FloBiz

SE Internship Assignment

Contents

Anubhav Dhuliya

ad953@snu.edu.in

9899863458

Front-End Philosophy

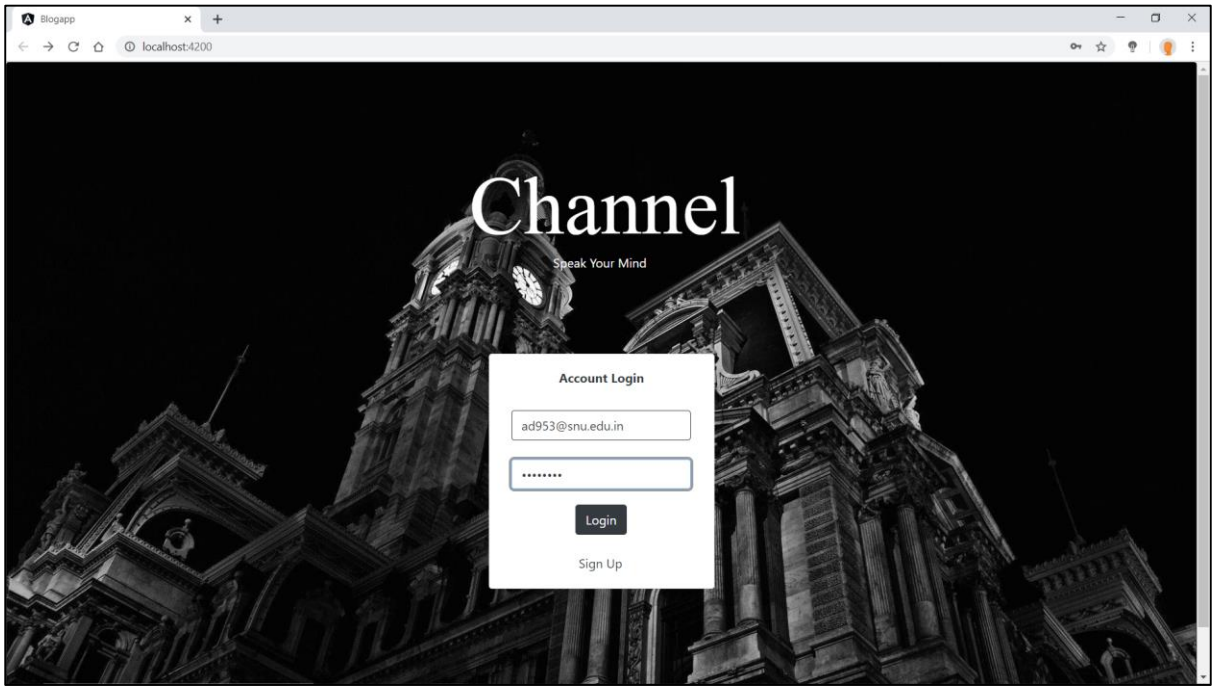
I have intentionally implemented a very simple and minimalistic user interface so that it's easier to use our services across all physical platforms attracting users from all walks of life.

My philosophy behind Channel was to make the services intuitive so that creators can focus on what's important, the content, and the gimmicks would never reach them.

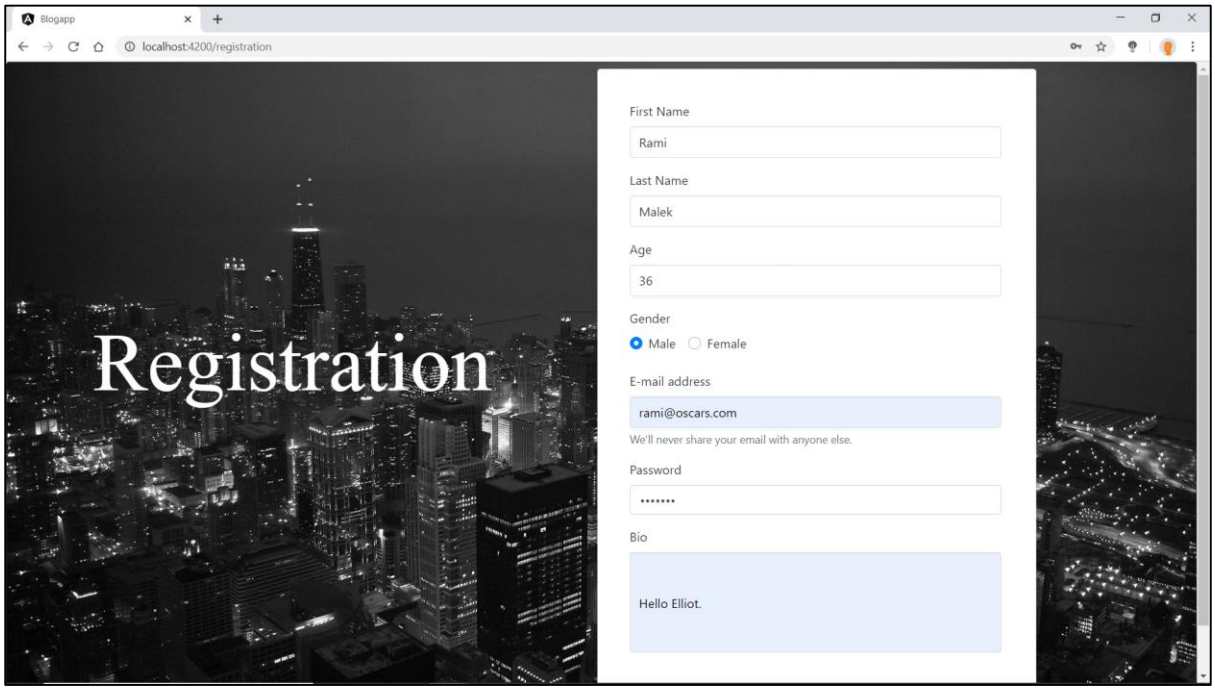
I've kept the pages very simple and focussed on enabling the user to customize their articles/blogs. It was implemented using Angular and Bootstrap.

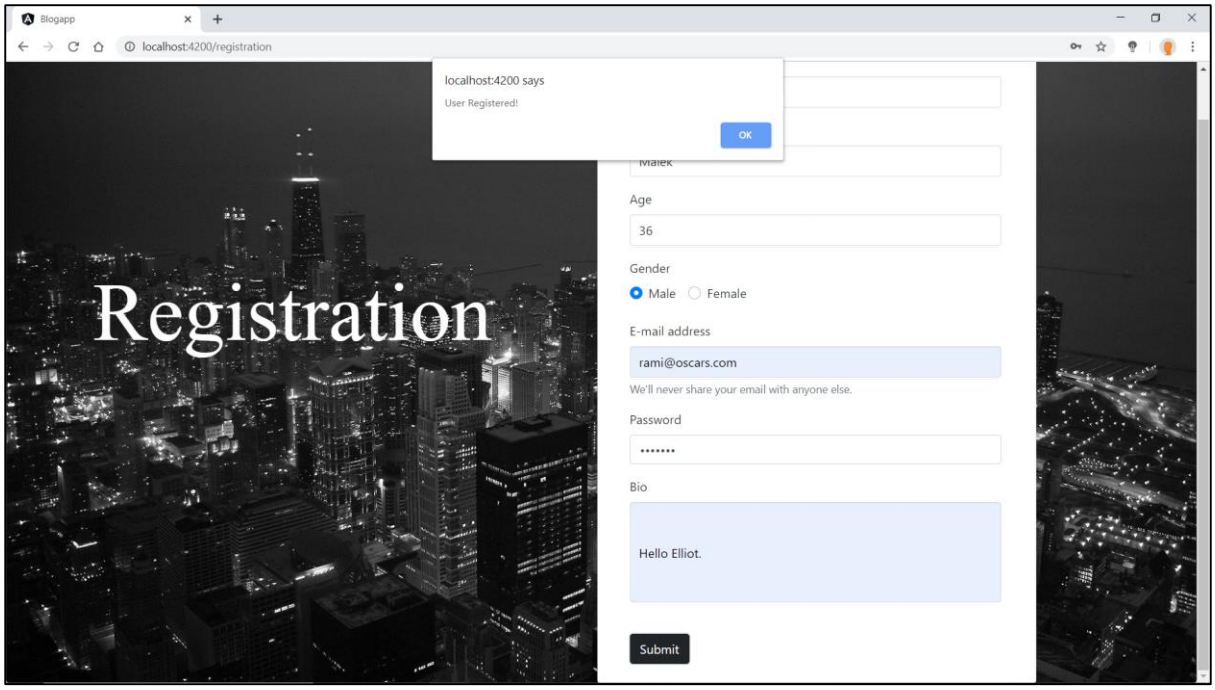
Front-End Implementation

1. **Login Page**

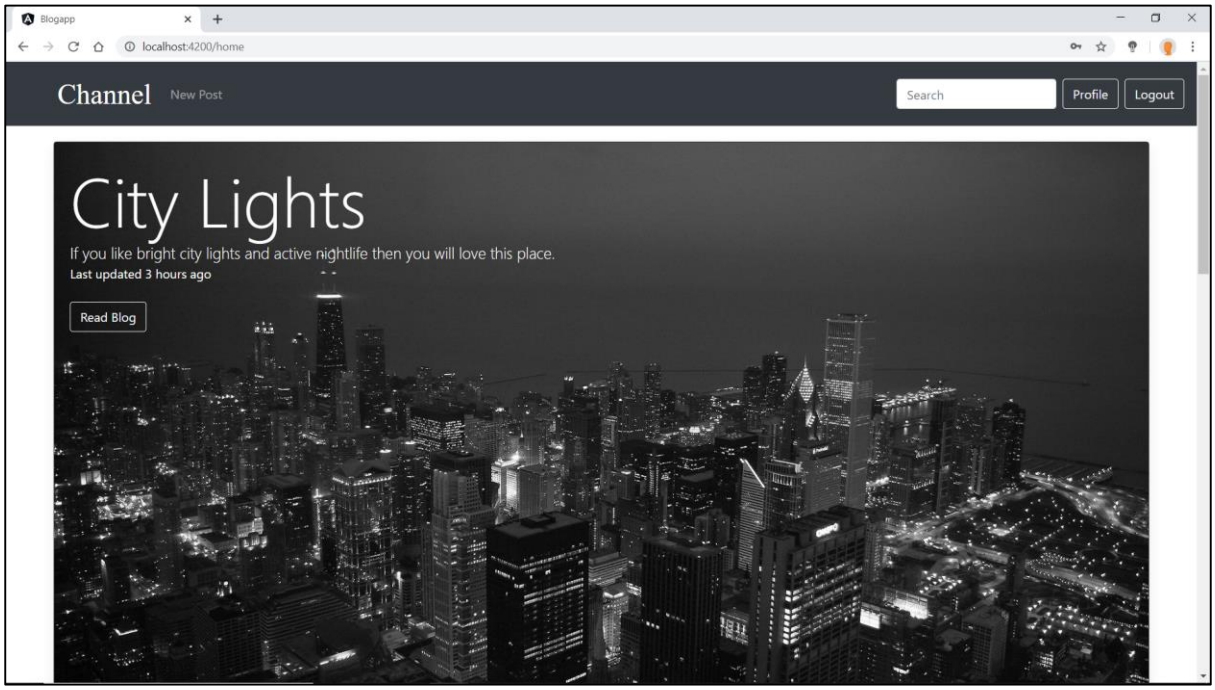


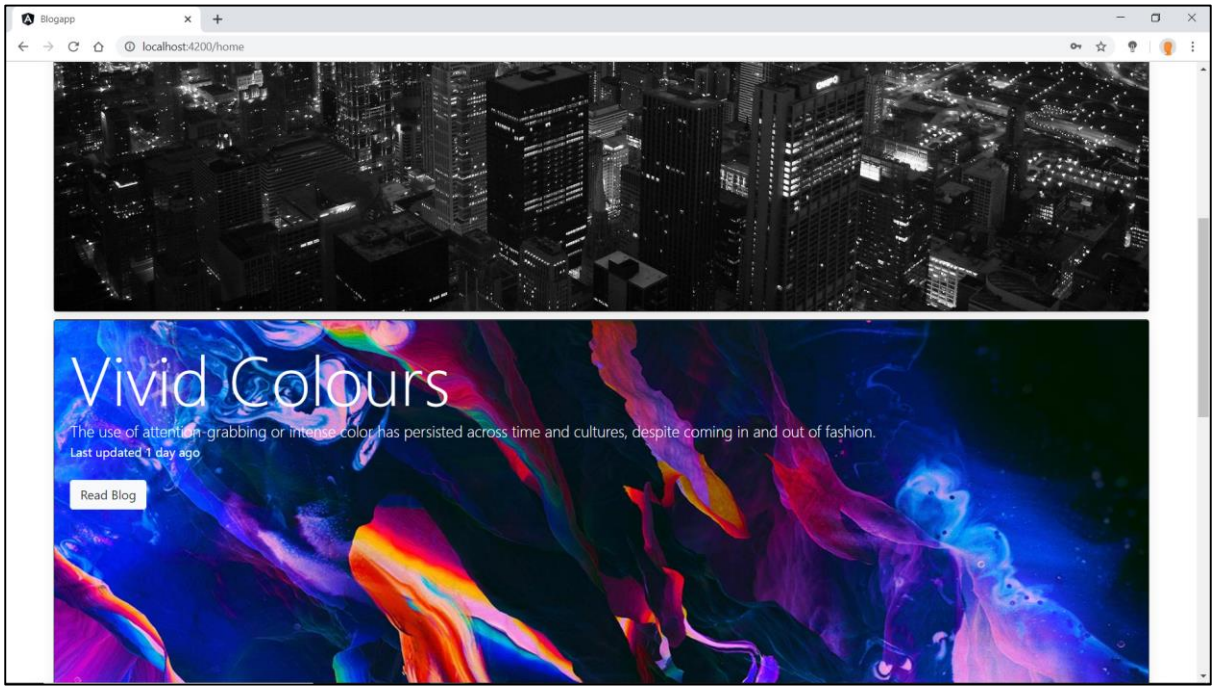
2. Registration Page



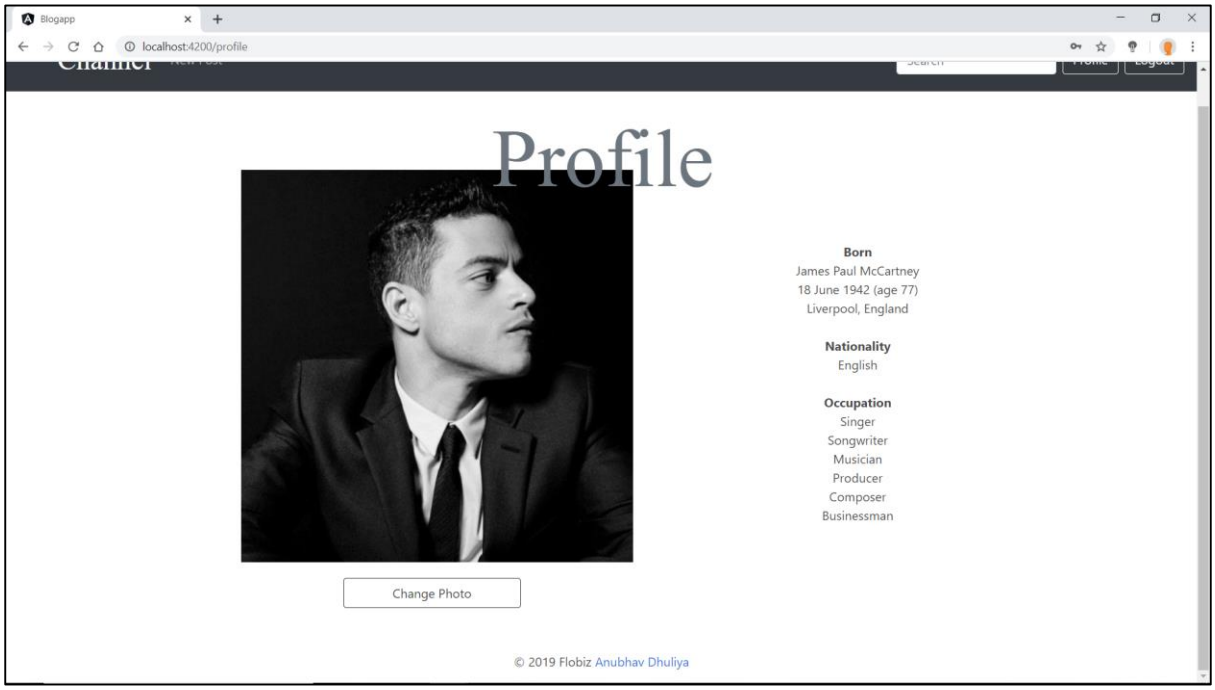


3. Home/Feed

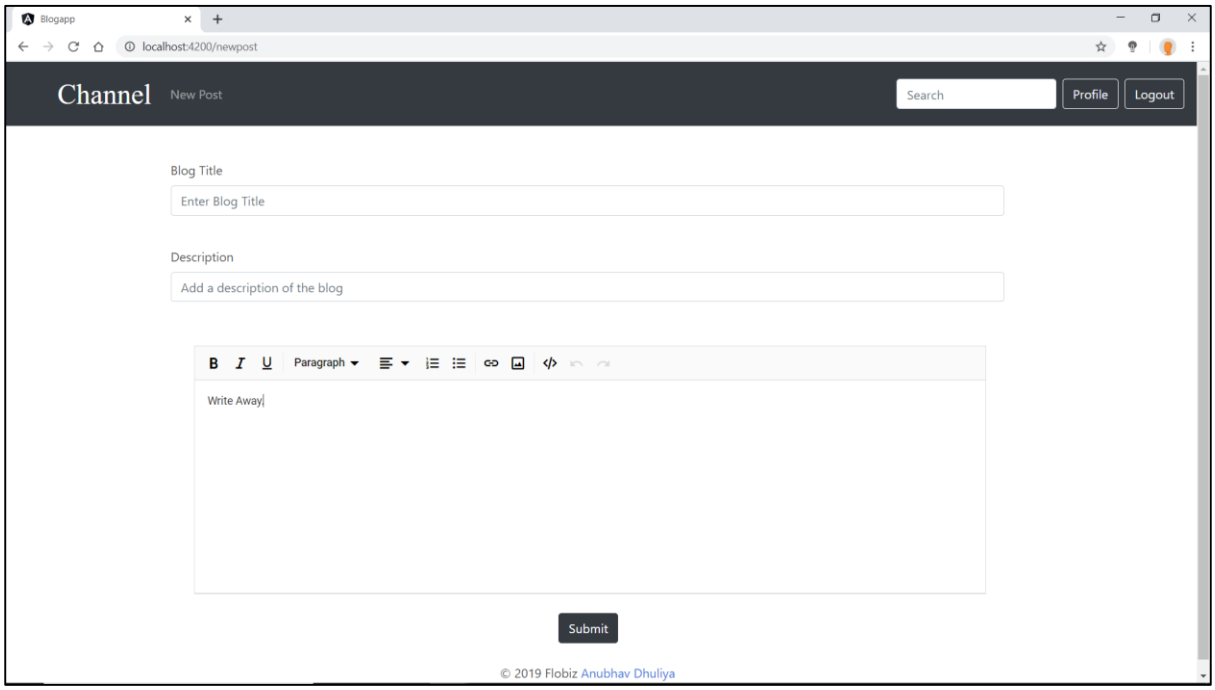

Blogapp
localhost:4200/home
Channel
New Post
Search
Profile
Logout
City Lights
If you like bright city lights and active nightlife then you will love this place.
Last updated 3 hours ago
Read Blog

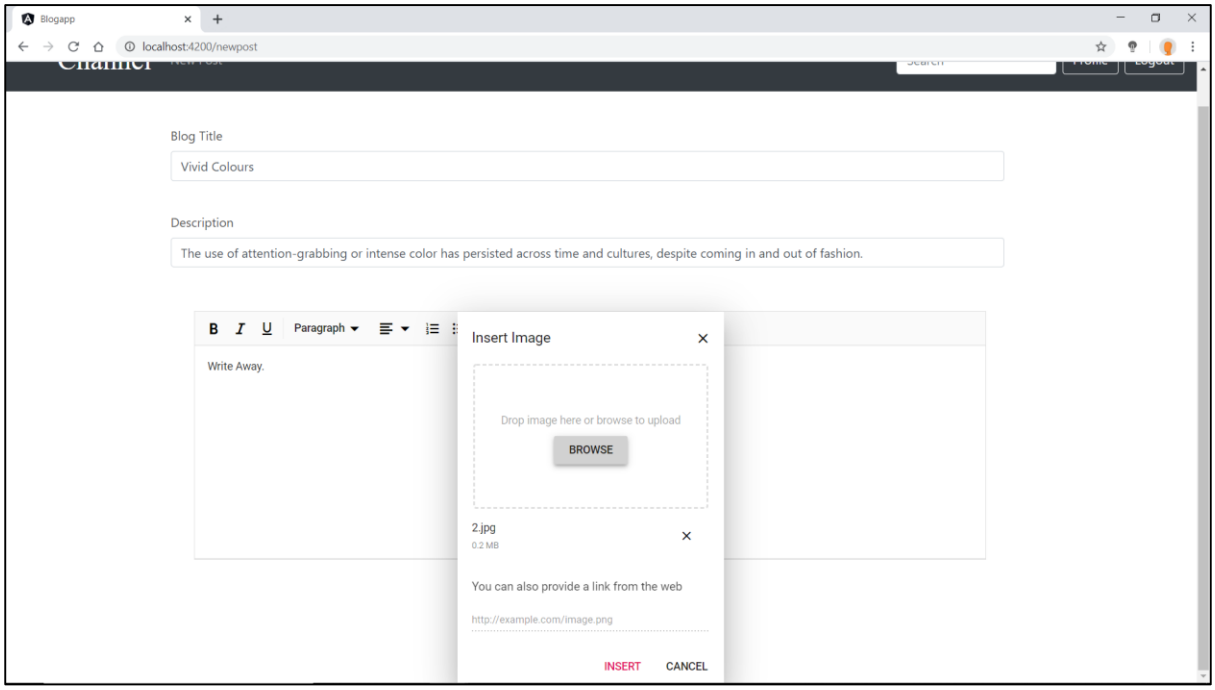

Blogapp
localhost:4200/home
Vivid Colours
The use of attention-grabbing or intense color has persisted across time and cultures, despite coming in and out of fashion.
Last updated 1 day ago
Read Blog

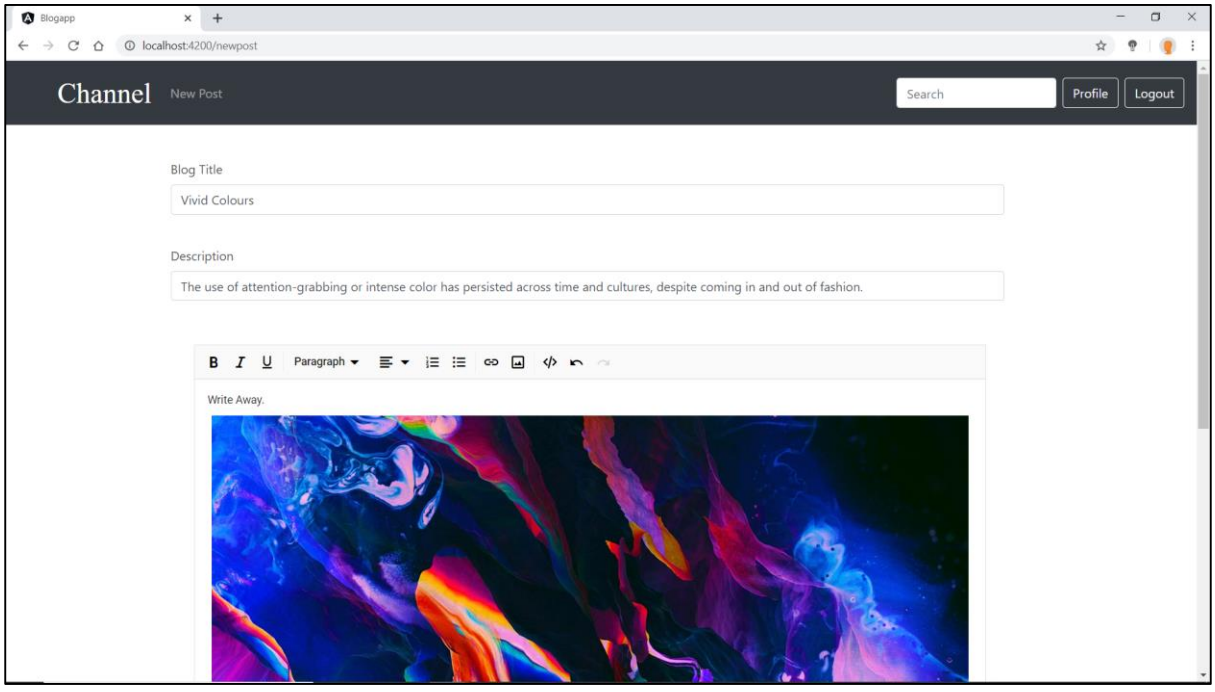
4. Profile



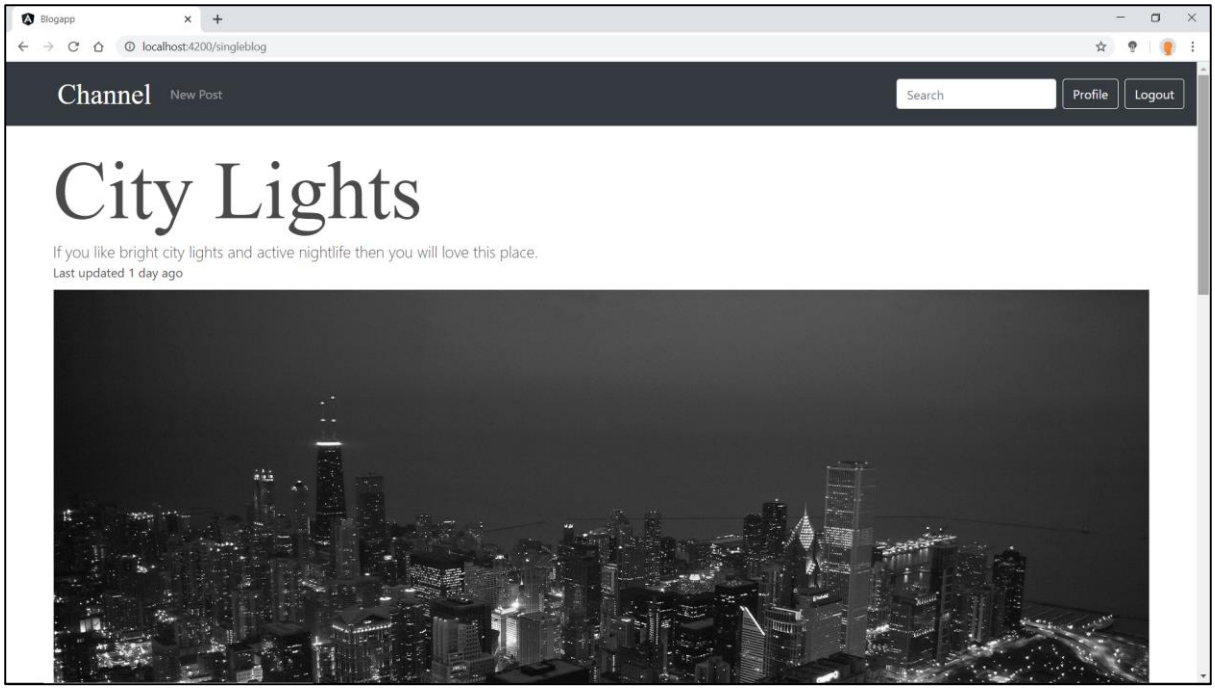
5. New Post

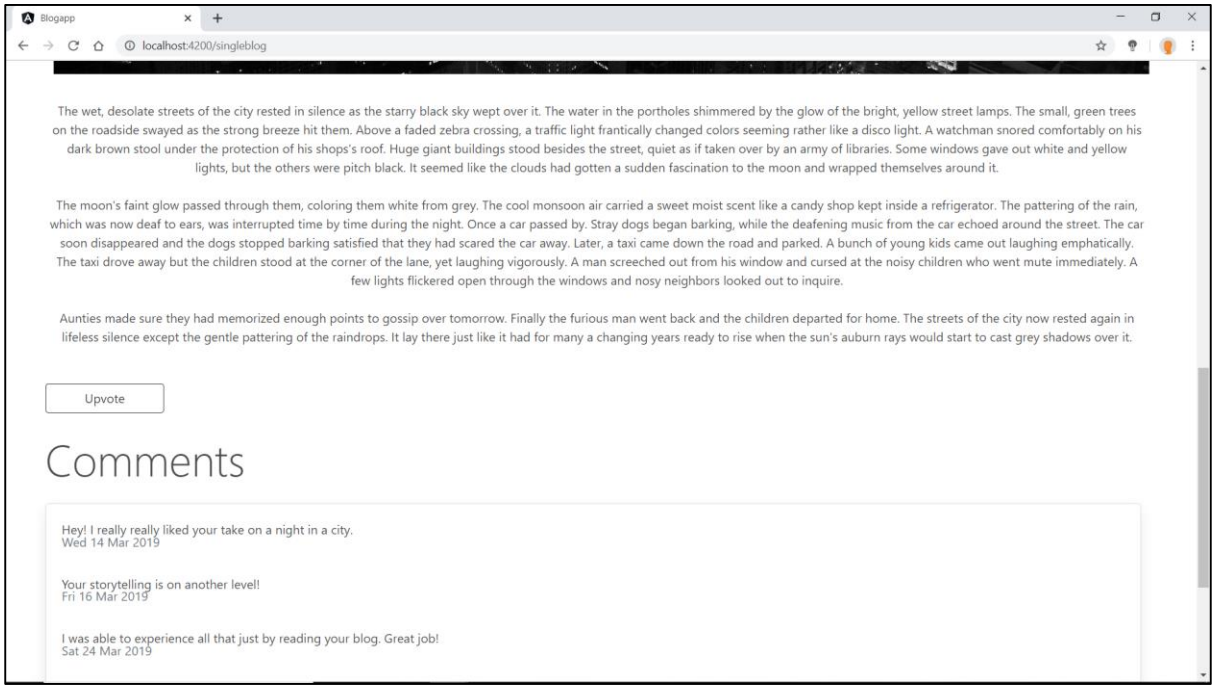


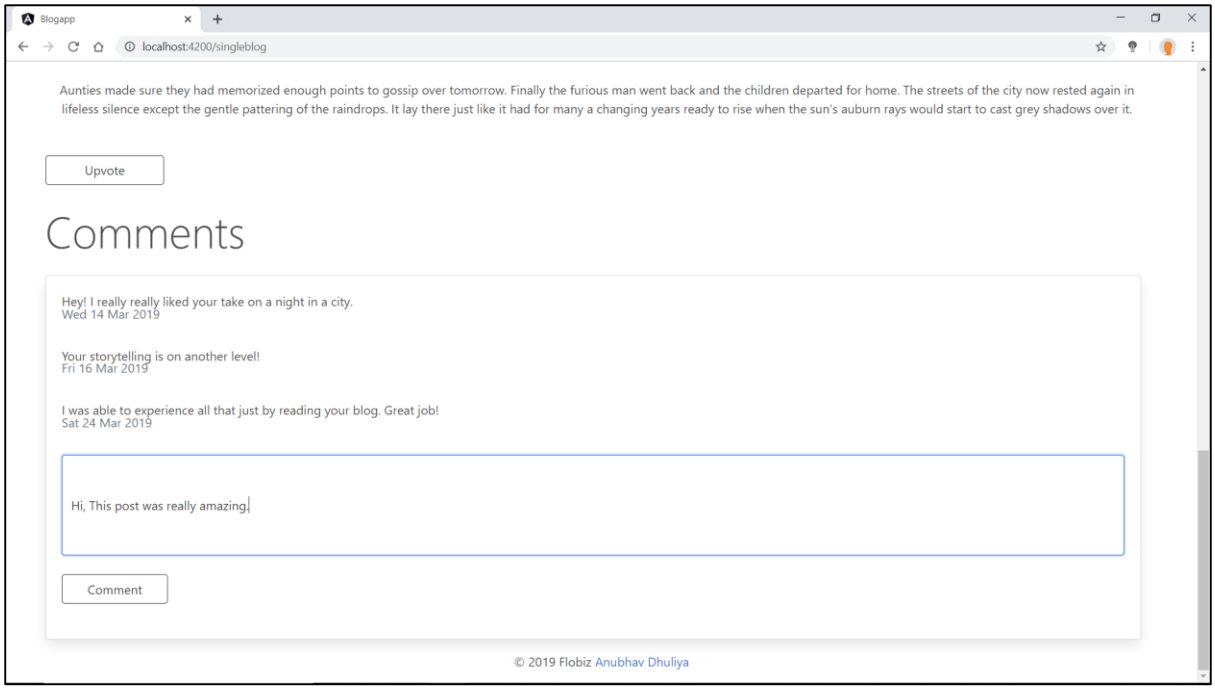

Blogapp
localhost:4200/newpost
Blog Title
Vivid Colours
Description
The use of attention-grabbing or intense color has persisted across time and cultures, despite coming in and out of fashion.
Paragraph
Write Away.
Insert Image
Drop image here or browse to upload
BROWSE
2.jpg
0.2 MB
You can also provide a link from the web
http://example.com/image.png
INSERT
CANCEL


Blogapp
localhost:4200/newpost
Channel
New Post
Search
Profile
Logout
Blog Title
Vivid Colours
Description
The use of attention-grabbing or intense color has persisted across time and cultures, despite coming in and out of fashion.
Paragraph
Write Away.

6. Single Blog



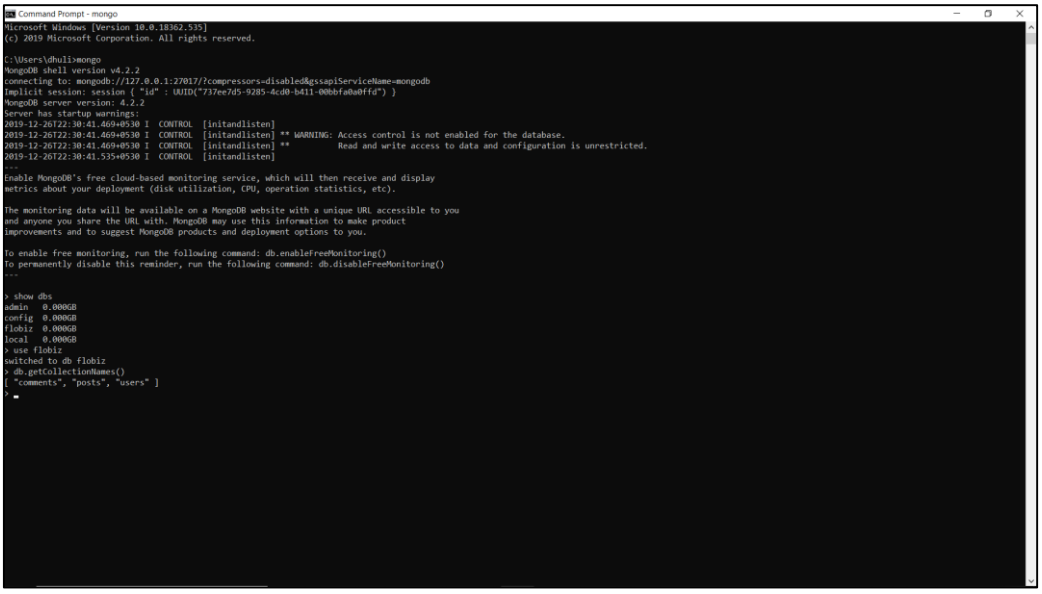




Back-End Implementation

The APIs were made using NodeJs and were distributed with regard to the resources, i.e. users, posts and, comments.

I used Express as the server framework. The database program was MongoDB.



API Testing Document

Users API

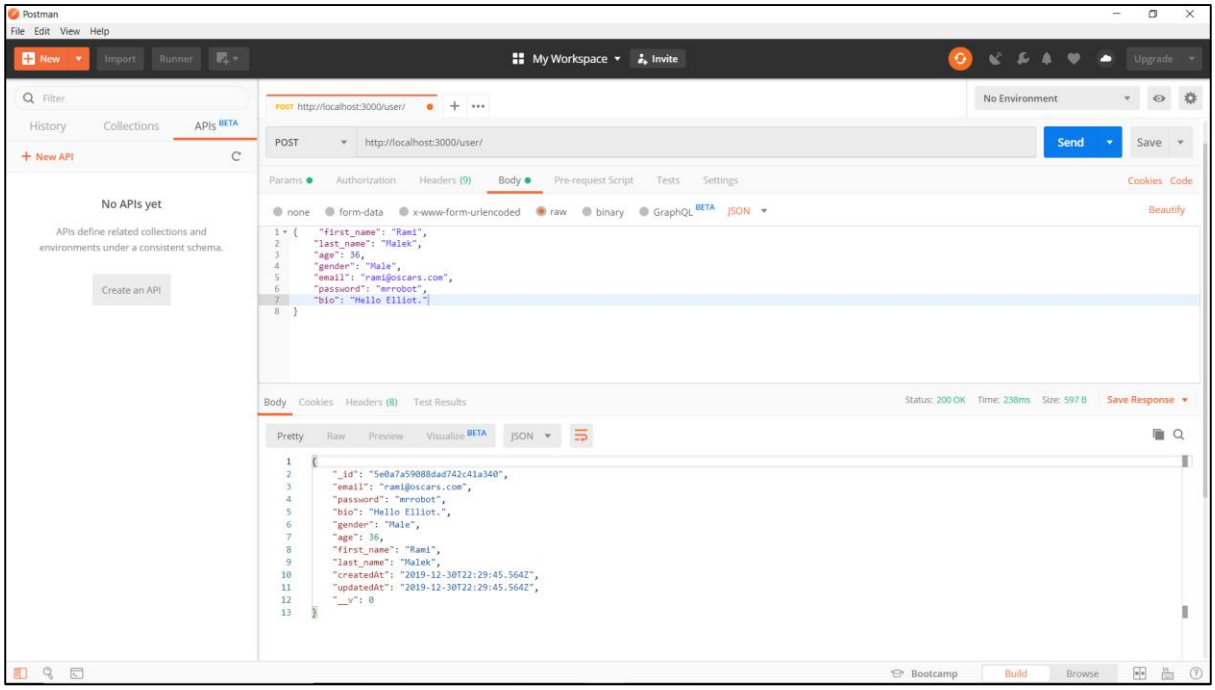
1. **POST /user/**

Creates a user.

Model Inputs :

first_name: String
last_name: String
age: Number
gender: String
email: String
password: String
bio: String

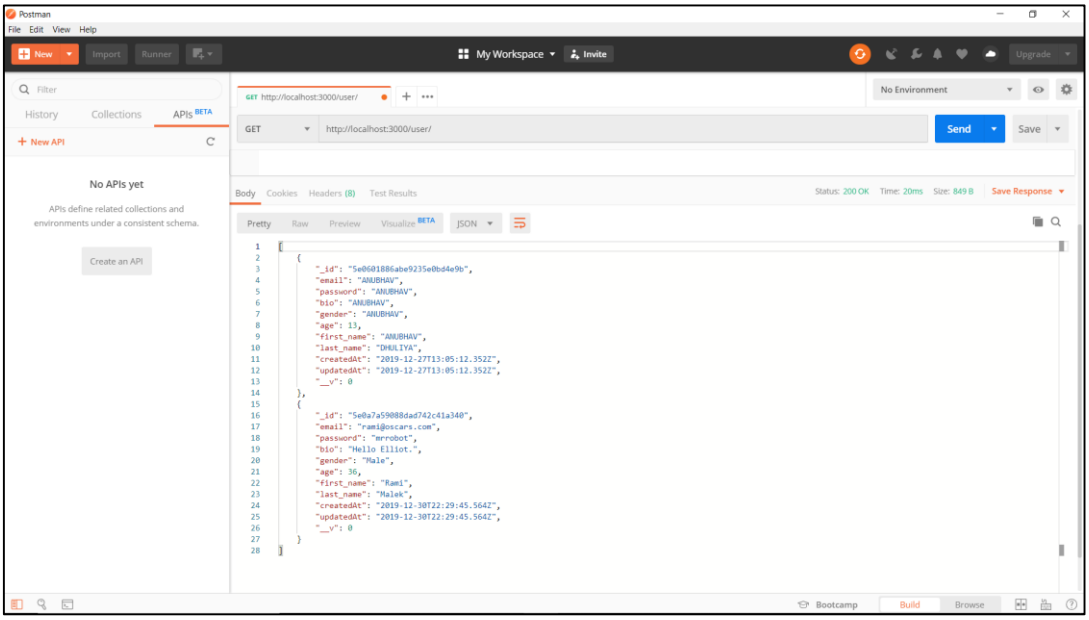
Output :



2. **GET /user/**

Fetches all the users.

Output :

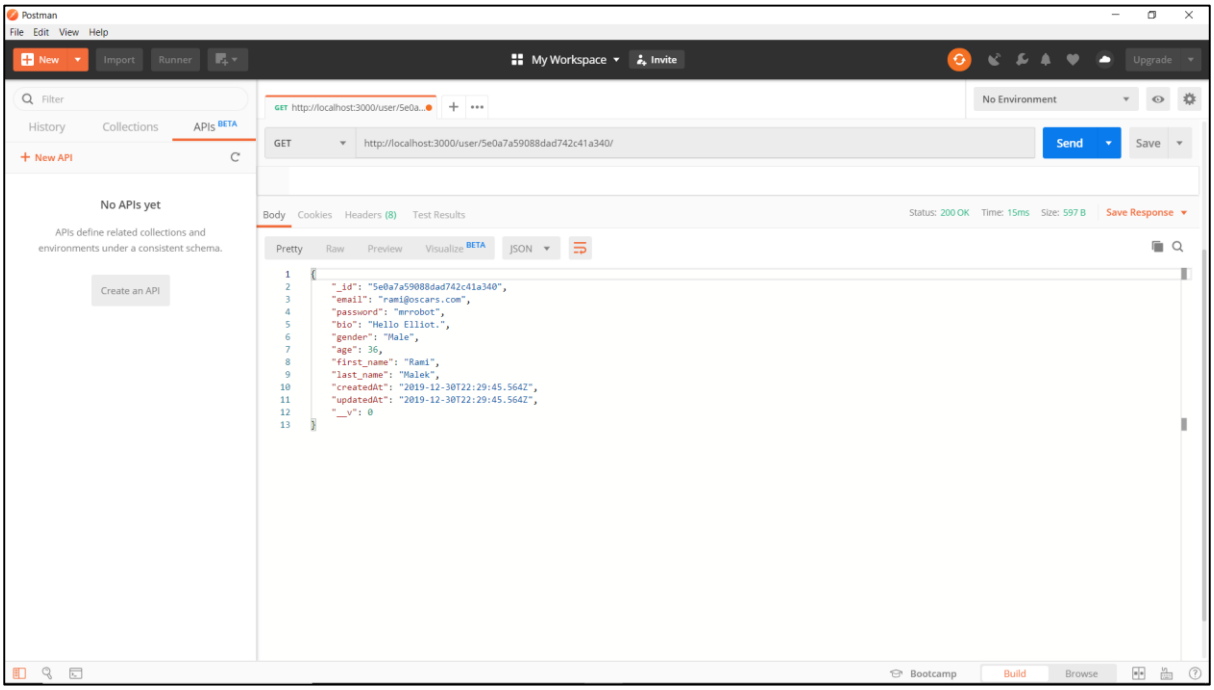


3. **GET /:userId**

Fetches a particular user.

Params : userId //the user's id generated by MongoDB

Output :



4. **PUT /:userId**

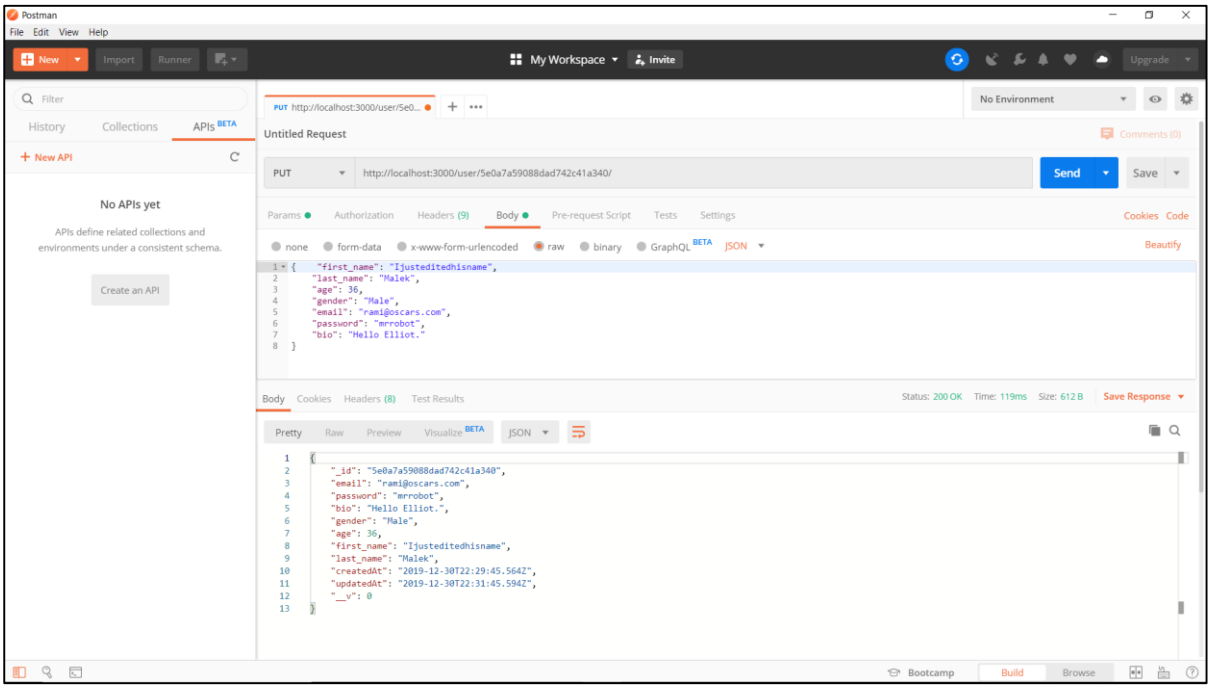
Updating a user's information.

Params : userId //the user's id generated by MongoDB.

Model Input :

first_name: String
last_name: String
age: Number
gender: String
email: String
password: String
bio: String

Output :

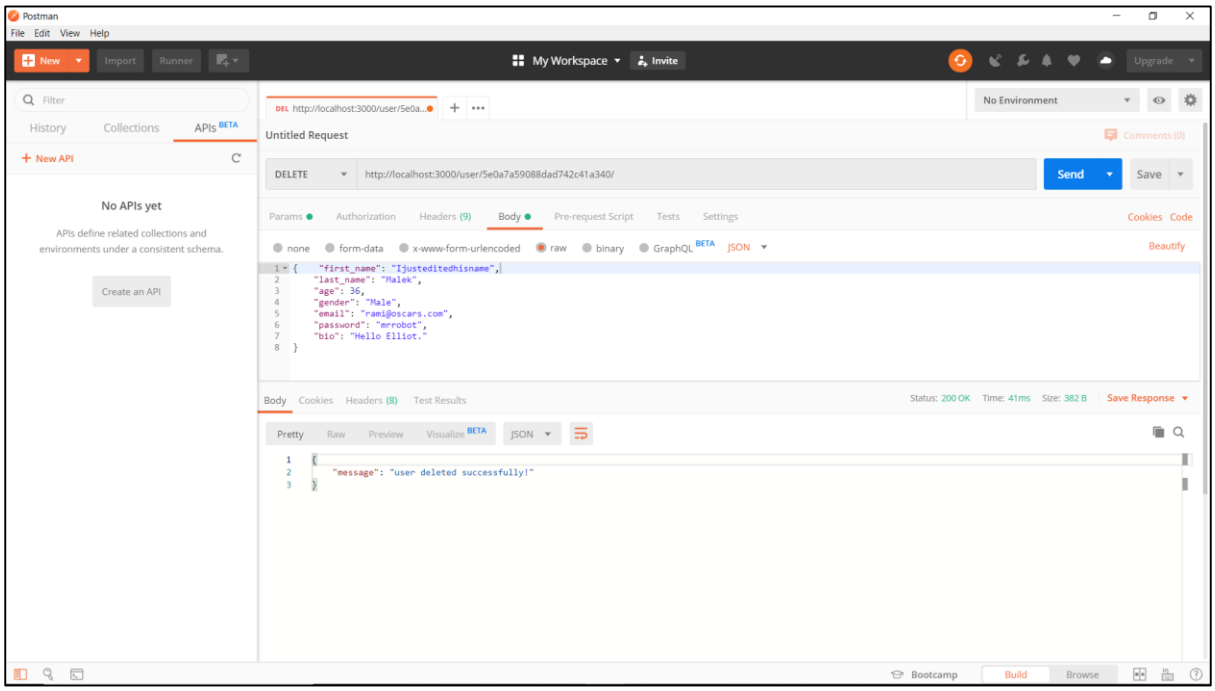


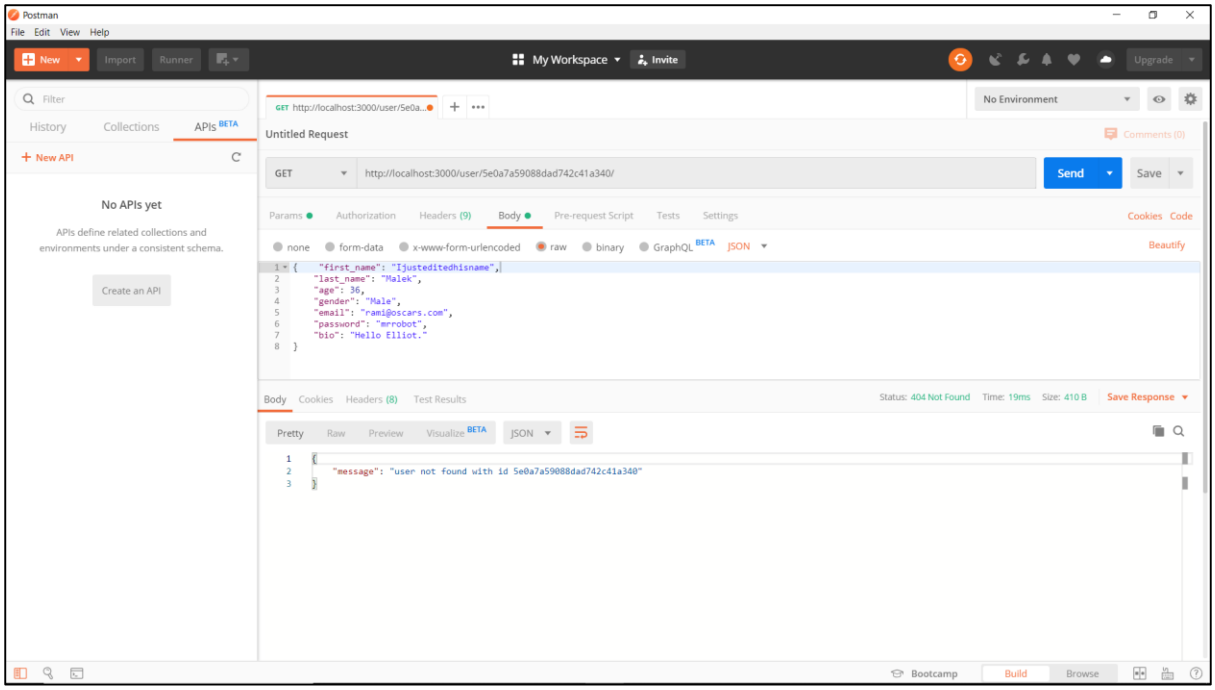
5. **DELETE /:userId**

Deletes a user.

Params : userId //the user's id generated by MongoDB.

Output :





Posts API

1. **POST /user/:userId/posts/**

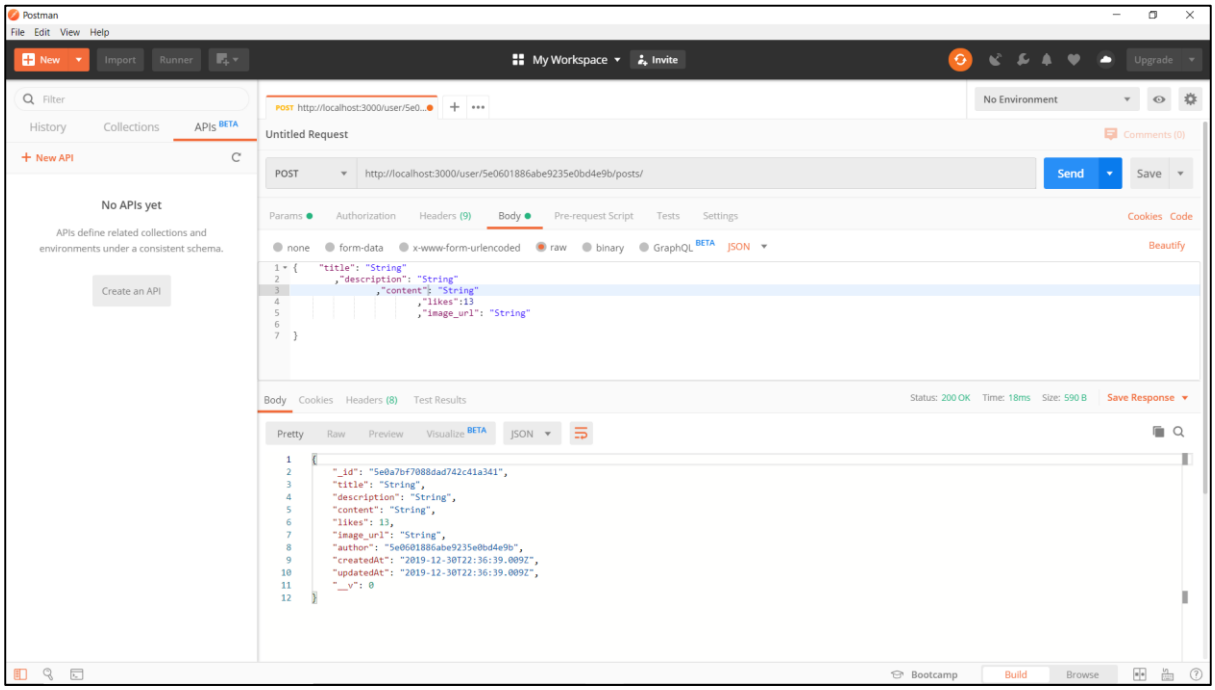
Creates a post/blog for a user.

Params : userId //the user's id generated by MongoDB.

Model Input :

title: String
description: String
content: String
likes: Number //sent over after aggregating all the likes through MongoDB
image_url: String
author: String //userId of the creator.

Output :

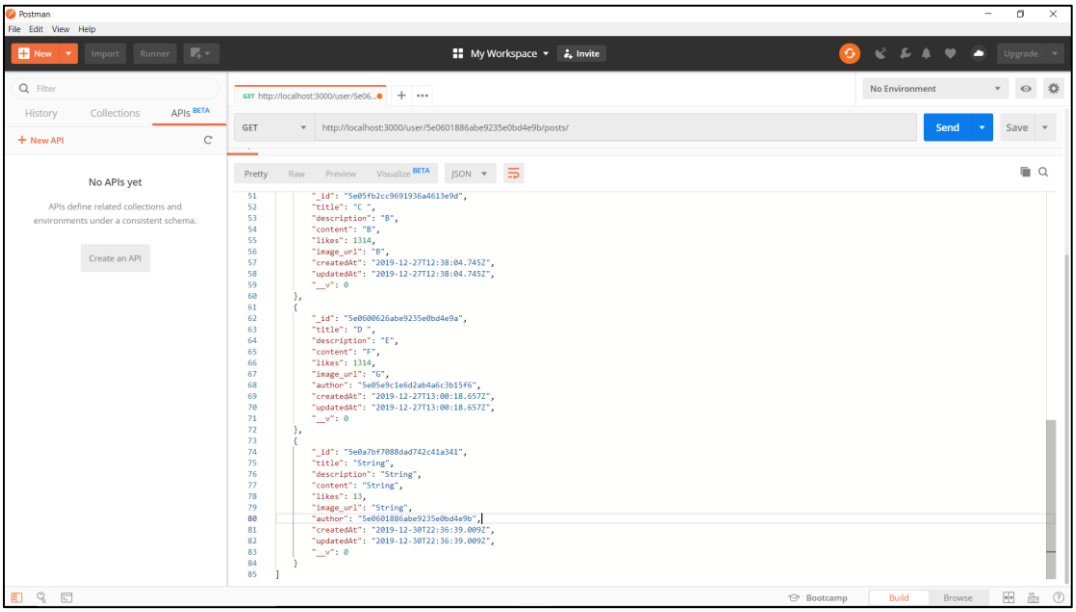


2. **GET /user/:userId/posts/**

Fetches all the posts/blogs by a particular user.

Params : userId //the user's id generated by MongoDB.

Output :

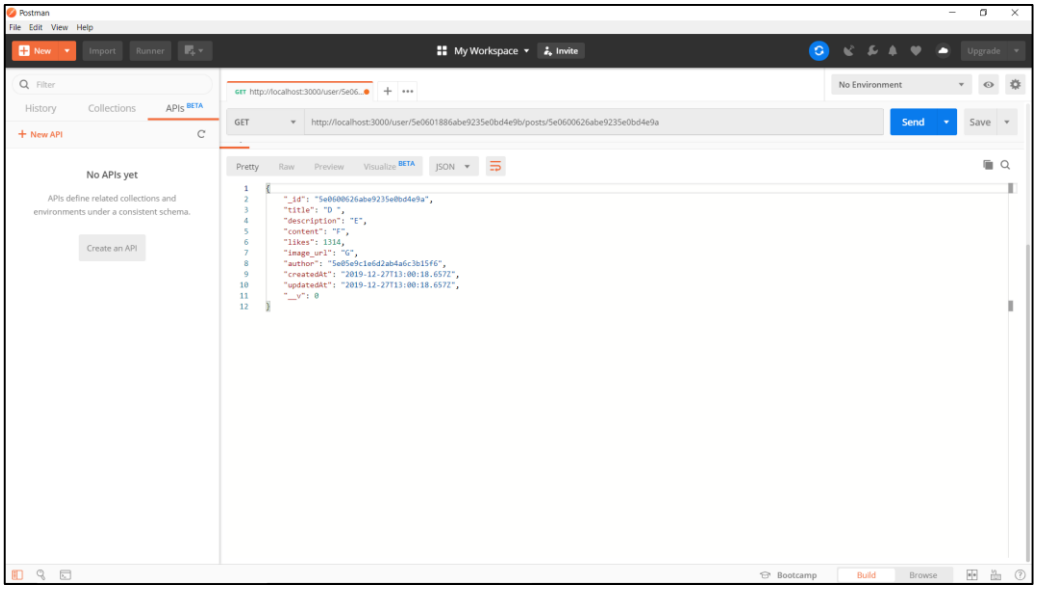


3. **GET /user/:userId/posts/:postId**

Fetches a particular post by a particular user.

Params : userId //the user's id generated by MongoDB.
postId //the post's id generated by MongoDB.

Output :



4. **PUT /user/:userId/posts/:postId**

Updates a particular post by a user.

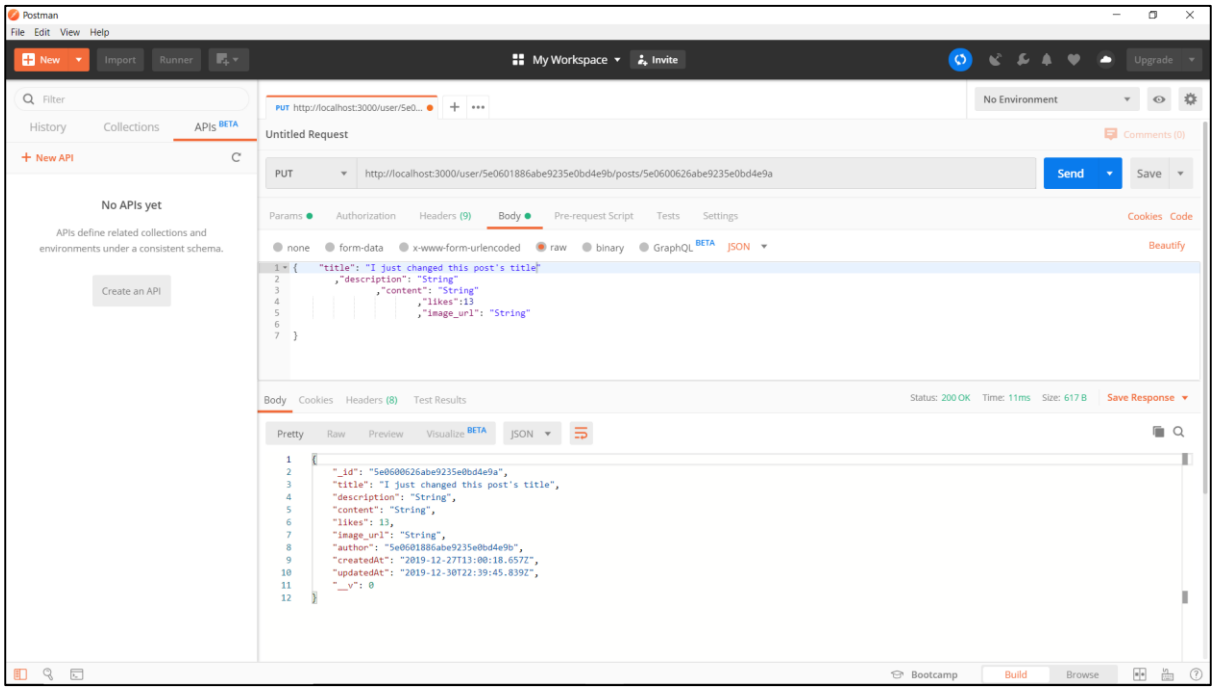
Params : userId //the user's id generated by MongoDB.
postId //the post's id generated by MongoDB.

Model Input :

title: String
description: String
content: String
likes: Number //sent over after aggregating all the likes through MongoDB

image_url: String
author: String //userId of the creator.

Output :

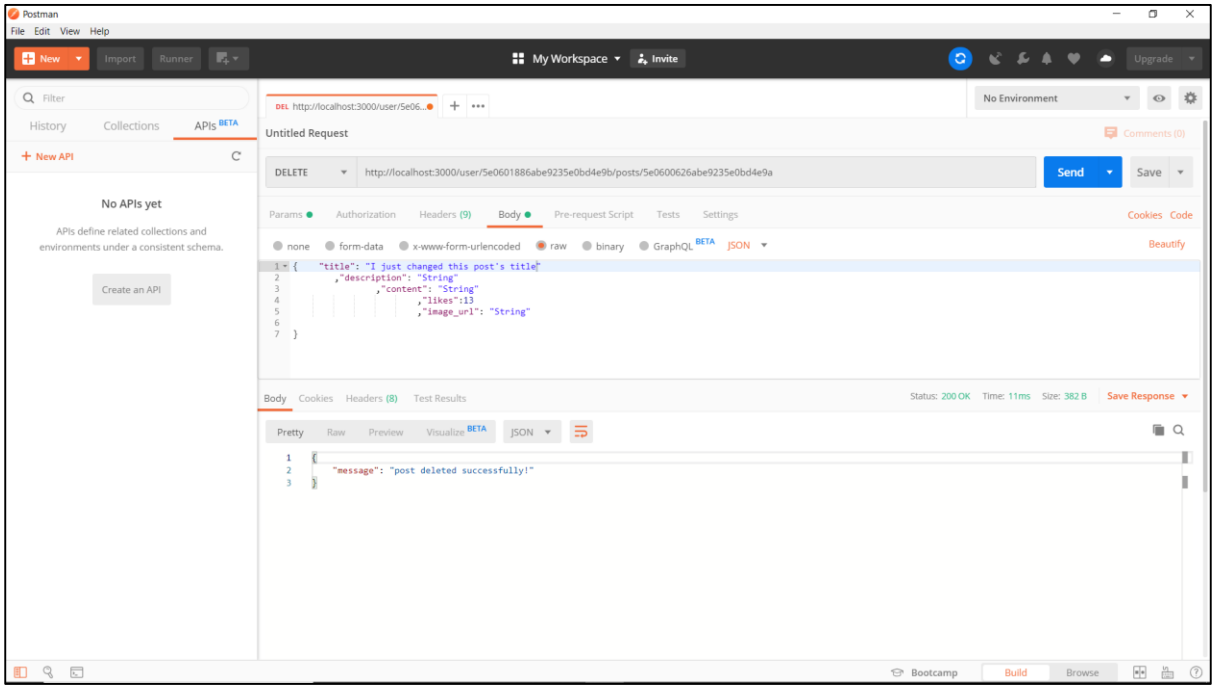


5. **DELETE /user/:userId/posts/:postId**

Deletes a post for a user.

Params : userId //the user's id generated by MongoDB.
postId //the post's id generated by MongoDB.

Output :



Comments API

1. **POST /user/:userId/posts/:postId/comments/**

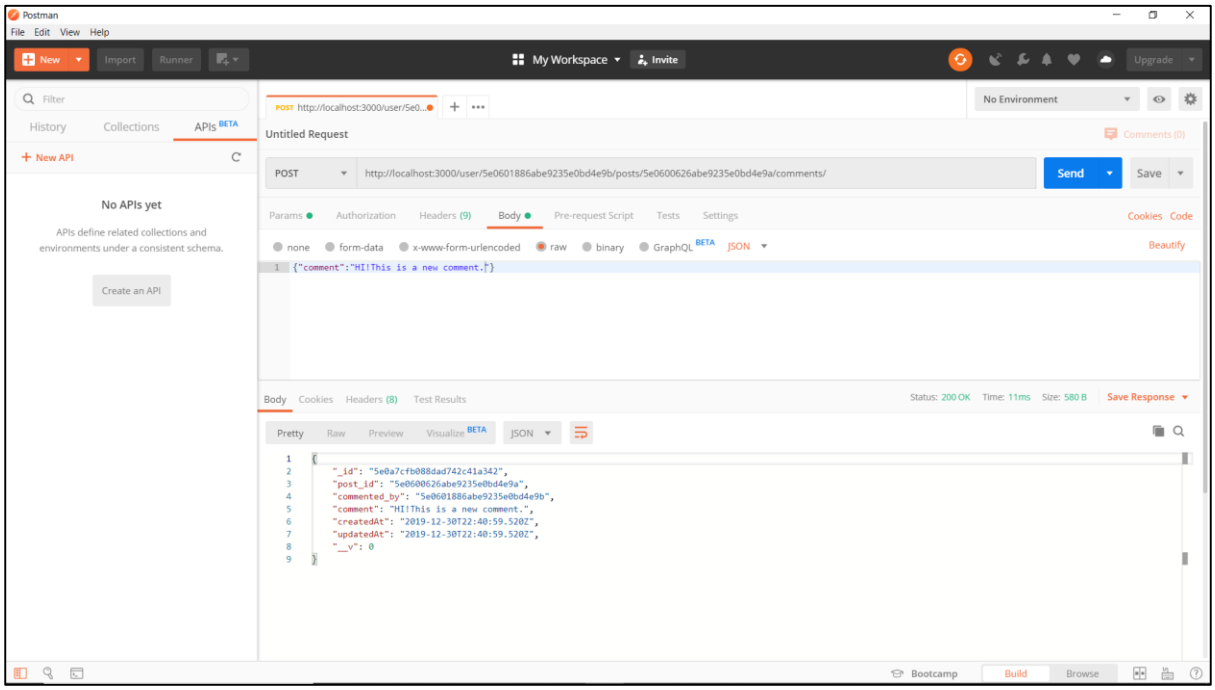
Creates a comment for a post.

Params : userId //the user's id generated by MongoDB.
postId //the post's id generated by MongoDB.

Model Input :

post_id: String//id of the post the comment belongs to
commented_by:String//id of the poster
comment: String

Output :

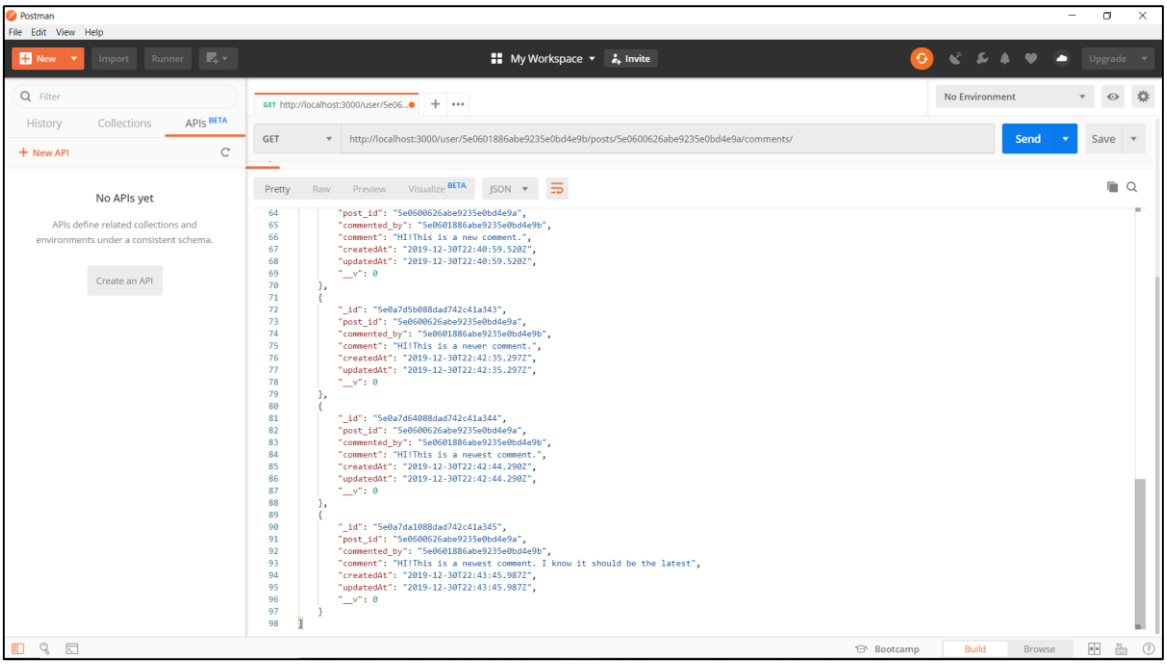


2. **GET /user/:userId/posts/:postId/comments**

Fetches all the comments on a post.

Params : userId //the user's id generated by MongoDB.
postId //the post's id generated by MongoDB.

Output :

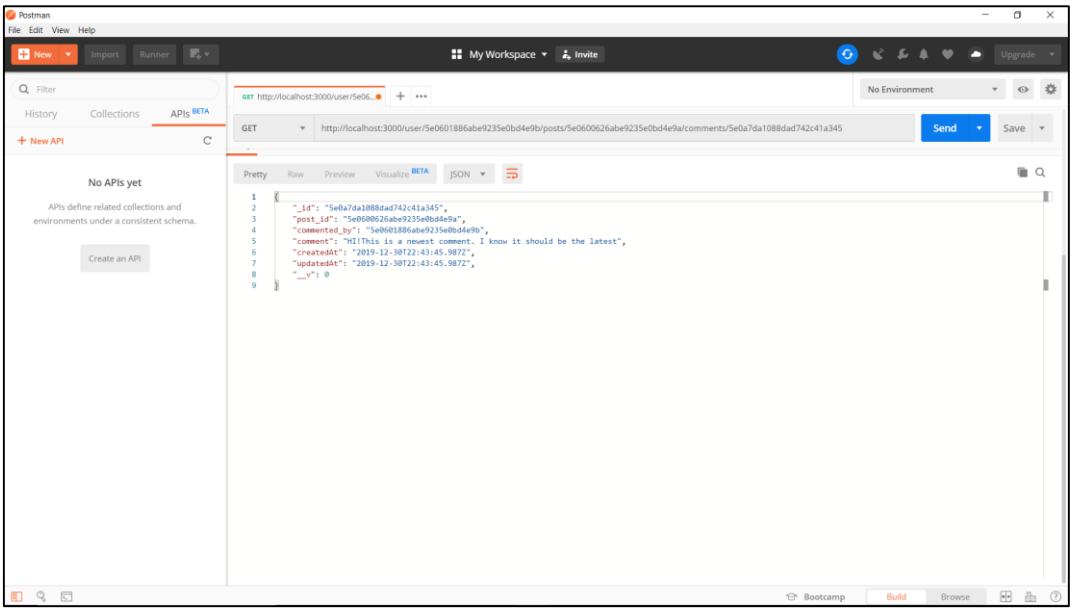


3. GET /user/:userId/posts/:postId/comments/:commentId

Fetches a particular comment.

Params : userId //the user's id generated by MongoDB.
postId //the post's id generated by MongoDB.
comentId //the comment's id generated by MongoDB.

Output :



4. **PUT /user/:userId/posts/:postId**

Updating a particular comment.

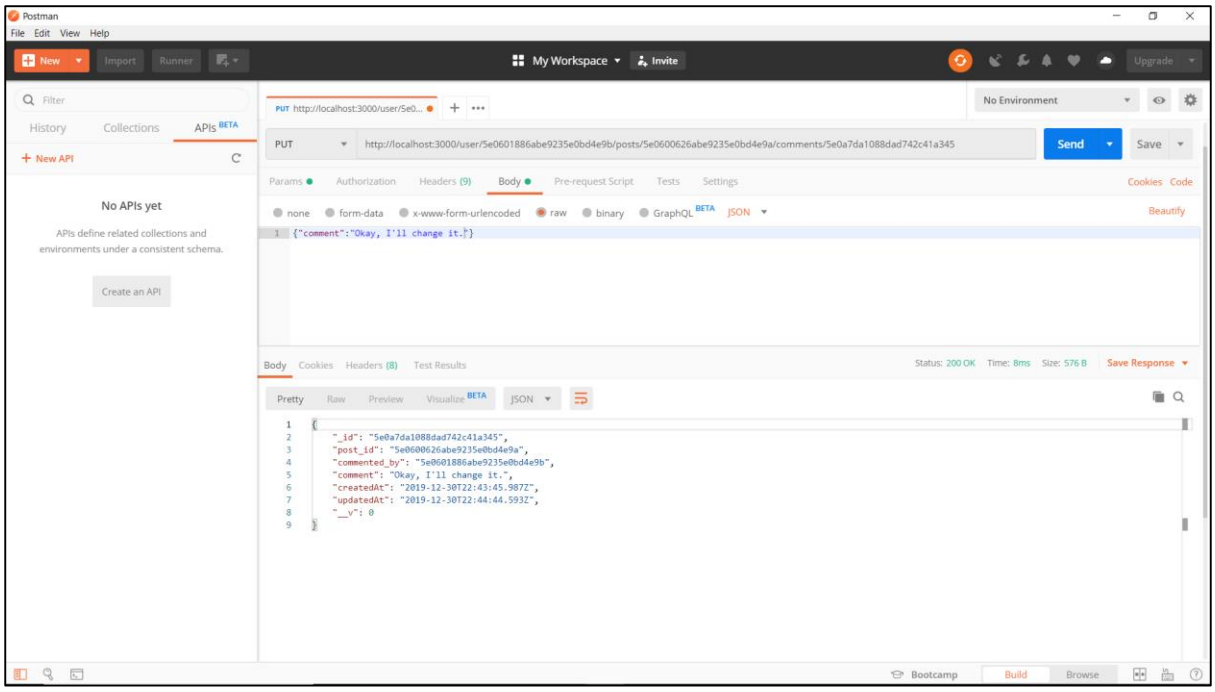
Params :

userId //the user's id generated by MongoDB.
postId //the post's id generated by MongoDB.
commentId //the comment's id generated by MongoDB.

Model Input :

post_id: String//id of the post the comment belongs to
commented_by:String//id of the poster
comment: String

Output :



5. **DELETE /user/:userId/posts/:postId**

Deletes a comment.

Params : userId //the user's id generated by MongoDB.
postId //the post's id generated by MongoDB.
commentId //the comment's id generated by MongoDB.

Output :