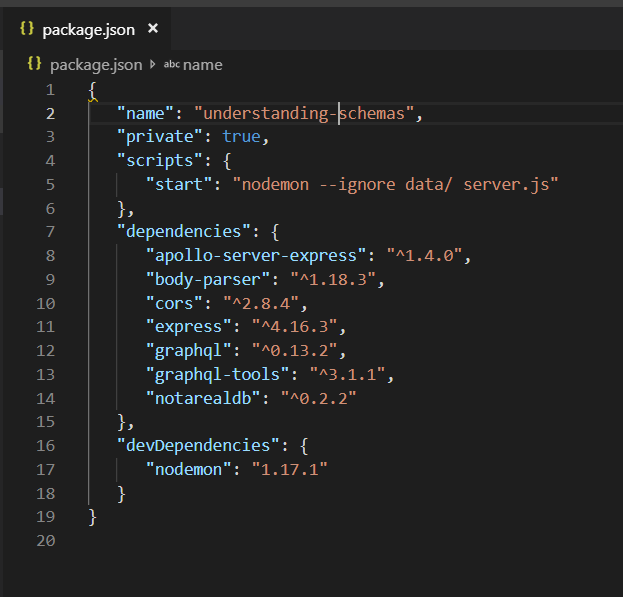
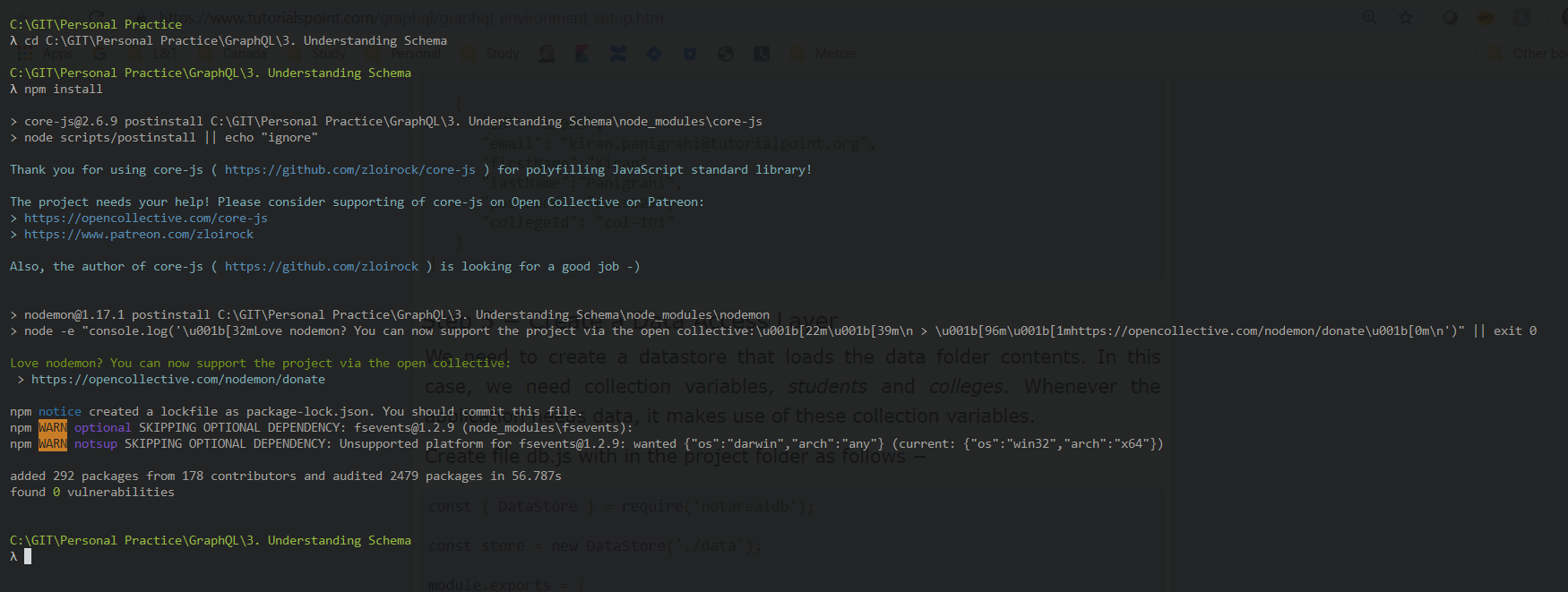
The custom validation logic will be a part of the resolver function.

Let us understand this with the help of an example.

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

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

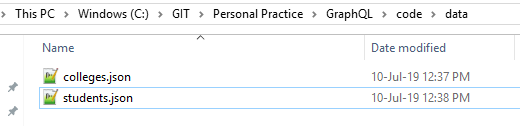


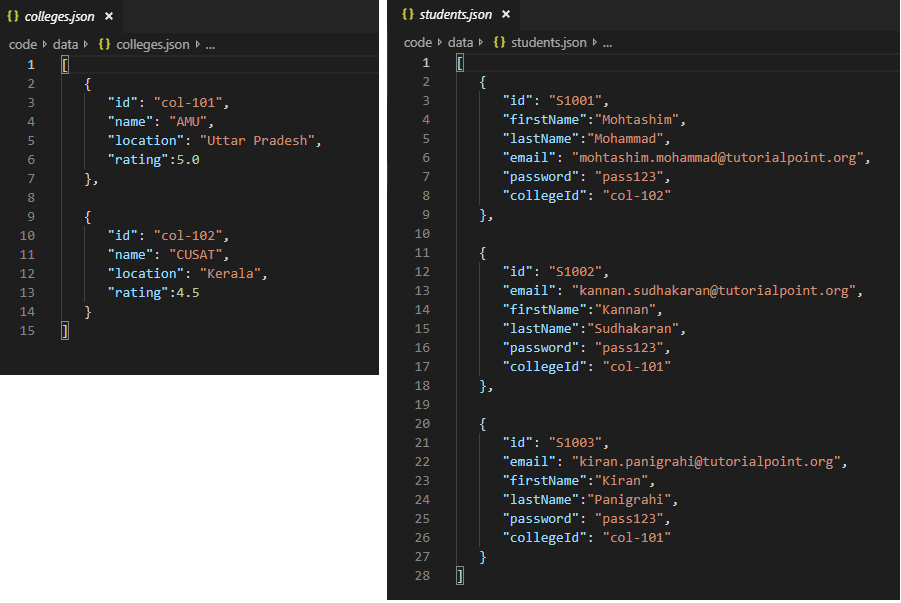


**Create Flat File Database in Data Folder**

we use flat files to store and retrieve data.

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





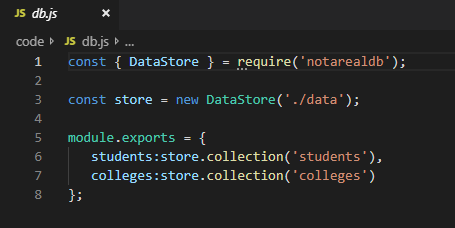
**Create a Data Access Layer**

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.

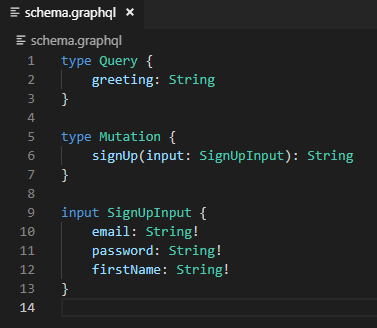
Create file db.js with in the project folder as follows



## Step 2 − Create a Schema

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

The schema file already has the student field. Let us add a field college and define its type.

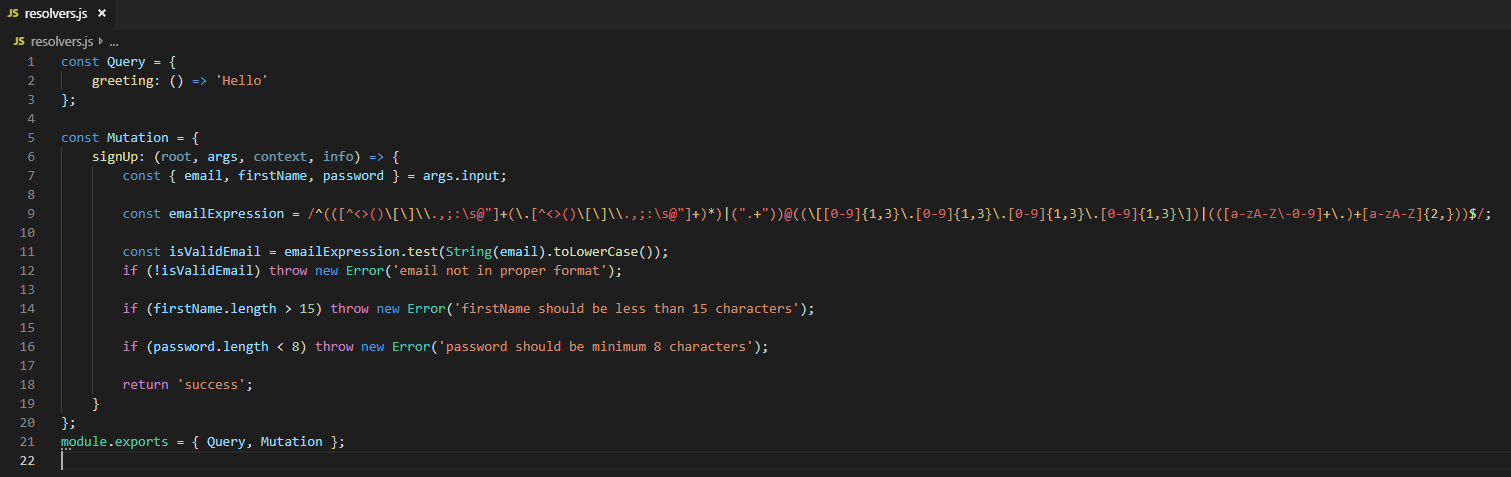


**Note**

We can use the input type SignUpInput to reduce the number of parameters in signUp function. So, signUp function takes only one parameter of type SignUpInput.

## Step 3 − Create Resolver

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

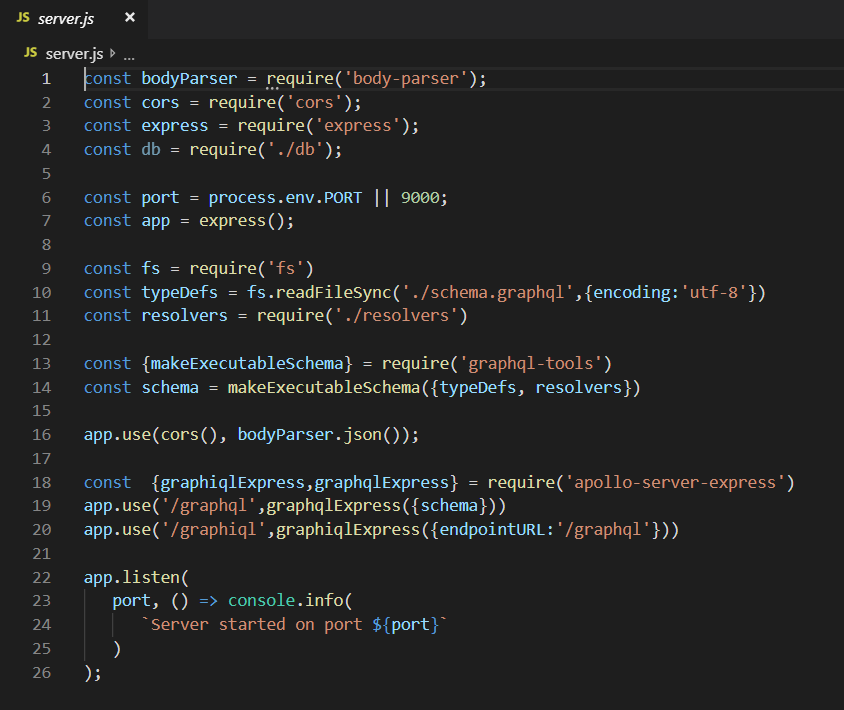


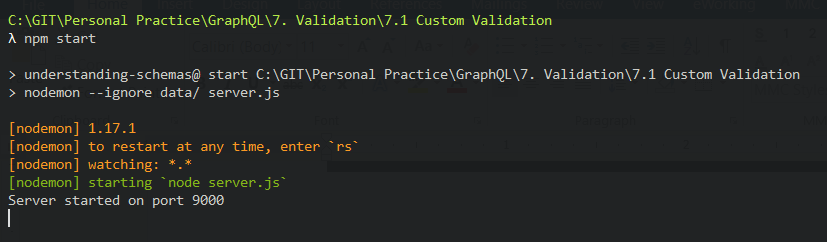
The resolver function, signUp accepts parameters email, password and firstName.

These will be passed through input variable so that it can be accessed through args.input.

## Step 4 − Run the Application

Create a server.js file.





Open the browser and enter the URL **http://localhost:9000/graphiql**. Type the following query in the editor −

mutation doSignUp($input:SignUpInput) {

signUp(input:$input)

}

## CASE 1

Since input to signup function is a complex type, we need to use query variables in graphiql.

For this, we need to first give a name to the query and call it doSignUp, the $input is a query variable.

The following query variable must be entered in query variables tab of graphiql –

{

"input":{

"email": "abc@abc",

"firstName": "kannan",

"password": "pass@1234"

}

}

The errors array contains the details of validation errors as shown below

{

"data": {

"signUp": null

},

"errors": [

{

"message": "email not in proper format",

"locations": [

{

"line": 2,

"column": 4

}

],

"path": [

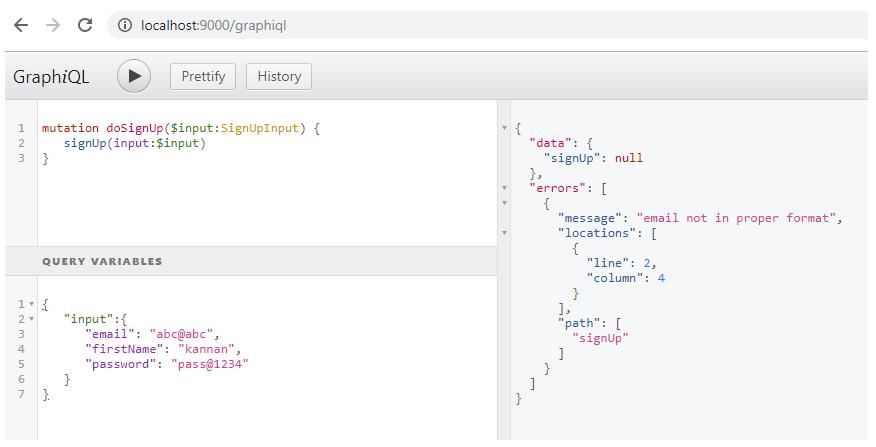
"signUp"

]

}

]

}



## CASE 2

We have to enter a proper input for each field as given below

{

"input":{

"email": "abc@abc.com",

"firstName": "kannan",

"password": "pass@1234"

}

}

The response is as follows

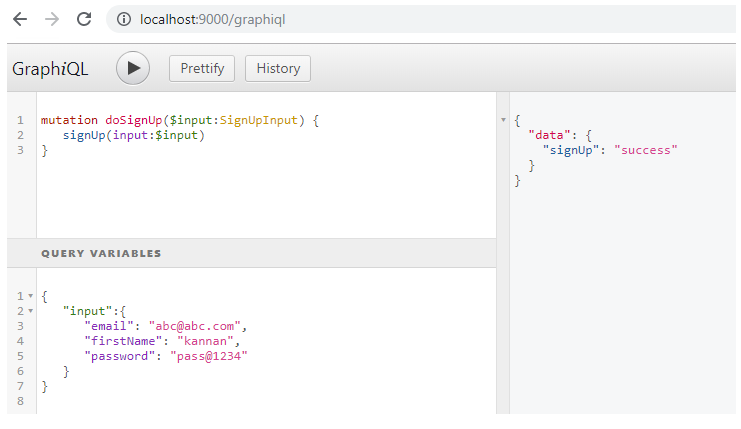
{

"data": {

"signUp": "success"

}

}



## CASE 3

Here, in the below query, we are not assigning any password.

{

"input":{

"email": "abc@abc.com",

"firstName": "kannan"

}

}

If a required field is not provided, then qraphql server will display the following error −

{

"errors": [

{

"message": "Variable \"$input\" got invalid value {\"email\":\"abc@abc.com\",\"firstName\":\"kannan\"}; Field value.password of required type String! was not provided.",

"locations": [

{

"line": 1,

"column": 19

}

]

}

]

}

