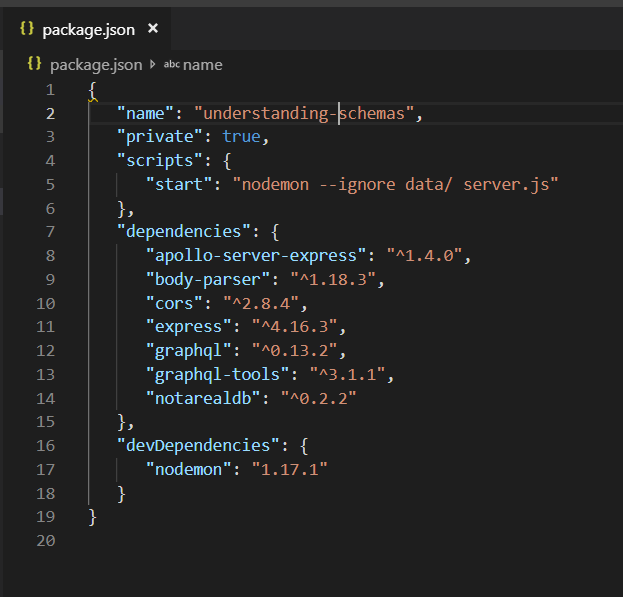
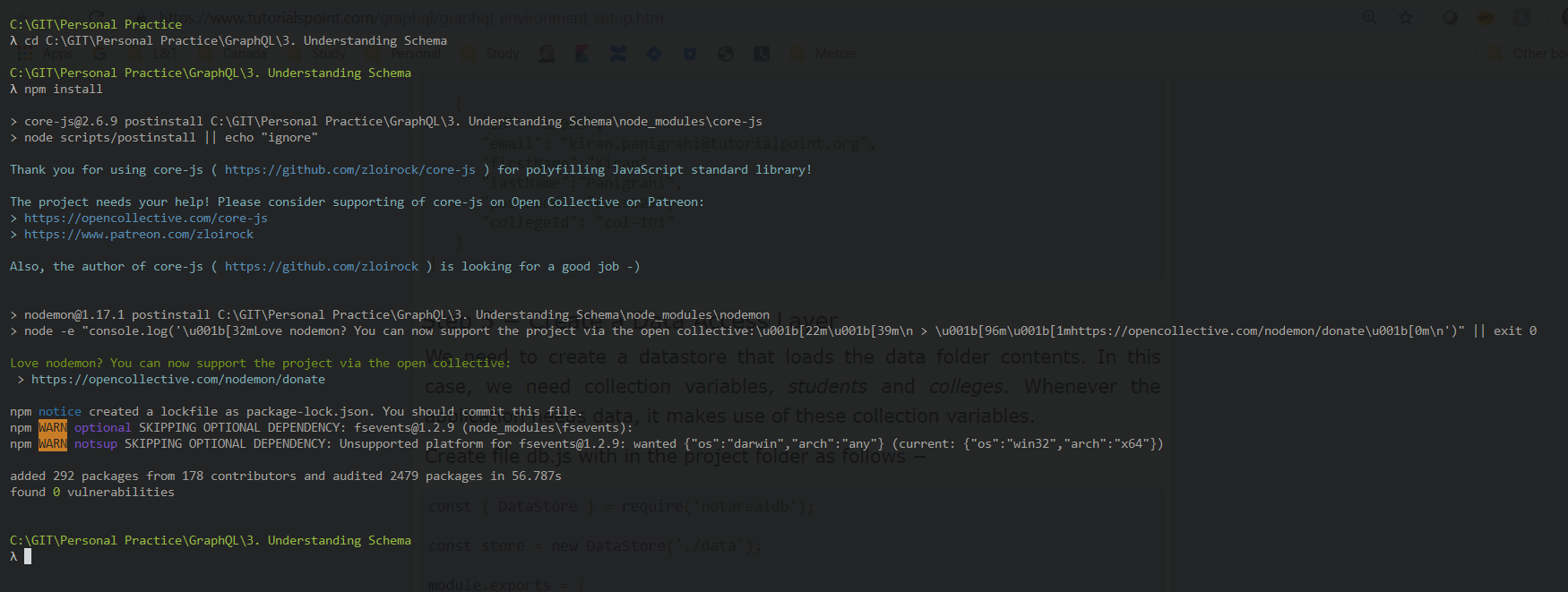
Let us see how to use Apollo Client to build client applications using the following steps

# 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.

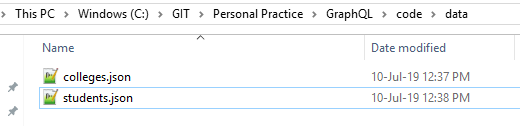


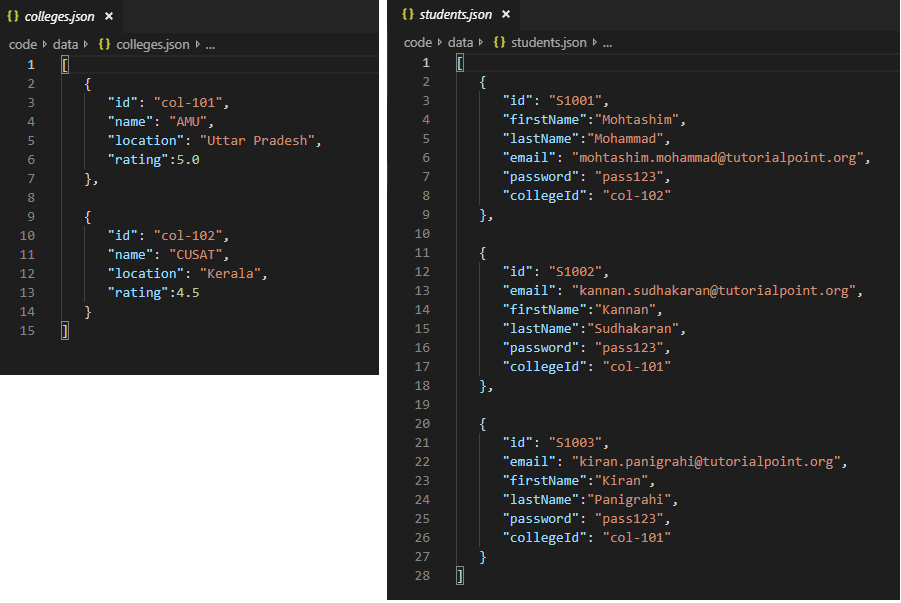


**Create Flat File Database in Data Folder**

we use flat files to store and retrieve data.

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





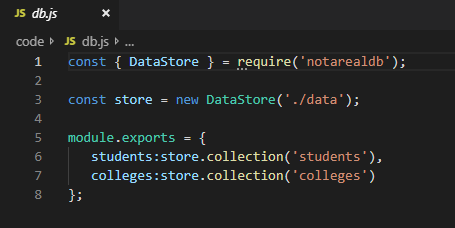
**Create a Data Access Layer**

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

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

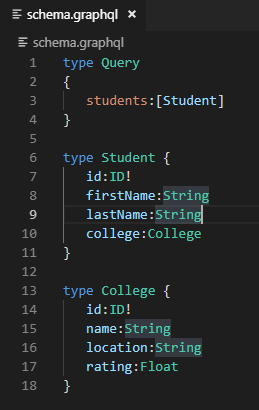
Whenever the application needs data, it makes use of these collection variables.

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



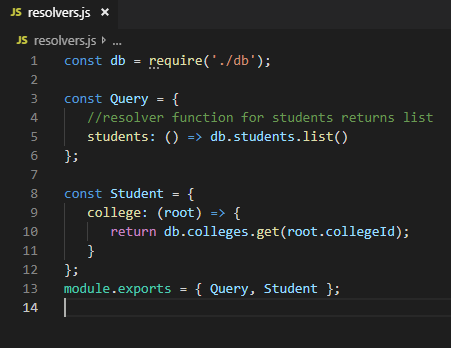
## Step 2 − Create a Schema

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



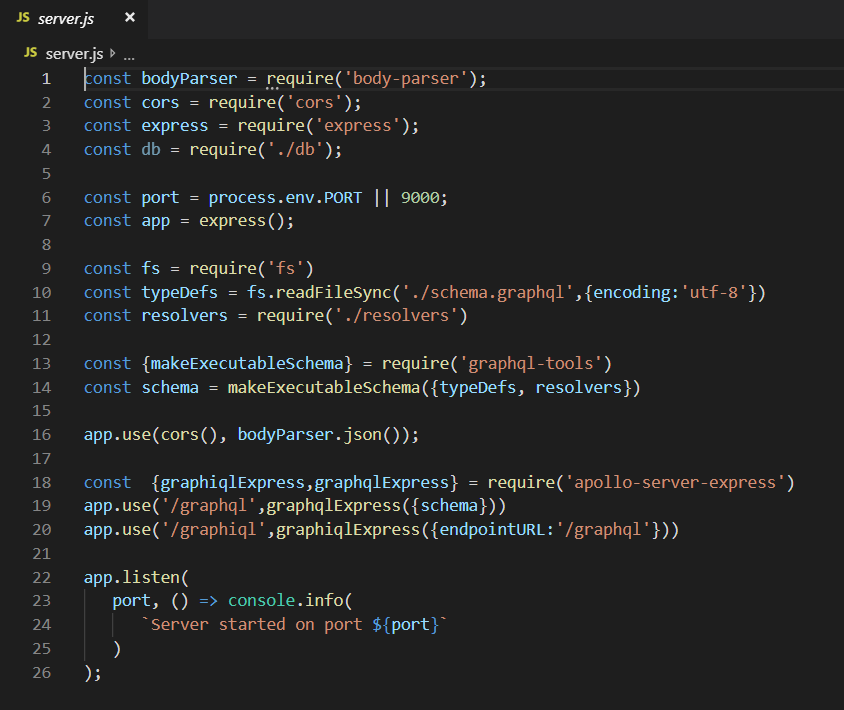
## Step 3 − Create Resolver

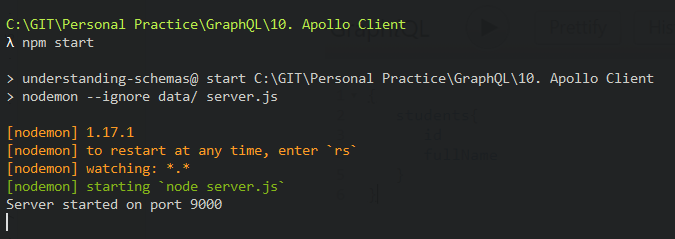
Create a file **resolvers.js** with following code

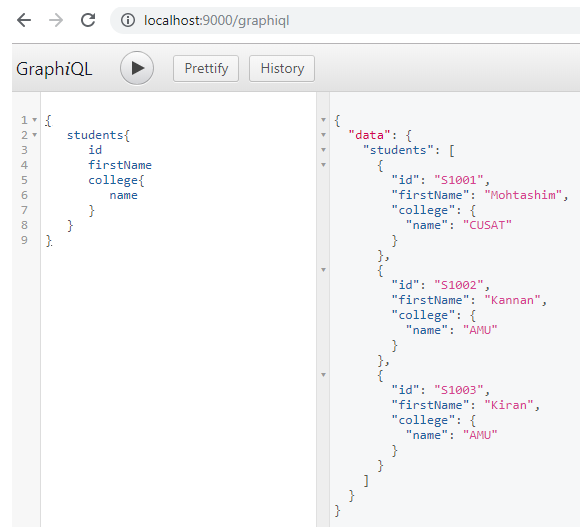


## Step 4 − Run the Application

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.

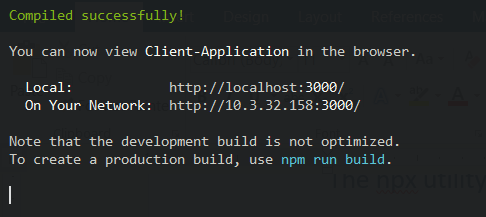
## Step 2 − Start hello-world-client

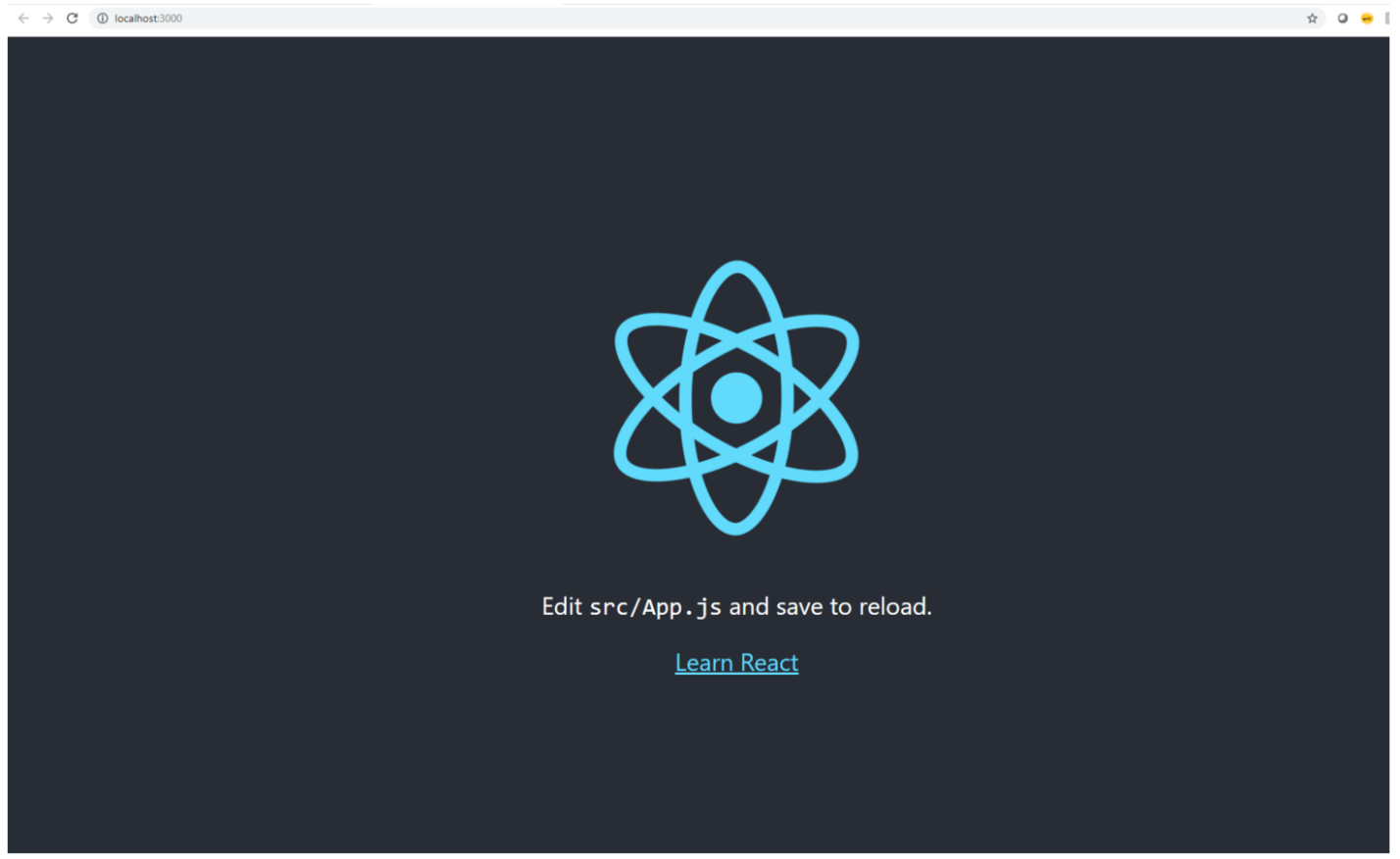
Change the current folder path in the terminal to **Client-Application**.

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.

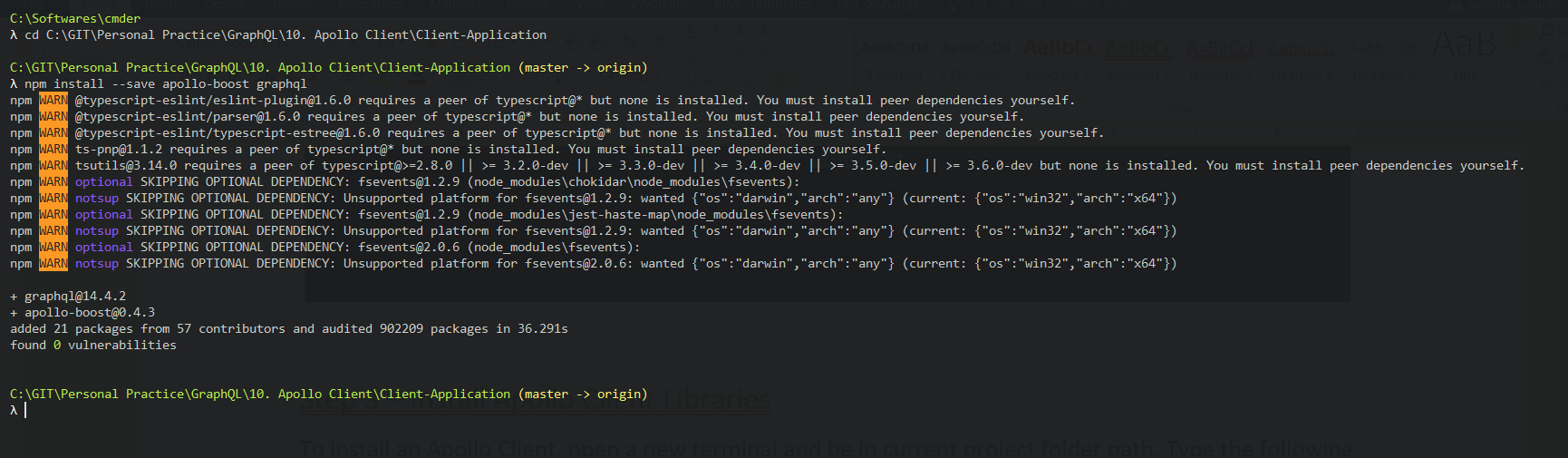
This is shown in the screenshot given below





## Step 3 − Install Apollo Client Libraries

To install an Apollo Client, open a new terminal and be in current project folder path. Type the following command



This will download the graphql libraries for client side and also the Apollo Boost package.

We can cross check this by typing npm view in apollo-boost dependencies.

This will have many dependencies as shown below

{

'apollo-cache': '^1.1.15',

'apollo-cache-inmemory': '^1.2.8',

'apollo-client': '^2.4.0',

'apollo-link': '^1.0.6',

'apollo-link-error': '^1.0.3',

'apollo-link-http': '^1.3.1',

'apollo-link-state': '^0.4.0',

'graphql-tag': '^2.4.2'

}

We can clearly see that Apollo-Client library is installed.

## Step 4 − Modify the App Component in index.js File

With Apollo Client, we can directly call server without the use of fetch API.

Also, the queries and mutations should not be embedded in a string made with back tick notation.

This is because, the **gql** function directly parses the queries.

This means, a programmer can directly write queries in the same way when writing queries in GraphiQL tool.

**gql** is a tag function which will parse the template string written in back tick notation to graphql query object.

The Apollo Client query method returns a promise.

Following code snippet shows how to import Apollo Client −

import {ApolloClient, HttpLink, InMemoryCache} from 'apollo-boost'

const endPointUrl = 'http://localhost:9000/graphql'

const client = new ApolloClient({

link: new HttpLink({uri:endPointUrl}),

cache:new InMemoryCache()

});

In the previous chapter, we discussed how to use fetch API for HTTP requests.

The following code shows how to use **gql** function.

The **loadStudentsAsync**function uses graphql client to query the server.

async function loadStudentsAsync() {

const query = gql`

{

students{

id

firstName

lastName

college{

name

}

}

}`

const {data} = await client.query({query}) ;

return data.students;

}

You only need to keep the **index.js** in **src** folder and index.html in public folder; all other files that are auto generated can be removed.

The directory structure is given below –

hello-world-client /

-->node\_modules

-->public

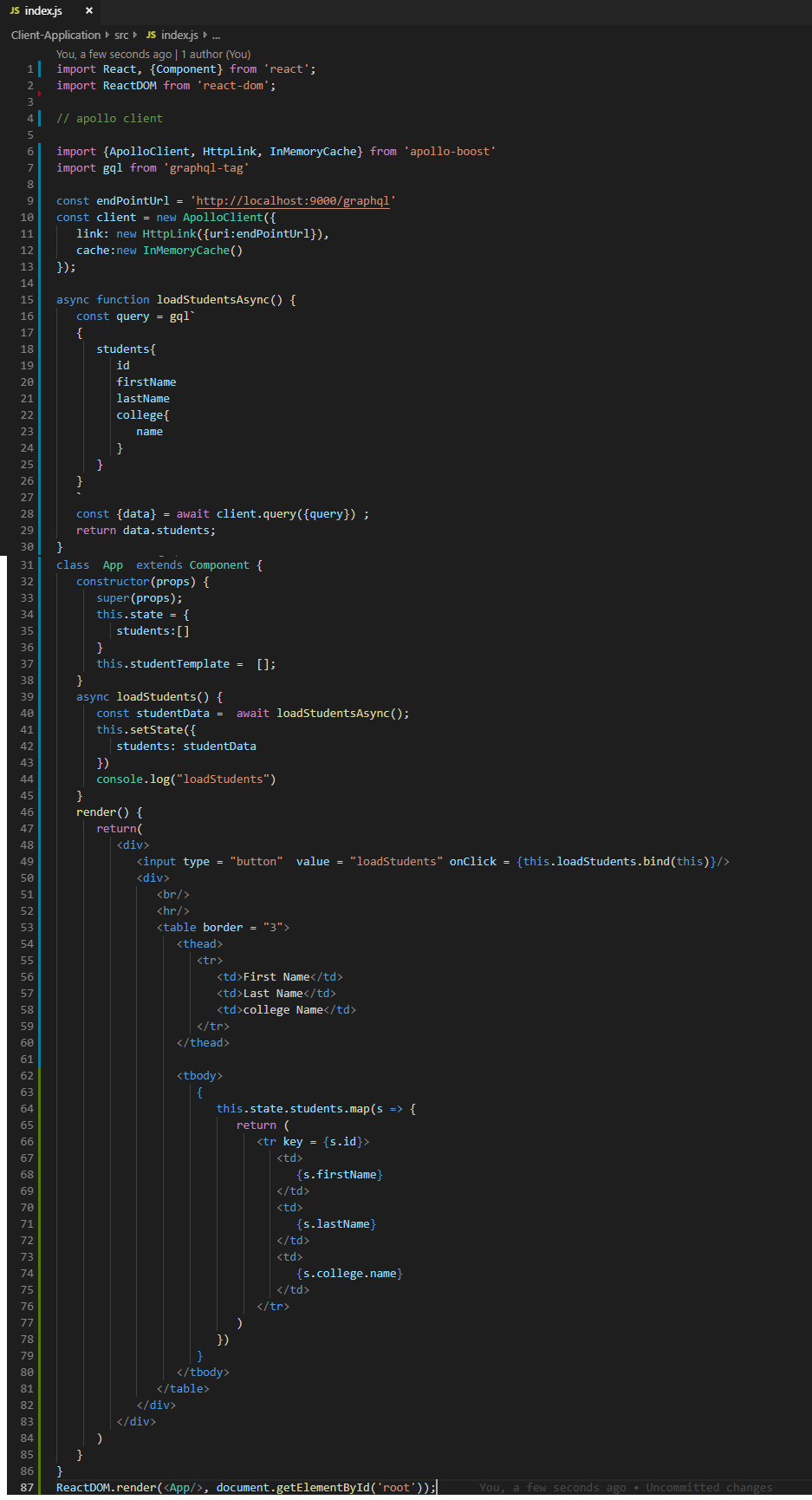
index.html

-->src

index.js

-->package.json

Following is the **index.js** in react application



The react application will load students from GraphQL server, once we click on loadStudents button as shown below −

