

Faculty of Science and Engineering

**SCHOOL OF MATHEMATICS AND COMPUTER SCIENCE**

**Photography Web Application**

**Final Report and User Manual**

**Submitted by:- Mahindra Maddipati(2013727)**

**Submitted to:- Alix Bergeret**

**Module:- 7CC005 – Web Technologies (report/web application)**

**Link to mi-linux: http://314f74f36219.ngrok.io/api/v1/**

This report consists of working and implementation of single-page web application designed for Photography site.

Photography site is built by a specific functionality for Client-side framework with Server-side API. Where server-side API’s and data are stored and integrated in mi-linux.wlv.ac.uk . This photography application was built for portfolio to my friend.

**Objective:-**

The main objective is to help my friend in his career. This portal was created with well-structured templates with responsive design. I included all social media links to this website and each template comes with integrated social media graphics, blogging, personal gallery. As for portfolio website I just included only subscribe option and also contact details option for visitor user to contact.

**Design:-**

This website consists of index items:-

1. Home
2. Gallery
3. Blog
4. Contact
5. And search bar(due to API token issues it was not seeded at present).

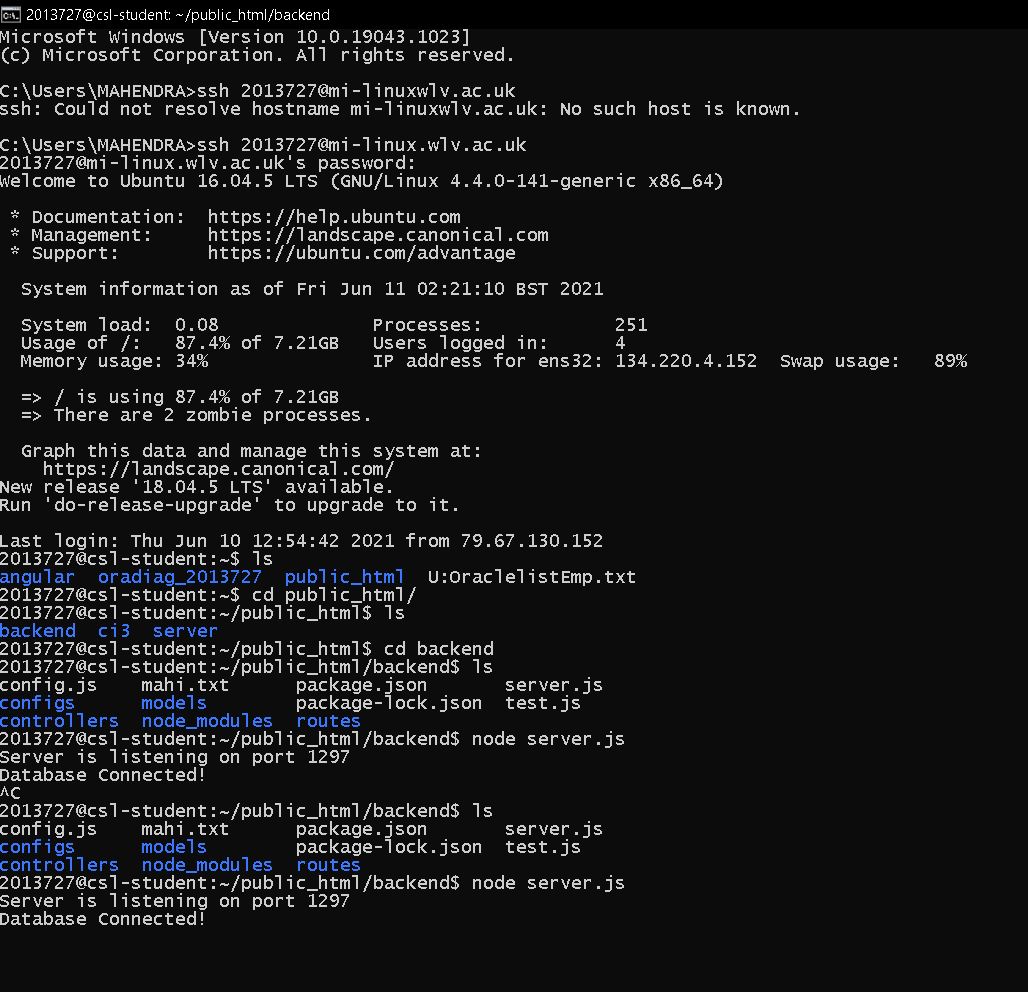
This portal I used 6 Rest-full API’s to integrate data from mi-linux.wlv.ac.uk. At present all credentials are stored in mi-linux MySql database.



**Execution Steps:-**

Instead of PHP I used Node JS for server-side scripting. In mi-linux(MySql database) provided by college is just providing local host credentials and also for PHP language it is more suitable for running locally but for node JS I can’t access it. Due to this, I used **ngrok** application where it allows me to tunnel the URL to my local running application. And also by this application we can run the test cases also.

* By ssh connect to college server.
* By using filezilla I transferred all the backend and ngrok files to mi-linux server.



* After connecting to database. Open another terminal and run the ngrok.
* Below ngrok running server port is defined in backend code in database only accessed or changed by .env(environment) command as nano command.

But in backend file to connect confidentially to mi-linux server. And also express JS for routing angular.

const dbConn = mysql.createConnection({

host : 'localhost',

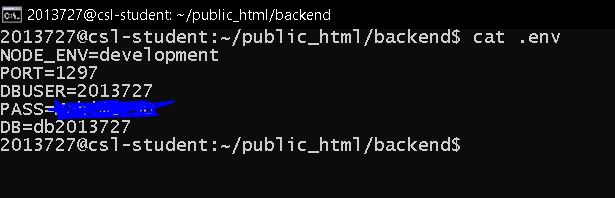
user : user,

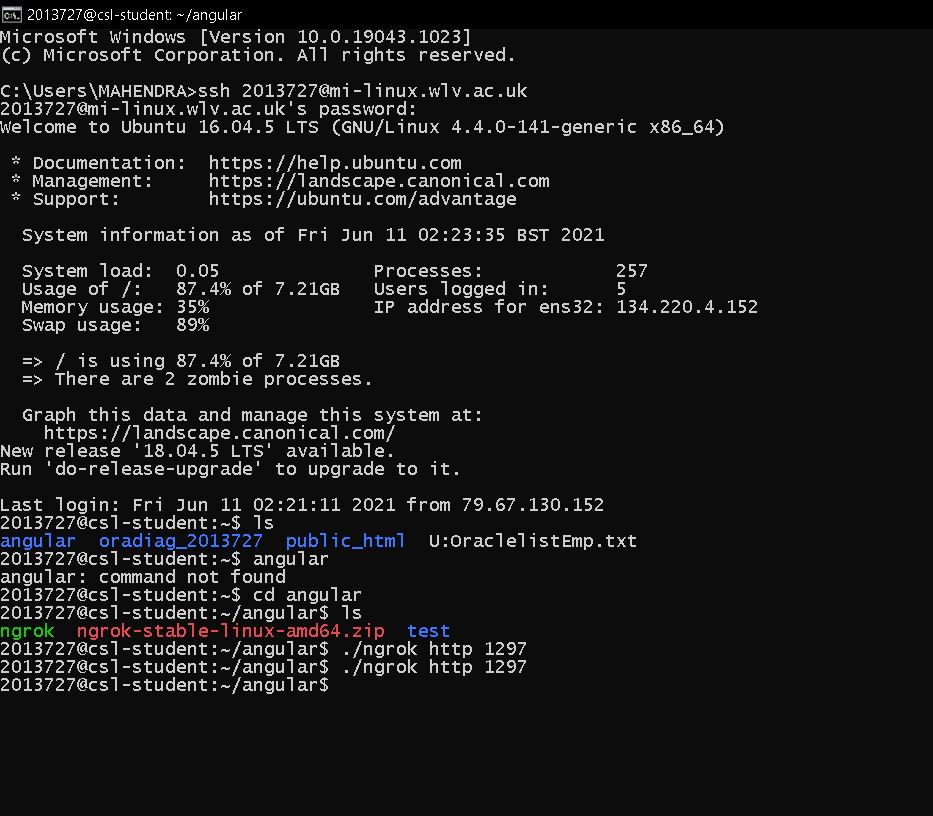
port : 3306,

password : pass,

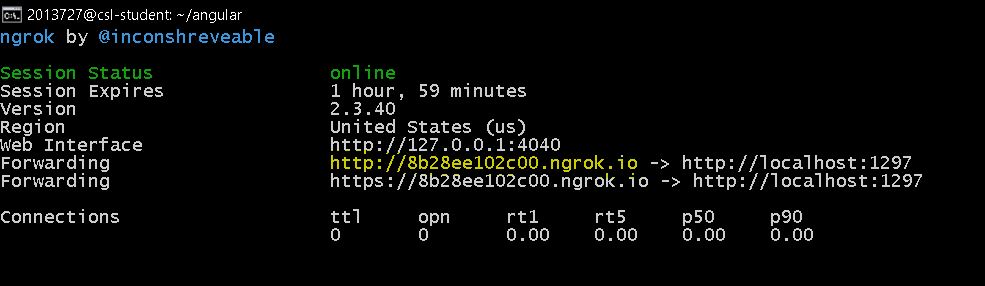
database : db

});

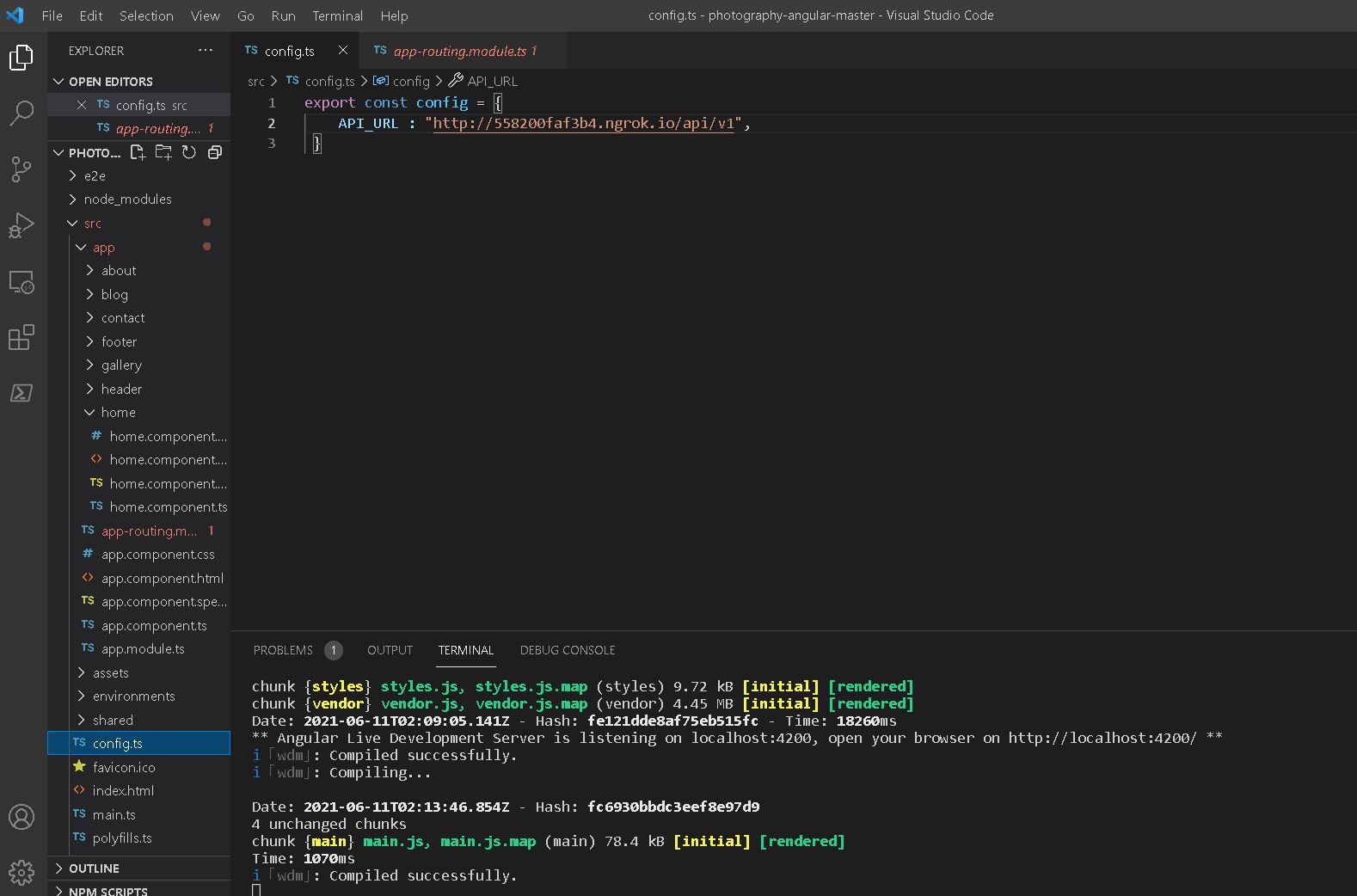




Then it will generate forwarding link(below image yellow color).



Now copy that link and paste in config.ts in and run the terminal with command ng serve.



**Sql script for seed data:-**

These commands I directly run on mysql. So as per indication I am mentioning sample file.

CREATE TABLE IF NOT EXISTS about (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`type` varchar(20) DEFAULT NULL,

`text` varchar(500) DEFAULT NULL,

`created\_at` DATETIME DEFAULT CURRENT\_TIMESTAMP,

`updated\_at` DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

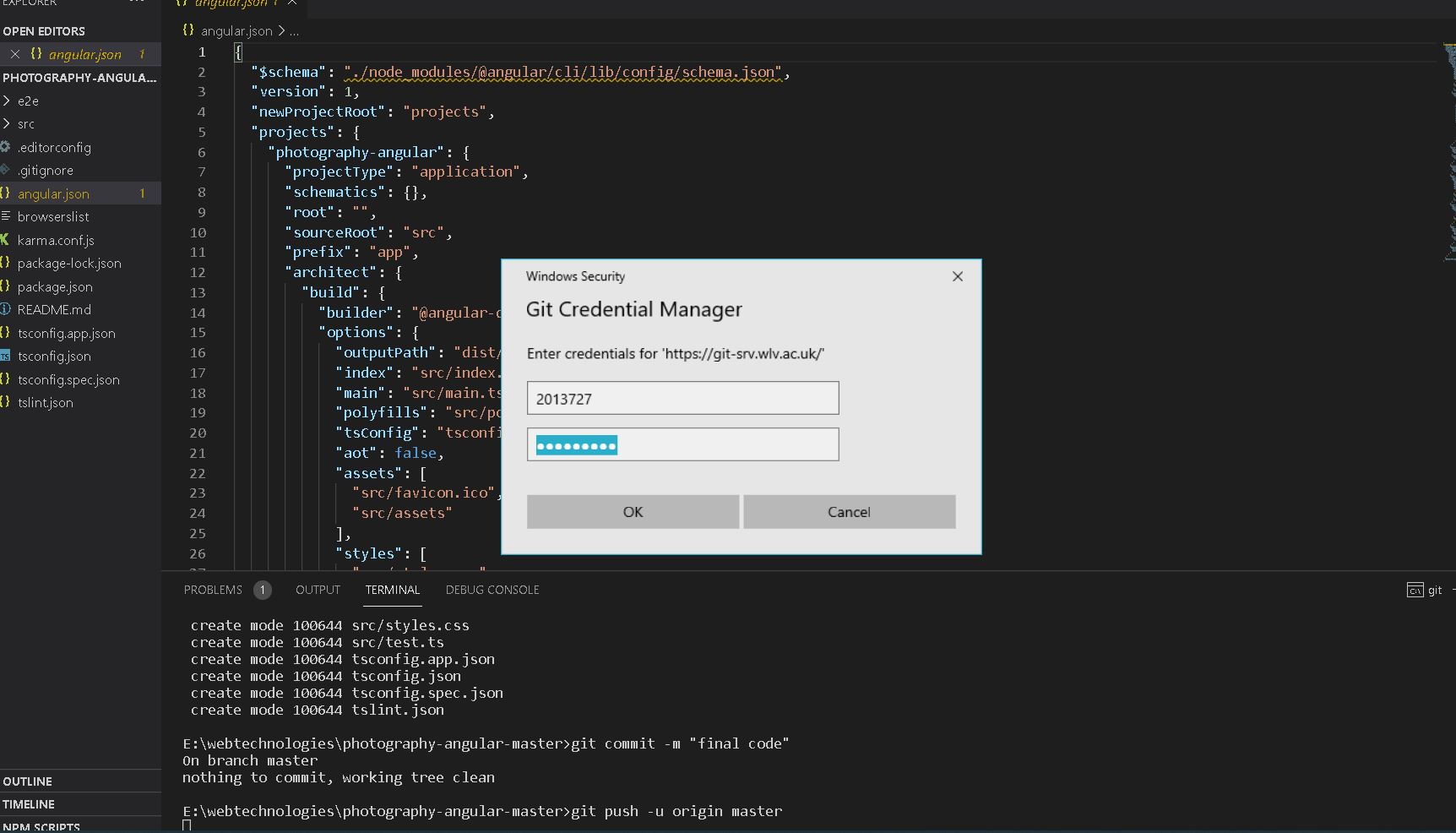
PRIMARY KEY (`id`)

);

**Pushing to gitlab:-**

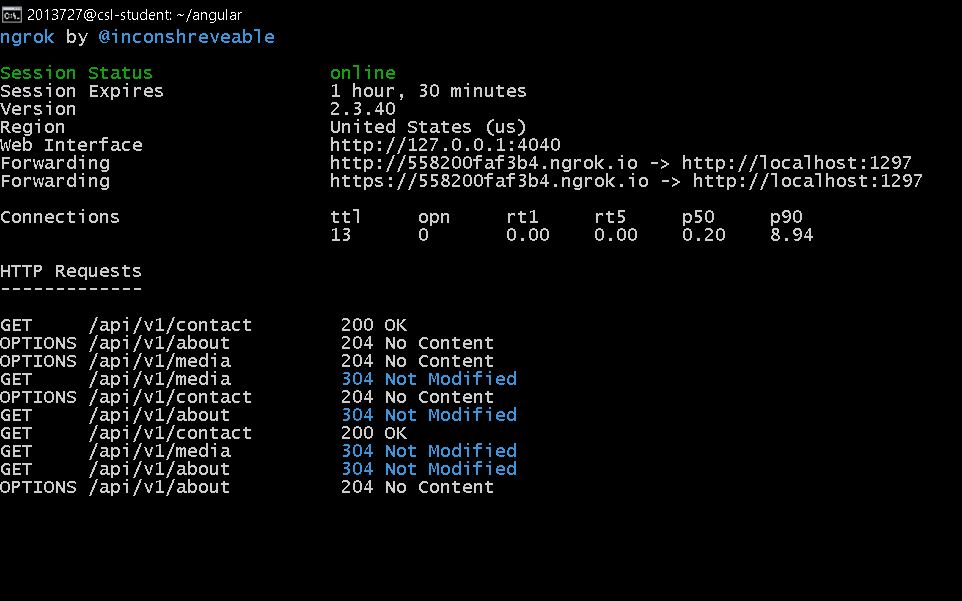
In terminal following commands:-

1. git config --global user.name "2013727"
2. git config --global user.email [m.maddipati@wlv.ac.uk](mailto:m.maddipati@wlv.ac.uk)
3. cd existing\_folder
4. git init
5. git remote add origin https://git-srv.wlv.ac.uk/gitlab/2013727/photography-angular.git
6. git add .
7. git commit -m "final code"



**Testing:-**

1. ngrok is also automatic testing tool which produce real-time web UI by HTTP traffic running over local server through tunnels. By changing the details in our website or CURD operation ngrok generate request and response details of parameters.



1. Postman is a fantastic tool for deconstructing RESTful APIs that have been created by others or for testing APIs that you have created yourself. It provides a simple user interface for making HTML queries, eliminating the need to write a large amount of code only to evaluate an API's operation. Consider the following scenario: I intended to send a GET call to a fan-made API for the video game Hearthstone in order to search for cards that have the word "archer" in their name. Instead of using Postman and instead writing code in something like Flask, I would have to create a completely new route and function to perform the request, then specify with more code what I wanted the response to look like, and finally either print out the response to the console or provide some other method of actually seeing the result of the request. Granted, I would have to write all of this anyway if I wanted to create a working app that used this API, but doing all of this to merely verify an API's operation is excessively onerous and time consuming when there is a tool like Postman available. A test of this kind becomes considerably simpler when using Postman. Simply entering the route into the URL bar, selecting GET as the response method from the selection box to its left, entering my API key in the “Headers” area, specifying that I want the response in “pretty” JSON format, and hitting submit is all it takes. After that, I get the return data in an easy-to-read JSON format, along with a status code of 200, certifying that the GET request was successfully accomplished. That is all there is to it!

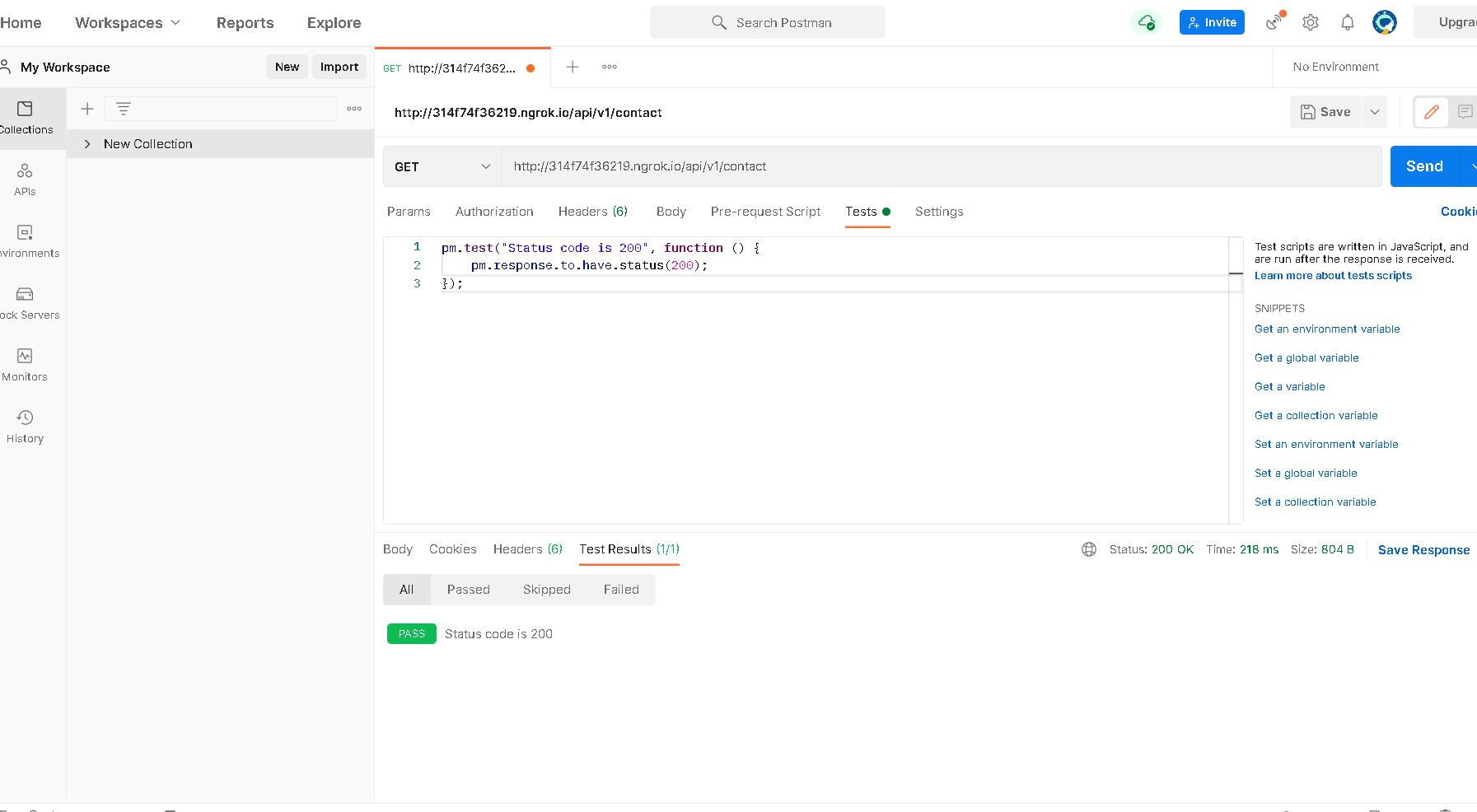
In postman paste following command in tests scripting.

pm.**test**("Status code is 200", **function** () {

    pm.response.to.have.status(200);

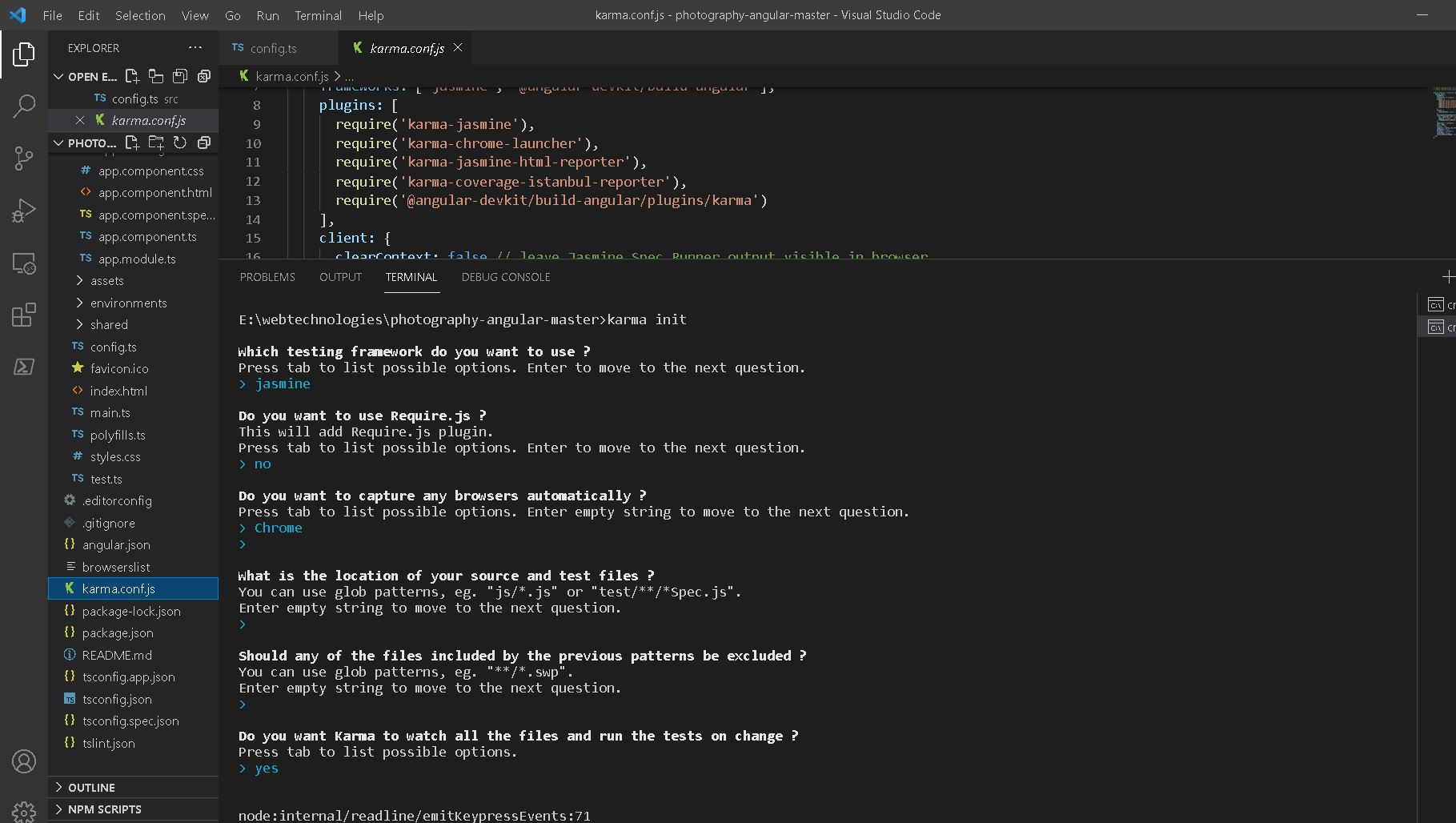
});

When you send or do any CRUD operations it will update the status.



In postman we can do all the CRUD operation from postman by calling the RESTful API’s. In this portal I created total 6 API’s. Where all the APIs are similar format.

1. The ability to conduct tests against actual browsers and actual devices is the primary selling point of Karma, since it makes your testing approach more robust and dependable. Because various browsers might have different DOM implementations, testing your application against all of them—or at the very least against the majority of the important ones—is crucial if you want to assure that your application will operate correctly for the vast majority of your customers. Furthermore, we live in an age in which not only a large number of browsers, but also a large number of devices are available. Phones, tablets, you name it, they are all available. Karma also gives you the option of running tests against them.



Karma also makes it simple for developers to run their tests continually without having to leave their terminals or integrated development environments (IDEs), since it can re-run the tests every time the files change.