React is a Javascript library for building user interfaces.

This example explains how one can integrate GraphQL with a React application.

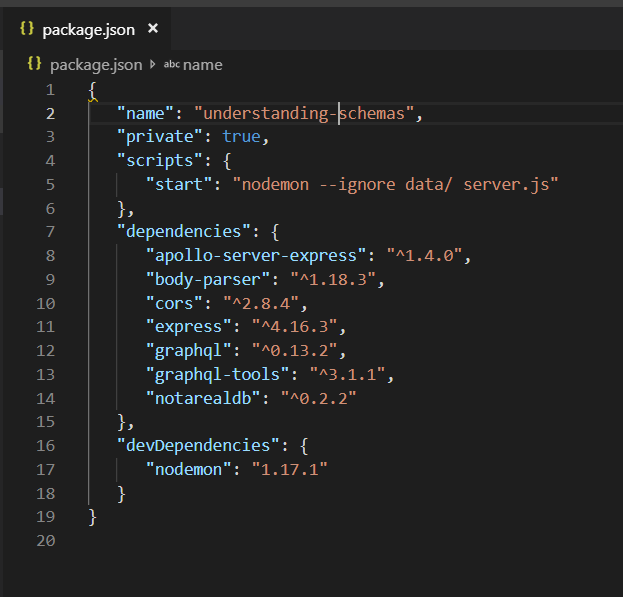
The quickest way to set up a react project is by using the *Create React App* tool.

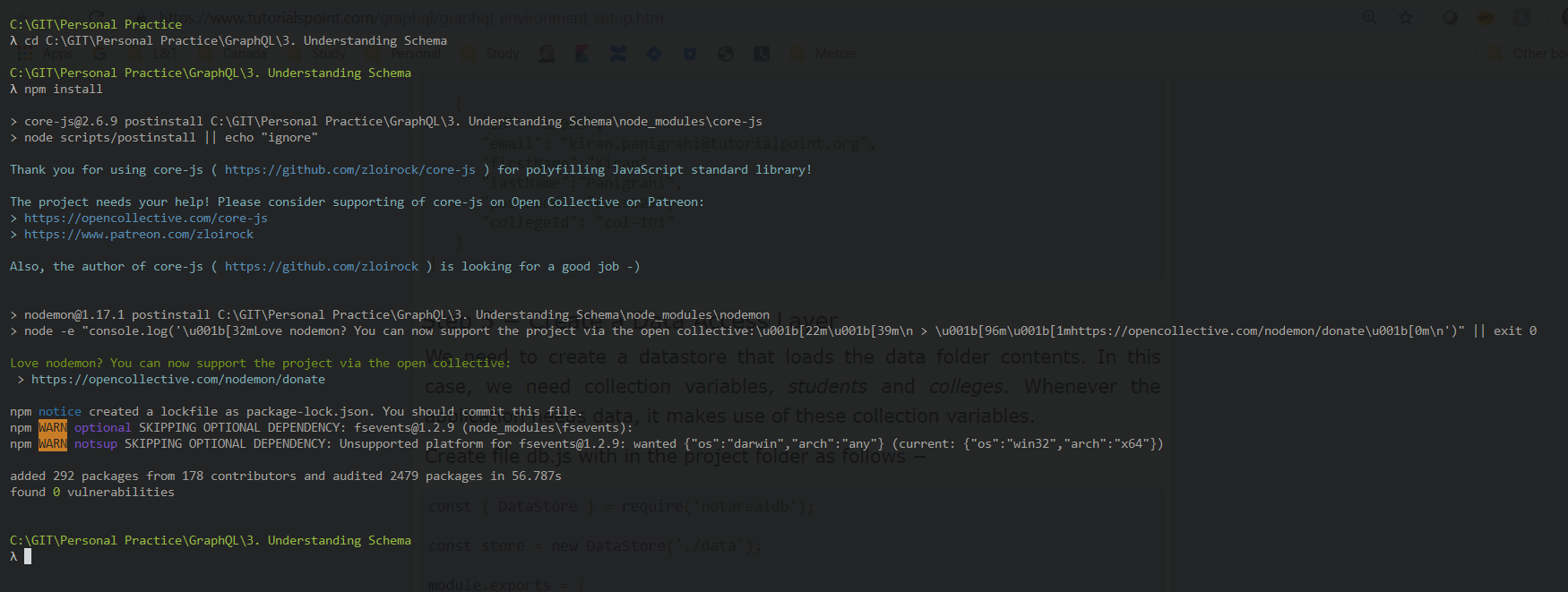
In the subsequent sections, we will learn how to set up both the Server and the Client.

# Setting up Server

## Step 1 − Download and Install Required Dependencies for the Project

Create new package.json file with below code and install all dependencies.

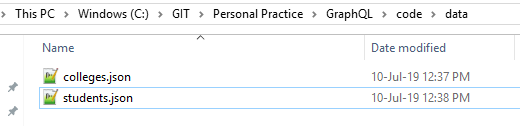


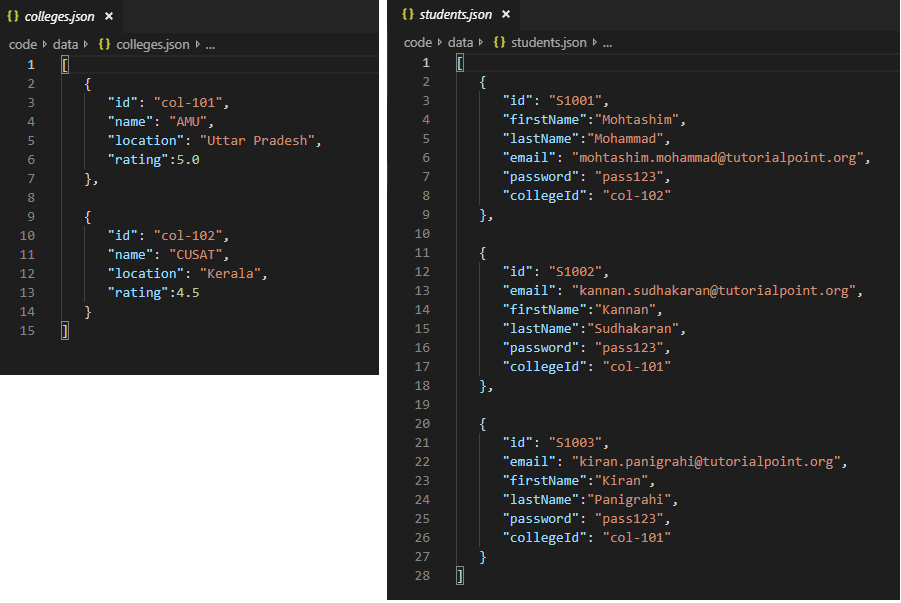


**Create Flat File Database in Data Folder**

we use flat files to store and retrieve data.

Create a folder data and add two files **students.json** and **colleges.json**.





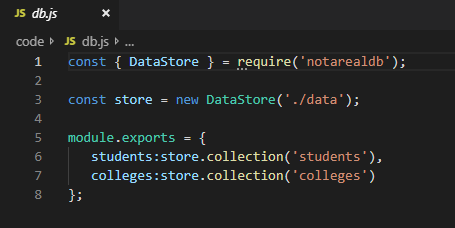
**Create a Data Access Layer**

We need to create a datastore that loads the data folder contents.

In this case, we need collection variables, *students* and *colleges*.

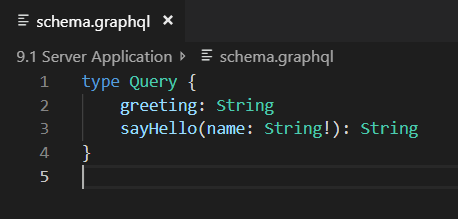
Whenever the application needs data, it makes use of these collection variables.

Create file db.js with in the project folder as follows



## Step 2 − Create a Schema

Add **schema.graphql** file with following code



The file has defined two queries –

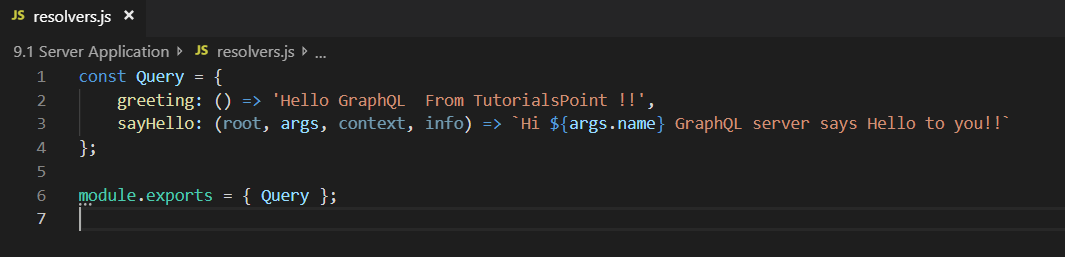
* greeting and
* sayHello.

The sayHello query accepts a string parameter and returns another string.

The parameter to the sayHello() function is not null.

## Step 3 − Create Resolver

Create a file **resolvers.js** with following code



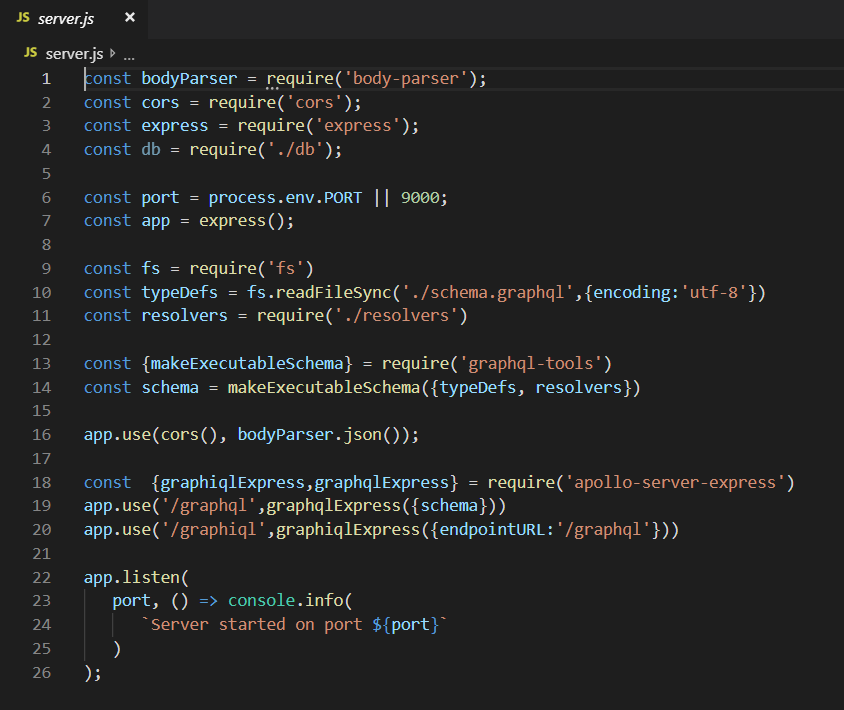
Here greeting and sayHello are two resolvers.

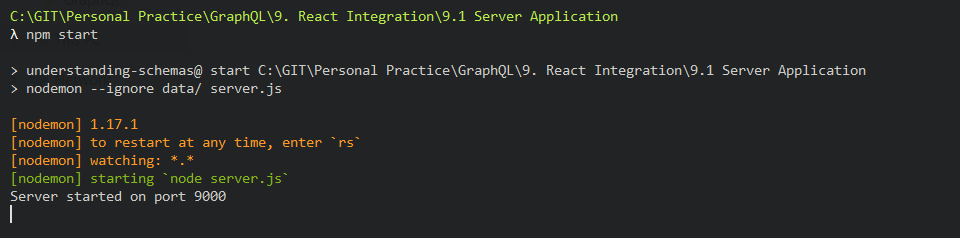
In the sayHello resolver, the value passed to the name parameter can be accessed through args.

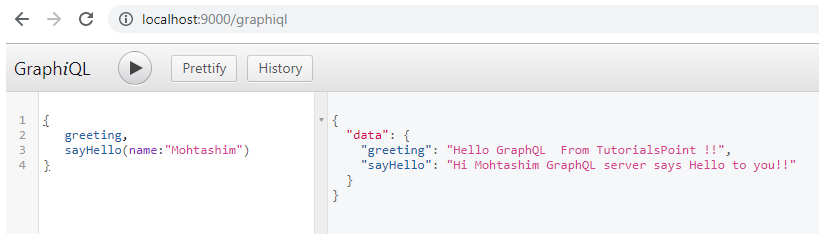
To access resolver functions outside the module, Query object has to be exported using module.exports.

## Step 4 − Run the Application

Create a server.js file.







# Setting up the Client

Open a new terminal for client.

The server terminal should be kept running before executing the client application.

* **React application** will be running on port number 3000 and
* **Server application** on port number 9000.

## Step 1 − Create a React Application

In the client terminal, type the following command

npx create-react-app Client-Application

This will install everything needed for a typical react application.

The npx utility and create-react-app tool create a project with name **Client-Application**.

Once the installation is completed, open the project in VSCode.

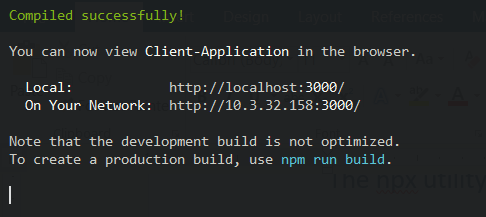
## Step 2 − Start Client-Application

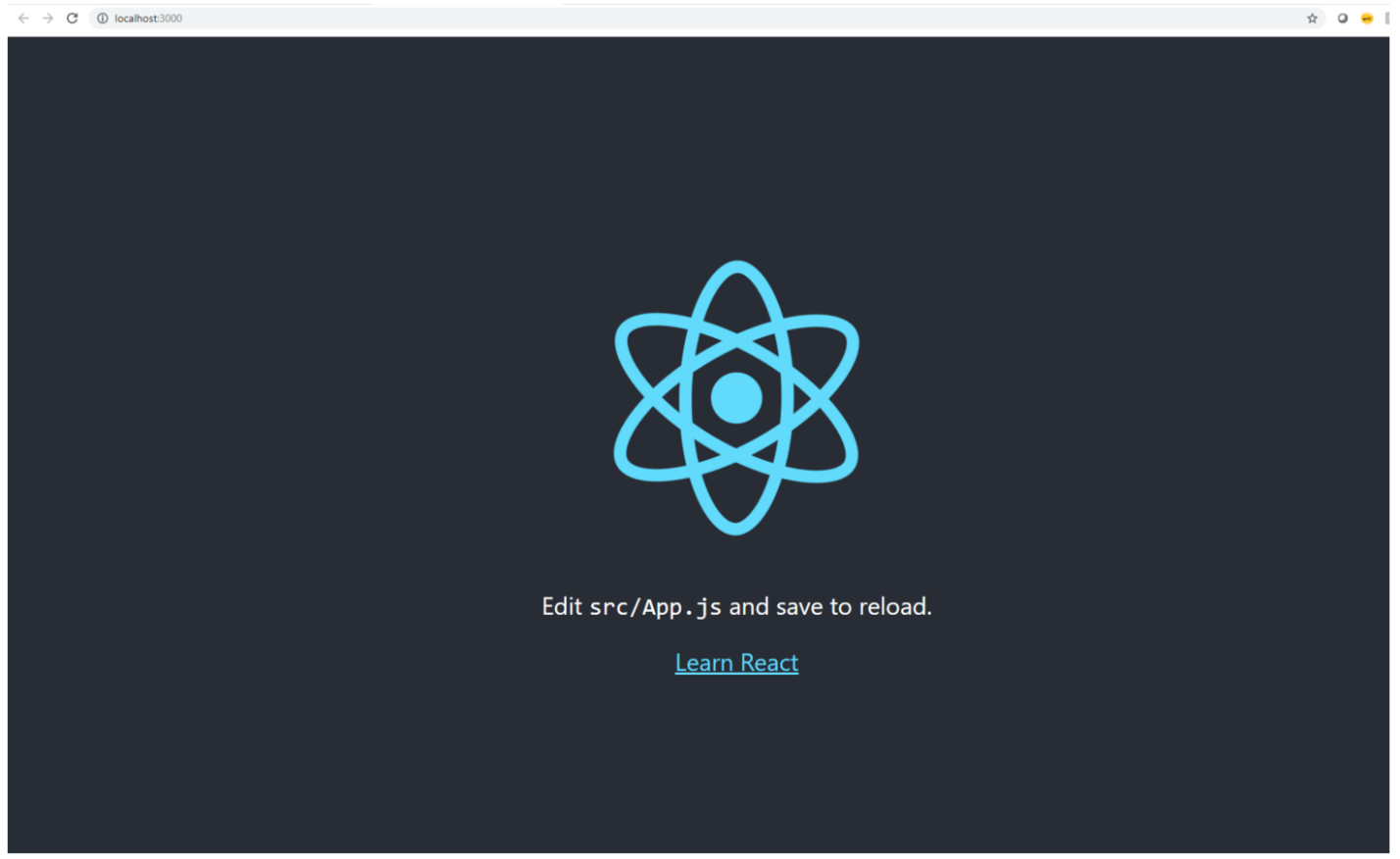
Change the current folder path in the terminal to **Client-Application**.

Type **npm start** to launch the project.

This will run a development server at port 3000 and will automatically open the browser and load the index page.

This is shown in the screenshot given below





## Step 3 − Modify the App Component

In the App.js inside src folder, add two functions, one to load greeting and another to load sayHello messages.

Following is the loadGreeting function which sends GraphQL query for greeting.

async function loadGreeting() {

const response = await fetch('http://localhost:9000/graphql', {

method:'POST',

headers:{'content-type':'application/json'},

body:JSON.stringify({query:'{greeting}'})

})

const rsponseBody = await response.json();

return rsponseBody.data.greeting;

console.log("end of function")

}

Following is the **loadSayhello** function which sends GraphQL query for sayHello −

async function loadSayhello(name) {

const response = await fetch('http://localhost:9000/graphql', {

method:'POST',

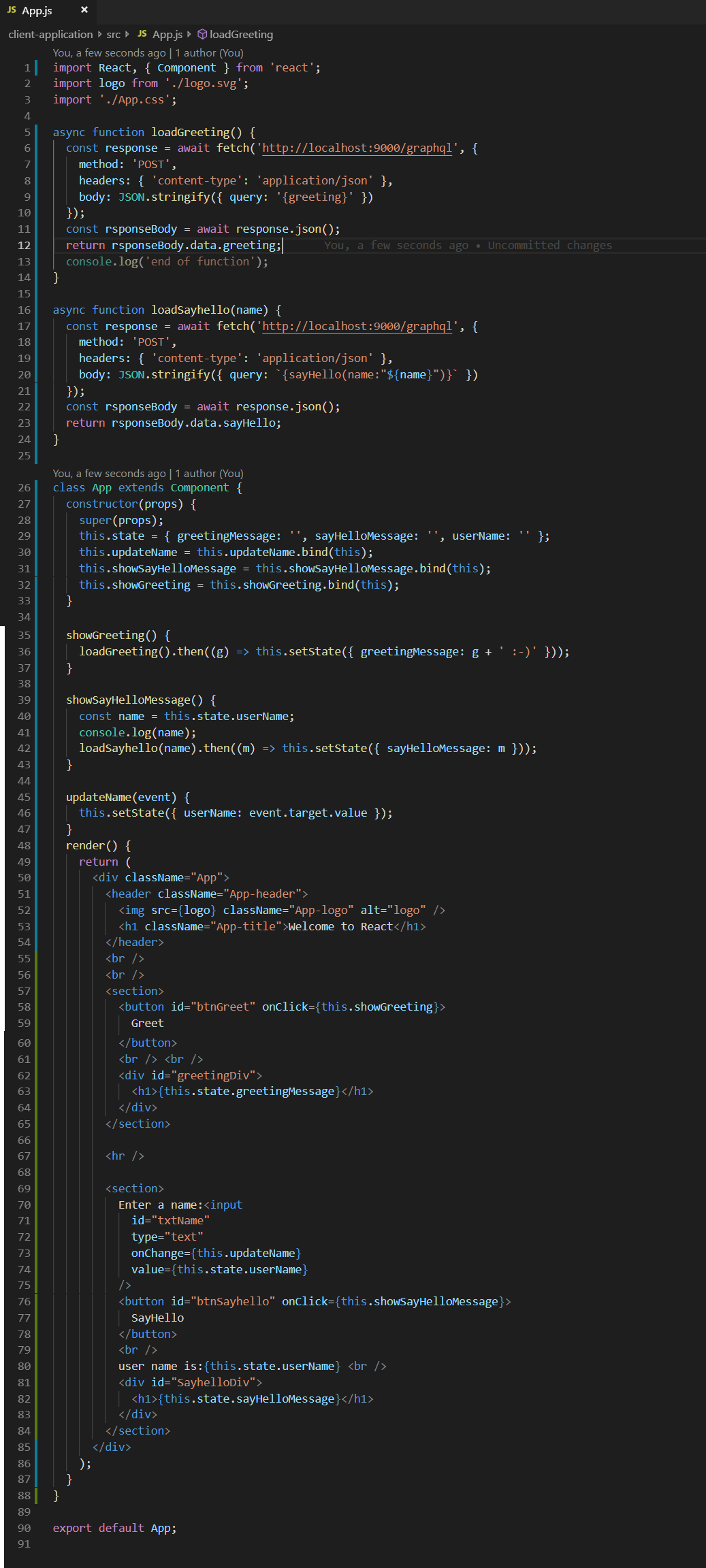
headers:{'content-type':'application/json'},

body:JSON.stringify({query:`{sayHello(name:"${name}")}`})

})

}

The complete **App.js** file is shown below −



Once both the applications are running, click on the greet button. Next, enter a name in the textbox and click on sayHello button. The output will be as given below –

