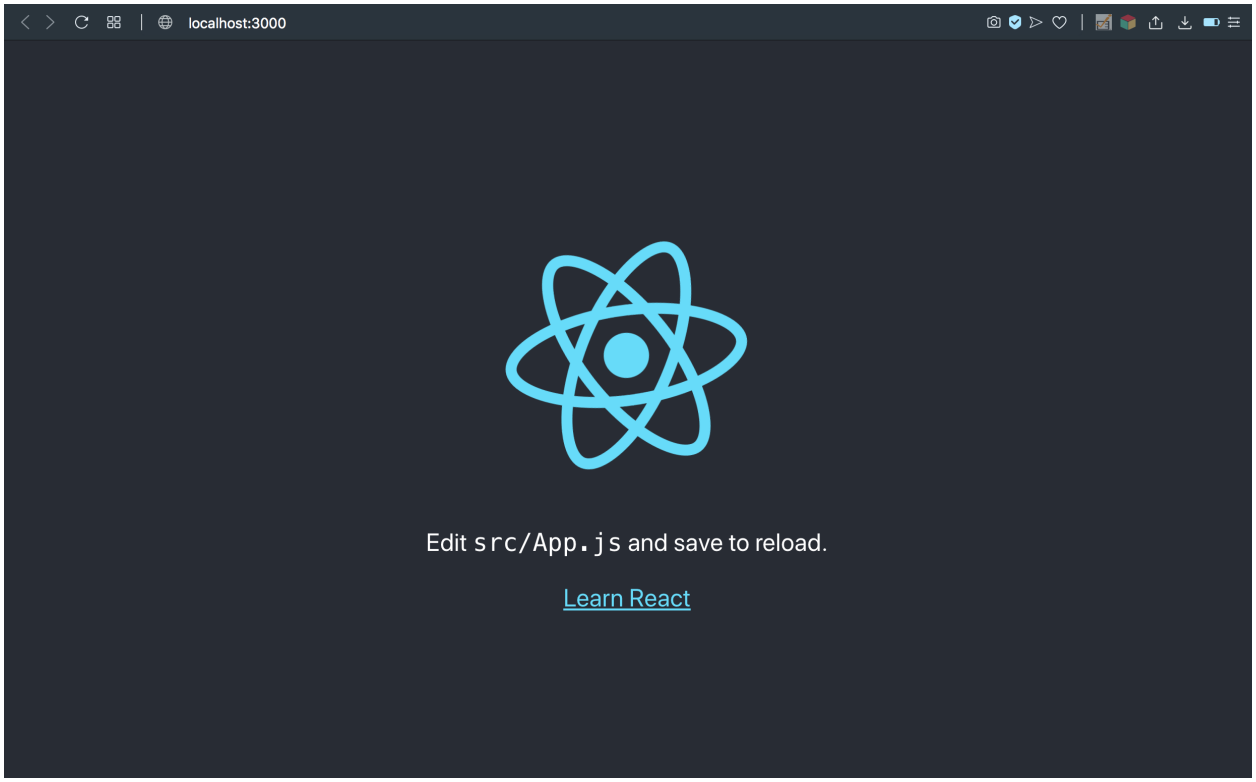
First, let's open the project in the terminal





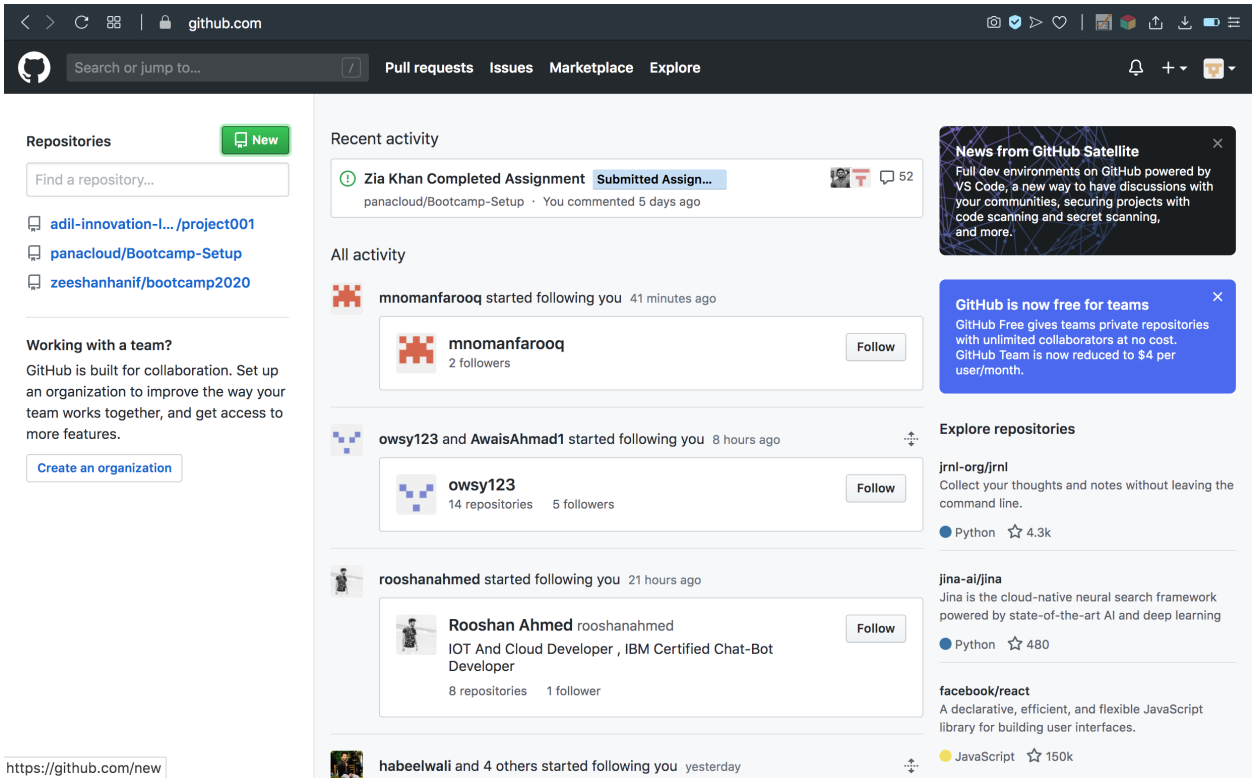

EXPLORER
OPEN EDITORS
REACT-EXPENSE-TRACKER
node_modules
public
src
.gitignore
package.json
README.md
yarn.lock
OUTLINE
TIMELINE
NPM SCRIPTS
DEBUG CONSOLE
TERMINAL
2: bash
Adil:react-expense-tracker adil$ yarn start
master
0 0


EXPLORER
OPEN EDITORS
REACT-EXPENSE-TRACKER
node_modules
public
src
.gitignore
package.json
README.md
yarn.lock
OUTLINE
TIMELINE
NPM SCRIPTS
DEBUG CONSOLE
TERMINAL
2: node
Compiled successfully!
You can now view react-expense-tracker in the browser.
Local: http://localhost:3000
On Your Network: http://192.168.1.4:3000
Note that the development build is not optimized.
To create a production build, use yarn build.
master
0 0

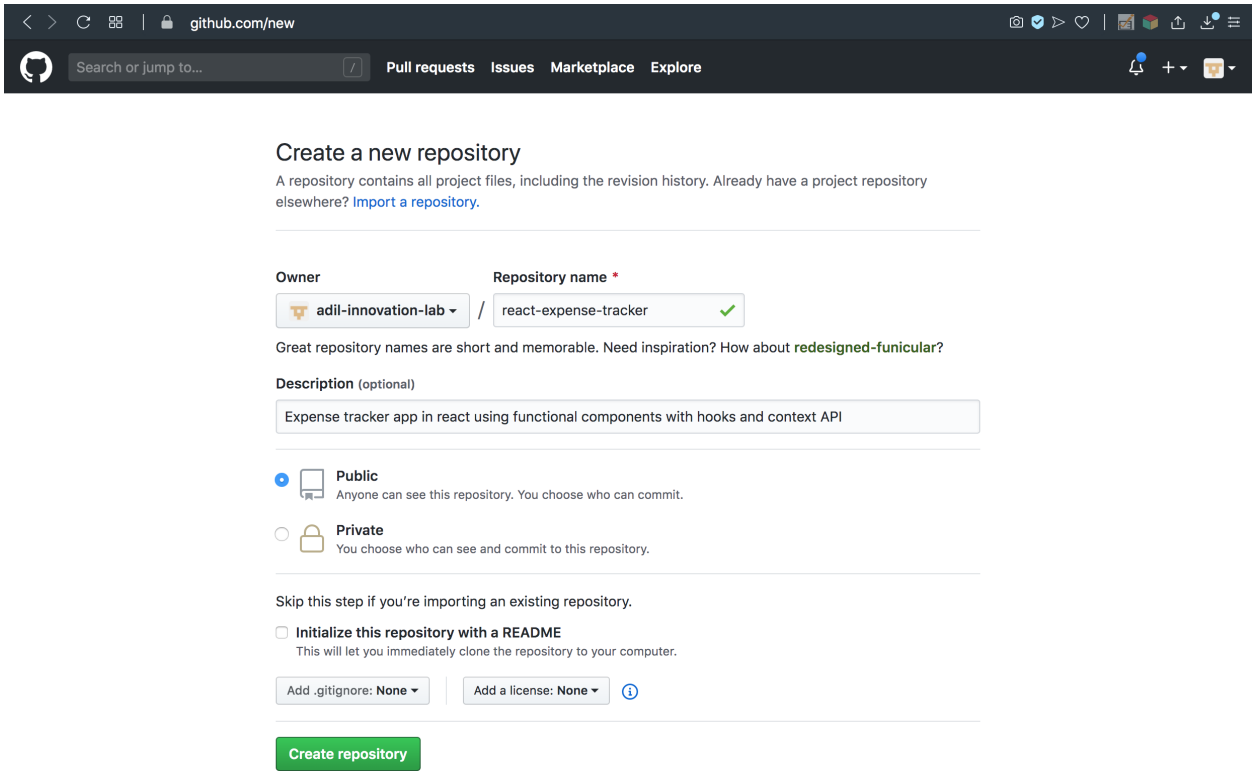

localhost:3000
Edit src/App.js and save to reload.
Learn React

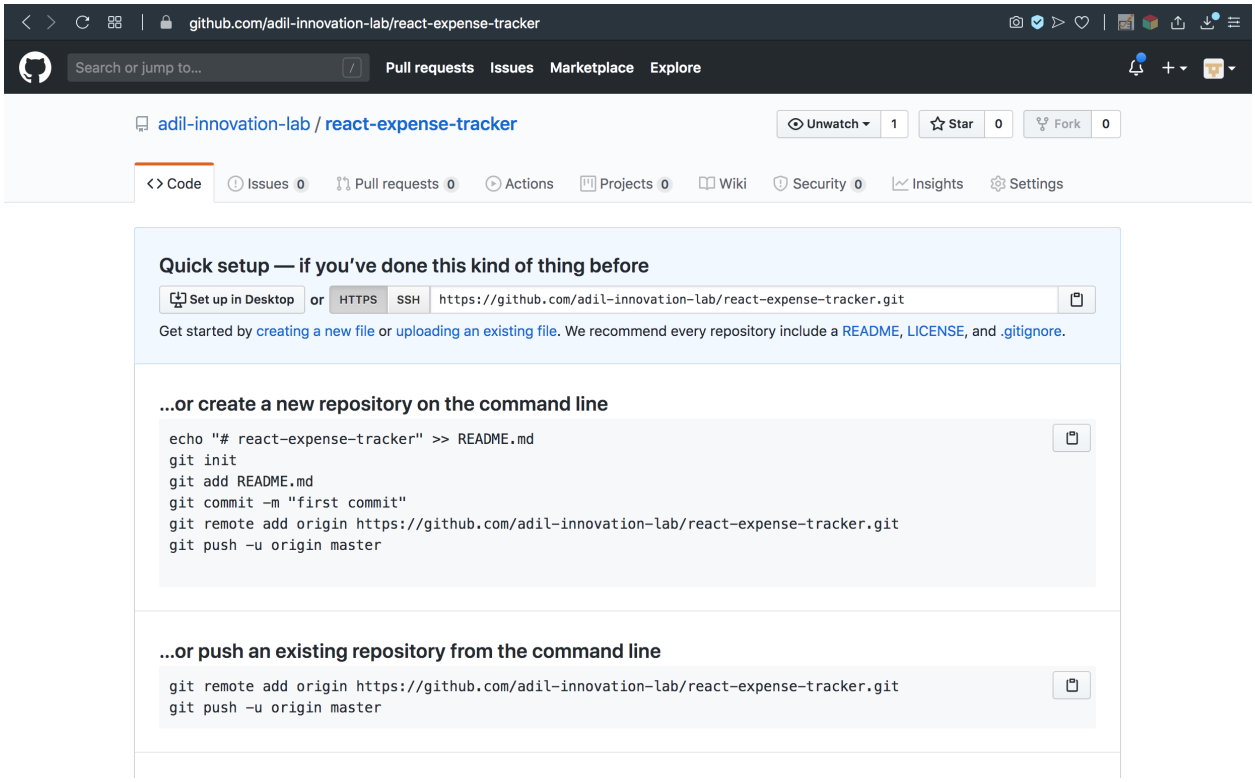
Part IV - Add Repository to GitHub with the CLI

Step 1: Create a new repository on GitHub

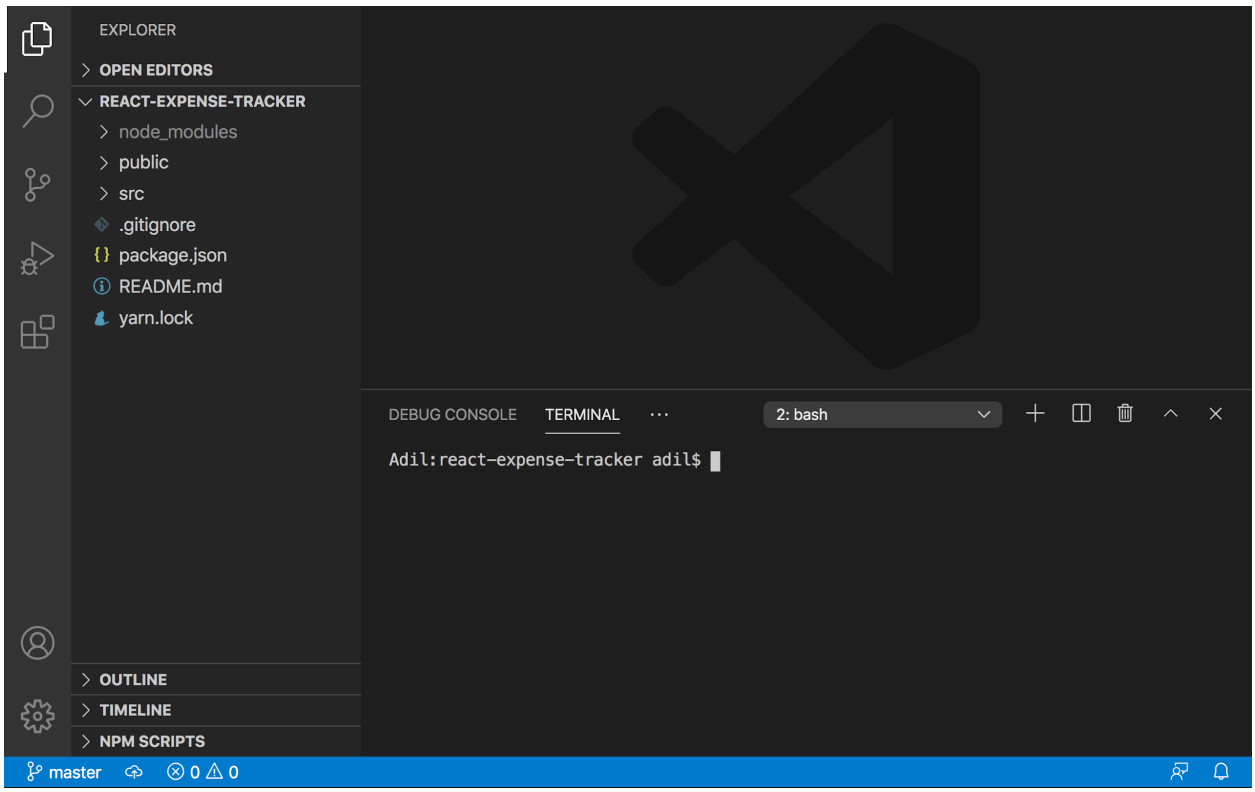


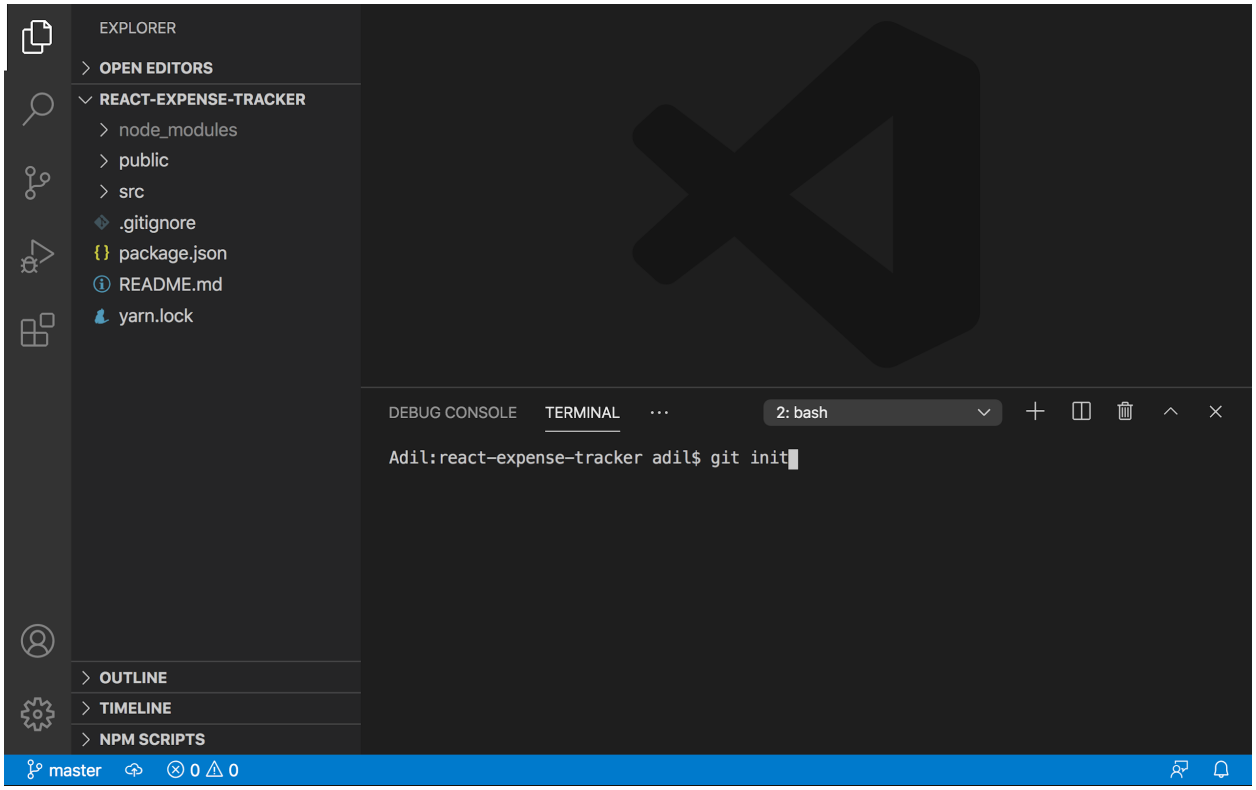
Do not initialize with a readme file, gitignore, or license:

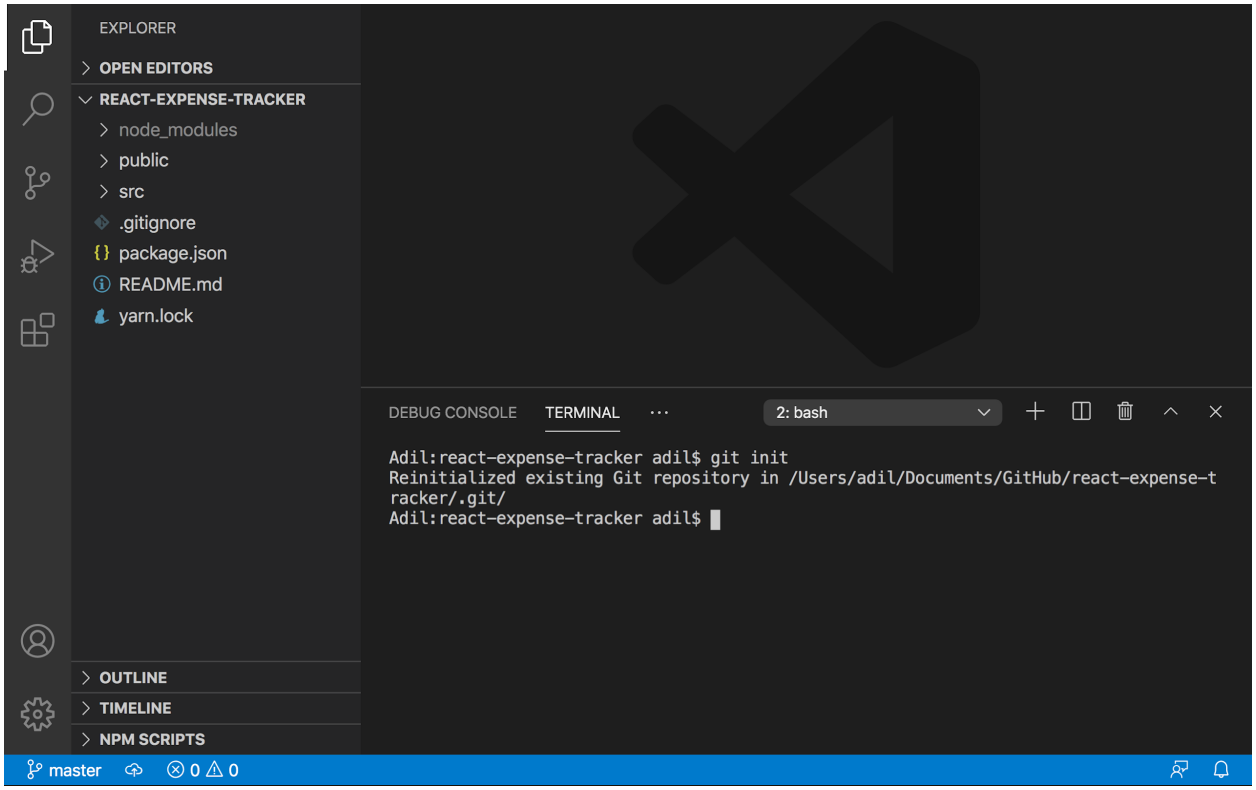




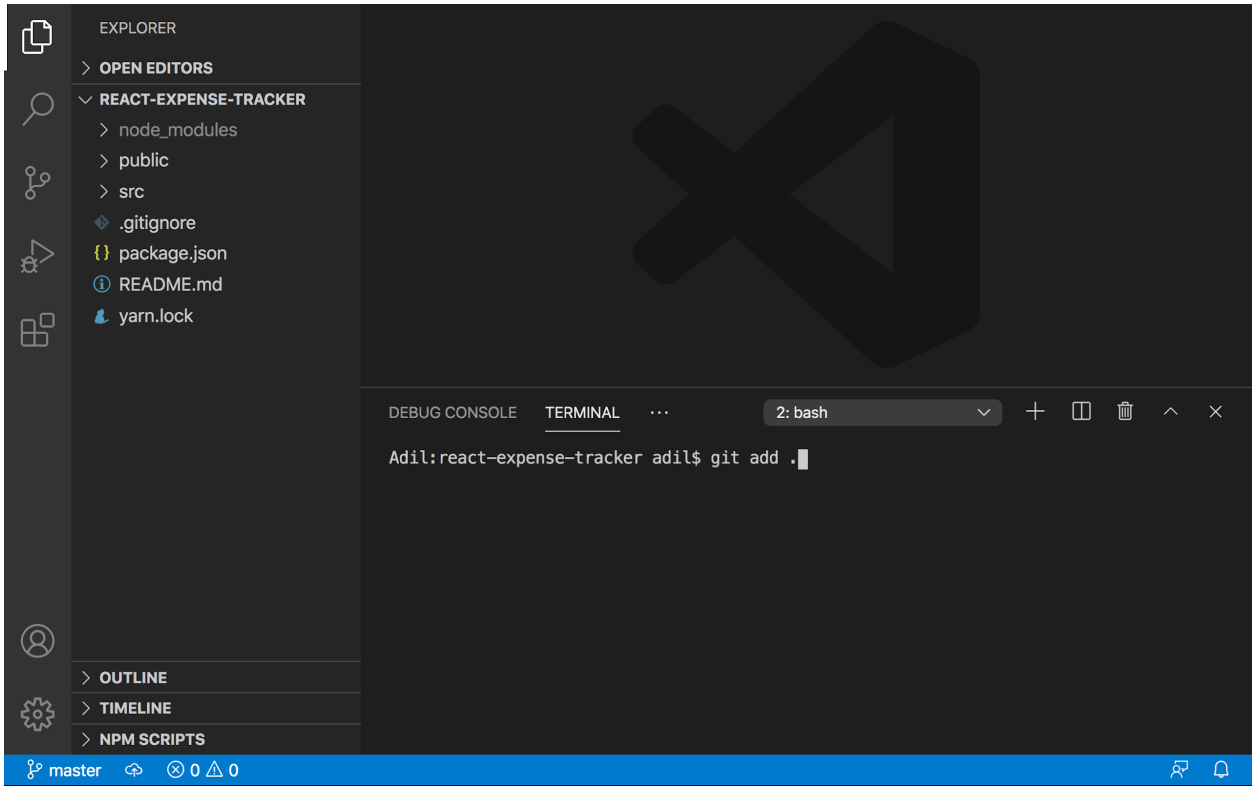
Step 2: Initialize the local directory as a git repository:
Open terminal in our local folder:



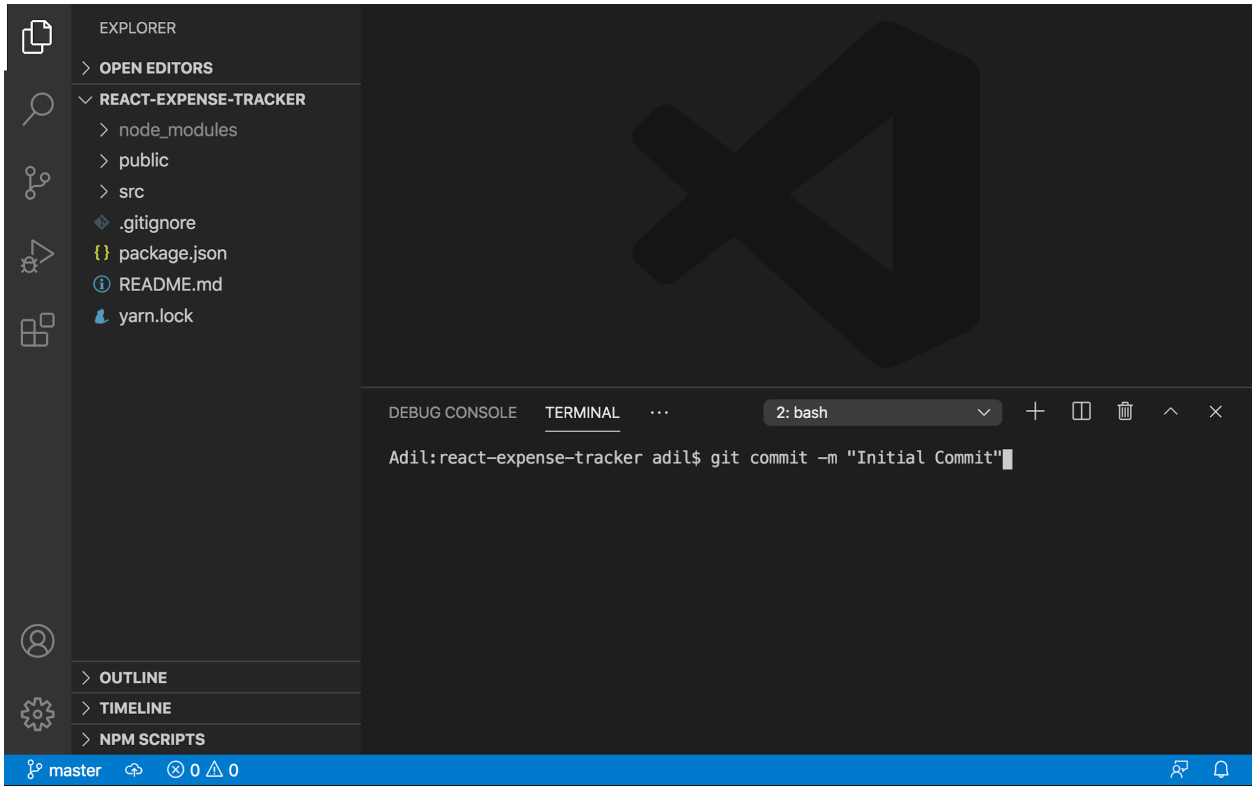

EXPLORER
OPEN EDITORS
REACT-EXPENSE-TRACKER
node_modules
public
src
.gitignore
package.json
README.md
yarn.lock
OUTLINE
TIMELINE
NPM SCRIPTS
DEBUG CONSOLE
TERMINAL
2: bash
Adil:react-expense-tracker adil$ git init
master
0 0

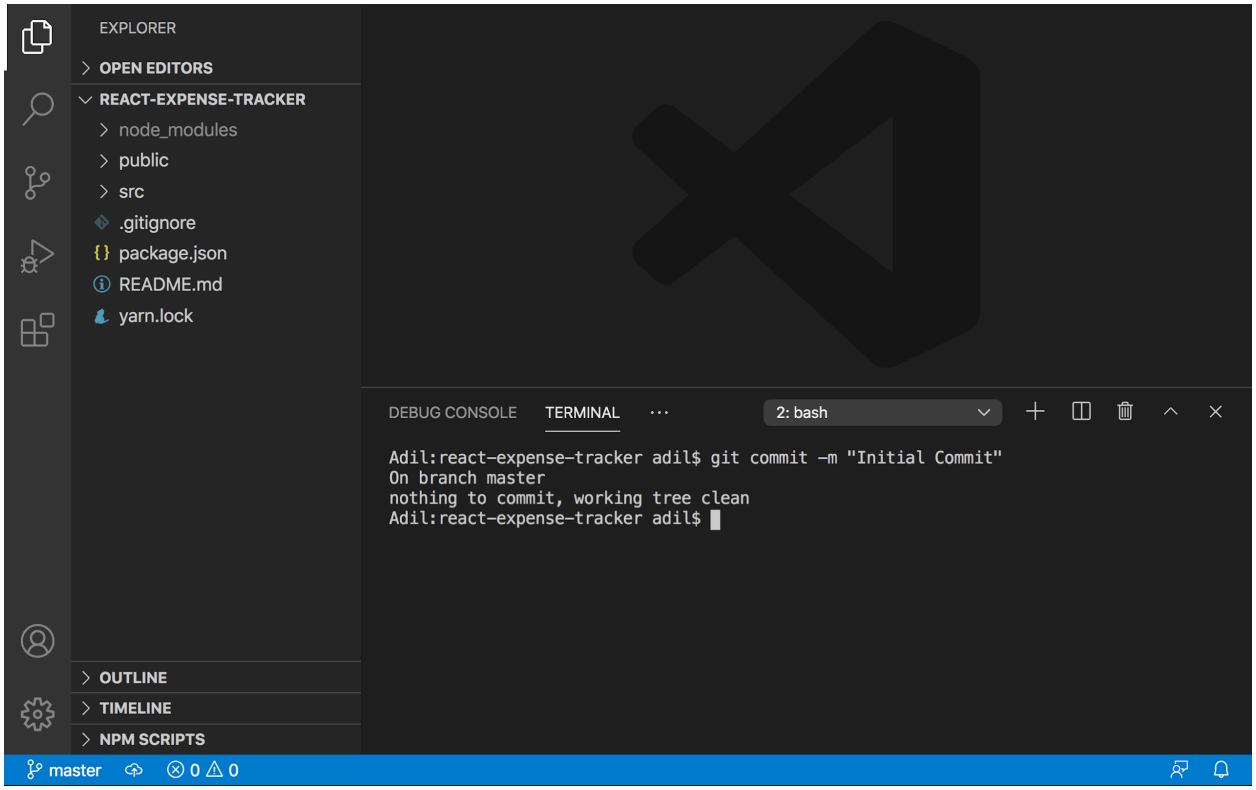

EXPLORER
OPEN EDITORS
REACT-EXPENSE-TRACKER
node_modules
public
src
.gitignore
package.json
README.md
yarn.lock
OUTLINE
TIMELINE
NPM SCRIPTS
DEBUG CONSOLE
TERMINAL
2: bash
Adil:react-expense-tracker adil$ git init
Reinitialized existing Git repository in /Users/adil/Documents/GitHub/react-expense-tracker/.git/
Adil:react-expense-tracker adil$
master
0 0

Step 3: Add files from local directory to git staging area to prepare for commit

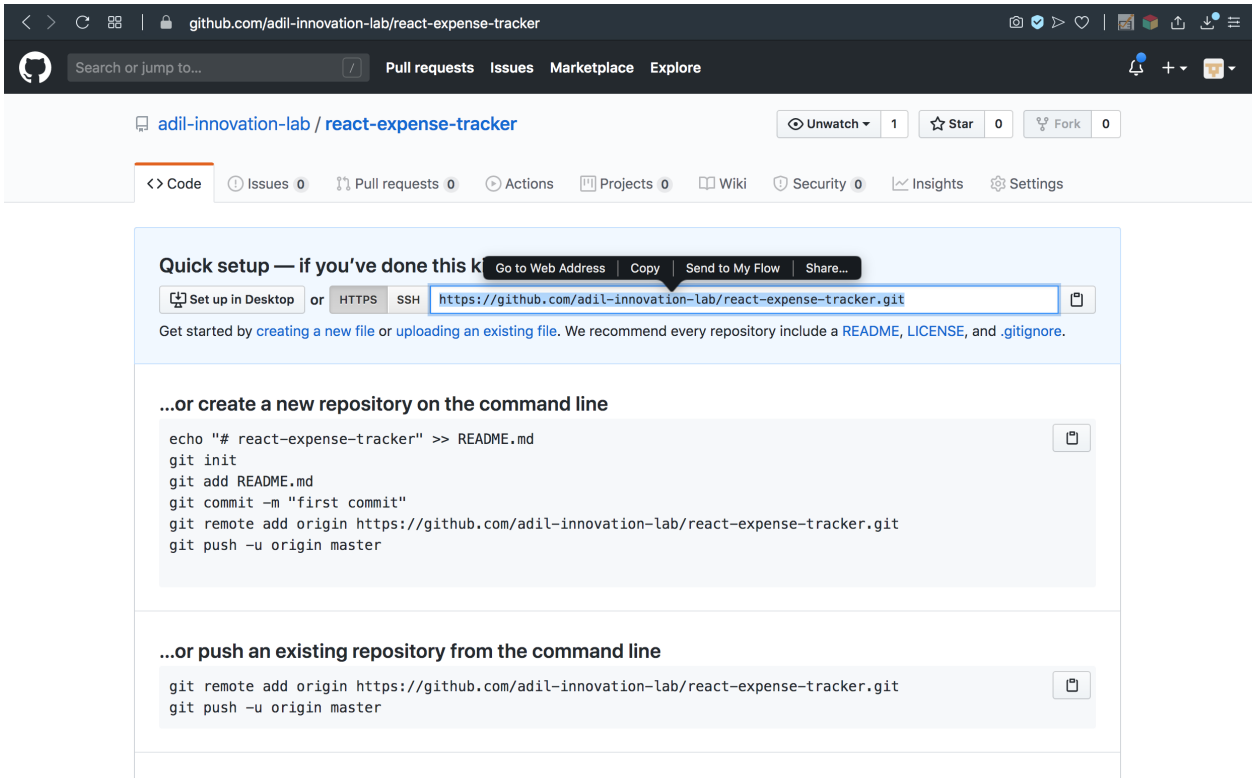


Step 4: Commit files from staging area to local git repository

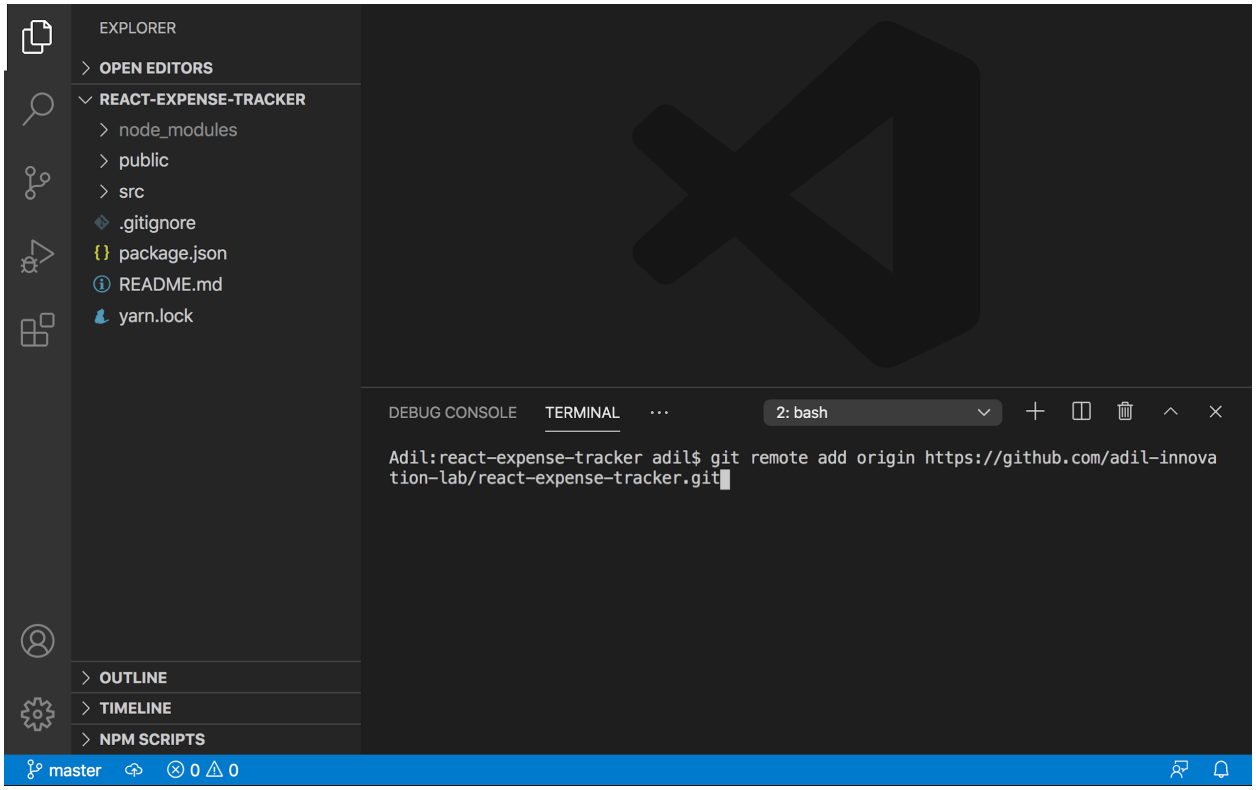




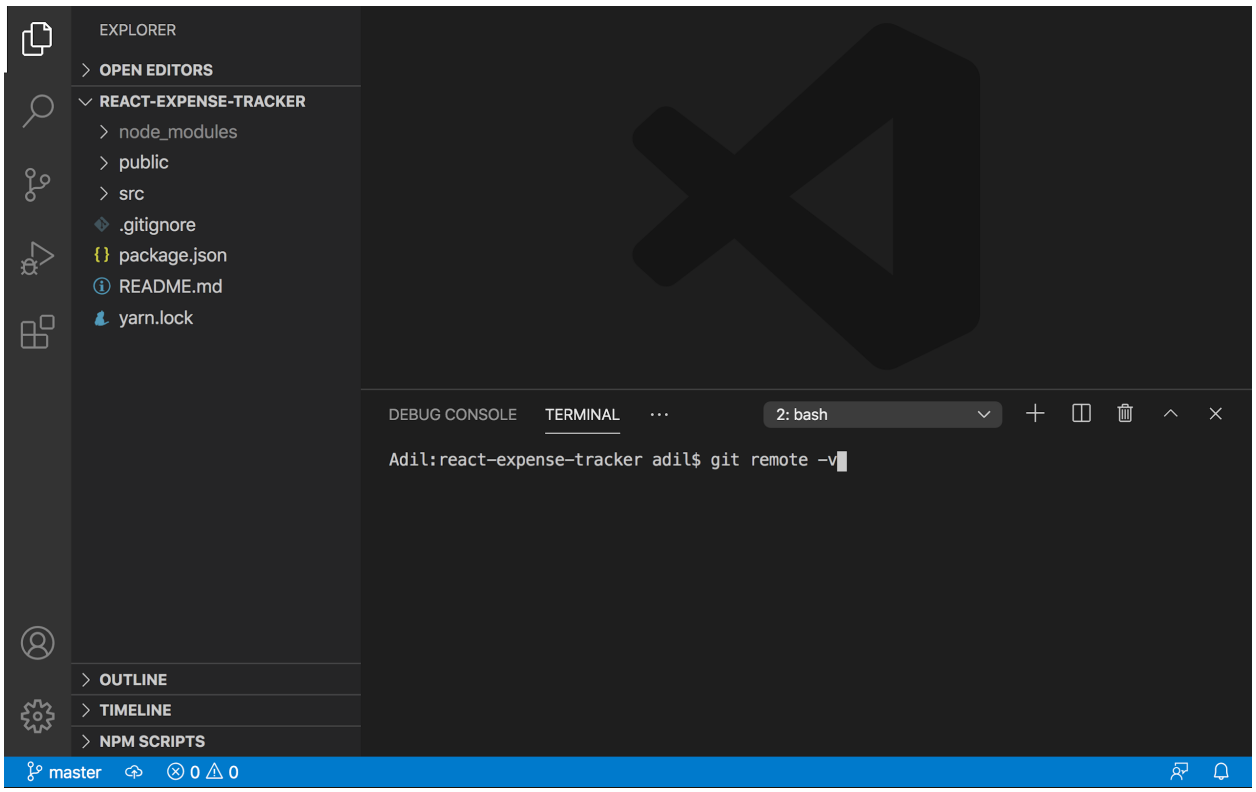
Step 5: Get the URL of the remote repository from GitHub:

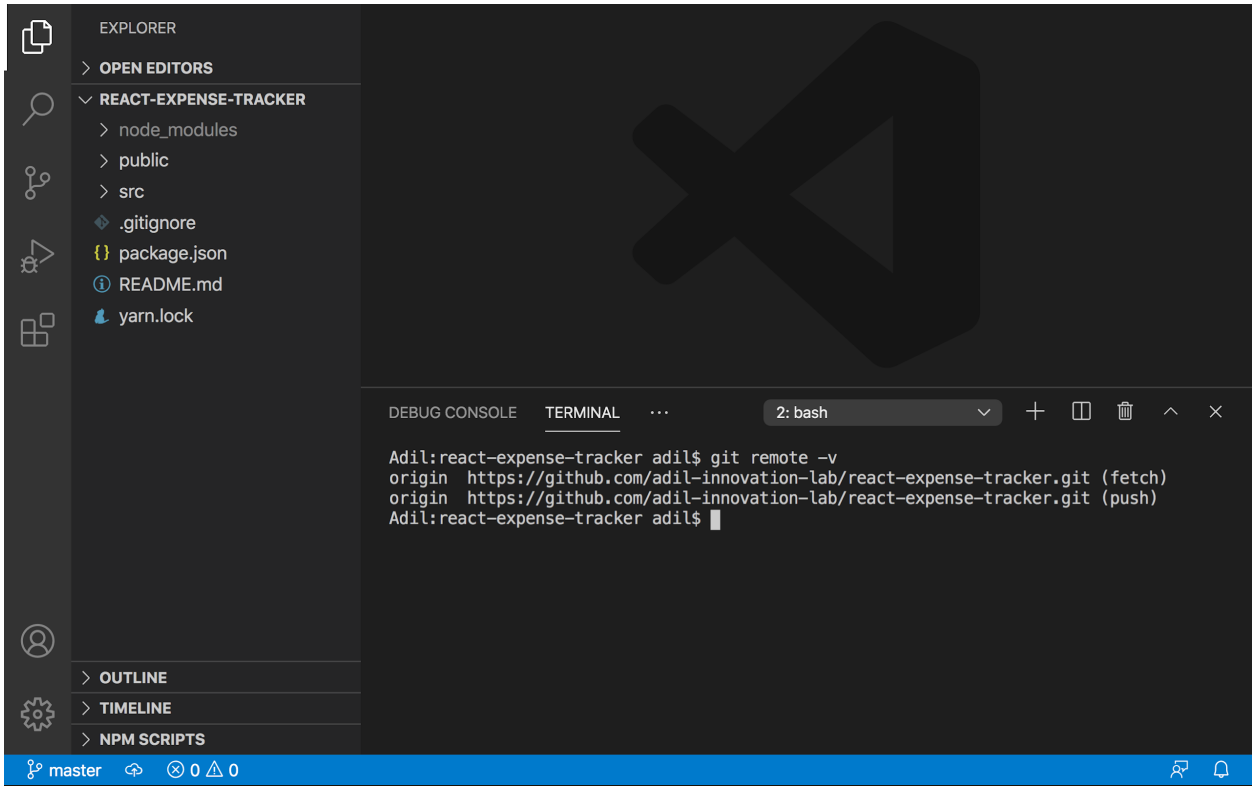


Step 6: Add remote URL to git via the terminal so local repository can be pushed to the right place

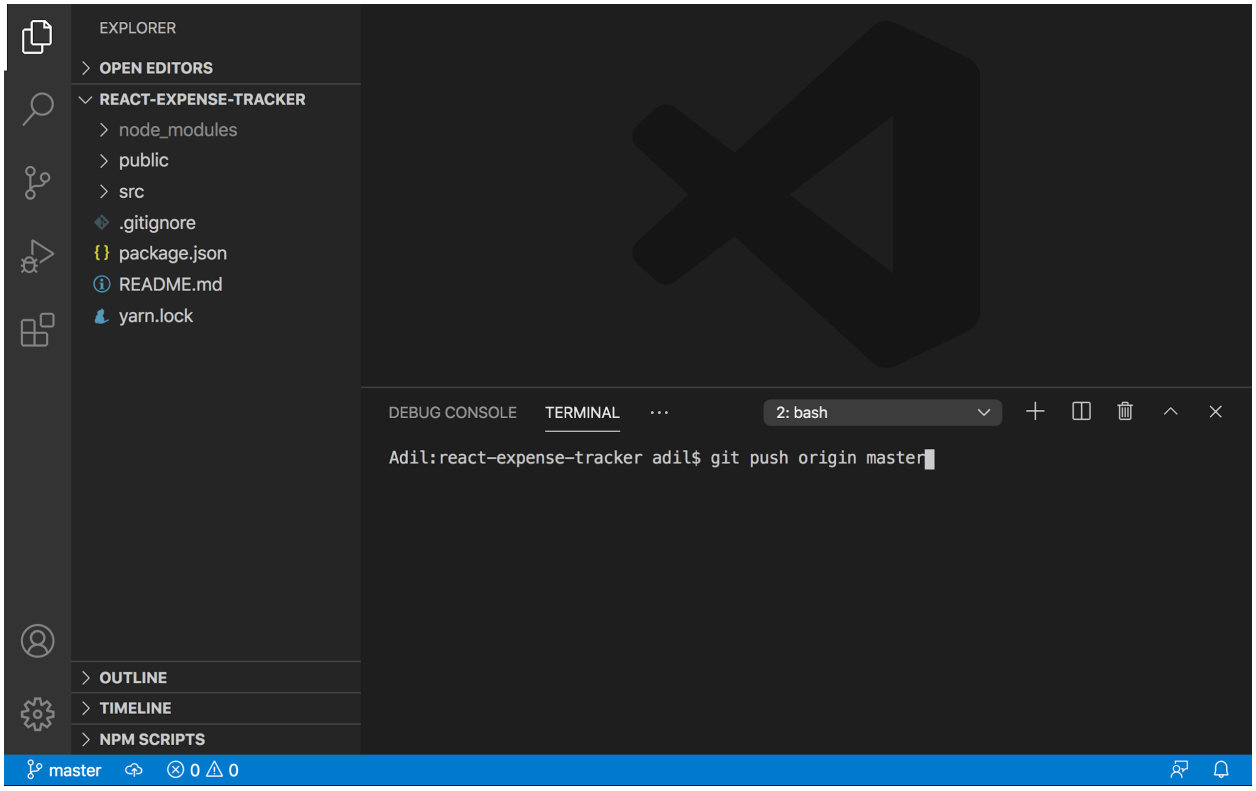


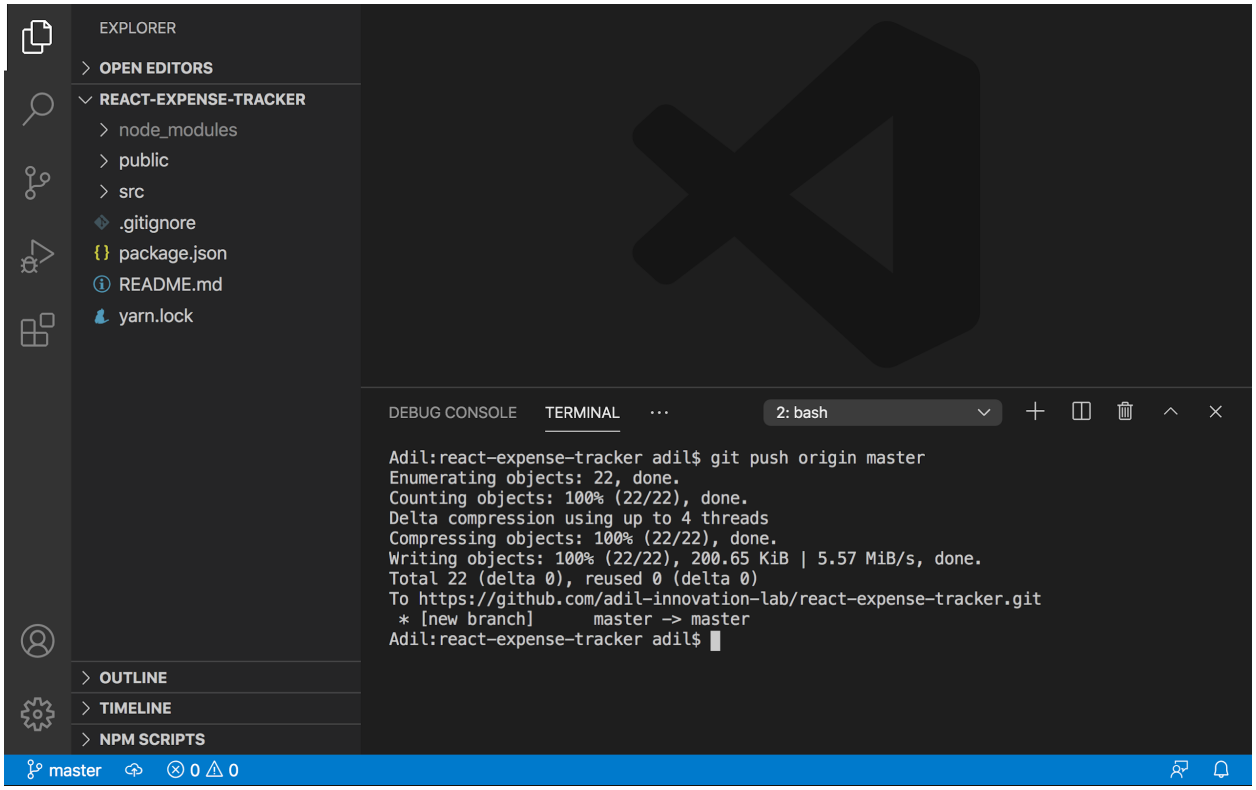
Verify the remote URL



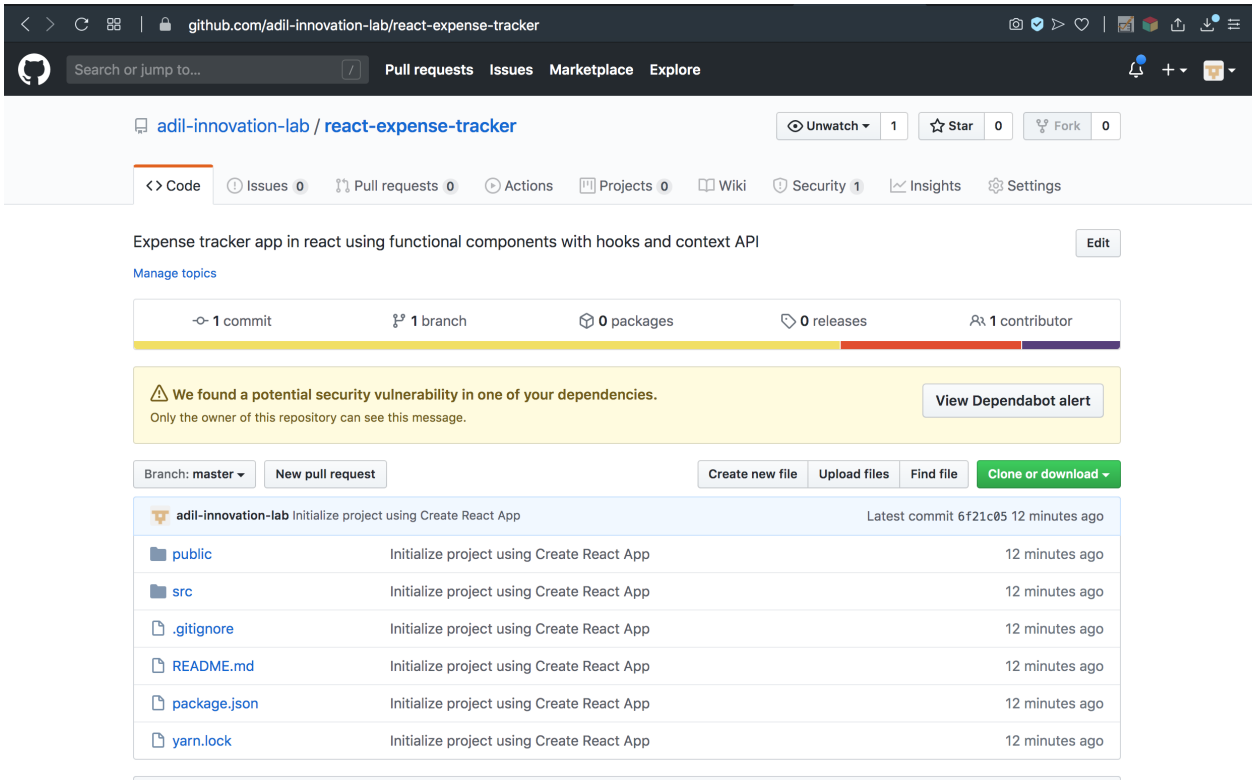


Step 7: Push the code from local git repository to the remote git repository



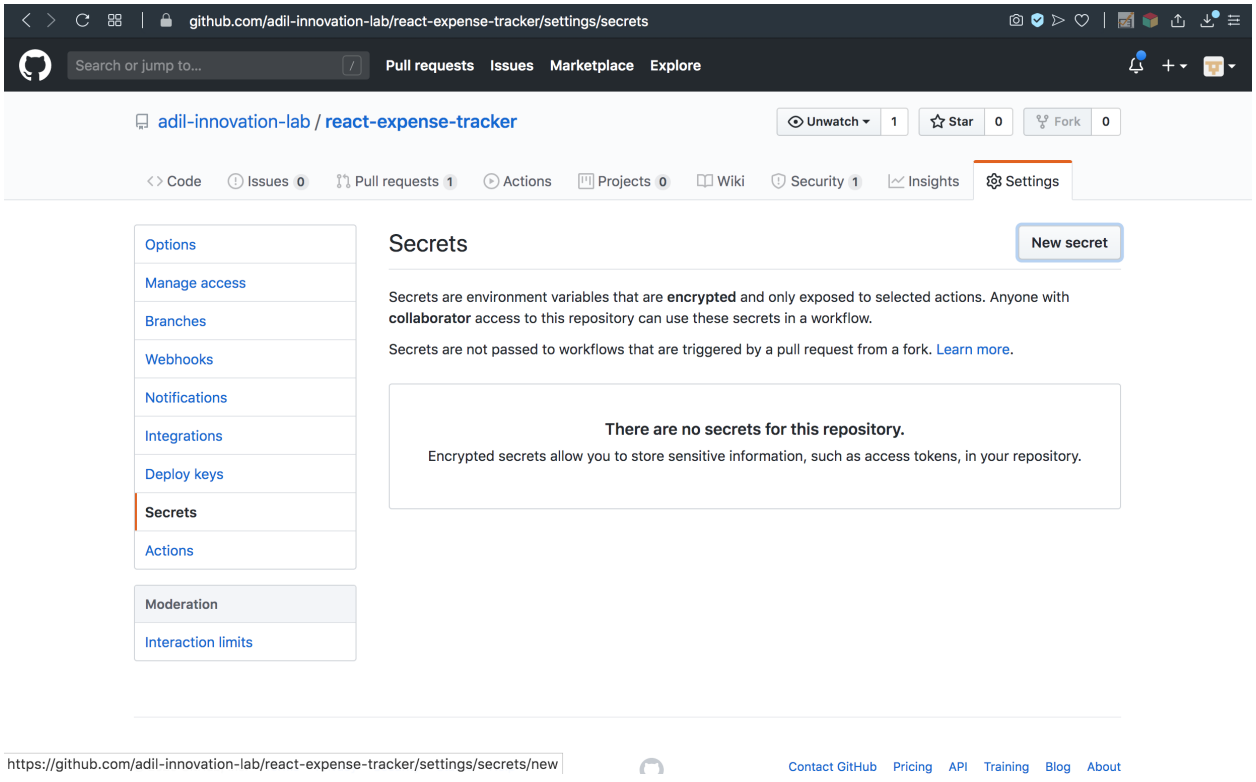


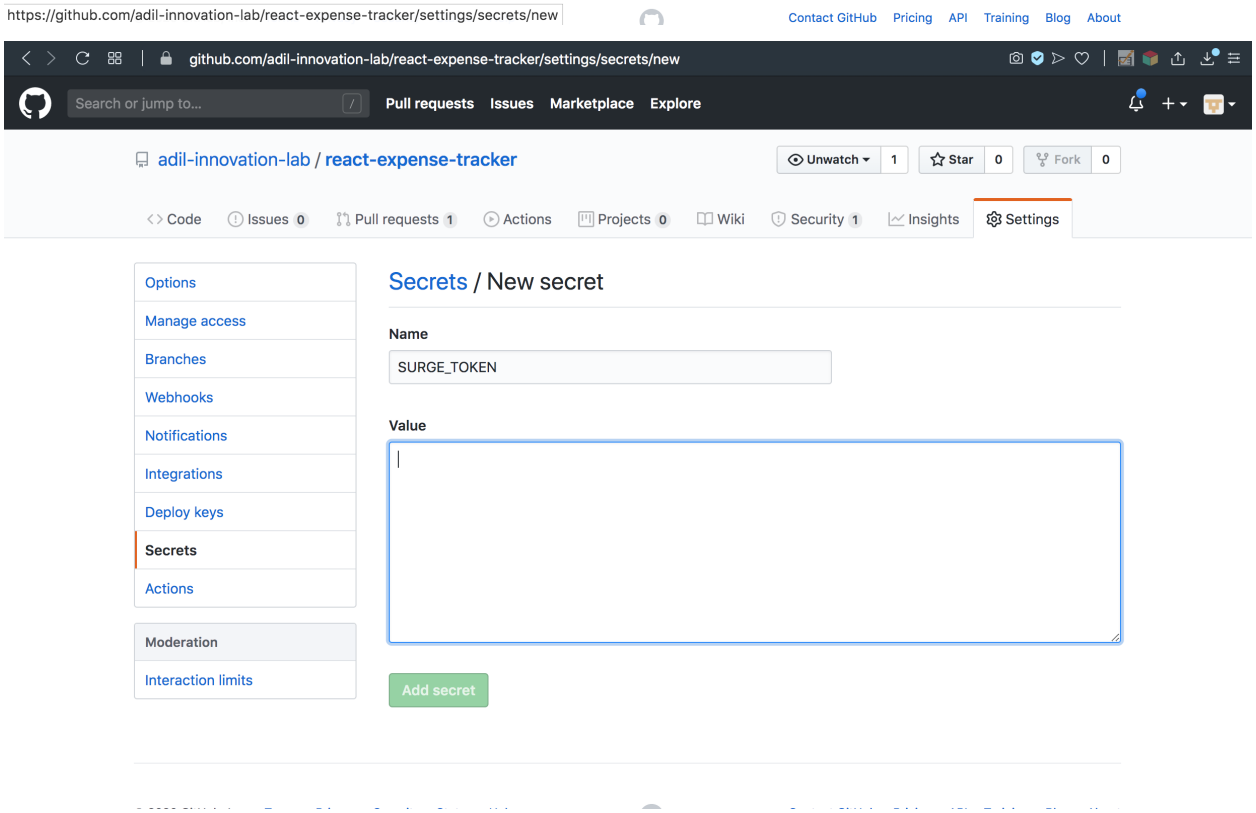
Step 8: Verify changes pushed to remote git repository



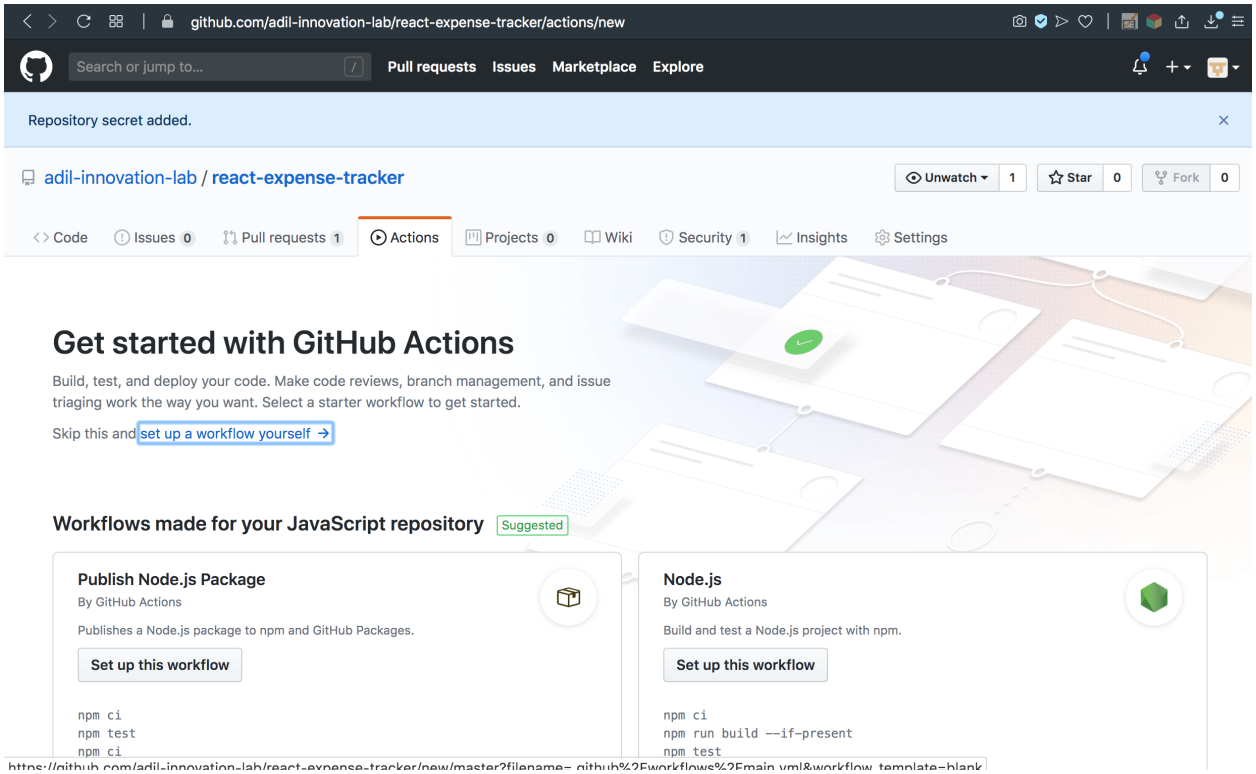
Part V - Deploy React App to Surge with a GitHub Workflow

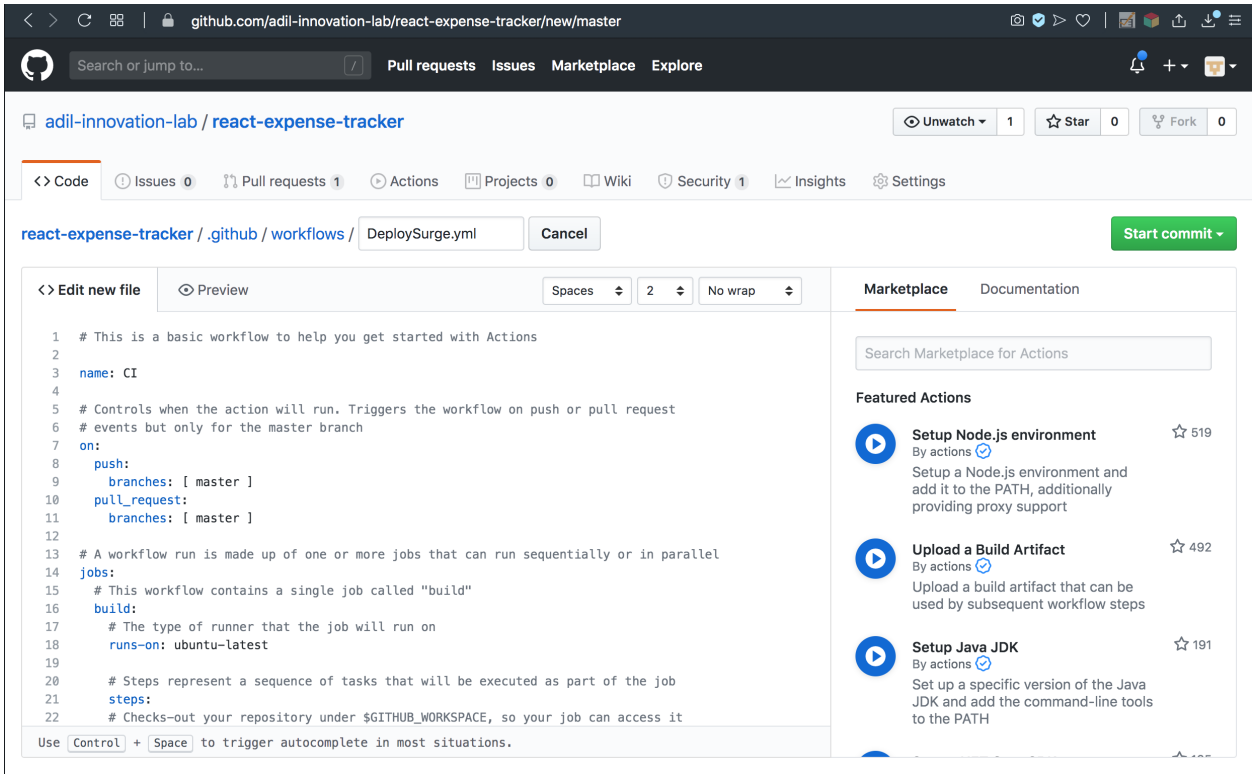
Step 1: Create surge token and add a secret to the GitHub repository



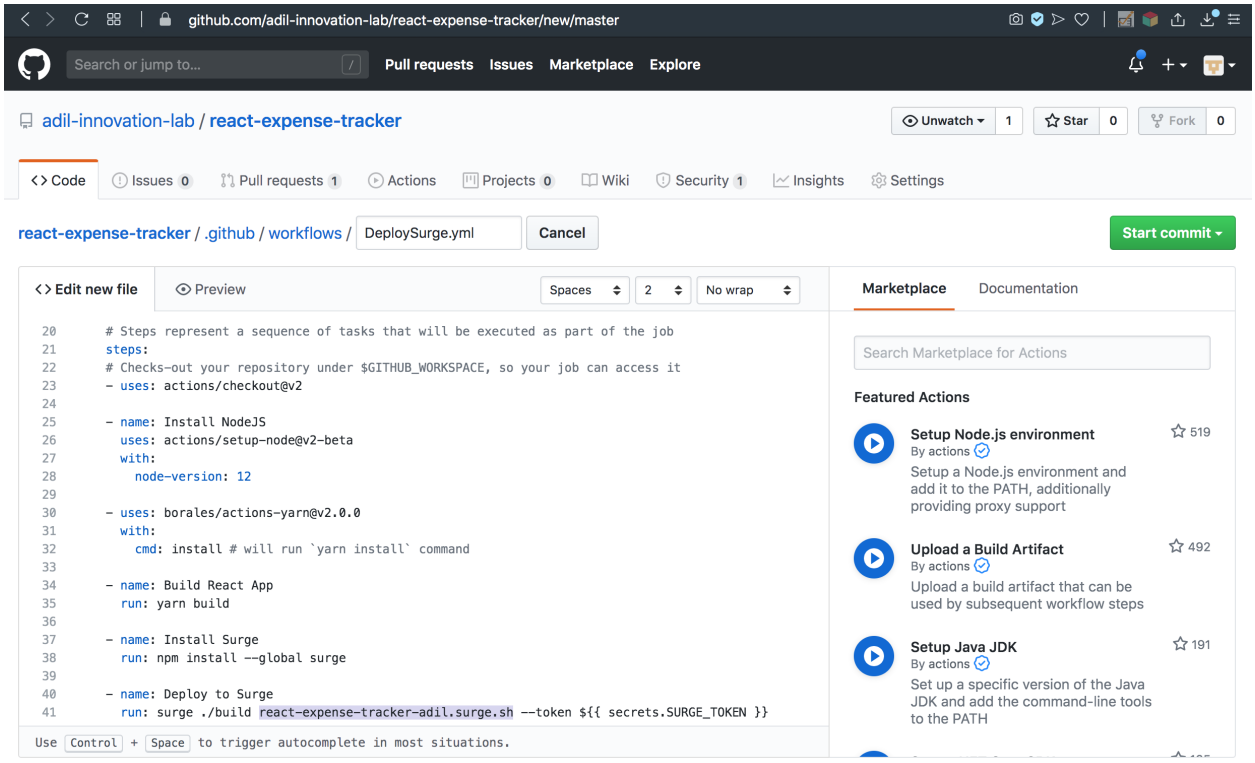


Step 2: Go to the Actions tab in the GitHub repository on github.com and click the set up a workflow yourself link

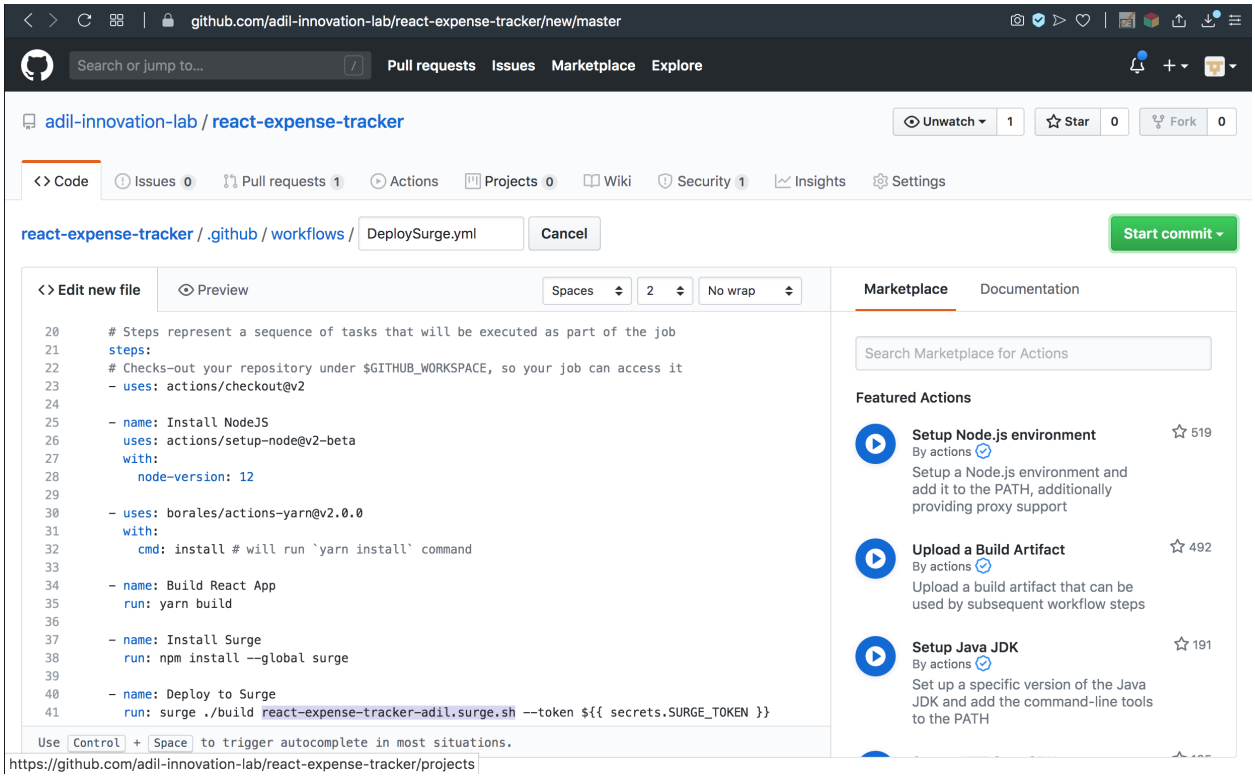




Step 3: Write the YAML for the workflow

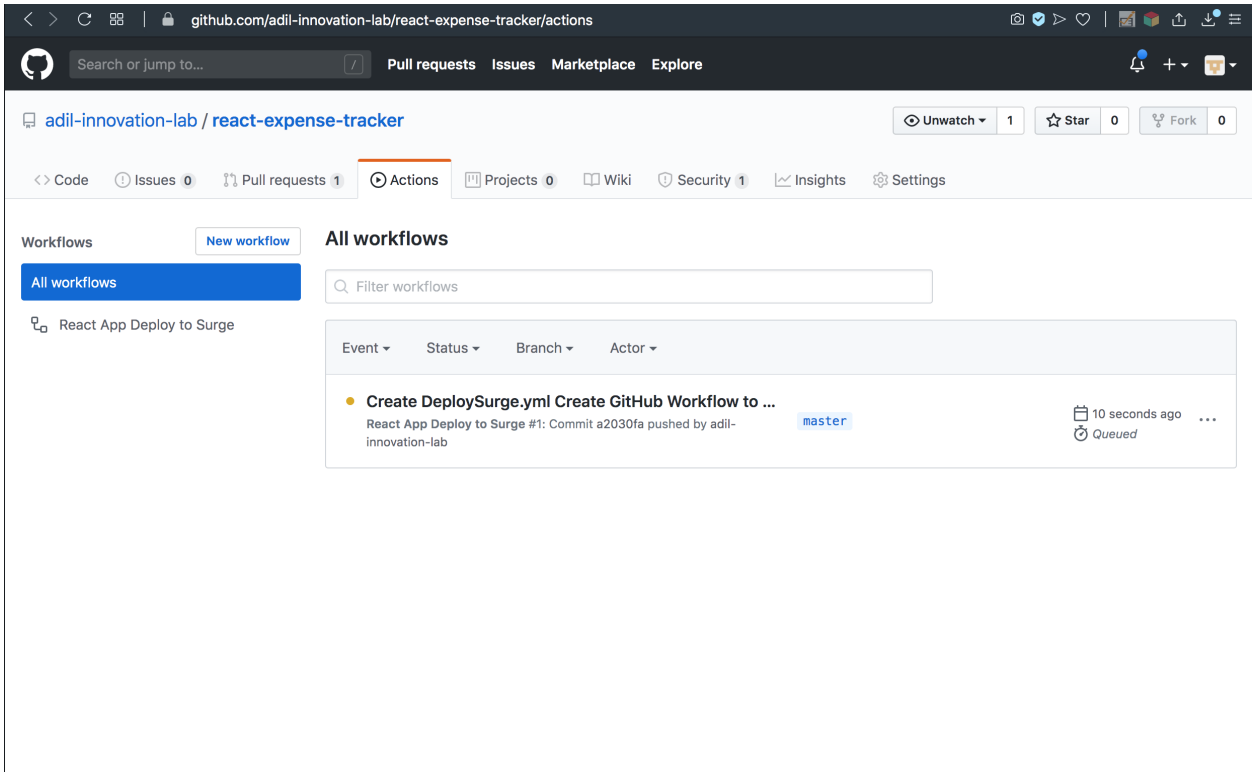


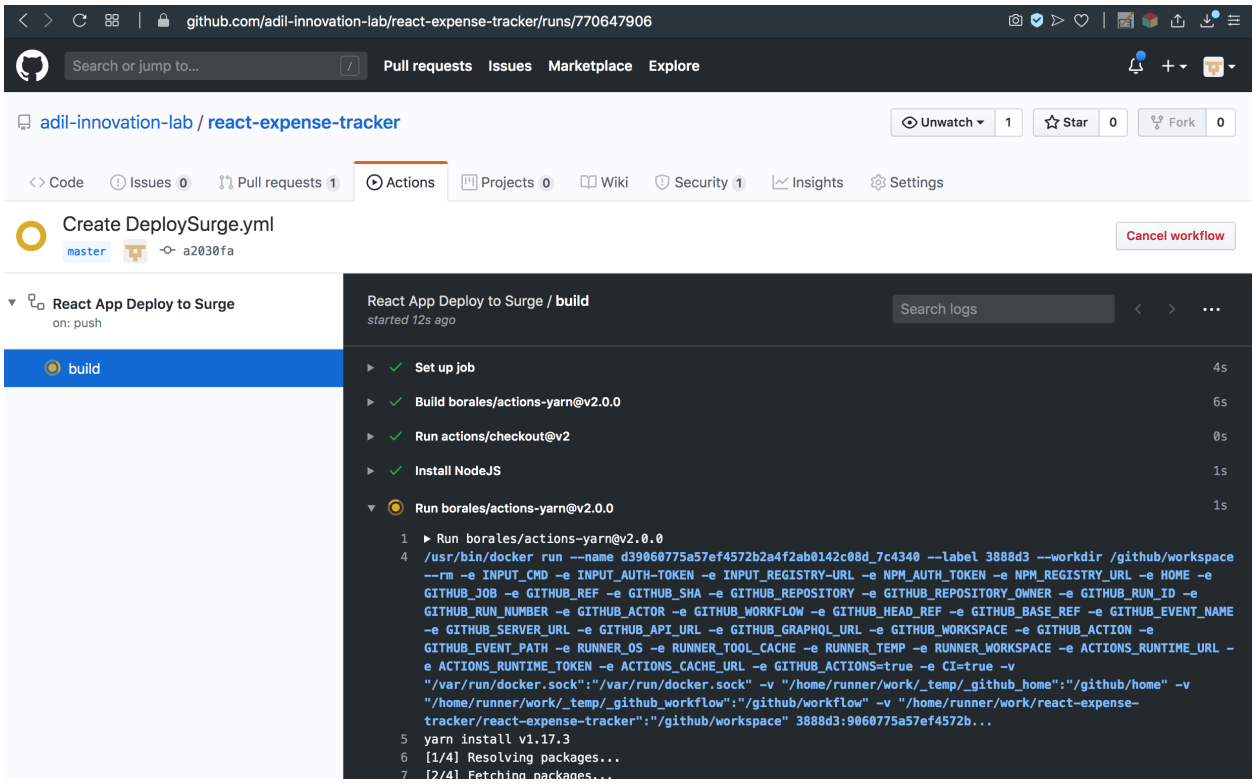
Step 4: Commit the workflow to the master branch

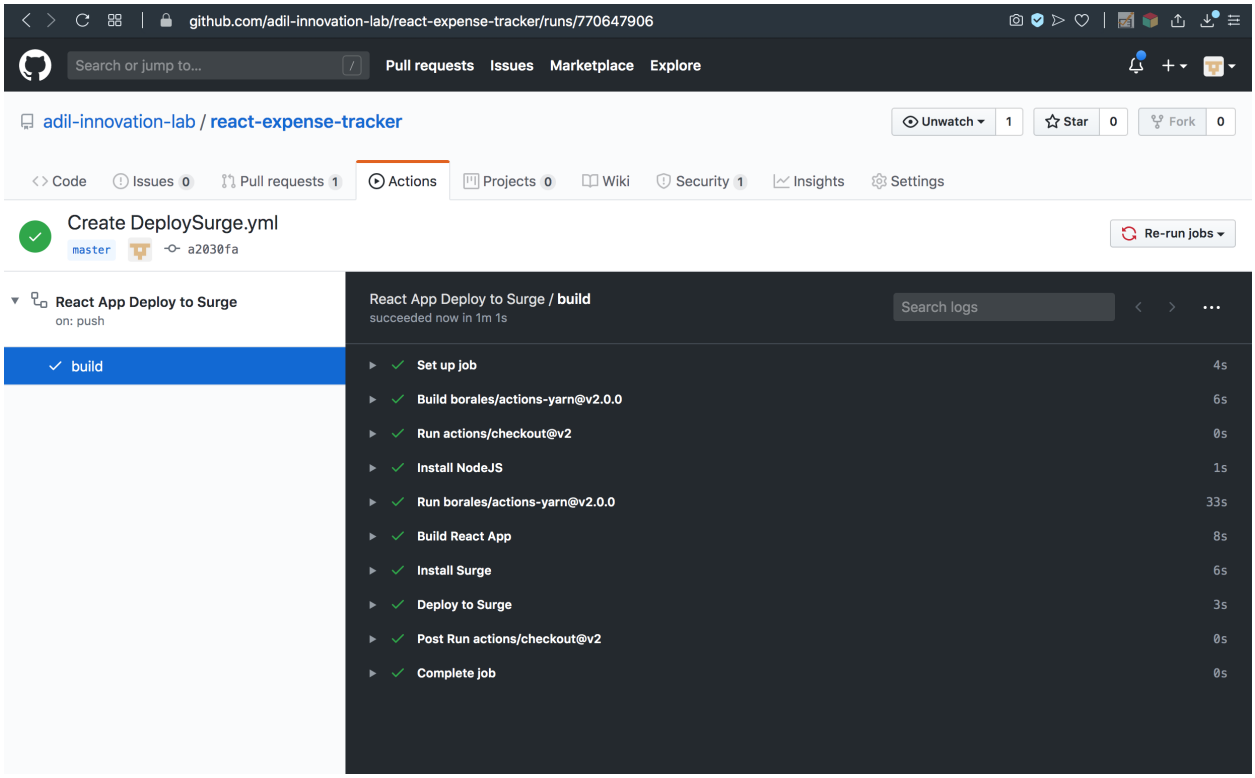


https://github.com/adil-innovation-lab/react-expense-tracker/projects

Step 5: Check the Actions tab for the workflow to deploy successfully

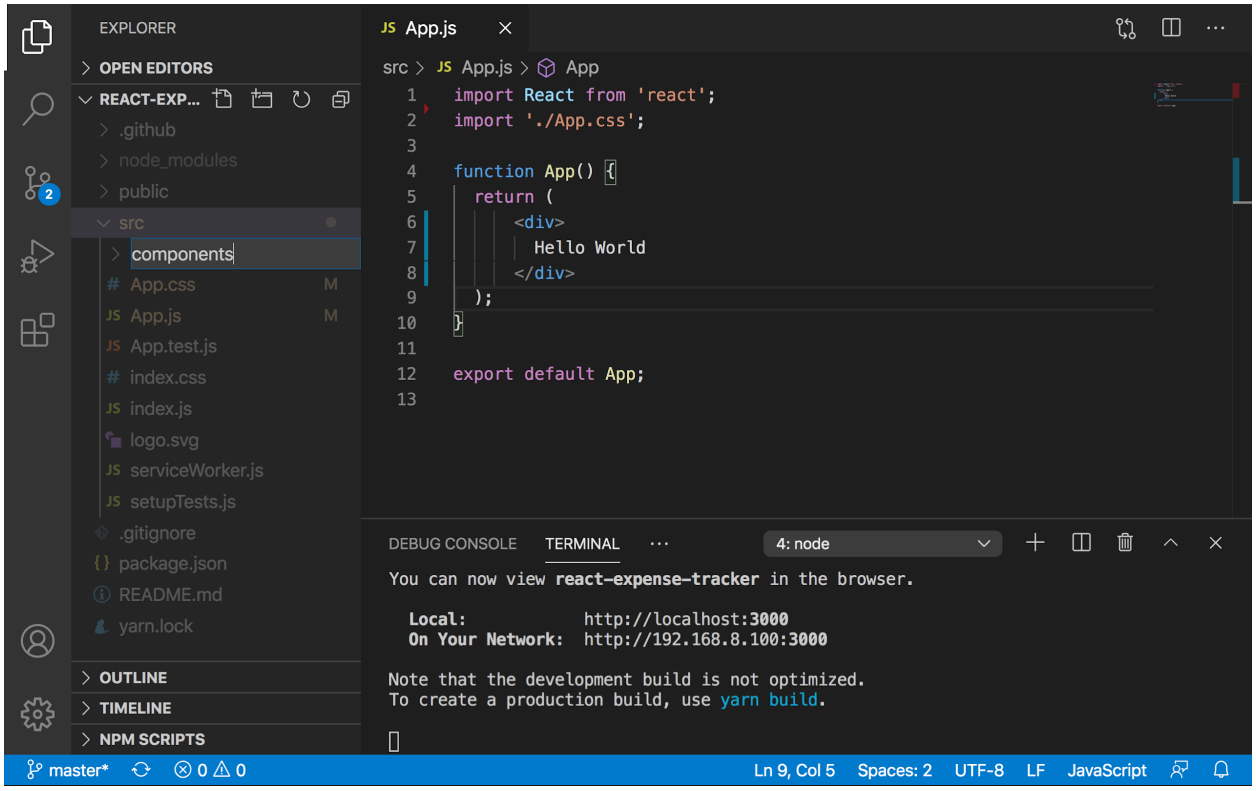




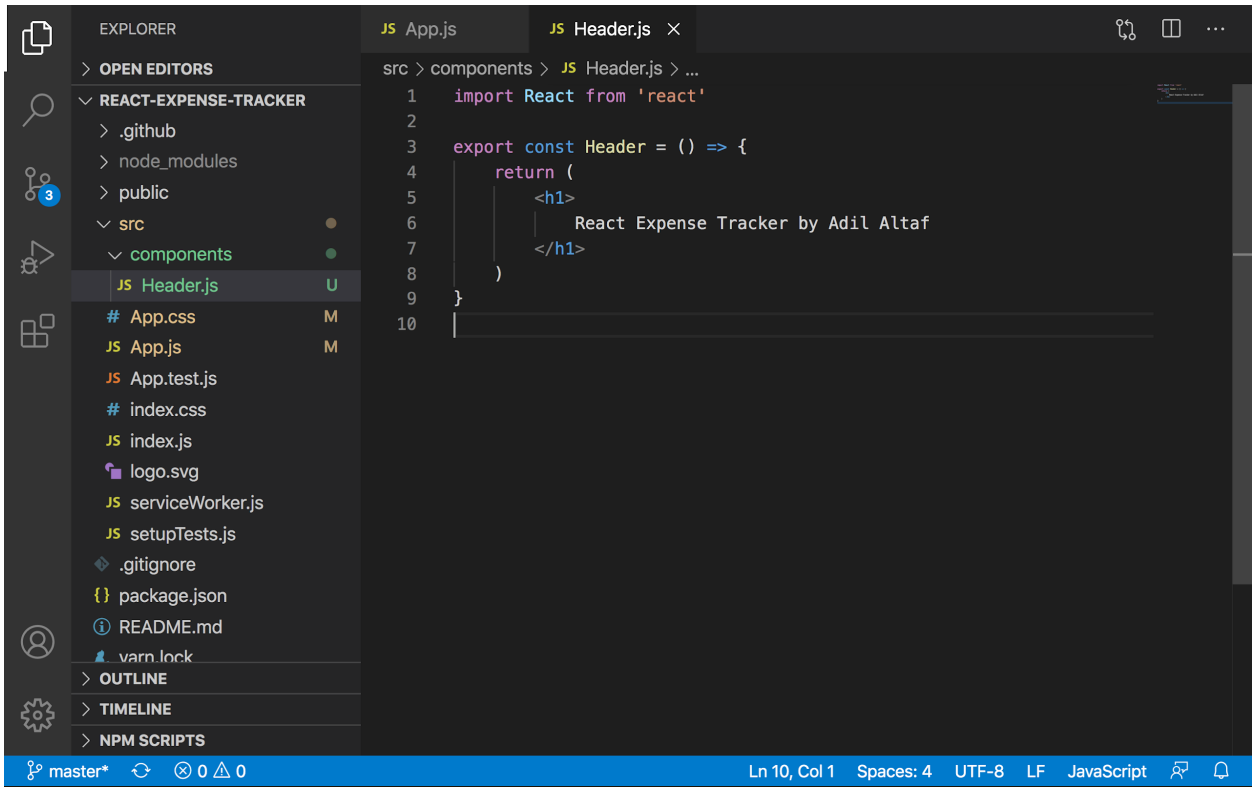

github.com/adil-innovation-lab/react-expense-tracker/runs/770647906
Search or jump to...
Pull requests
Issues
Marketplace
Explore
adil-innovation-lab / react-expense-tracker
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 1
Actions
Projects 0
Wiki
Security 1
Insights
Settings
Create DeploySurge.yml
master
a2030fa
Re-run jobs
React App Deploy to Surge
on: push
build
React App Deploy to Surge / build
succeeded now in 1m 1s
Search logs
Set up job 4s
Build borales/actions-yarn@v2.0.0 6s
Run actions/checkout@v2 0s
Install NodeJS 1s
Run borales/actions-yarn@v2.0.0 33s
Build React App 8s
Install Surge 6s
Deploy to Surge 3s
Post Run actions/checkout@v2 0s
Complete job 0s

Part VI - Create Components

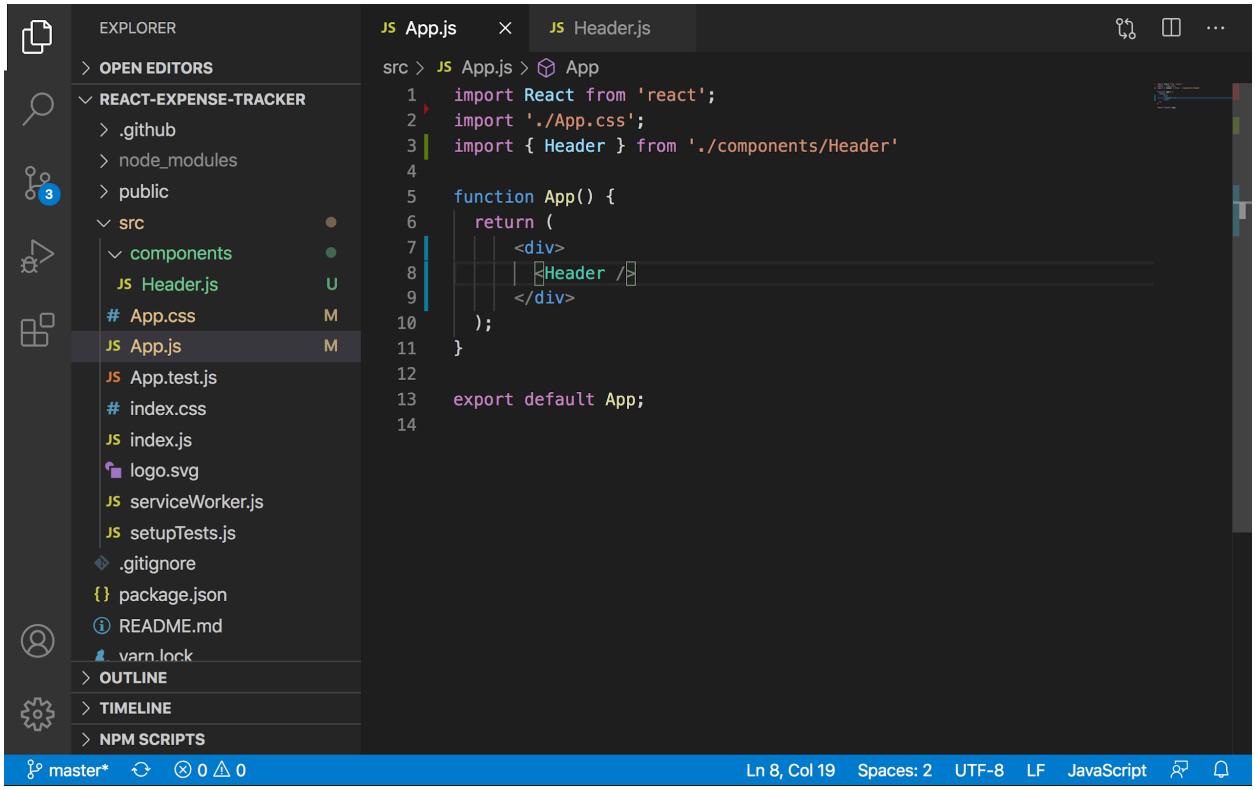
Step 1: Create a components folder in the src folder



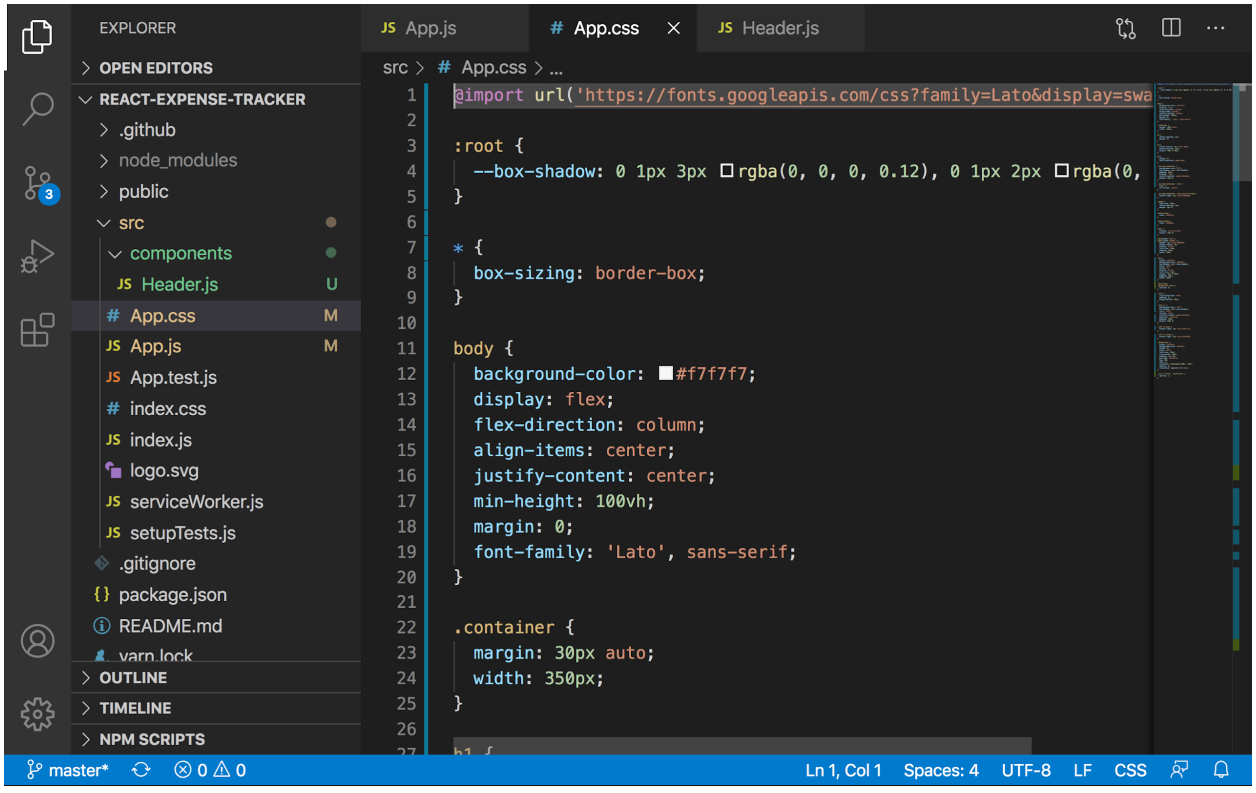
Step 2: Create Header Component



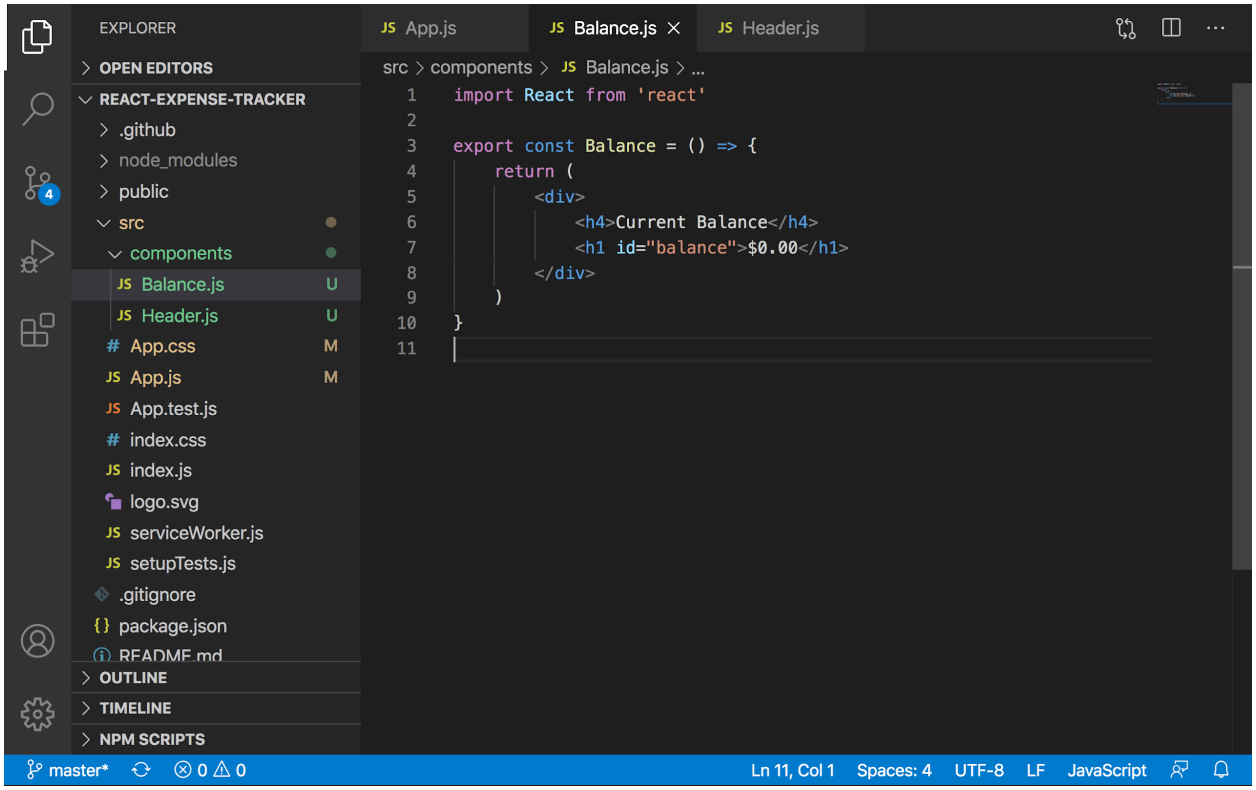
Step 3: Import the Header Component to App.js and add to App function



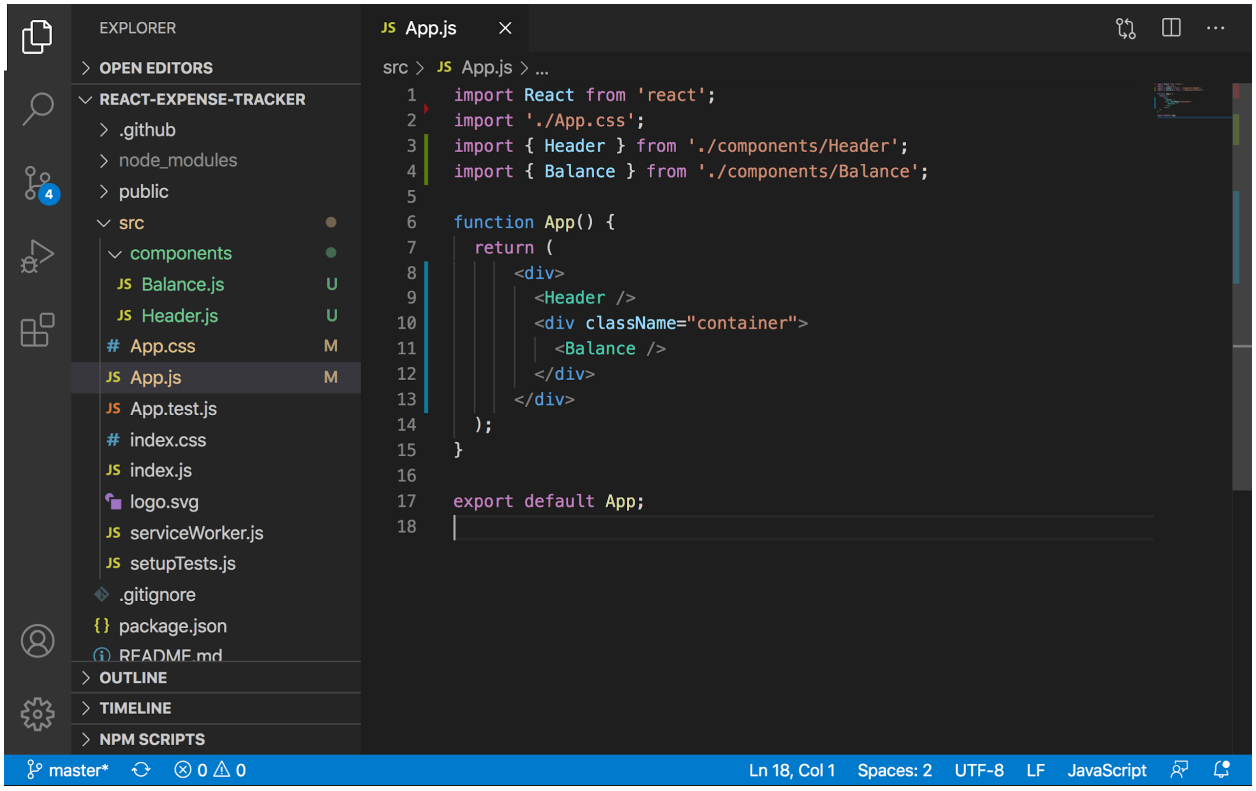
Step 4: Apply CSS to App.js from App.css



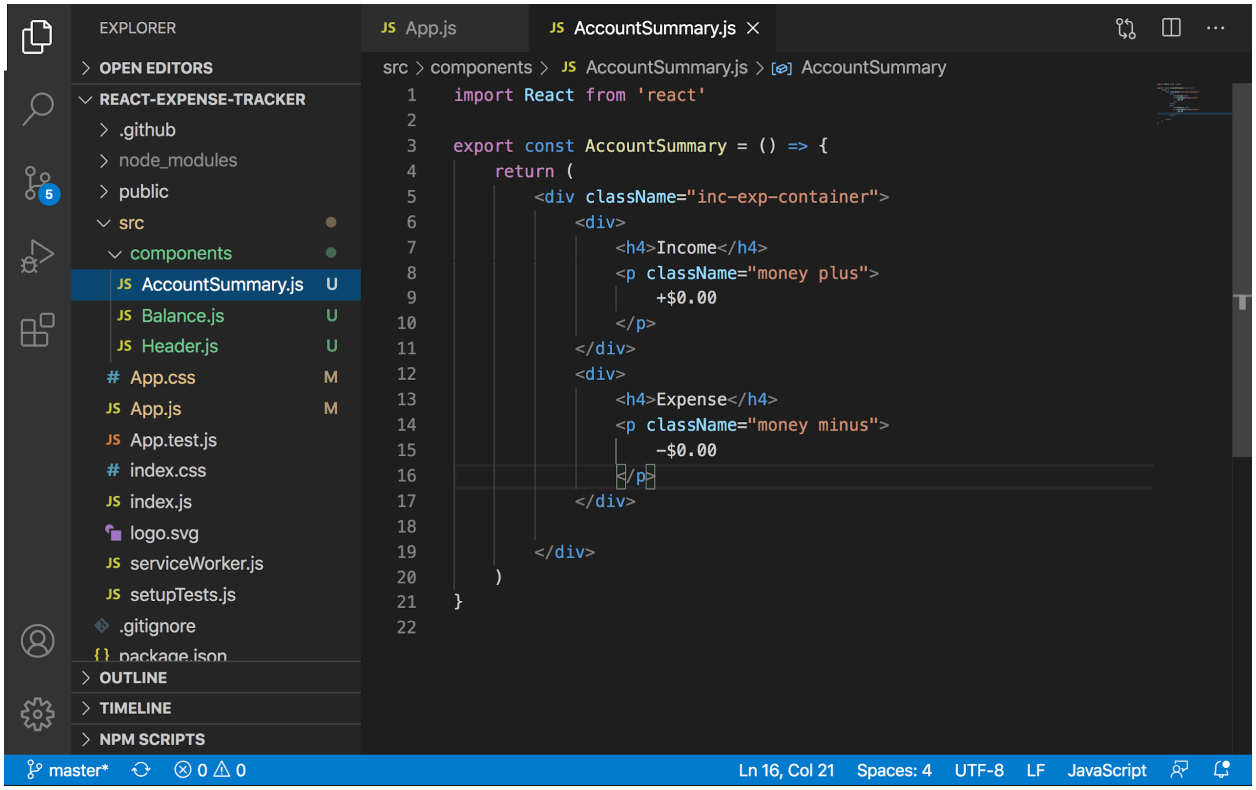
Step 5: Create Balance Component



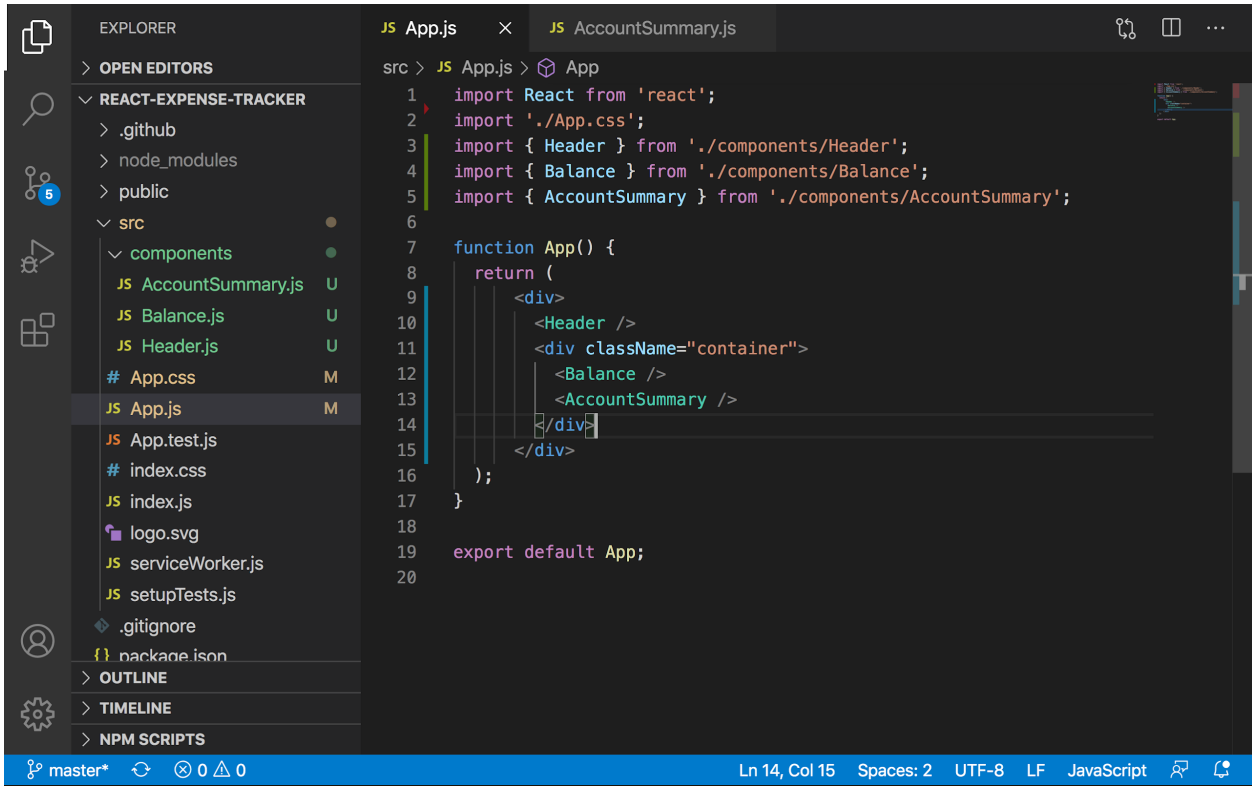
Step 6: Import the Balance Component to App.js and add to App function



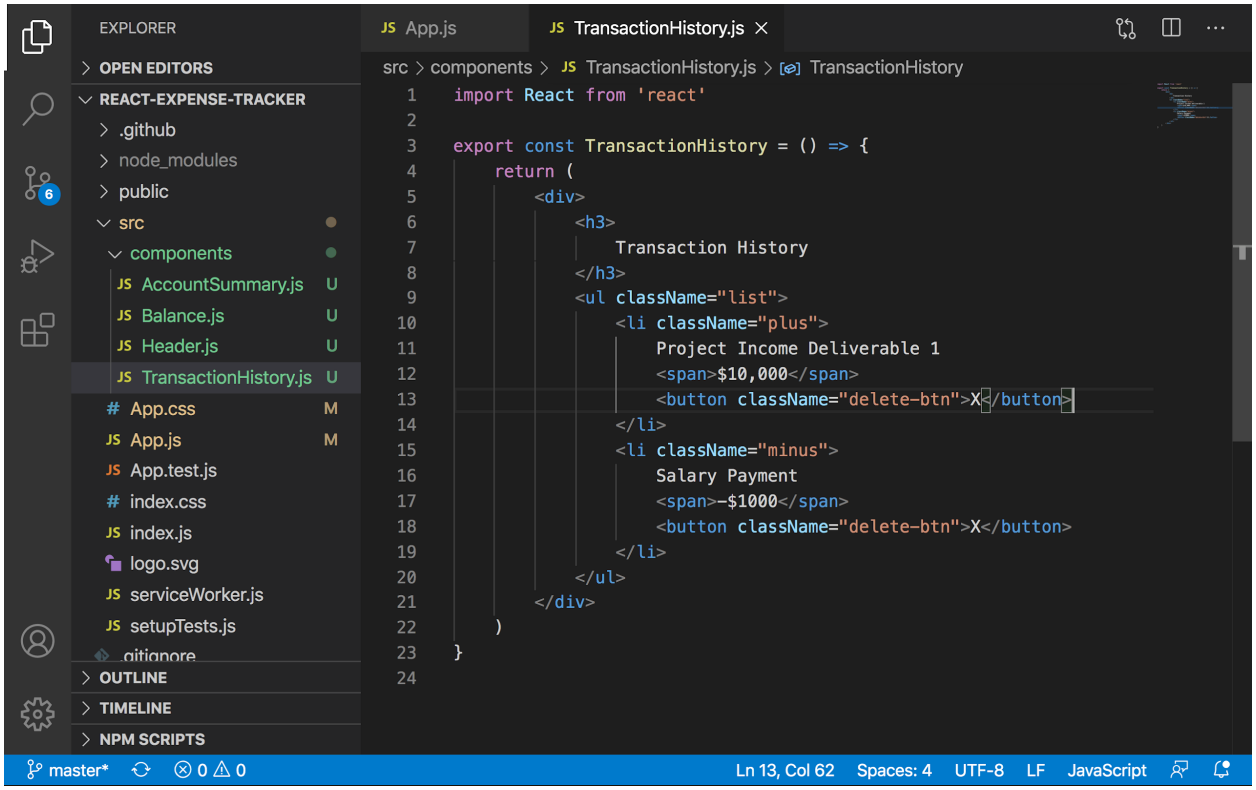
Step 7: Create the AccountSummary Component for Income and Expense Summary



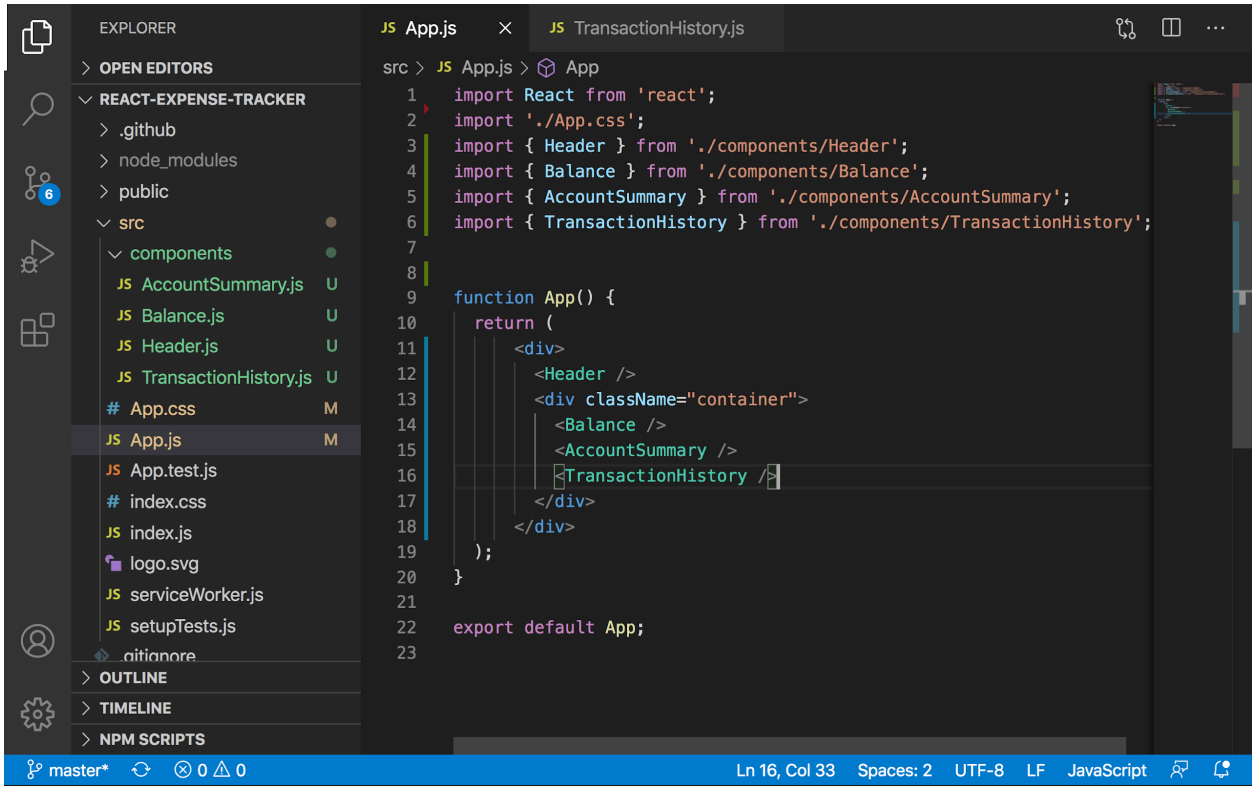
Step 8: Import the AccountSummary Component to App.js and add to App function



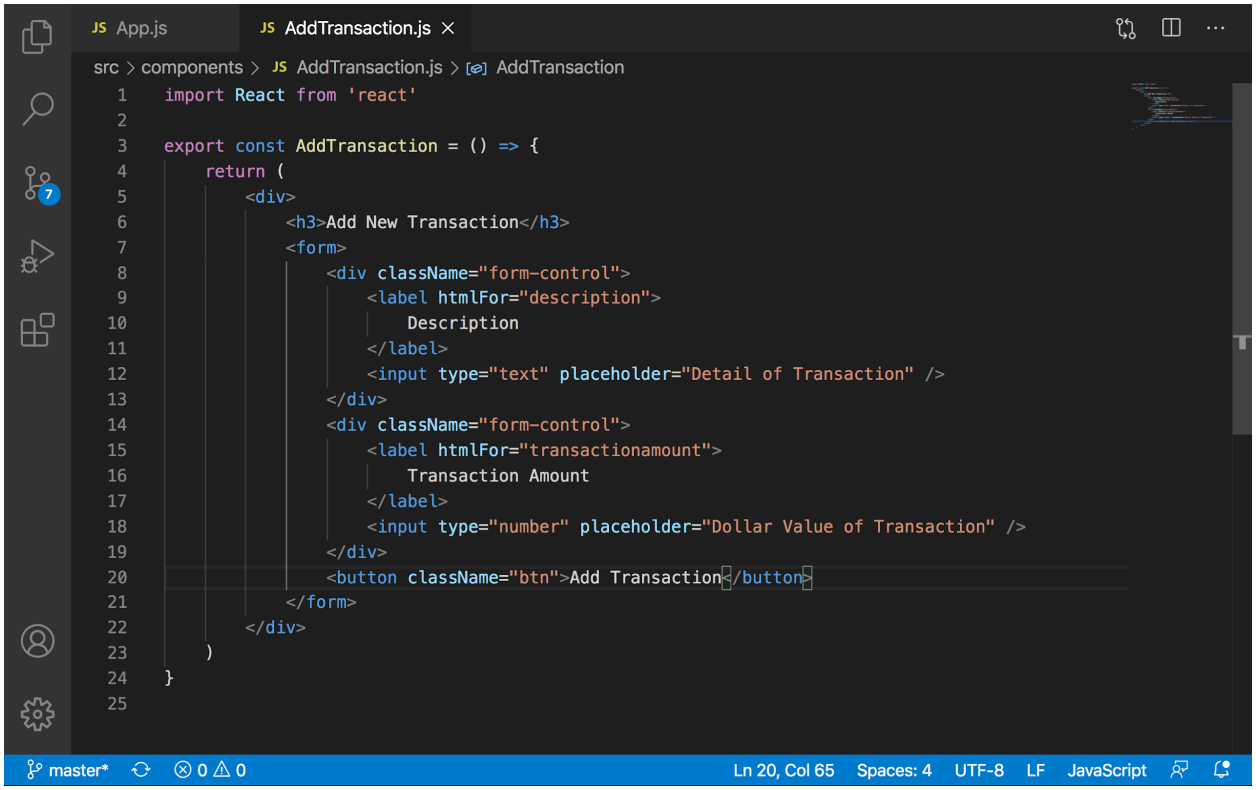
Step 9: Create the TransactionHistory Component for list of transactions



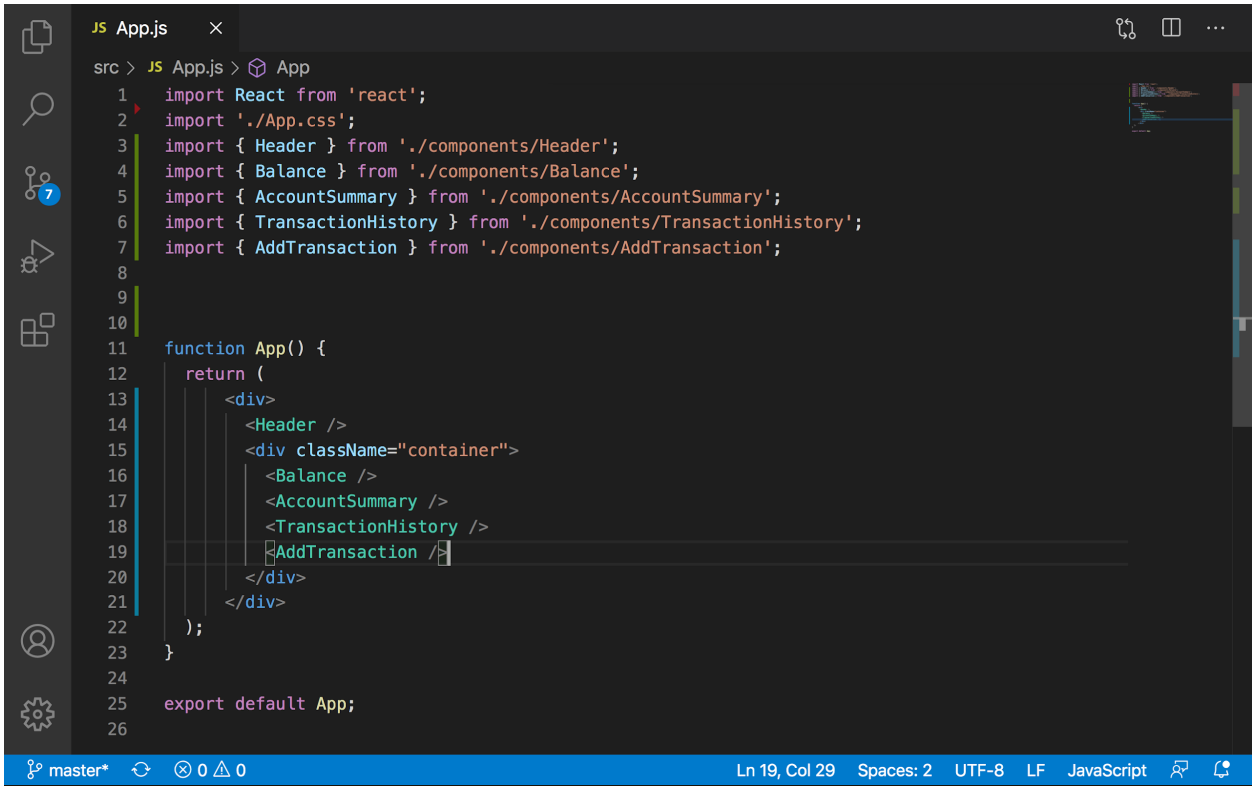
Step 10: Import the TransactionHistory Component to App.js and add to App function



Step 11: Create the AddTransaction Component for the form to add new transactions

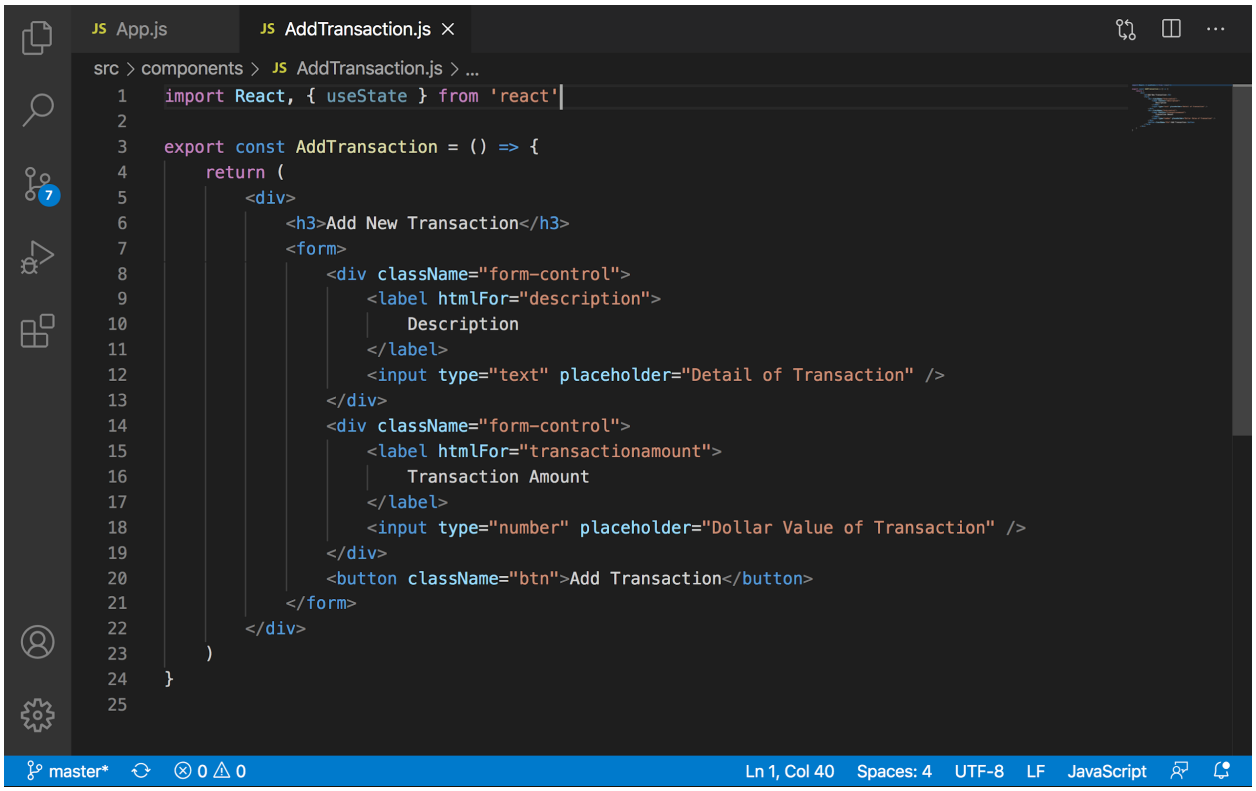


Step 12: Import the AddTransaction Component to App.js and add to App function

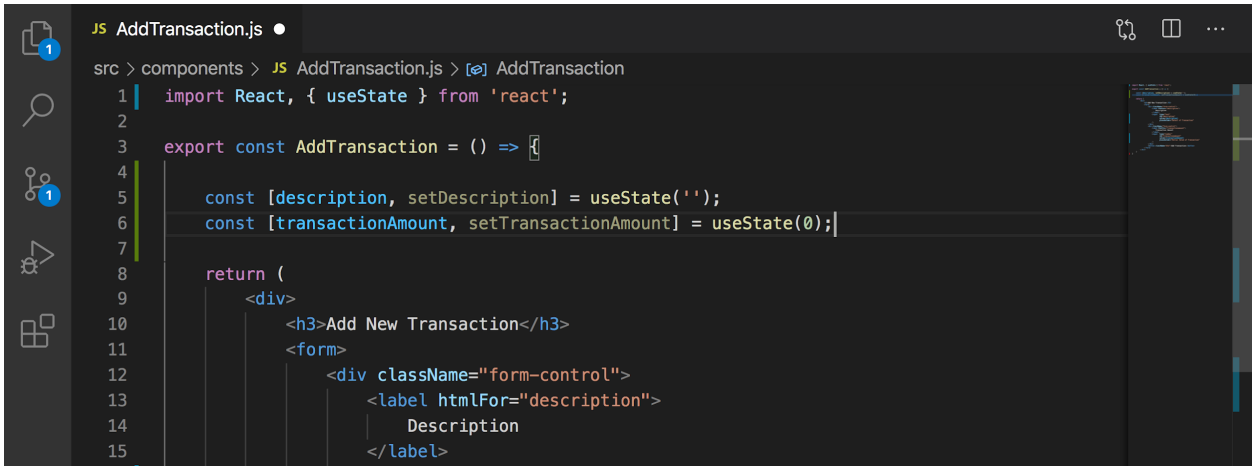


Part VII - Create Hook for Updating State with Add Transaction

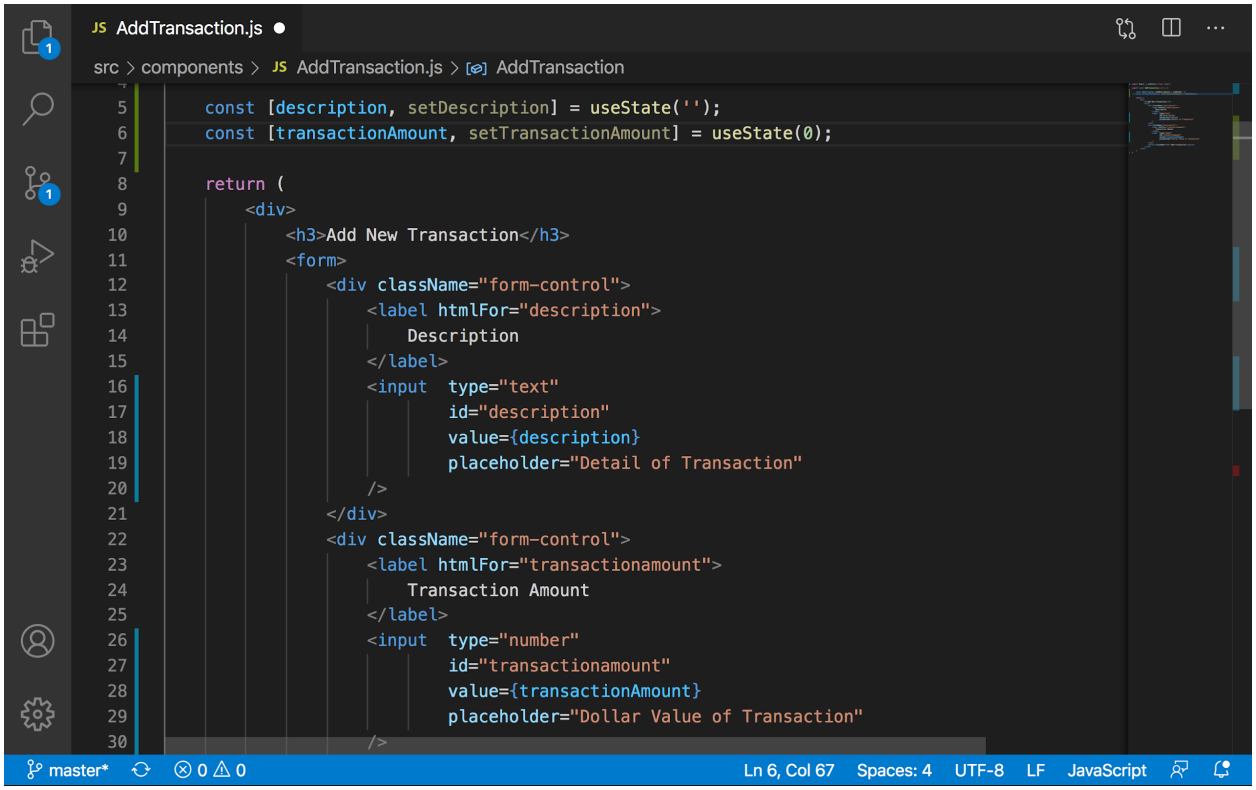
Step 1: Import useState



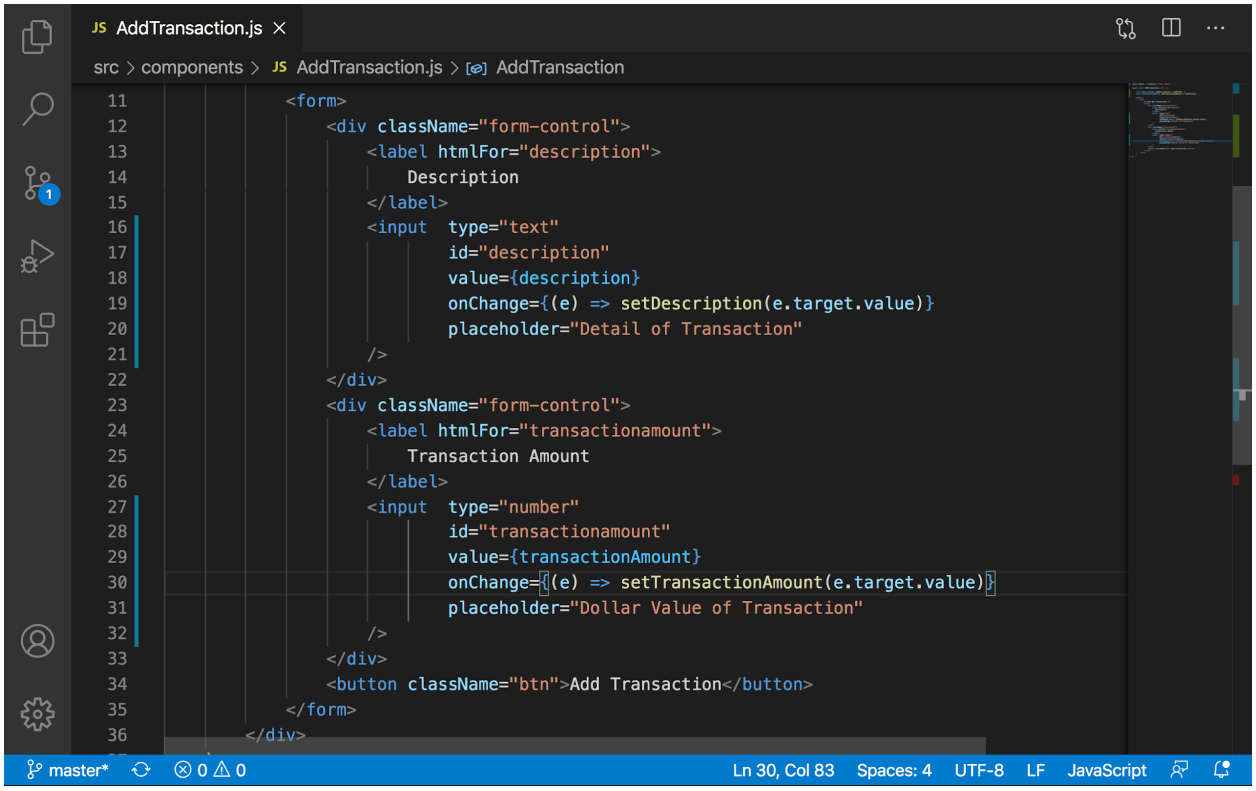
Step 2: Create state for description and transactionAmount



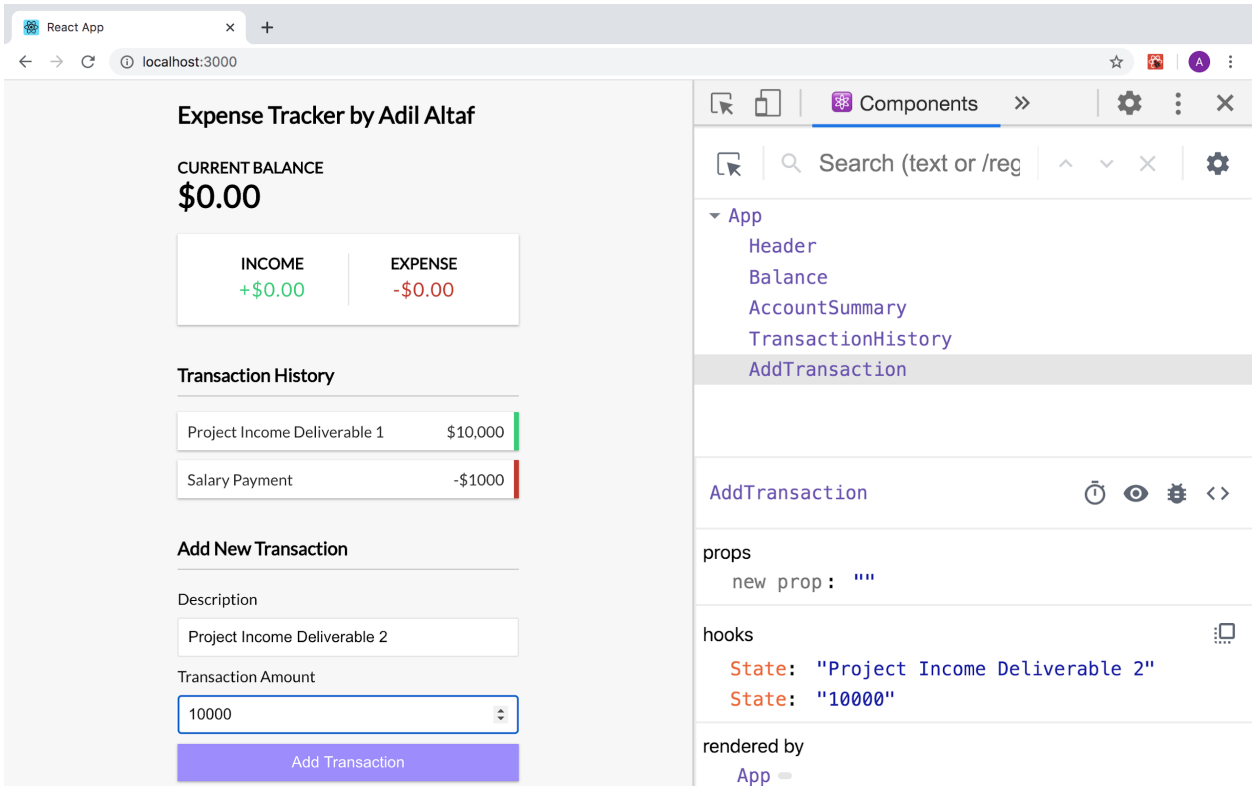
Step 3: Link the state to the form with the value attribute



Step 4: Update the state with onChange event

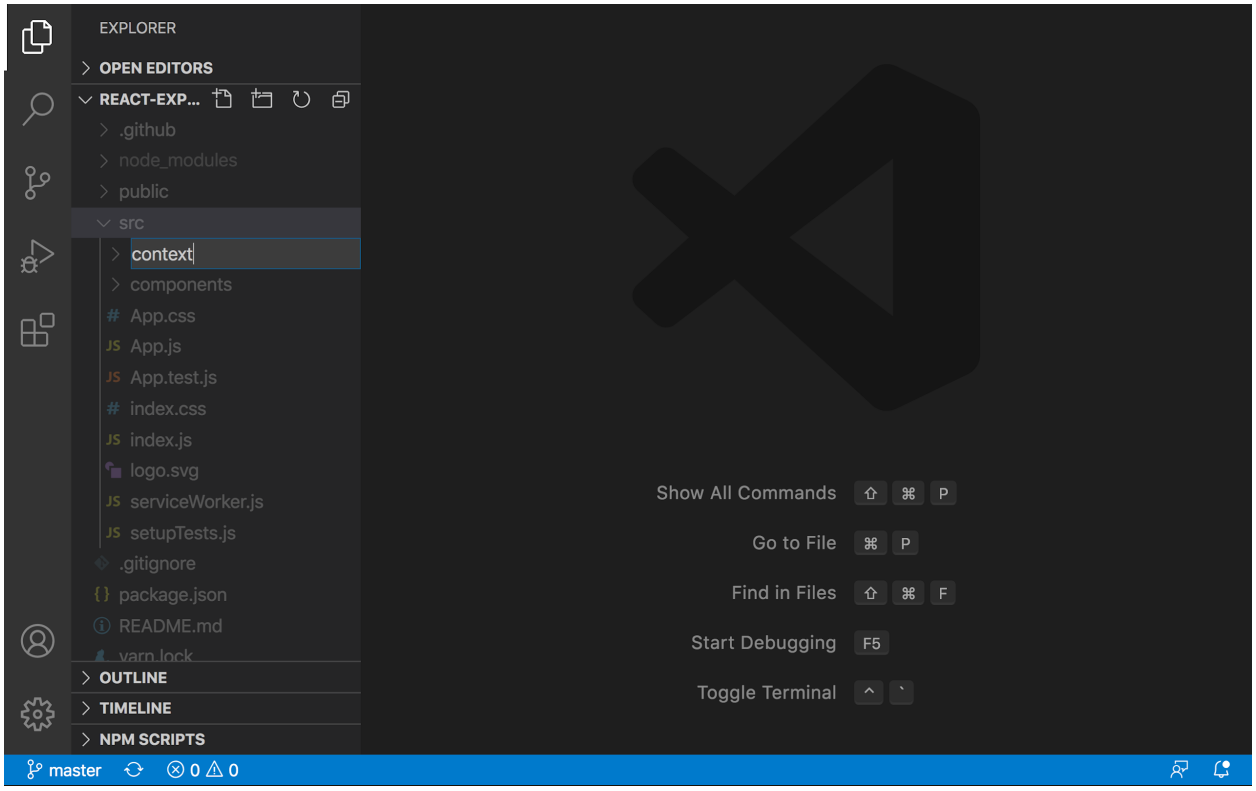


Step 5: Check the React Developer Tools in Chrome to see Components and check State

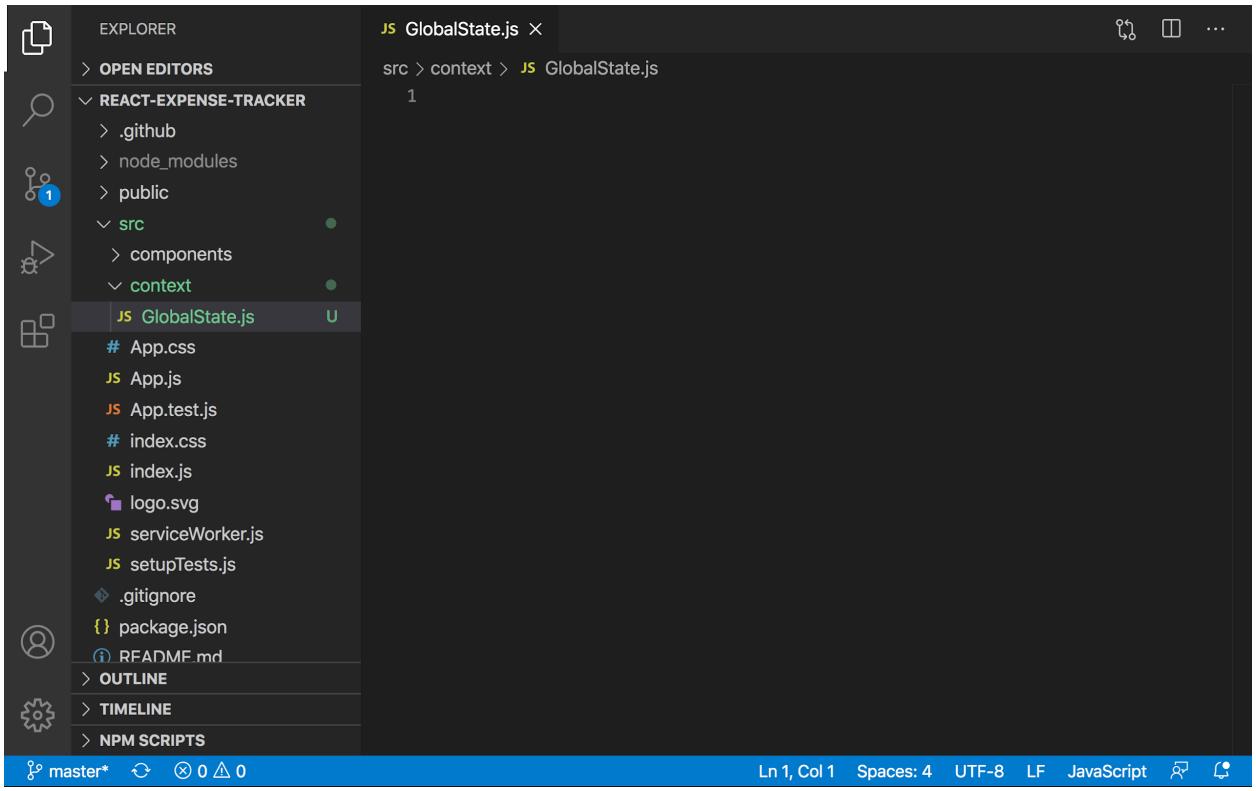


Part VIII - Create Context and Reducer

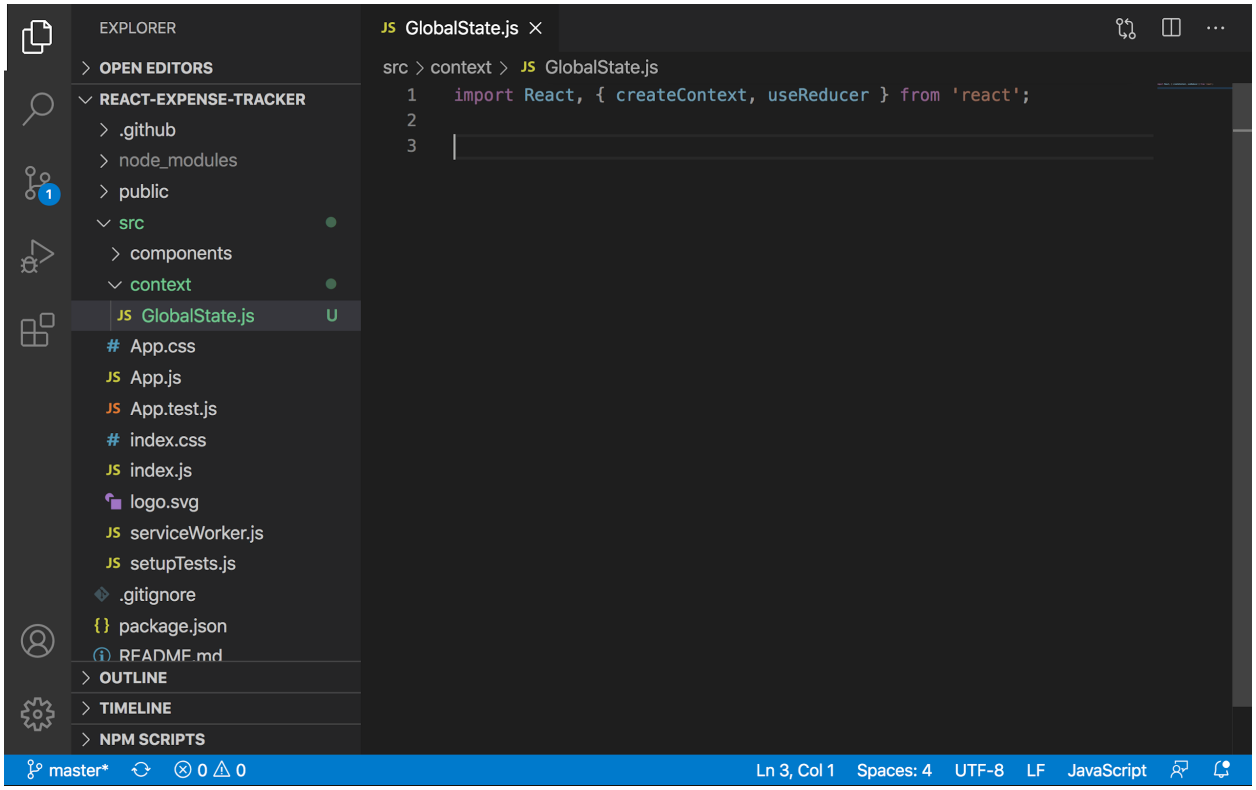
Step 1: Create a context folder in the src folder



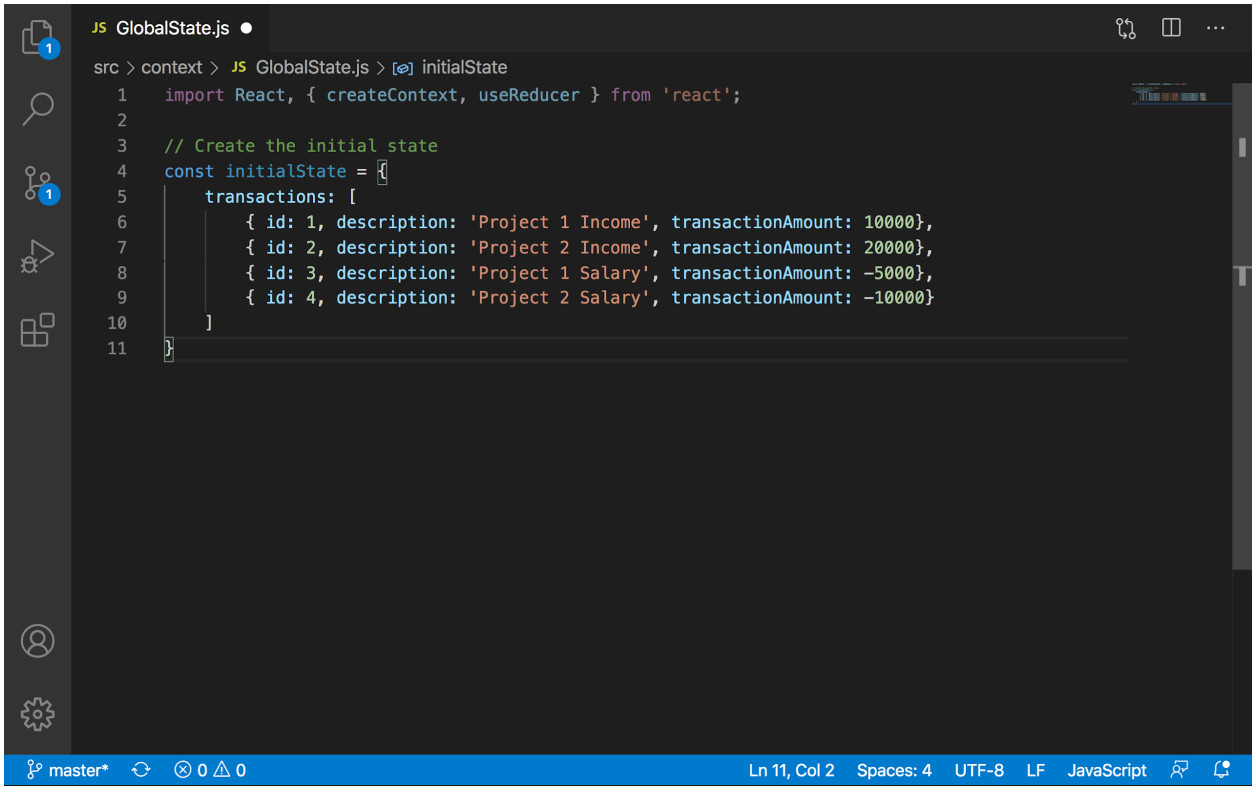
Step 2: Create GlobalState.js file for the context



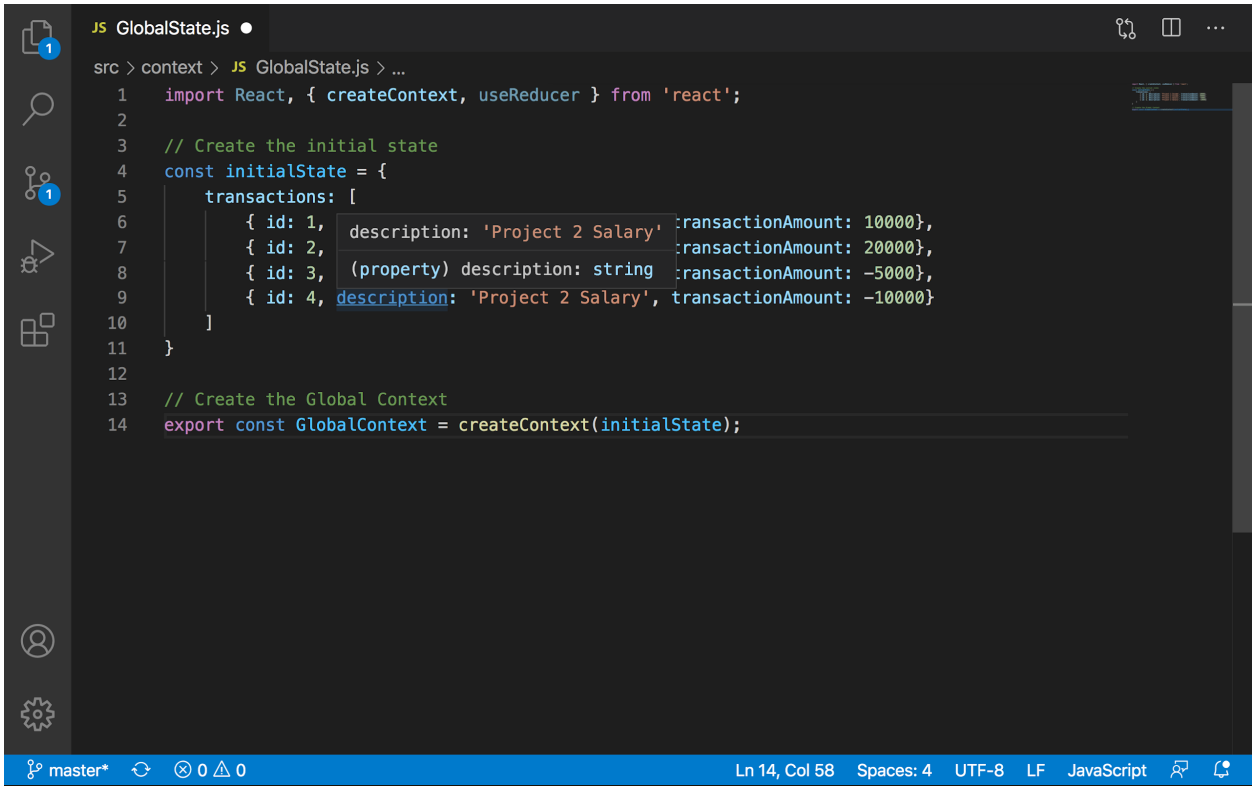
Step 3: Import React, { createContext, useReducer } from ‘react’



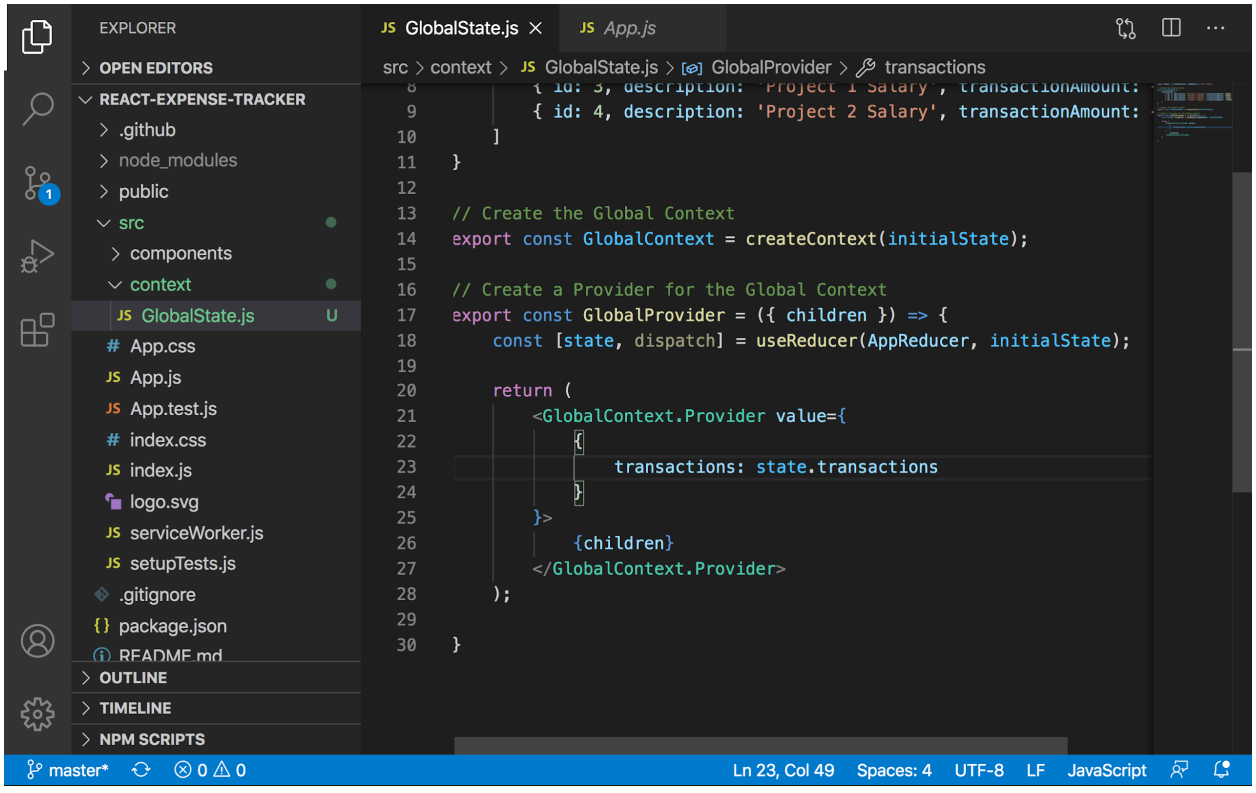
Step 4: Create the Initial State with placeholder transactions



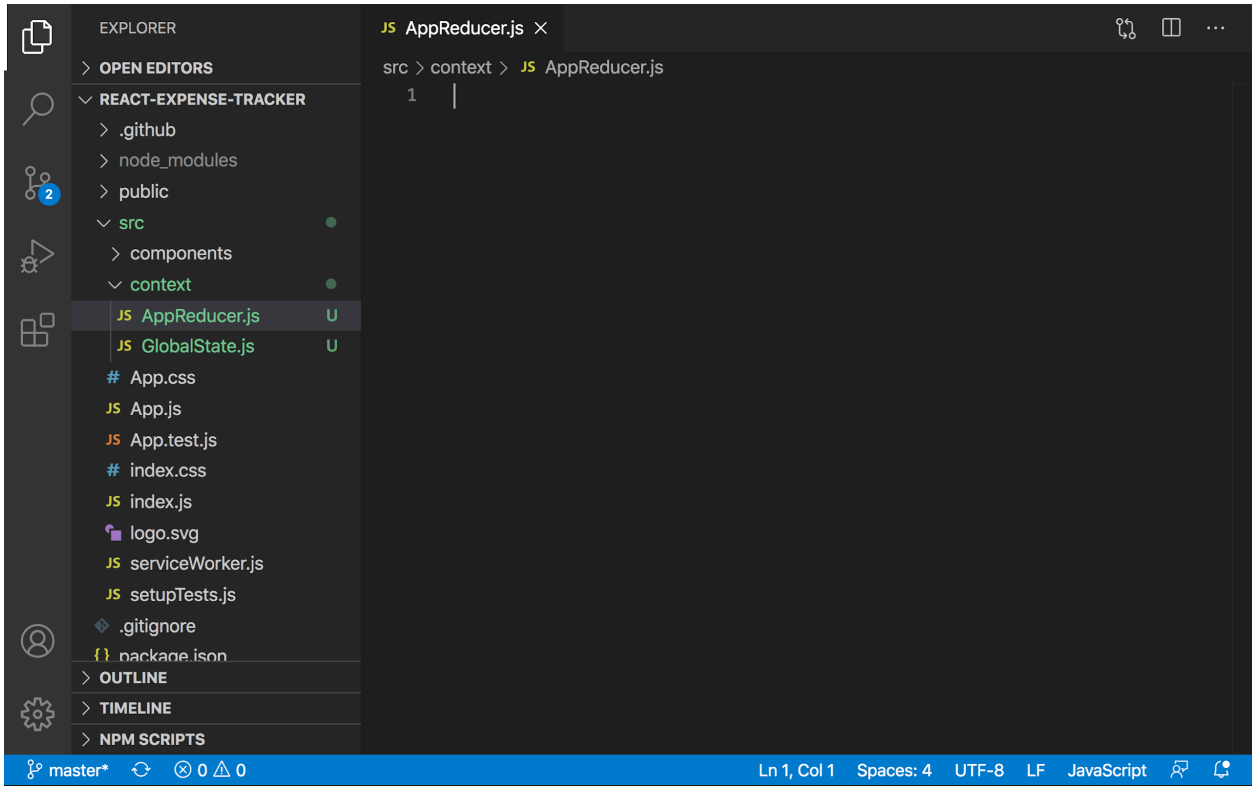
Step 5: Create the global context with the initialState as the default value:



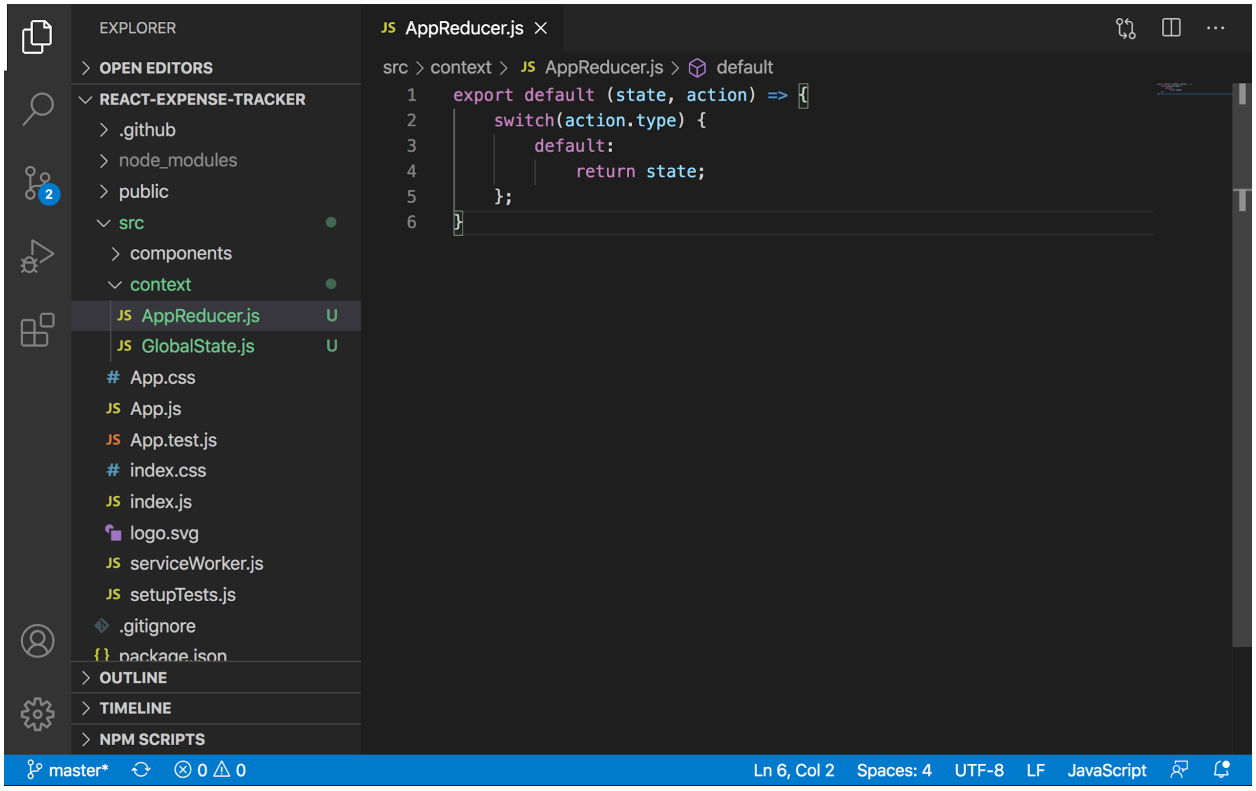
Step 6: Create the provider for the global context



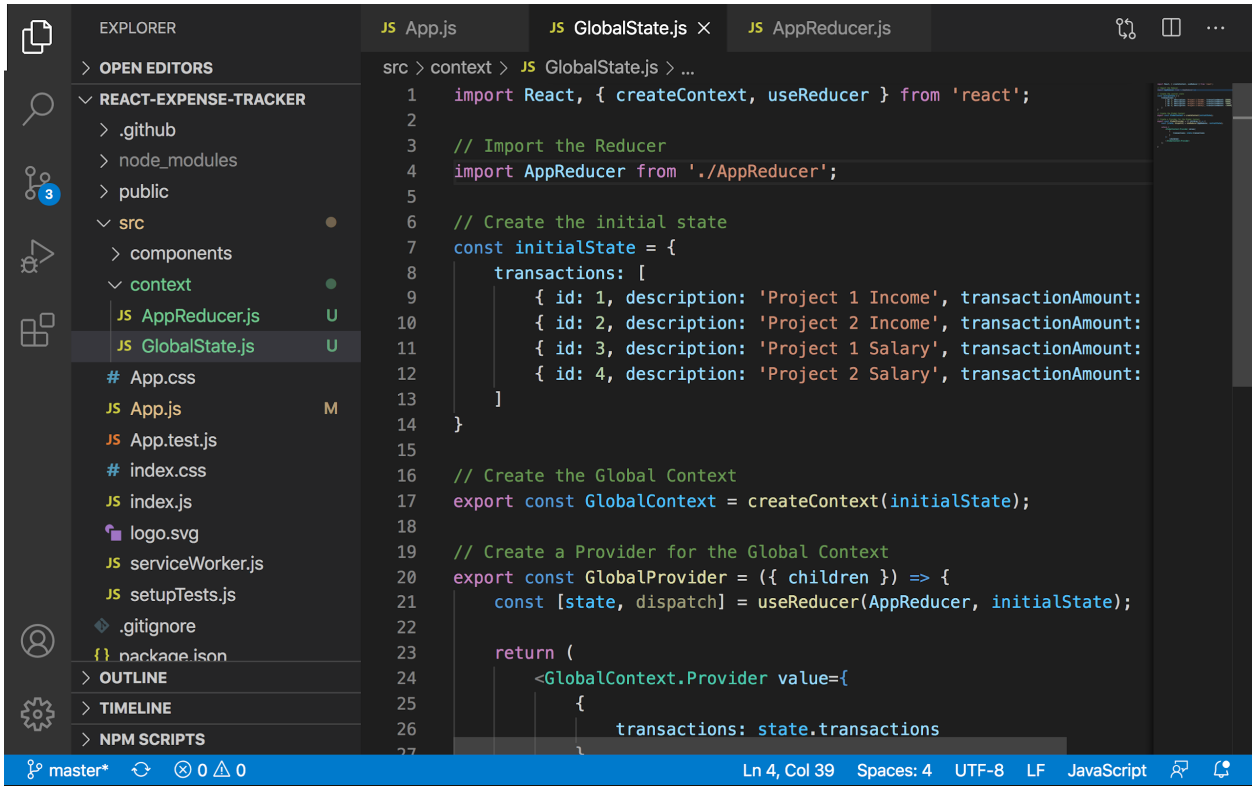
Step 7: Create the AppReducer.js file in the context folder



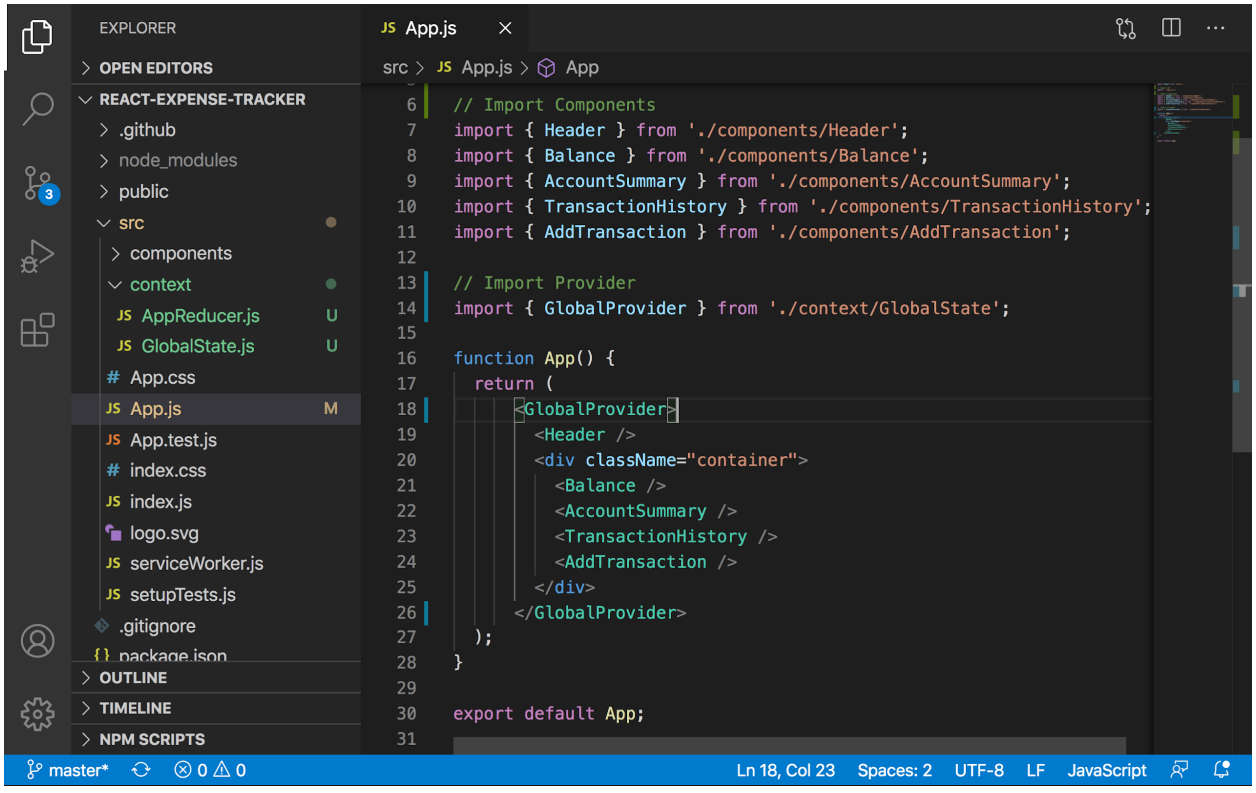
Step 8: Create the basic reducer to use action.type to return a the state if no action is performed



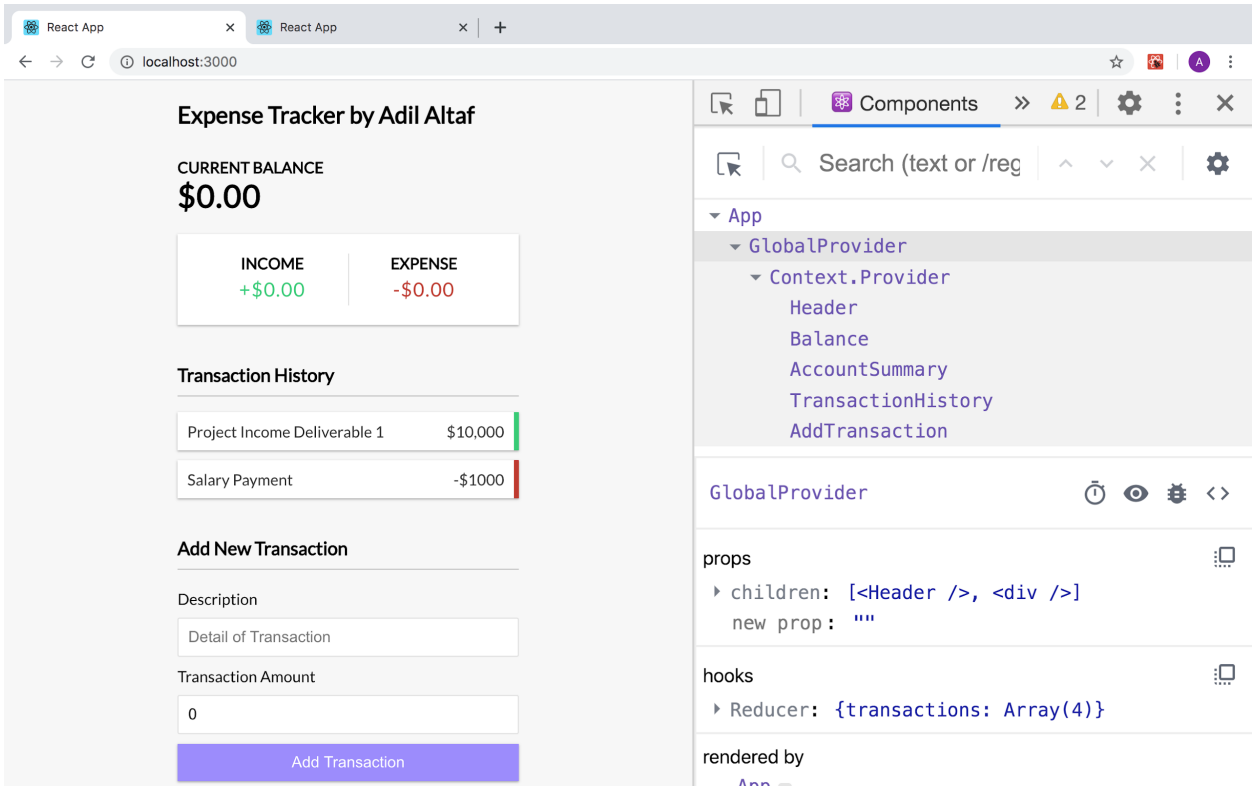
Step 9: Import the AppReducer into the GlobalState.js file



Step 10: Import the GlobalProvider to the App.js and wrap components with the GlobalProvider

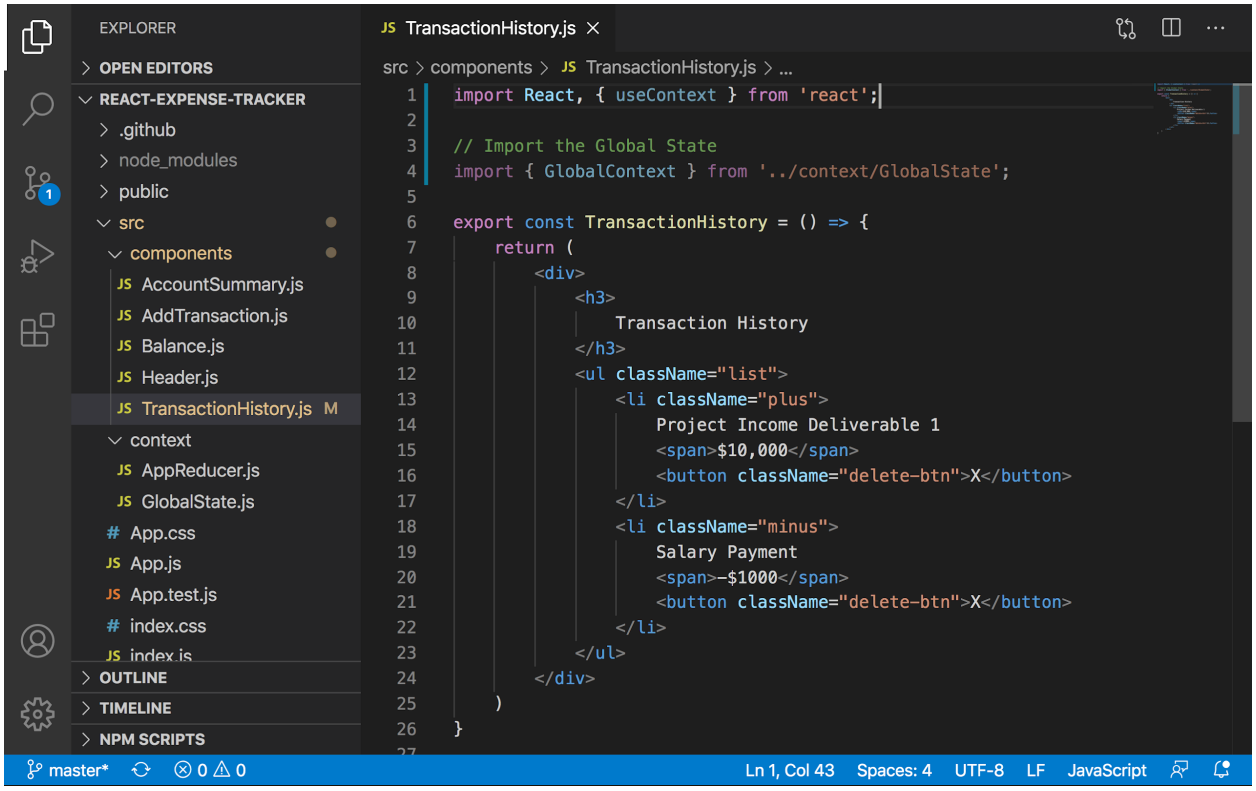


Step 11: Check the React Components in the Browser to see if reducer is present in the hooks and you can see the transactions from the initialState

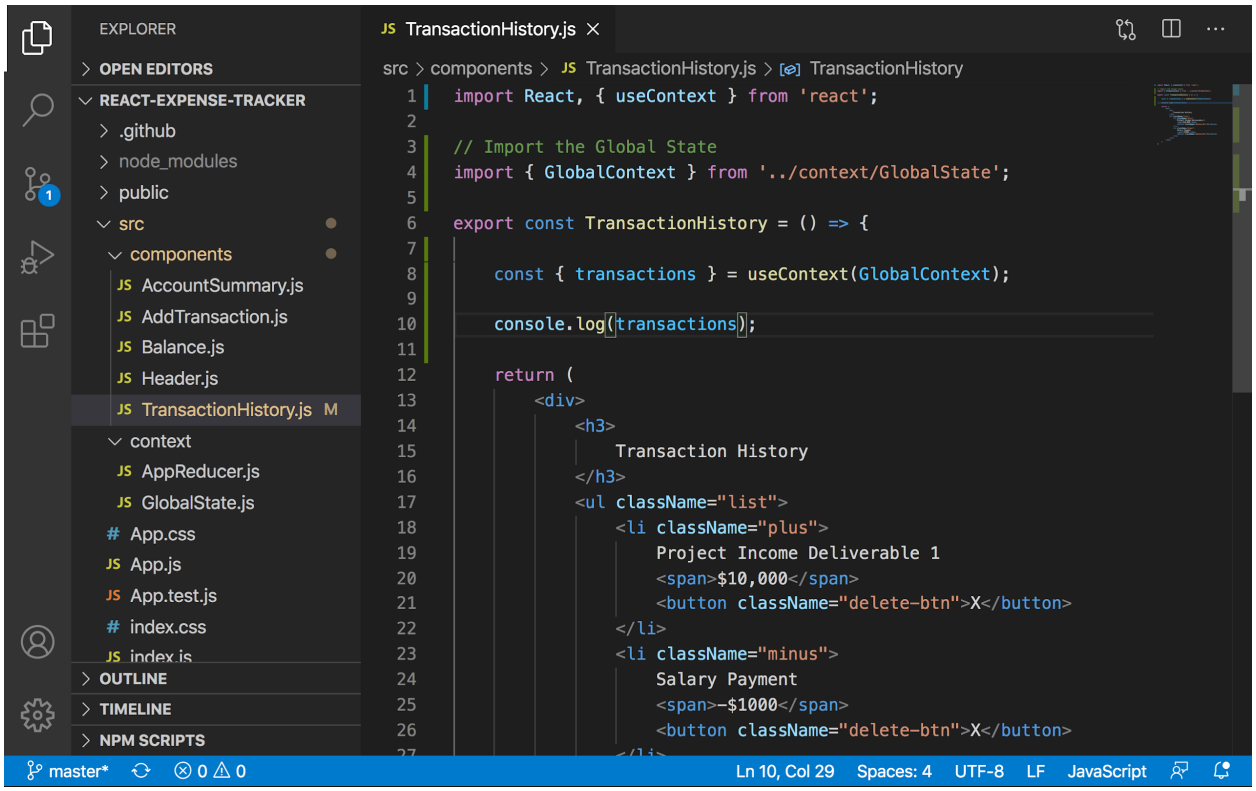


Part IX - Show Transactions from State the Transaction History

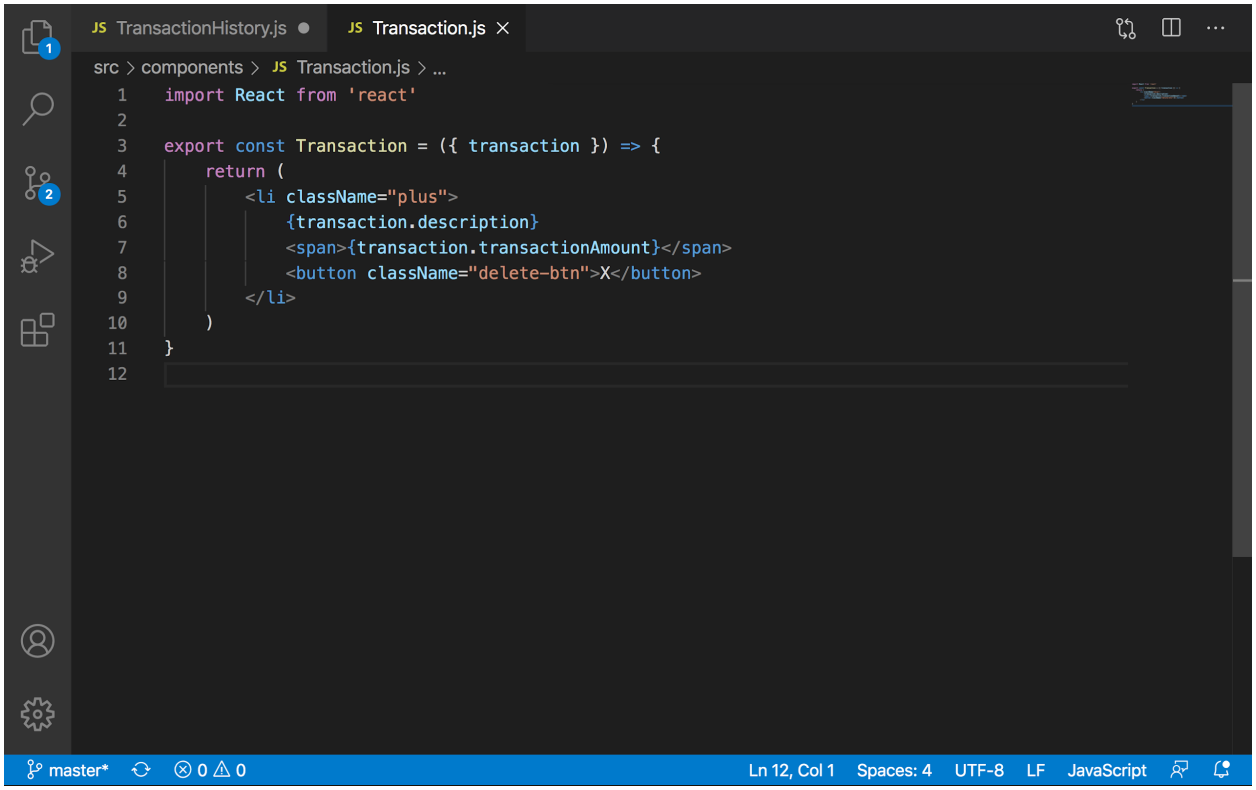
Step 1: Import the Global State from the GlobalContext file and the useContext hook so we can get the data from the context API



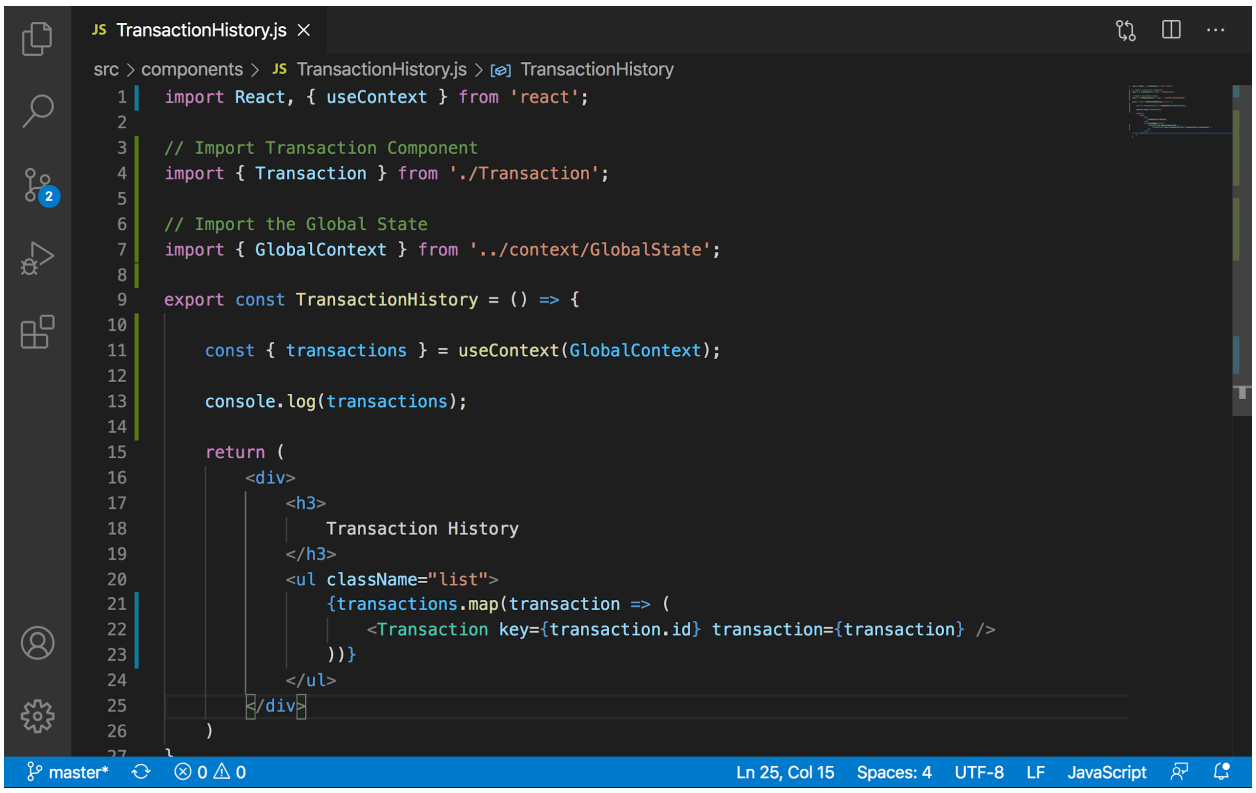
Step 2: Get access to transactions in the state using the GlobalContext



Step 3: Create the Transaction Component

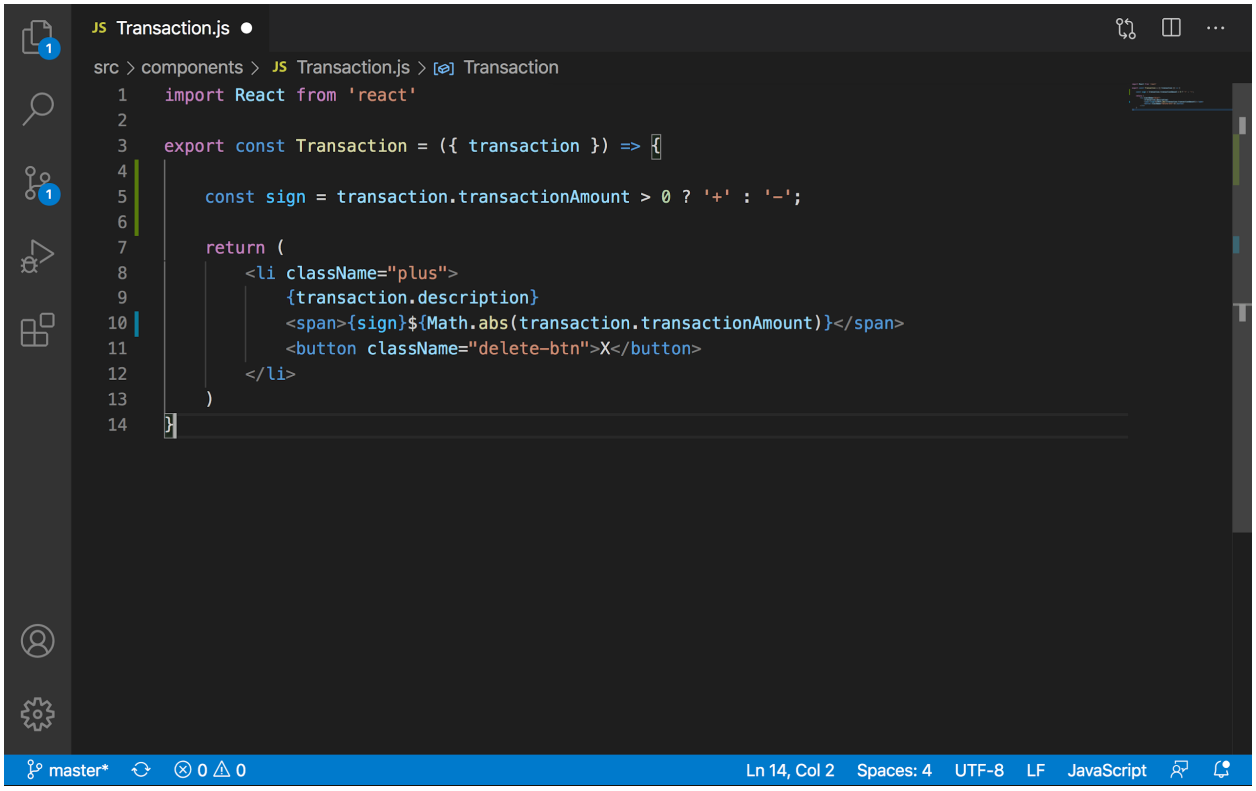


Step 4: Import the Transaction component and use the map function to iterate over the transactions and display them

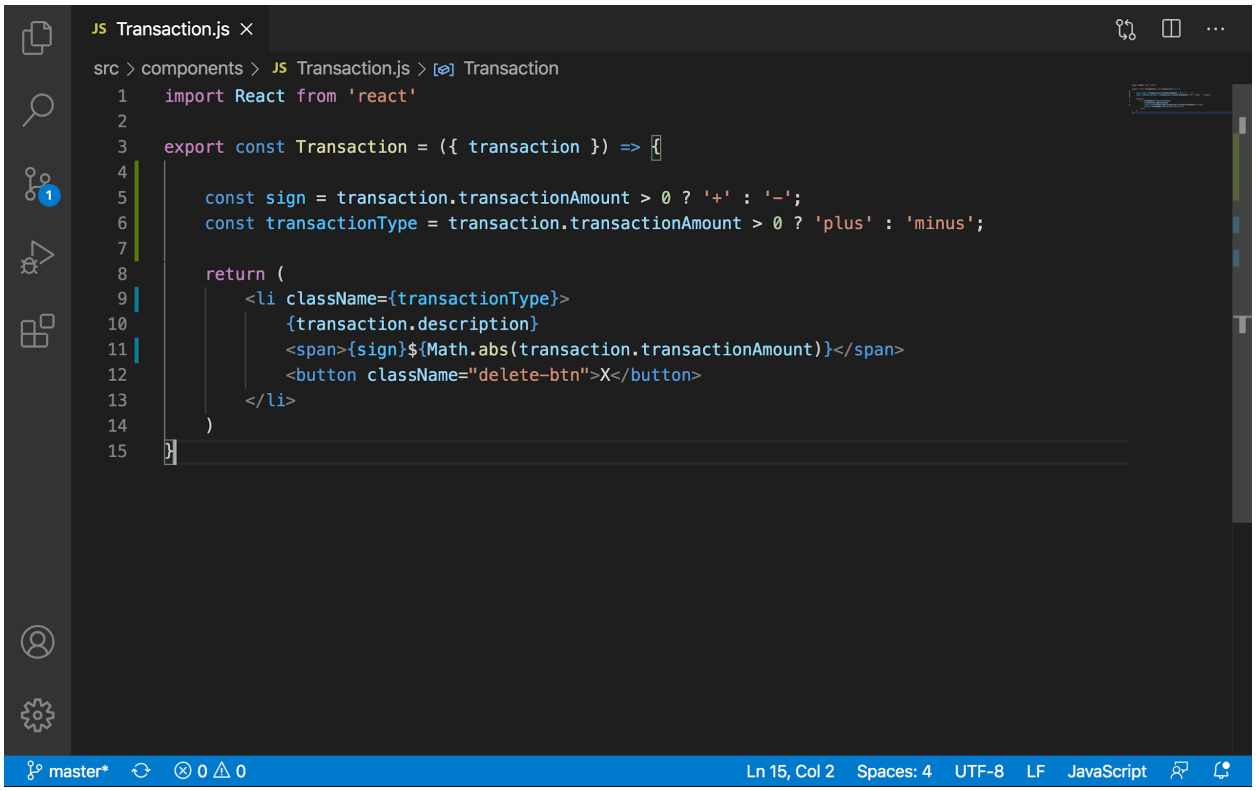


Part X - Differentiating between Income and Expense

Step 1: Using a ternary operator determine if the transaction amount is positive or negative and use the type to significy positive or negative transaction amount in the display

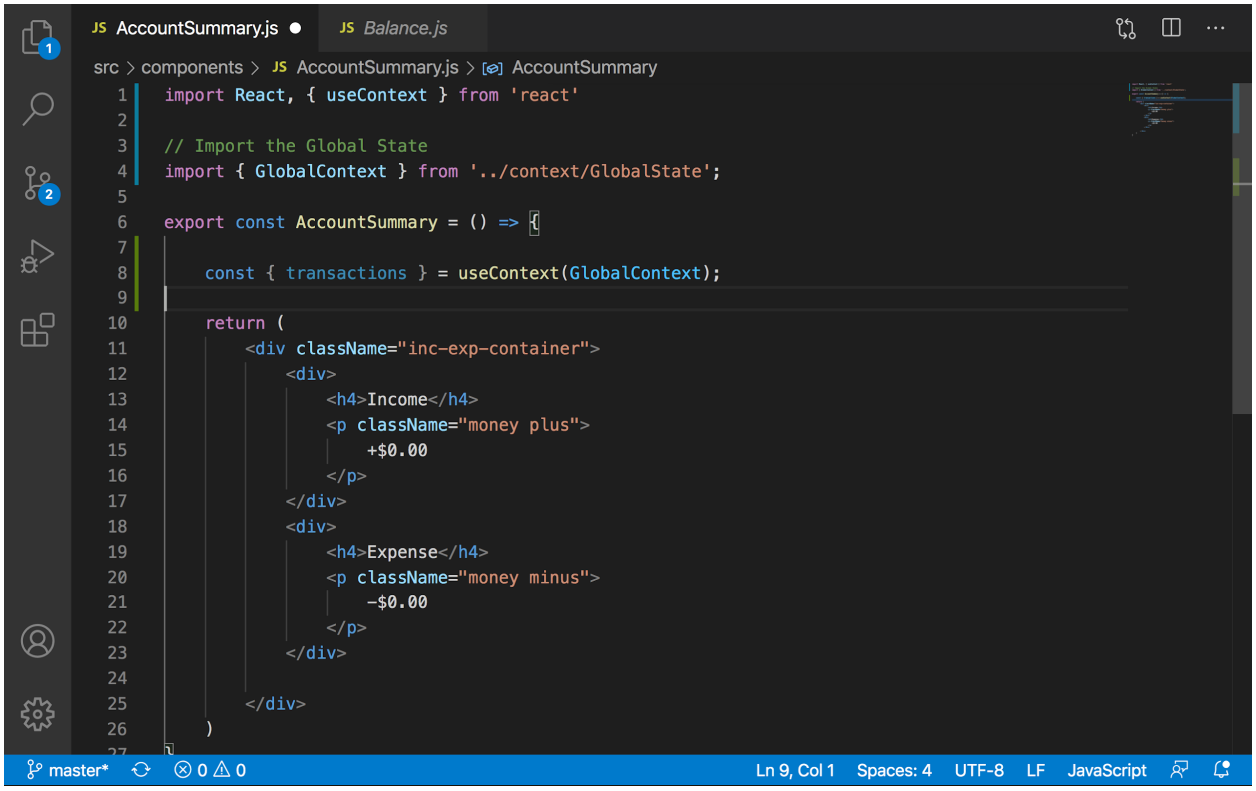


Step 2: Dynamically assign class to differentiate between income and expense

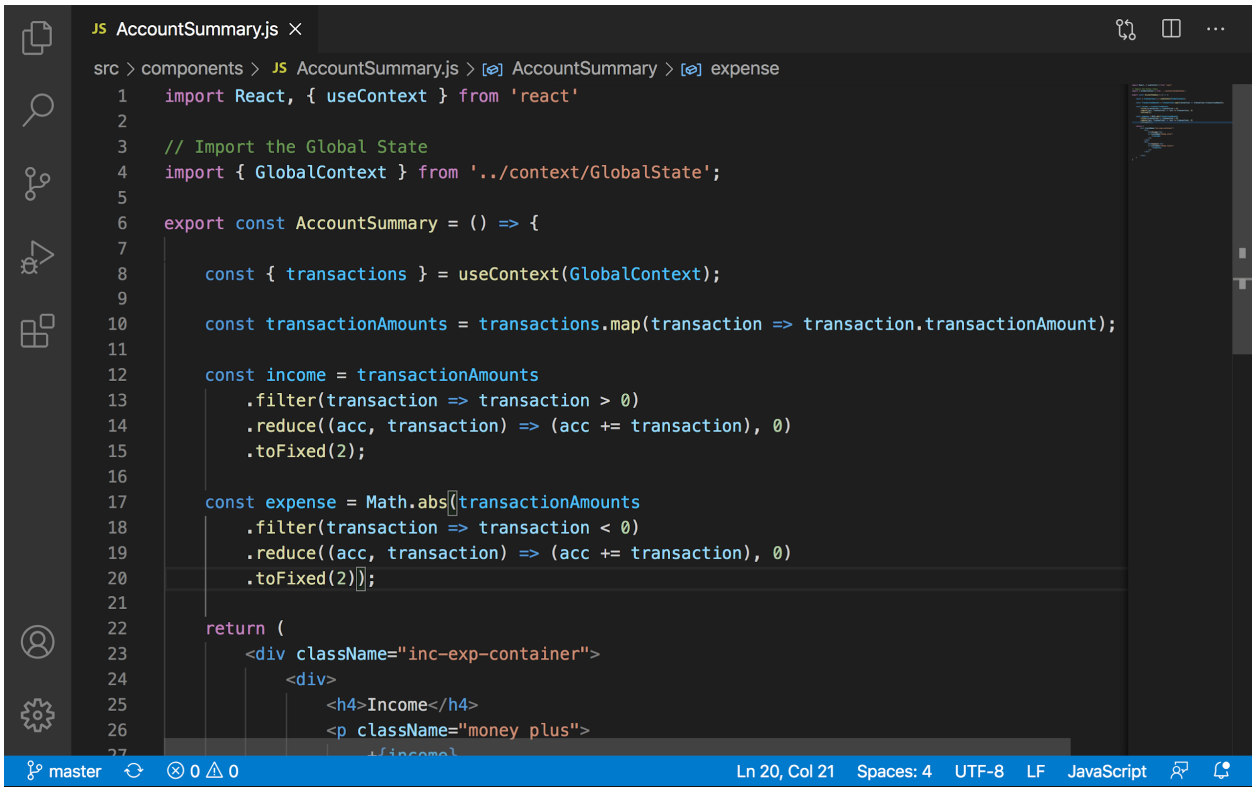


Part X - Updating the Account Summary

Step 1: Import the useContext, GlobalContext and get the transactions

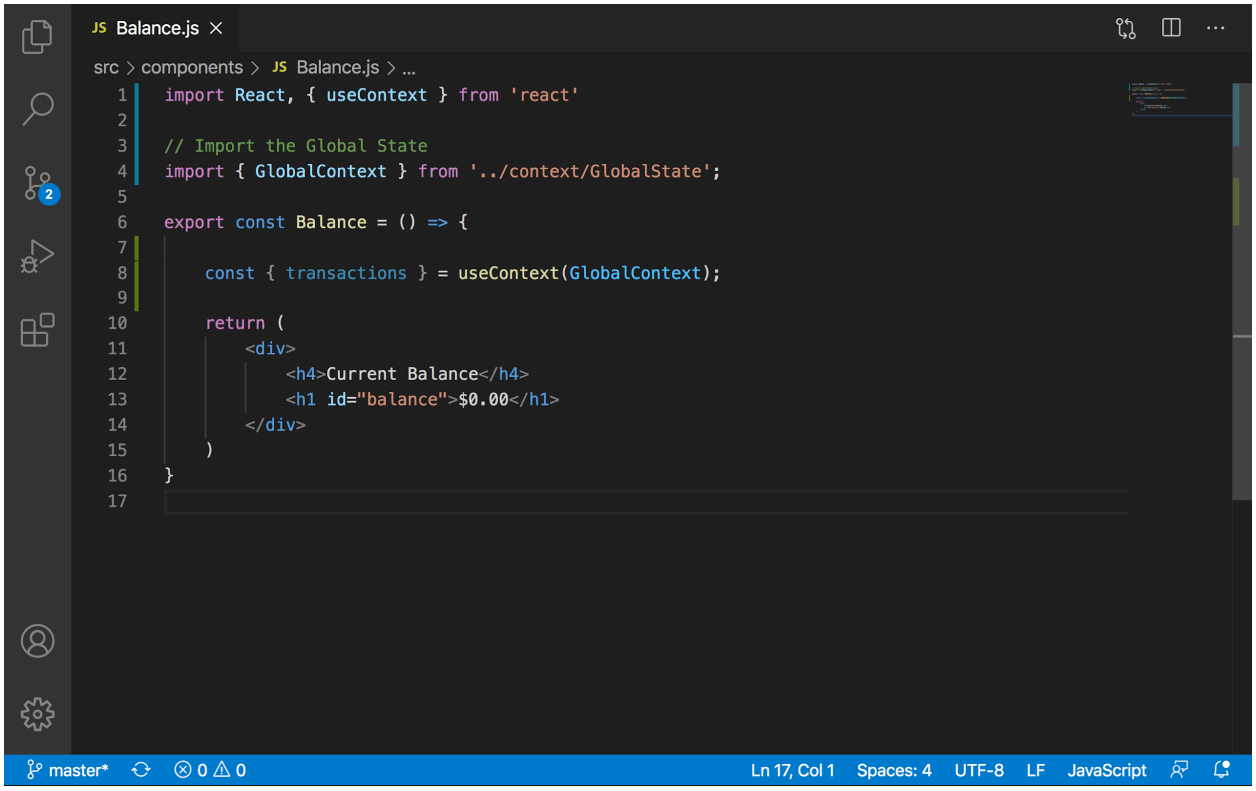


Step 2: Calculate the income and expense from the transactions array

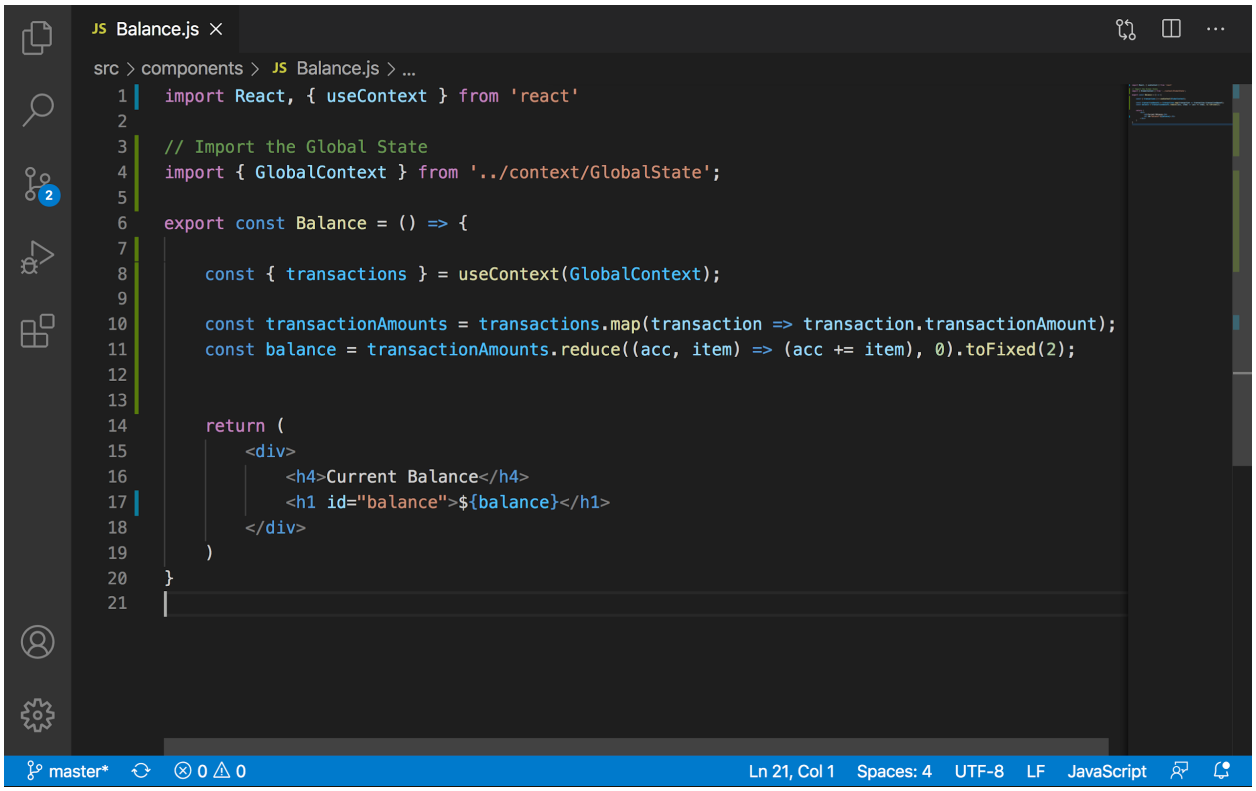


Part XI - Updating the Balance

Step 1: Import the useContext, GlobalContext and get the transactions

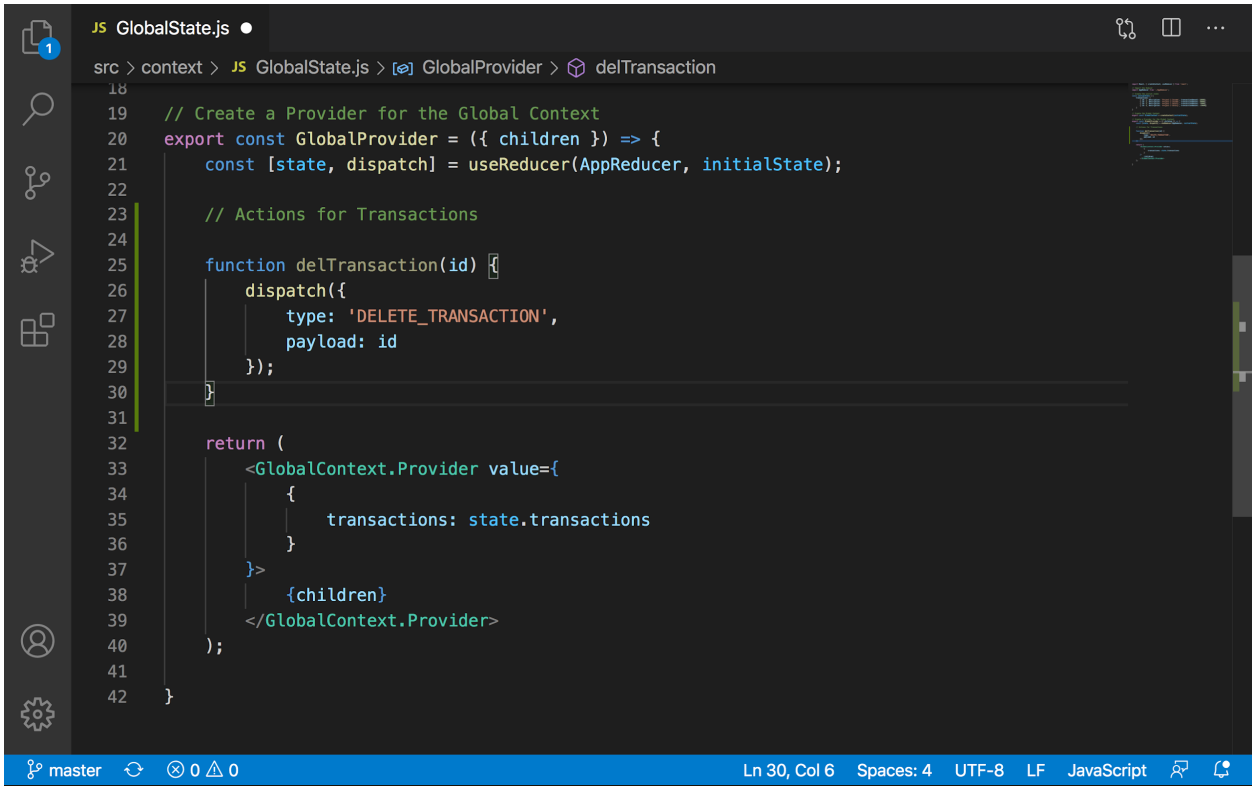


Step 2: Add the transaction amounts to get total balance using the reduce method

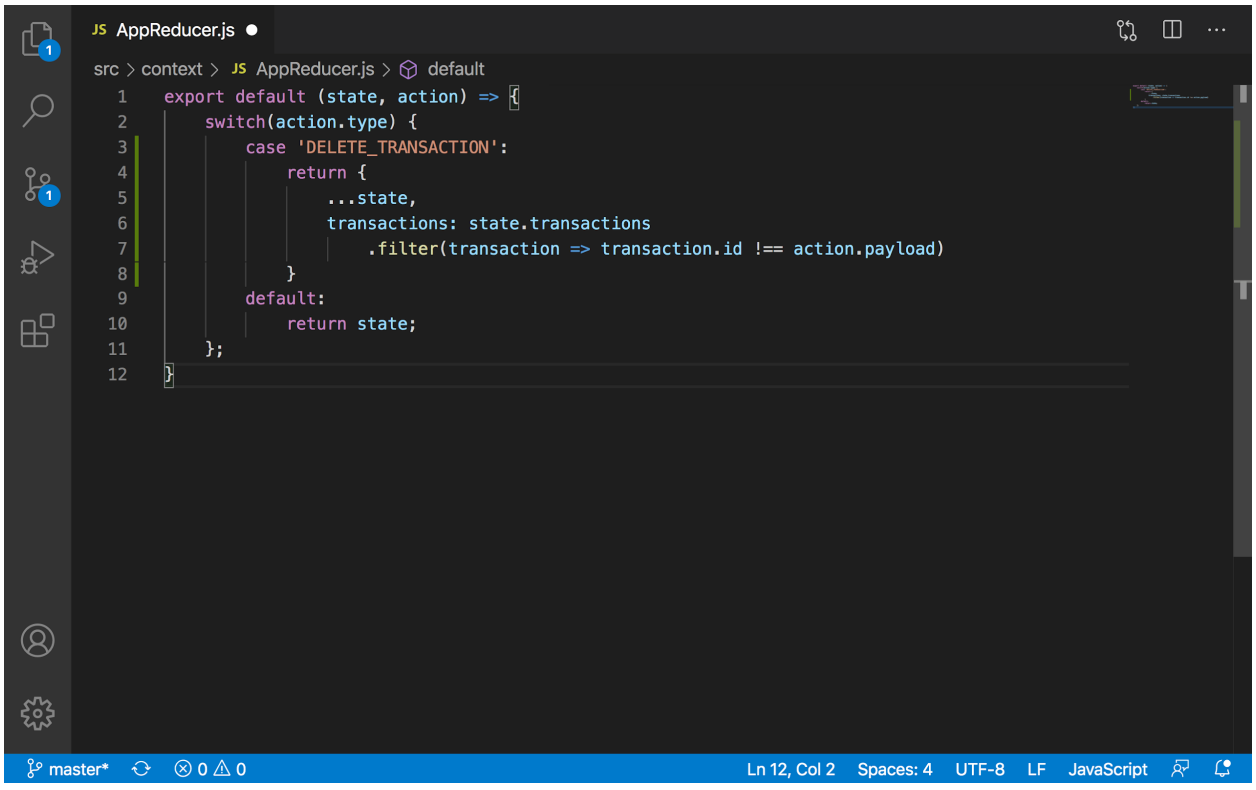


Part XII - Add an Action to Delete Existing Transactions

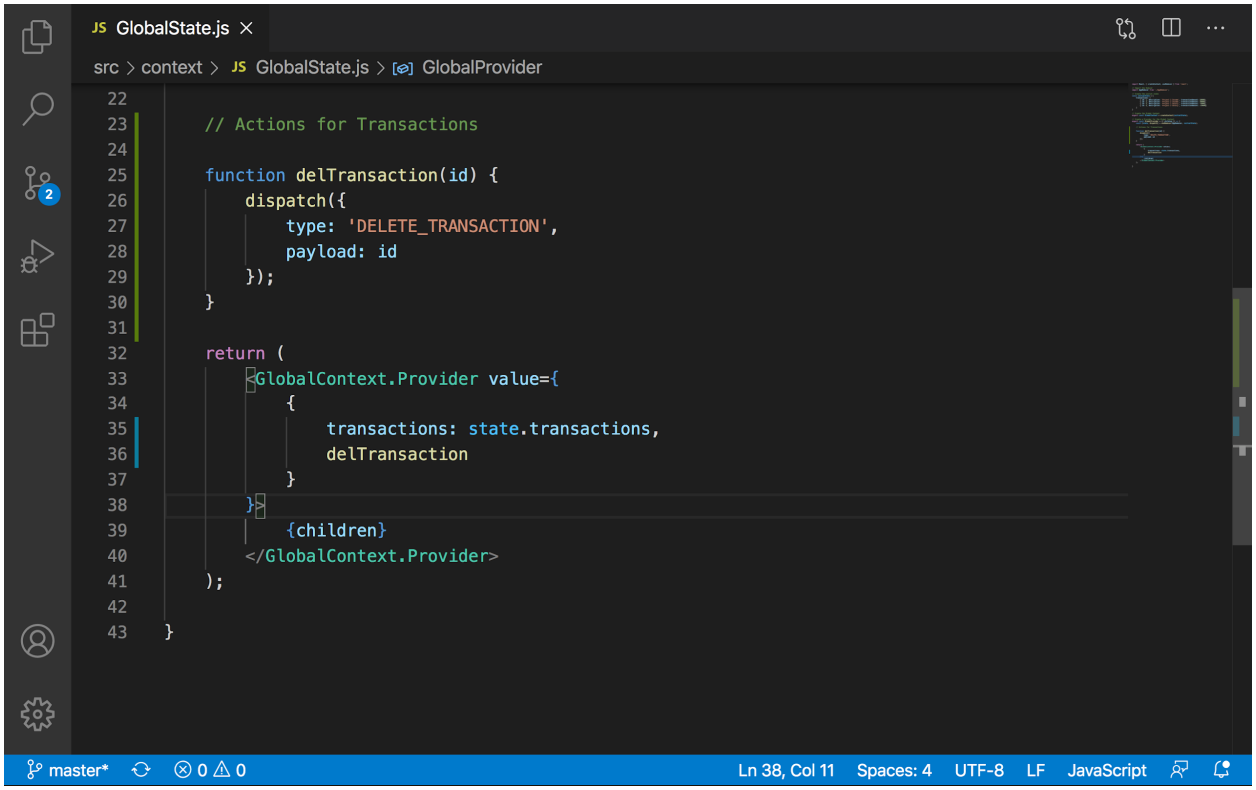
Step 1: Create an action in the GlobalProvider for the delTransaction feature



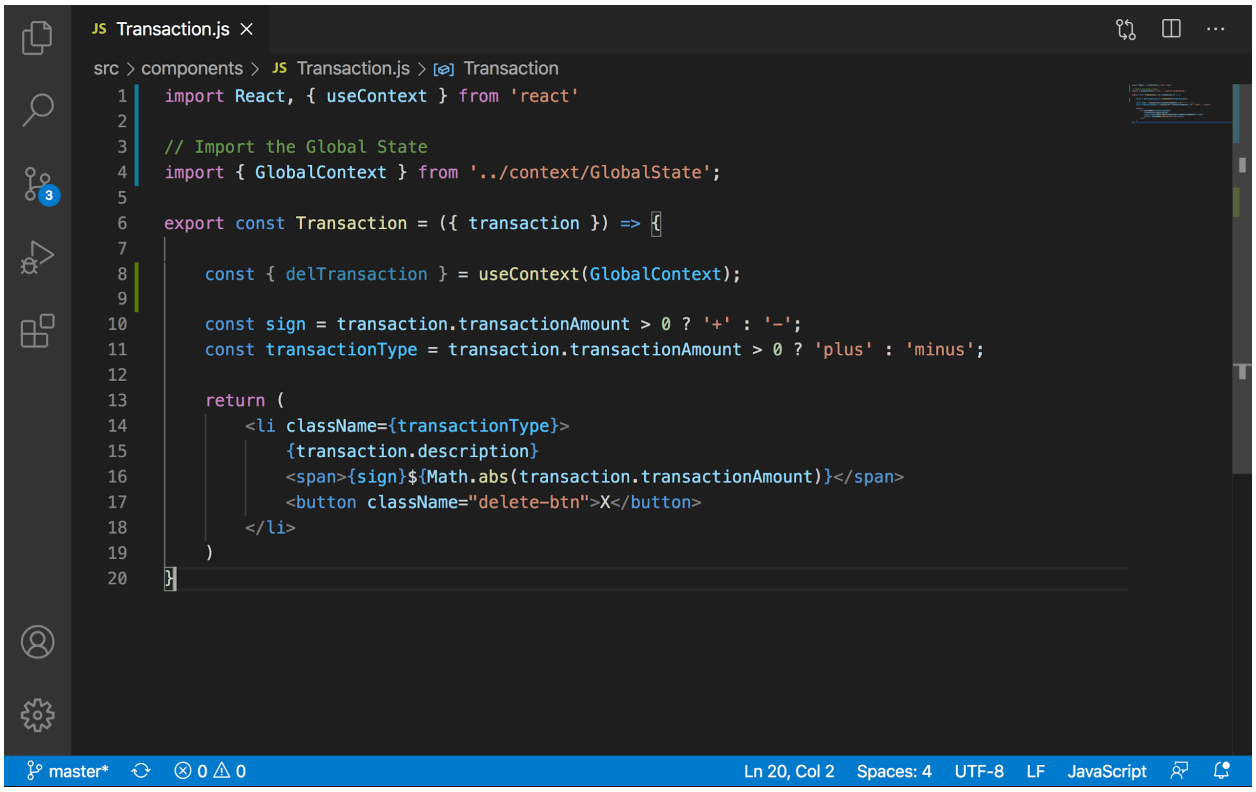
Step 2: Create a case for the delete transaction in the AppReducer to create a new state



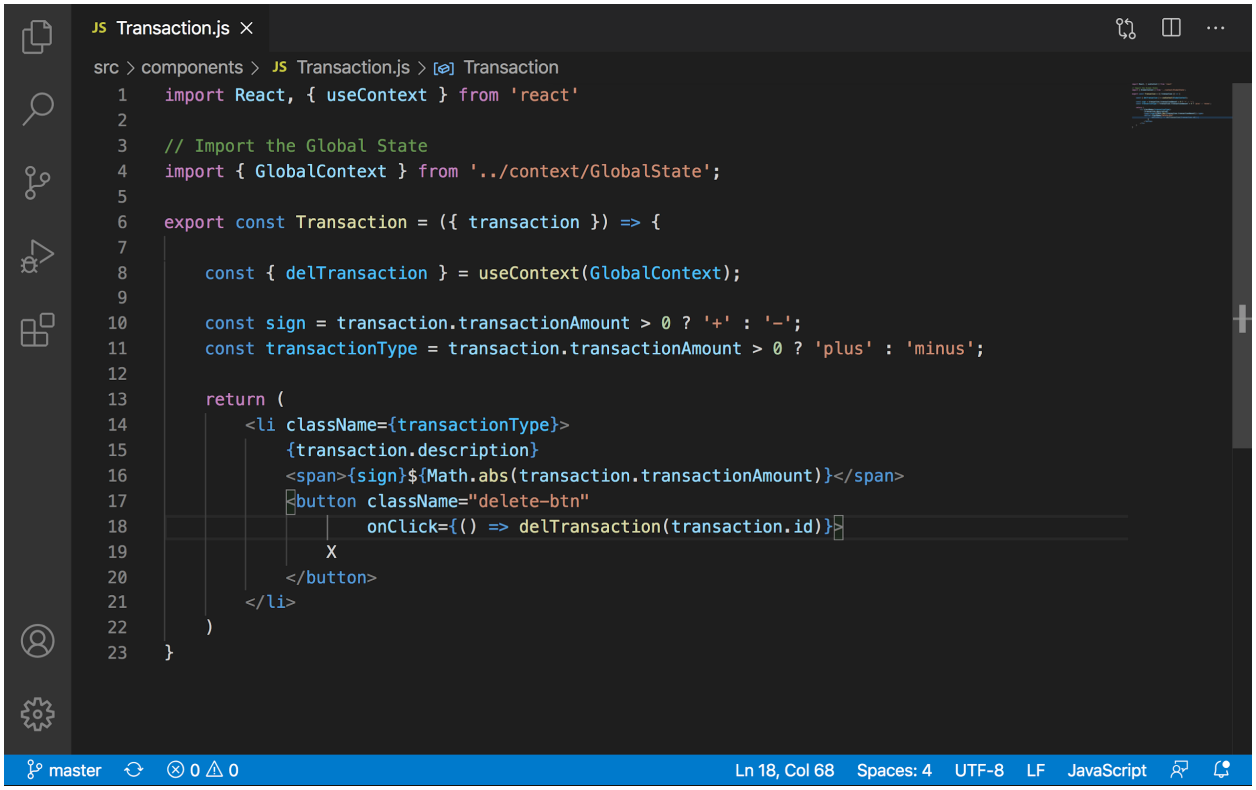
Step 3: Update the GlobalProvider to pass new delTransaction function to components



Step 4: Update the Transaction component with delTransaction
Import the useContext and GlobalContext
Get the delTransaction from the GlobalContext

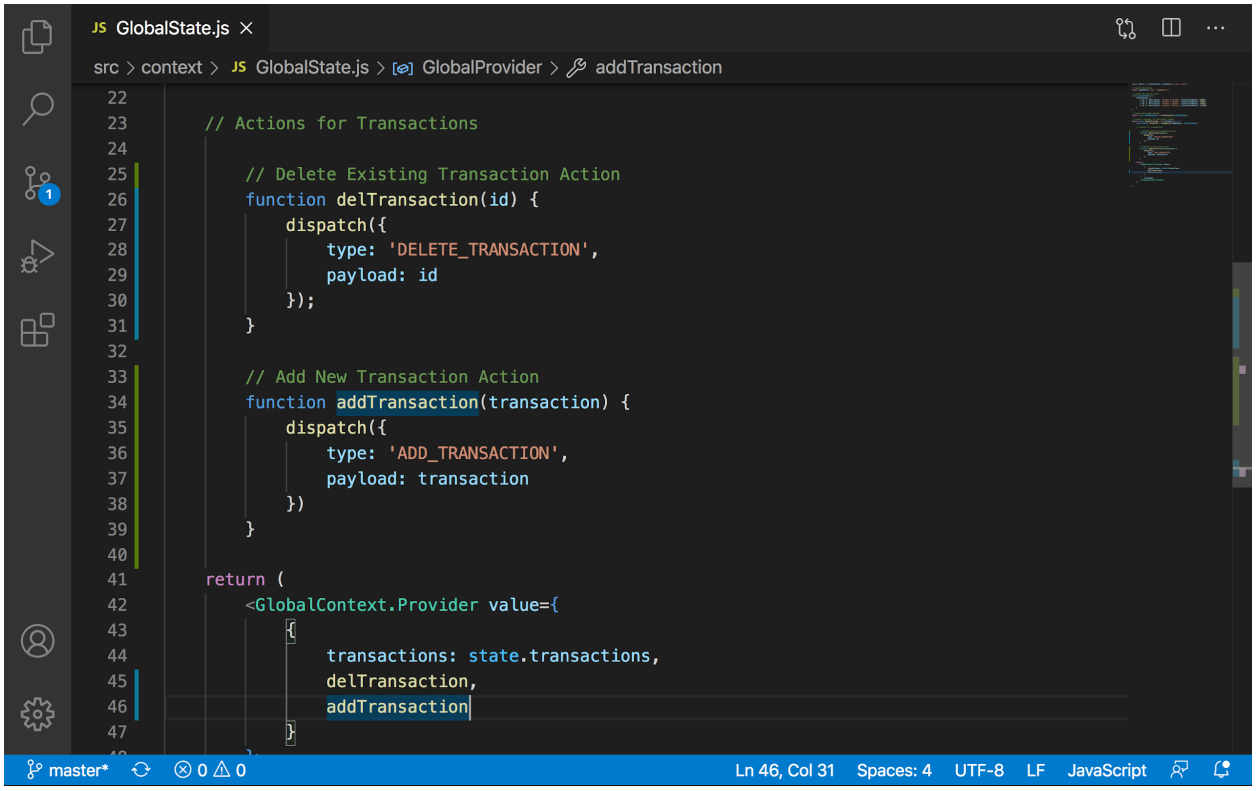


Step 5: Call the delTransaction function when clicking the x button

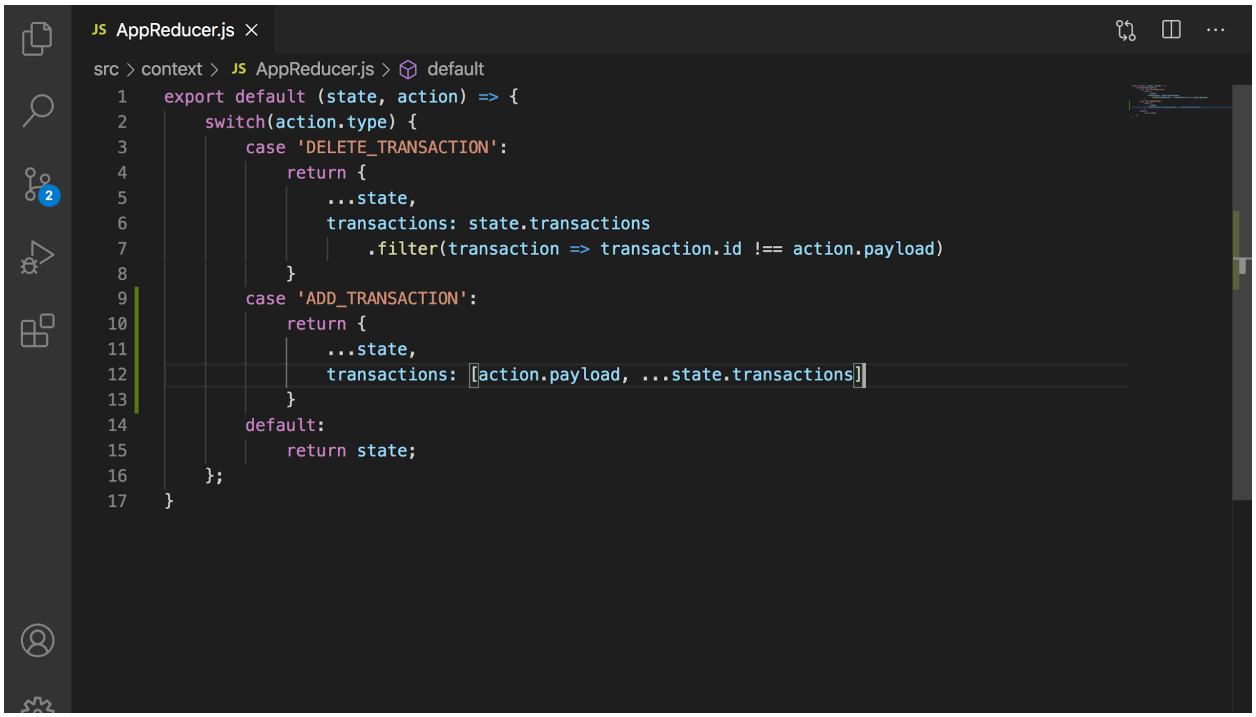


Part XIII - Add an Action to Add New Transactions

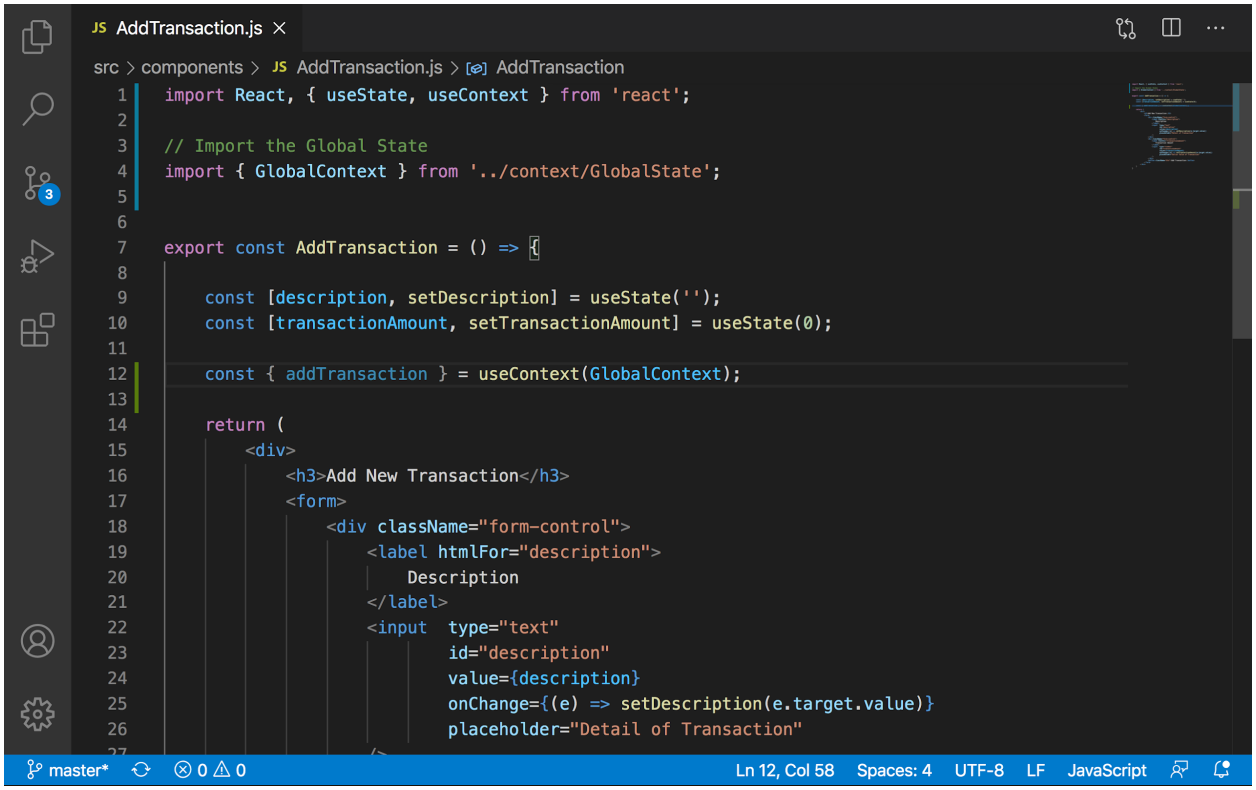
Step 1: Create an action in the GlobalProvider for the addTransaction feature and add to the Provider



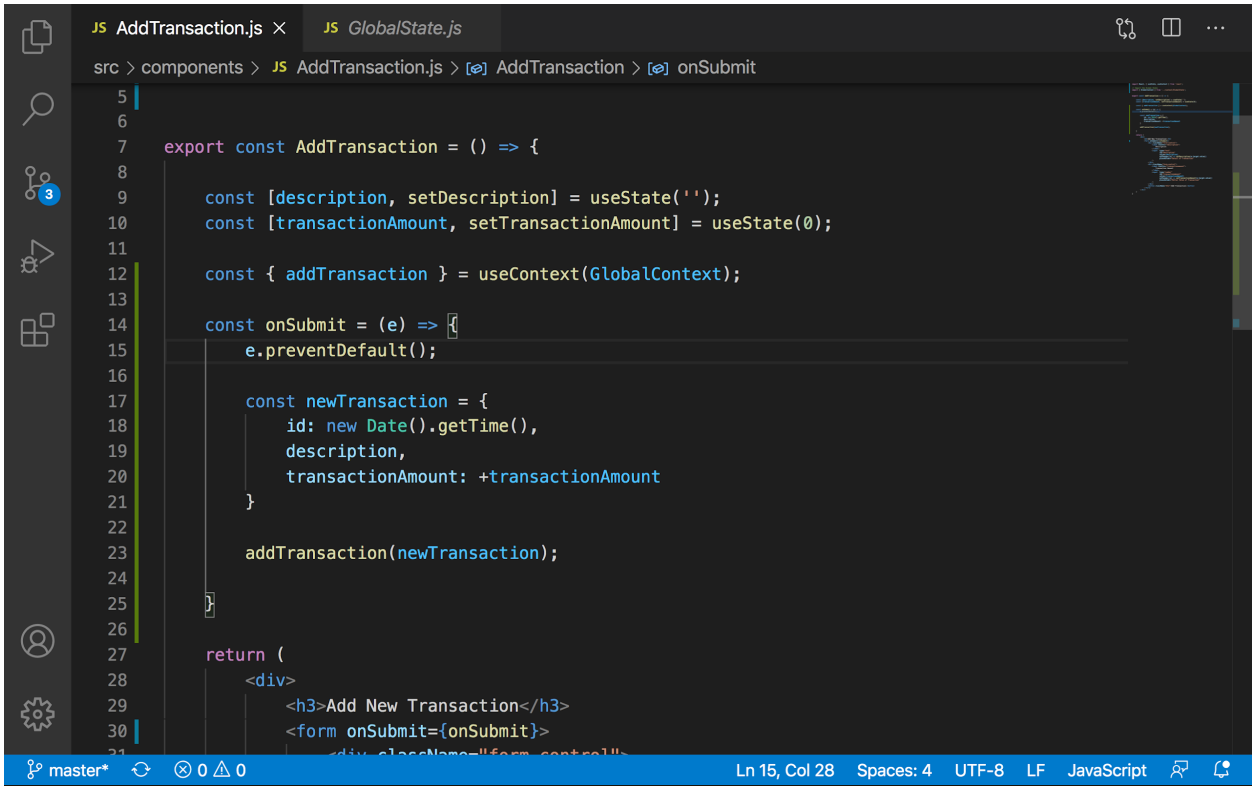
Step 2: Create a case for the add transaction in the AppReducer to create a new state



Step 3: Update the AddTransaction component with the new action to add new transactions
Import useContext and the GlobalContext and create the addTransaction const



Step 4: Create the onSubmit function and apply the onSubmit event in the form



Step 5: Remove initialState transactions from GlobalState