

FloBiz

SE Internship Assignment

Contents

1. Front-End Philosophy
2. Front-End Implementation
3. Back-End Implementation
4. API Testing Document

Anubhav Dhuliya

ad953@snu.edu.in

9899863458

Front-End Philosophy

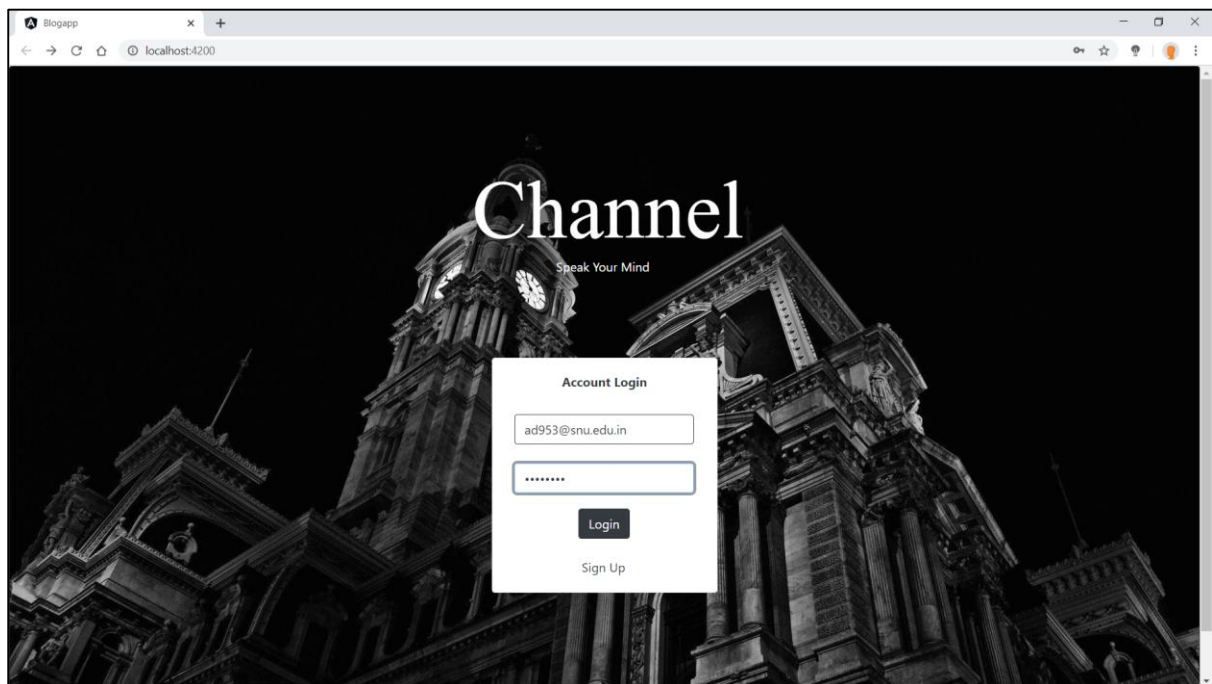
I have intentionally implemented a very simple and minimalistic user interface so that it's easier to use our services across all physical platforms attracting users from all walks of life.

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

Registration

First Name
Rami

Last Name
Malek

Age
36

Gender
☒ Male ☐ Female

E-mail address
rami@oscars.com

We'll never share your email with anyone else.

Password

Bio
Hello Elliot.

localhost:4200 says
User Registered!

OK

Registration

First Name
Rami

Last Name
Malek

Age
36

Gender
☒ Male ☐ Female

E-mail address
rami@oscars.com

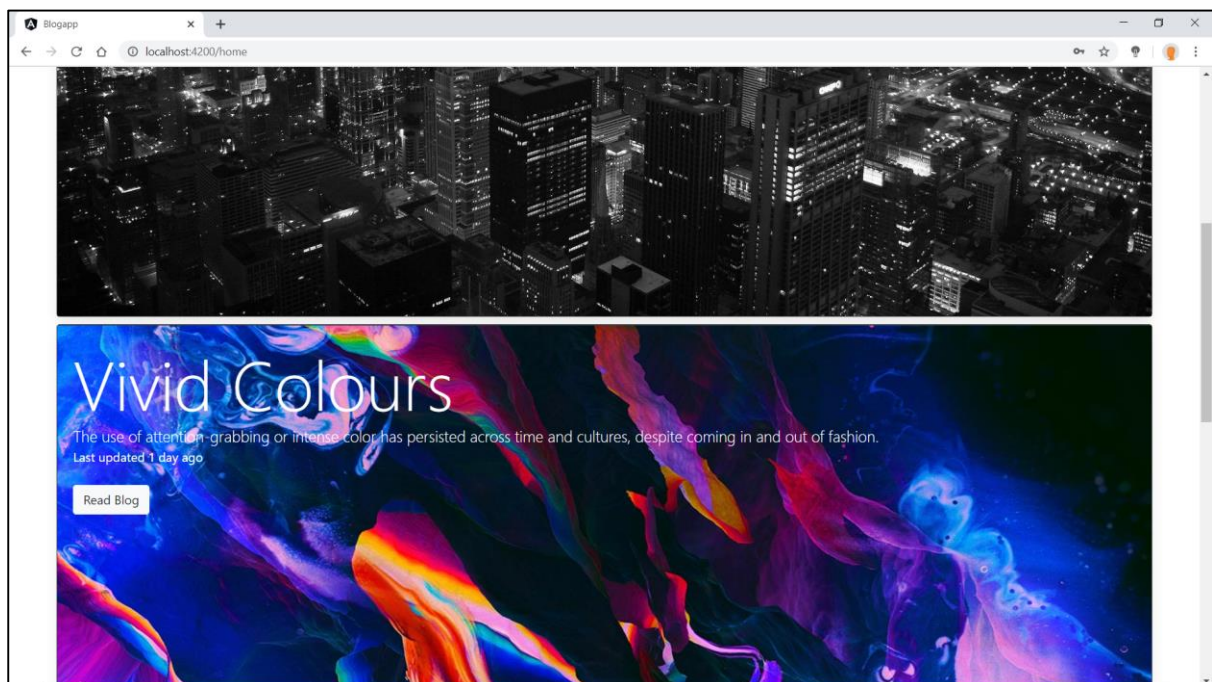
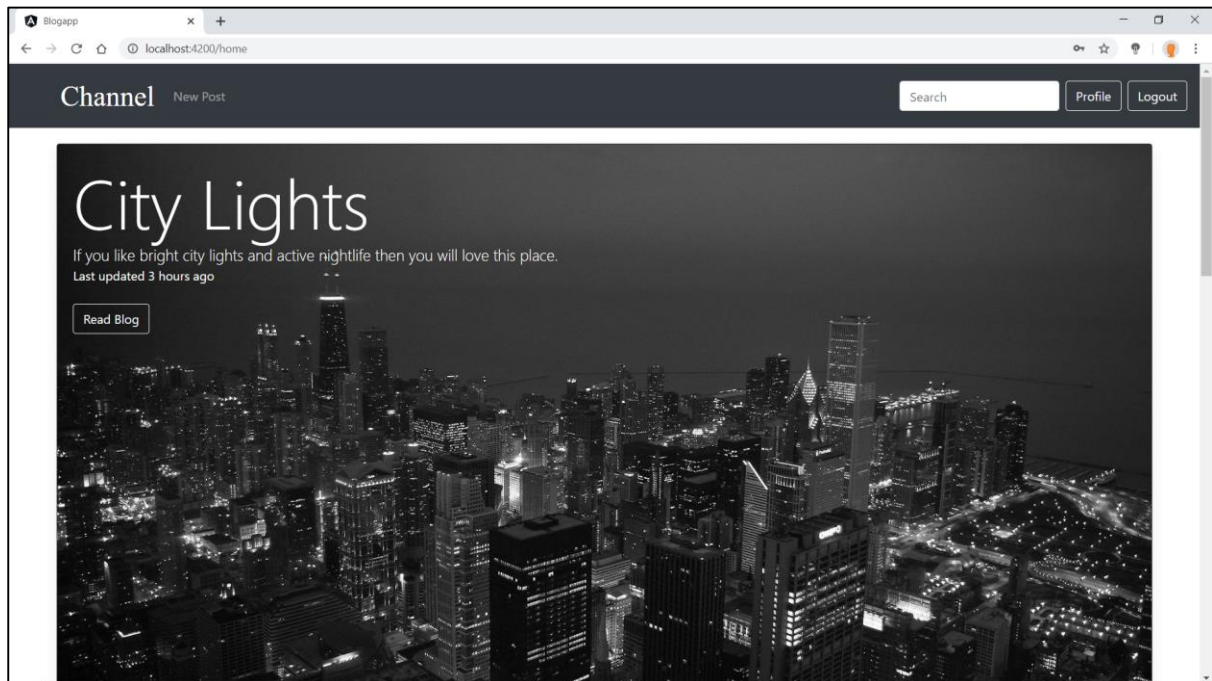
We'll never share your email with anyone else.

Password

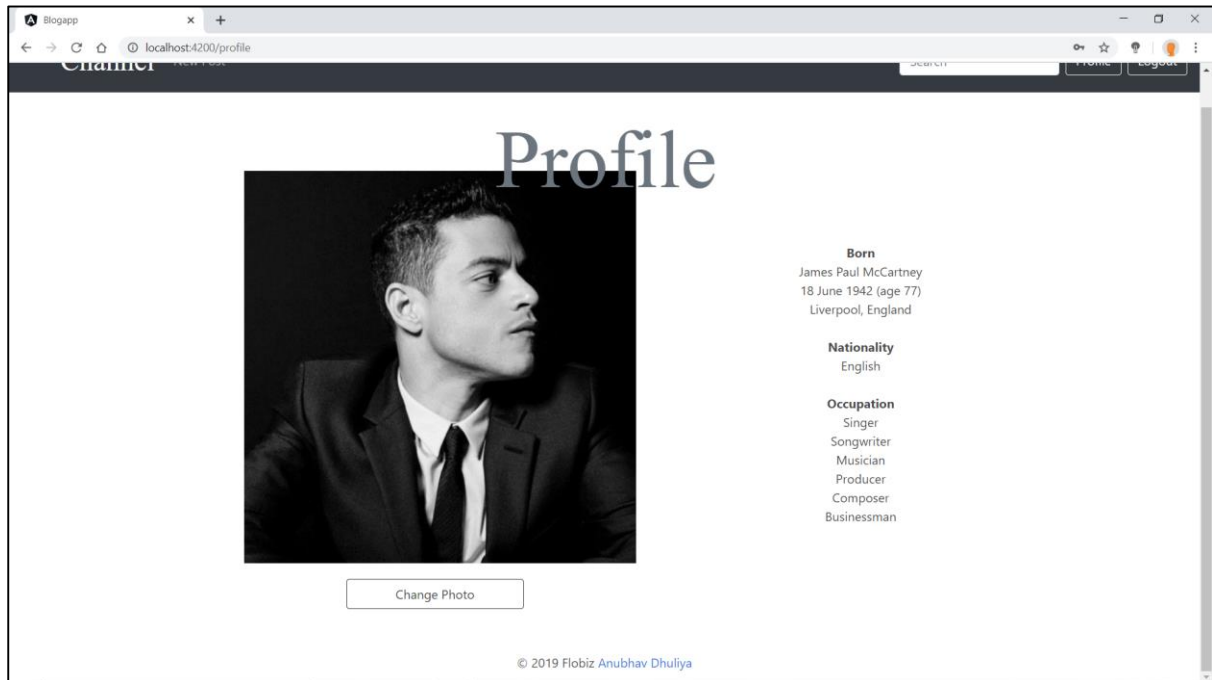
Bio
Hello Elliot.

Submit

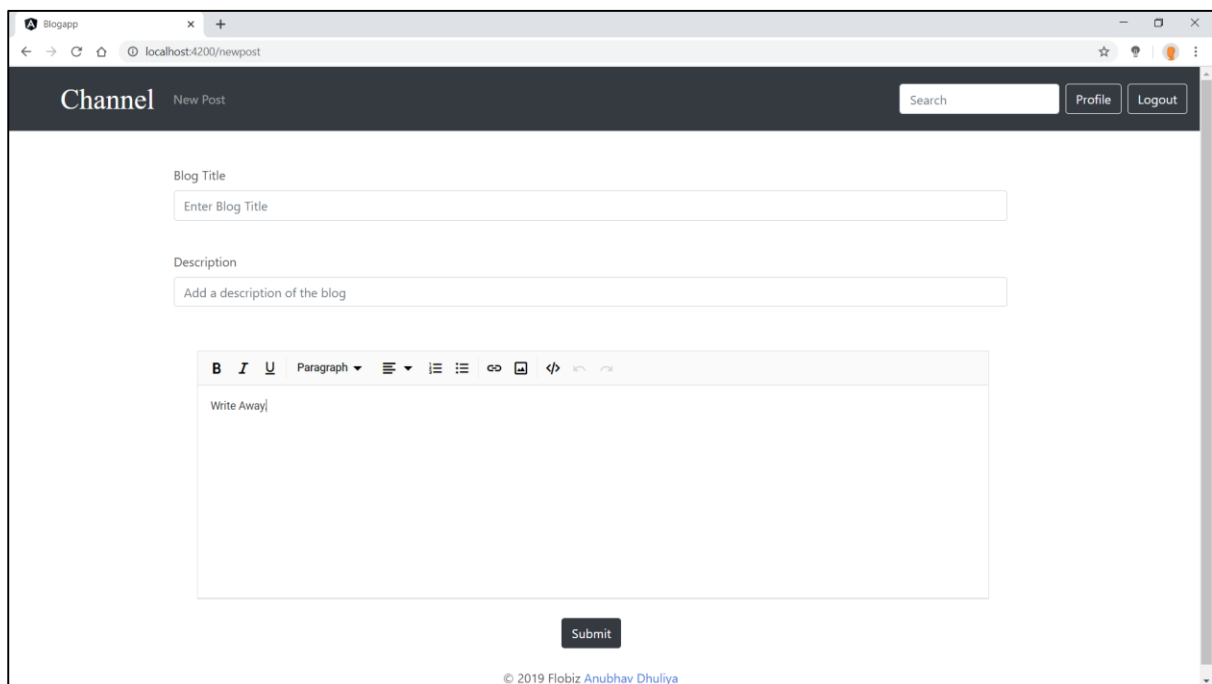
3. Home/Feed

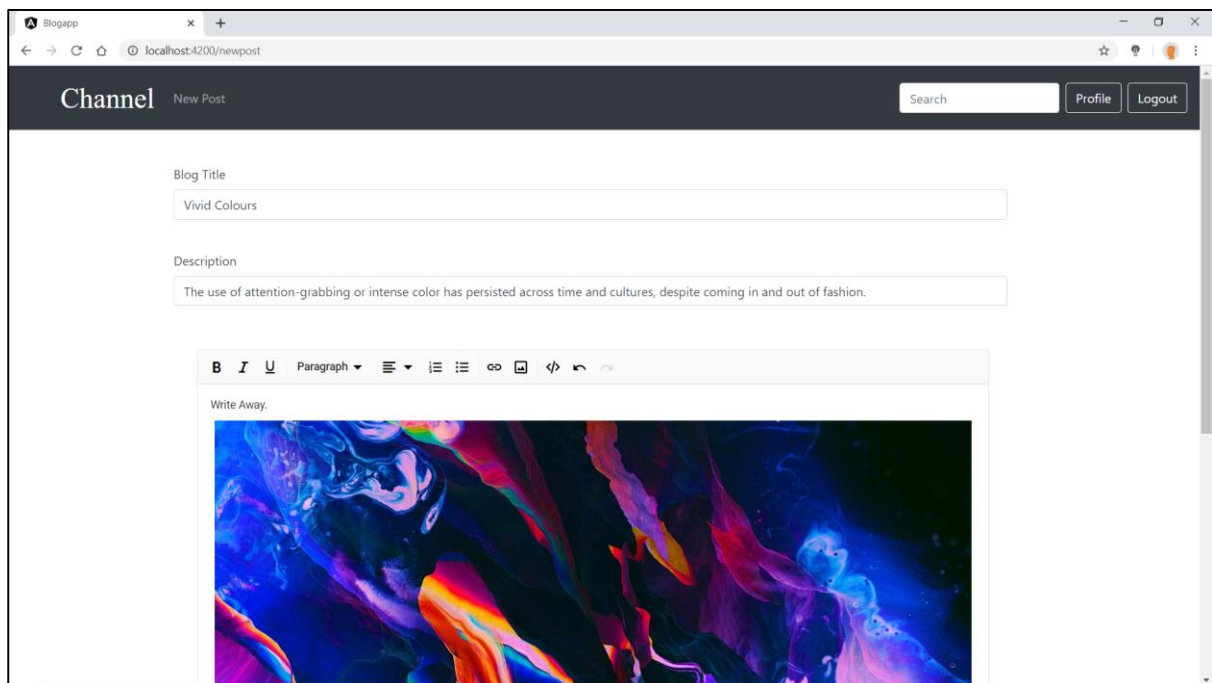
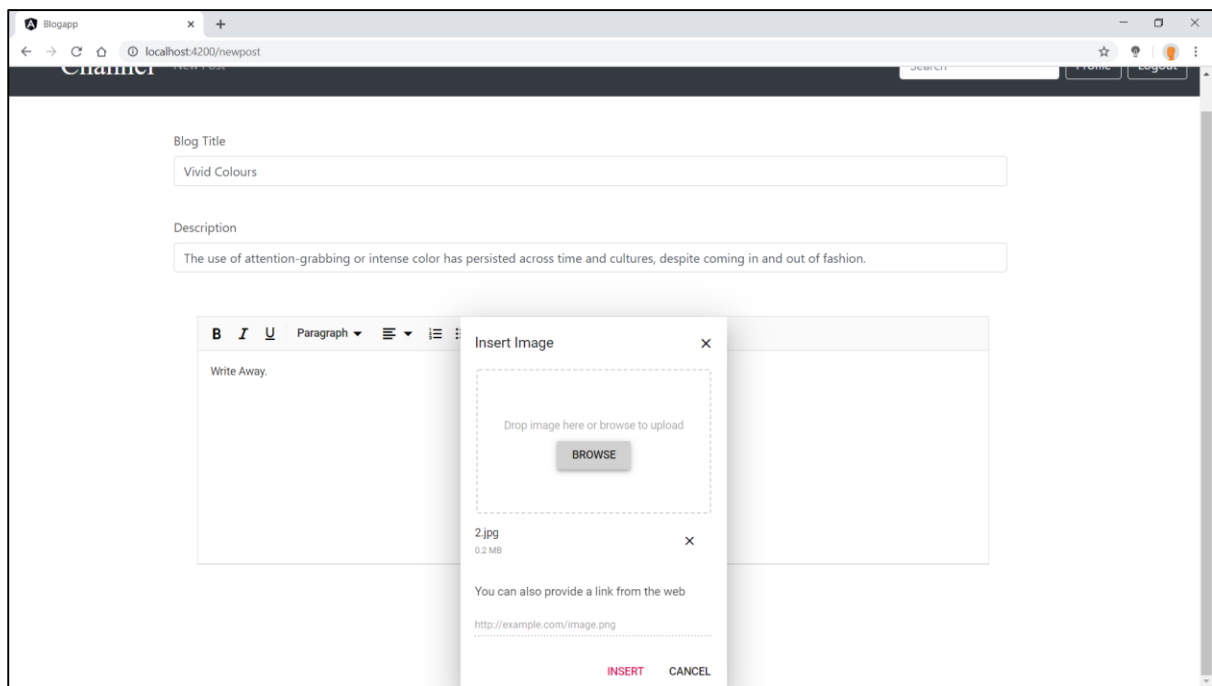


4. Profile

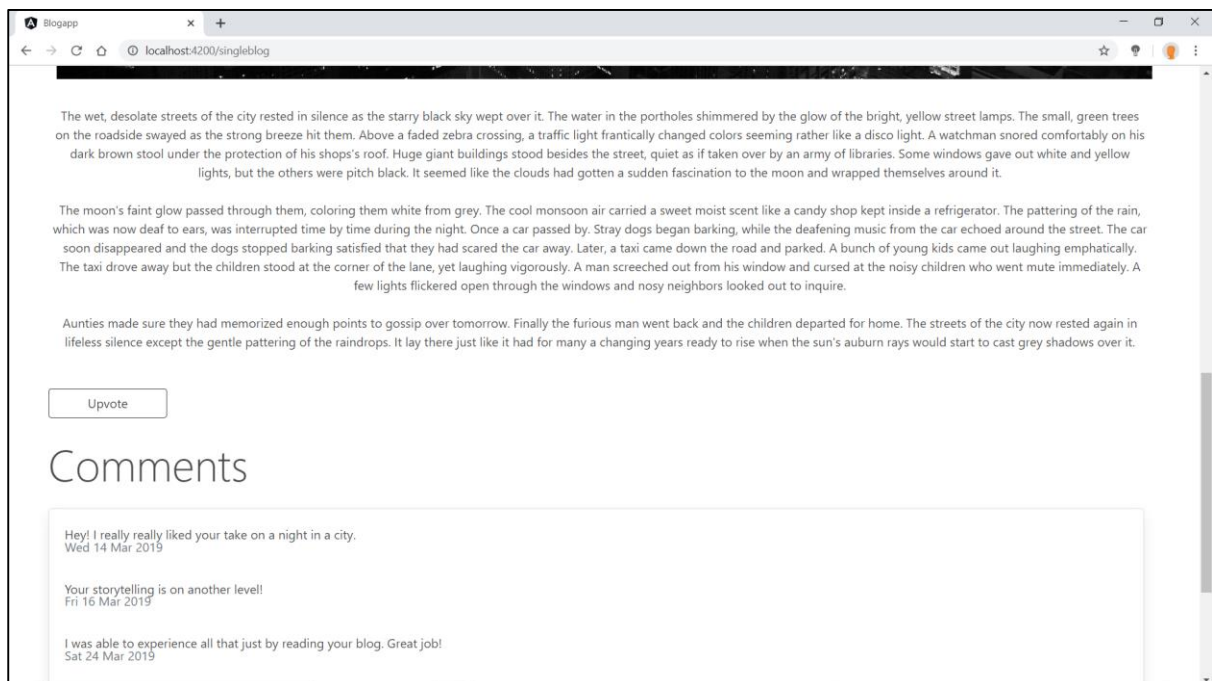
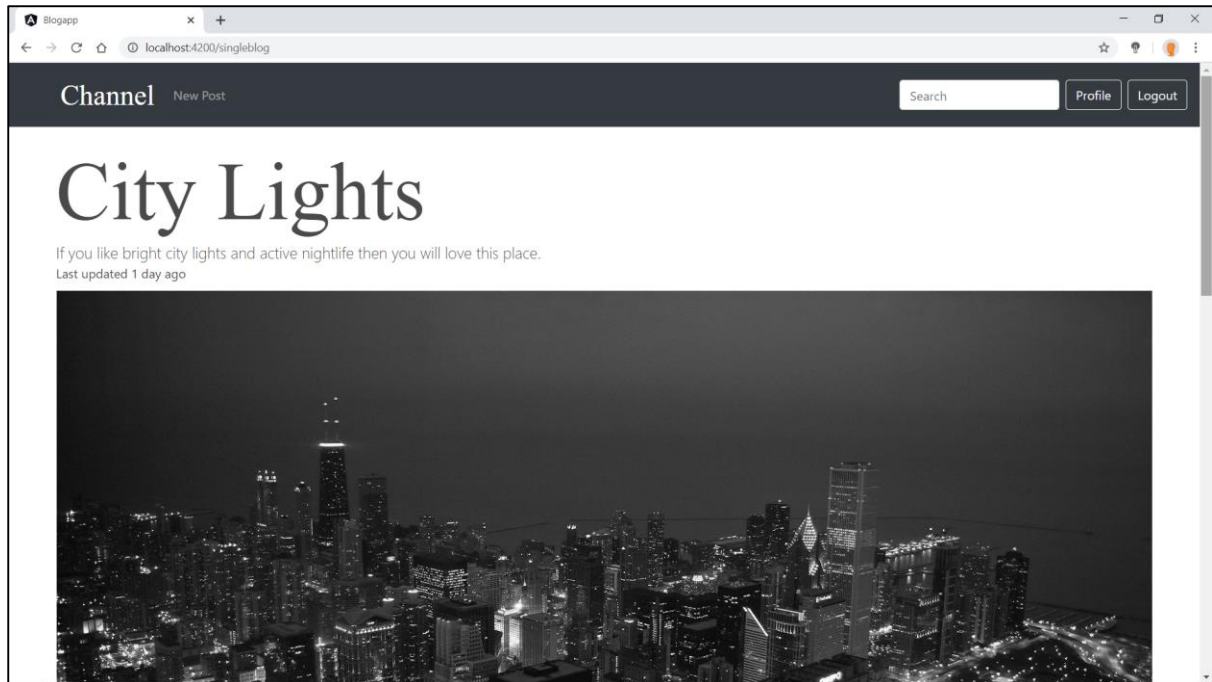


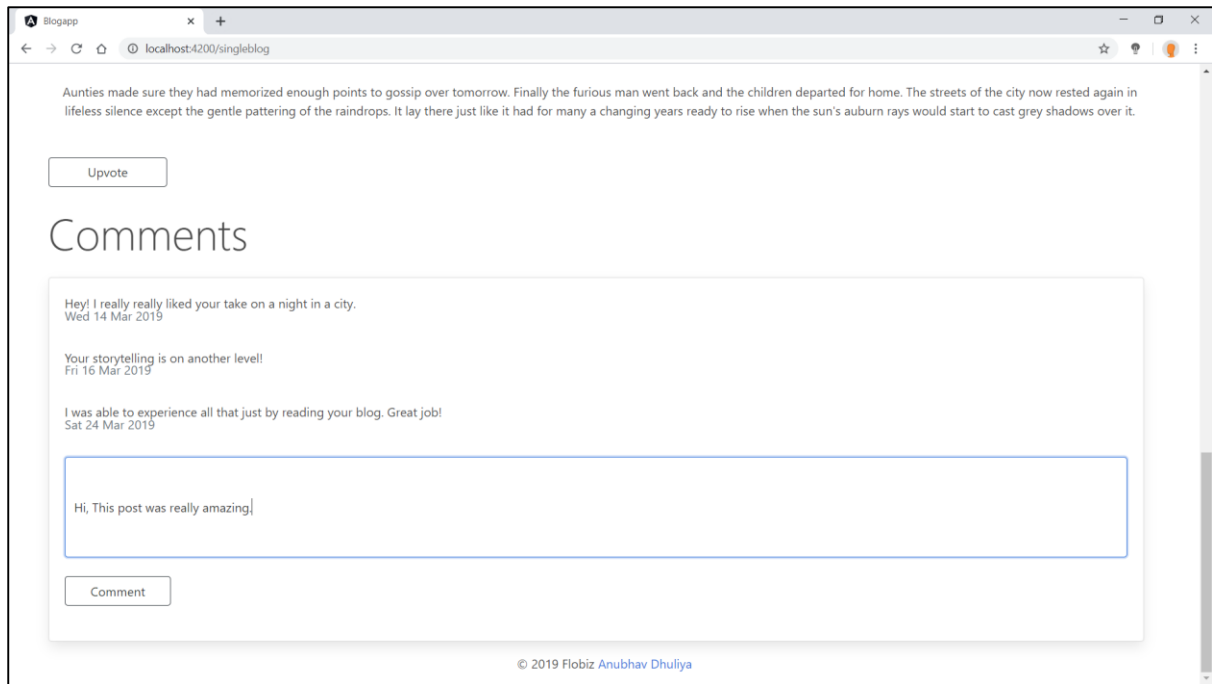
5. New Post





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.

```
Command Prompt - mongo
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\dhulimongo>
MongoDB shell version v4.2.2
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("737ee7d5-9285-4cd0-b411-006bfa0a0ff6") }
MongoDB server version: 4.2.2

Server has startup warnings:
2019-12-26T22:30:41.469+0530 I CONTROL [initandlisten]
2019-12-26T22:30:41.469+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-12-26T22:30:41.469+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2019-12-26T22:30:41.535+0530 I CONTROL [initandlisten]

Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
>

> show dbs
admin 0.000GB
config 0.000GB
flobiz 0.000GB
local 0.000GB
> use flobiz
switched to db flobiz
> db.getCollectionNames()
[ "comments", "posts", "users" ]
>
```


API Testing Document

Users API

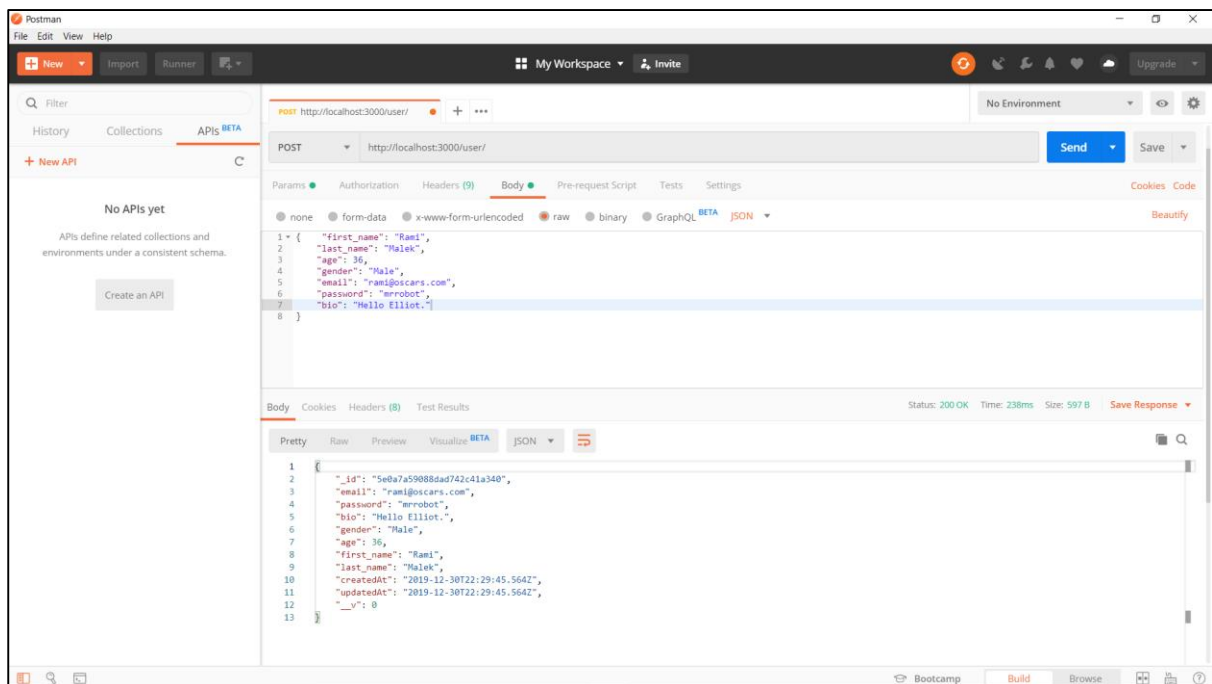
1. POST /user/

Creates a user.

Model Inputs :

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

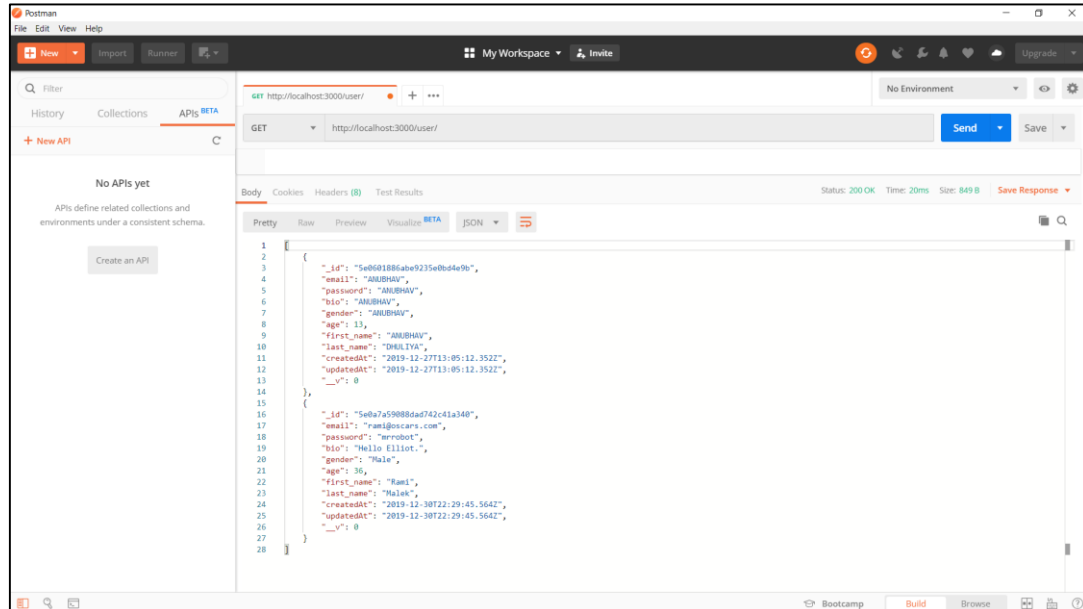
Output :



2. GET /user/

Fetches all the users.

Output :

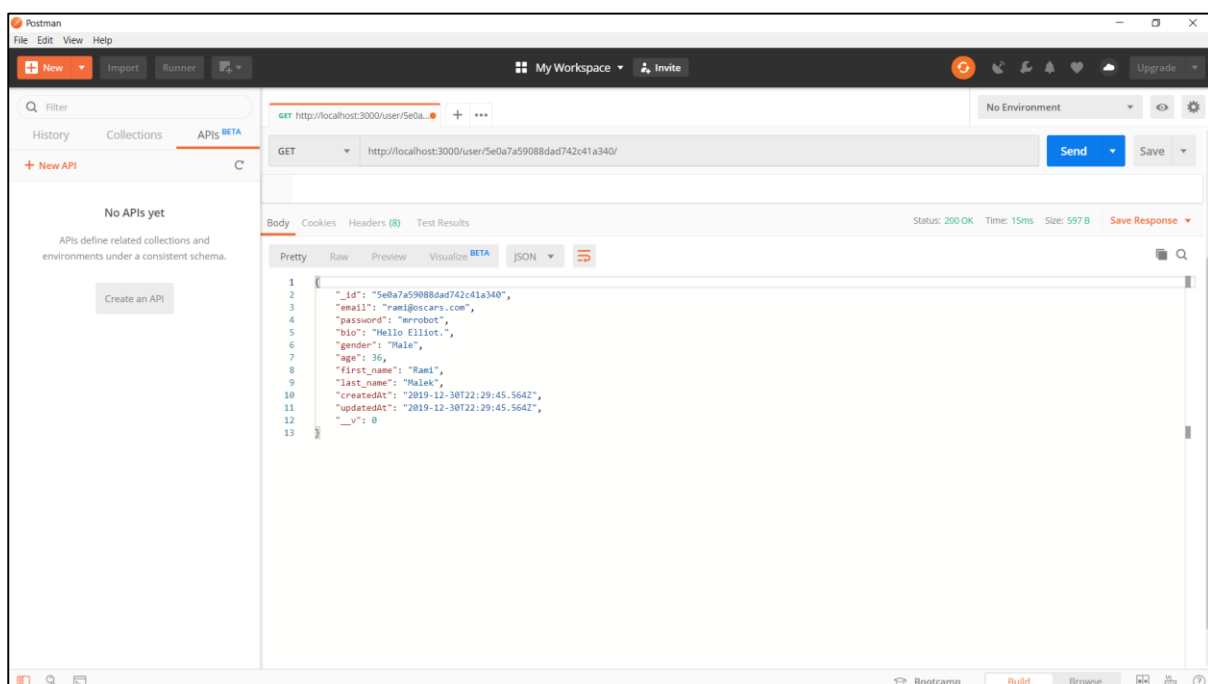


3. GET /:userId

Fetches a particular user.

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

Output :



4. PUT /:userId

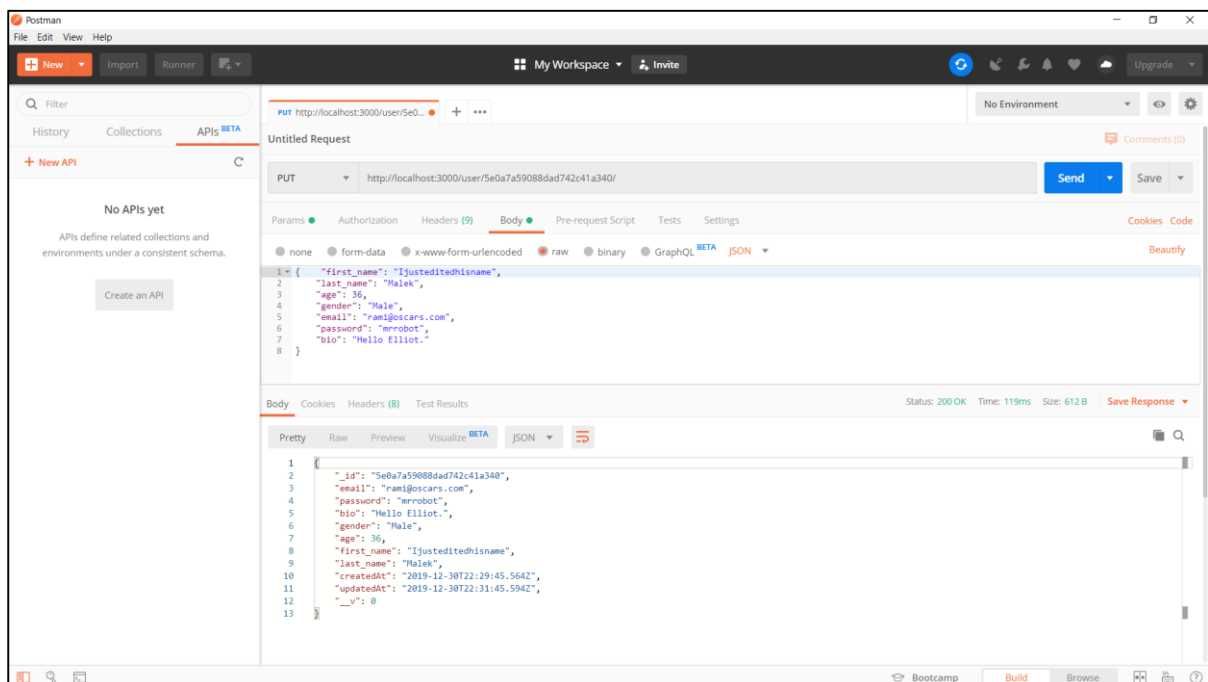
Updating a user's information.

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 :

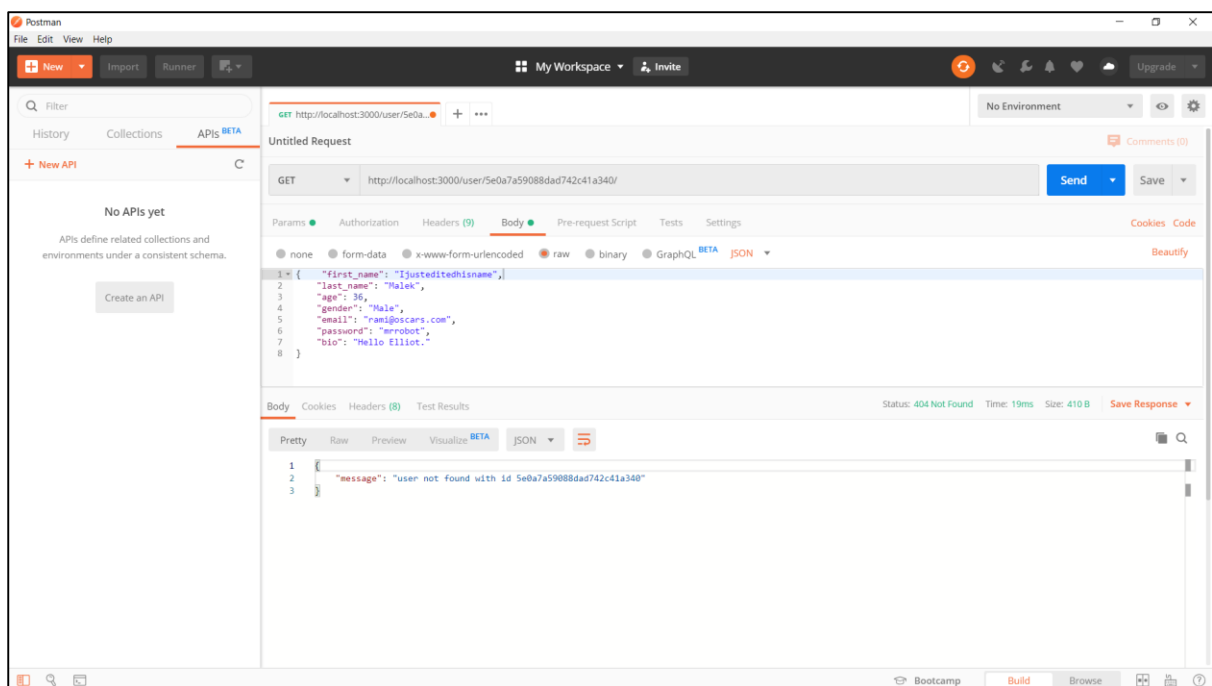
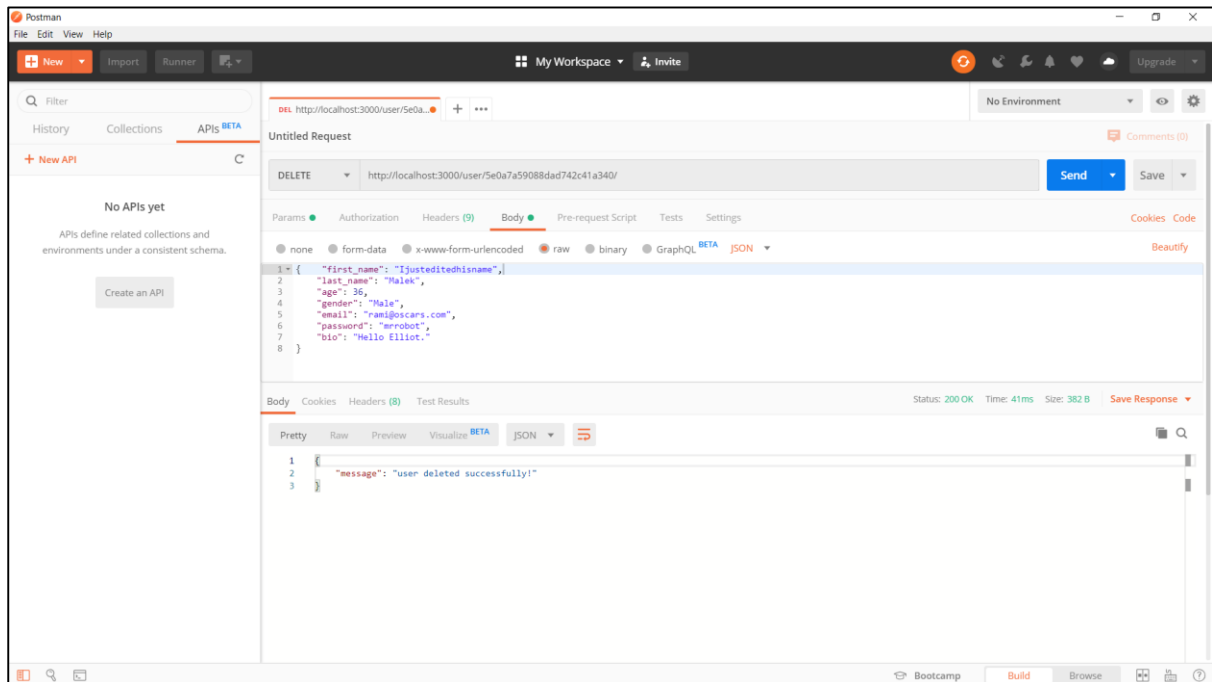


5. DELETE /:userId

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

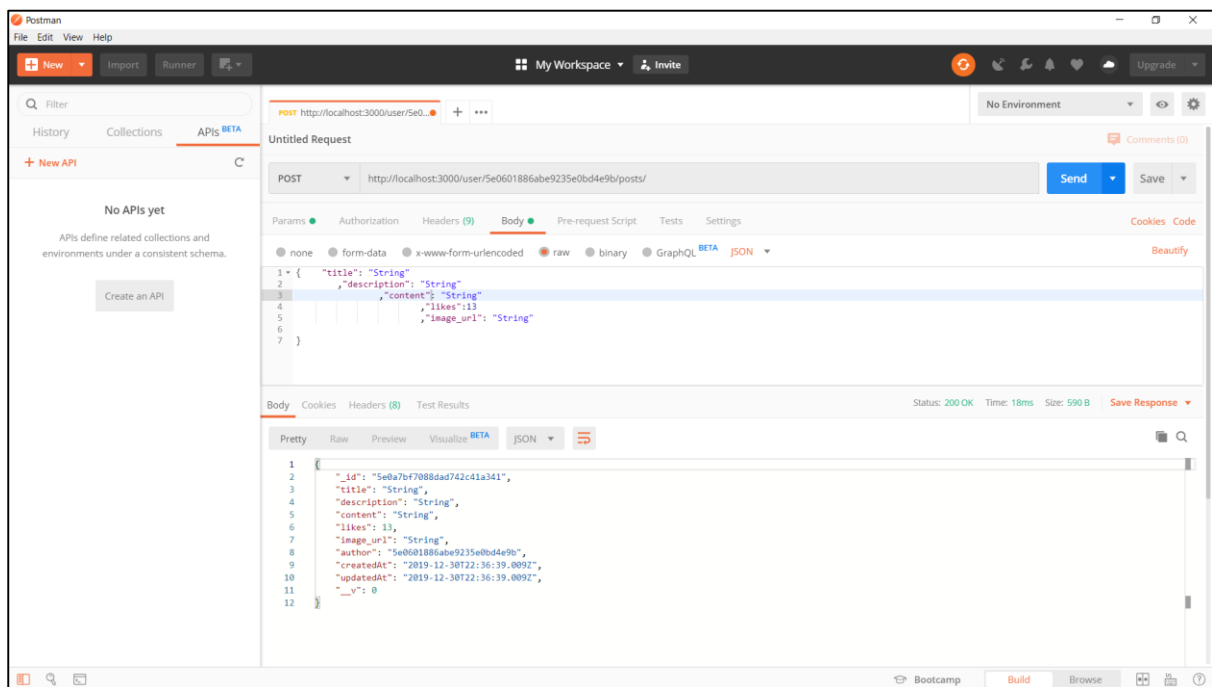
content: String

likes: Number //sent over after aggregating all the likes through MongoDB

image_url: String

author: String //userId of the creator.

Output :

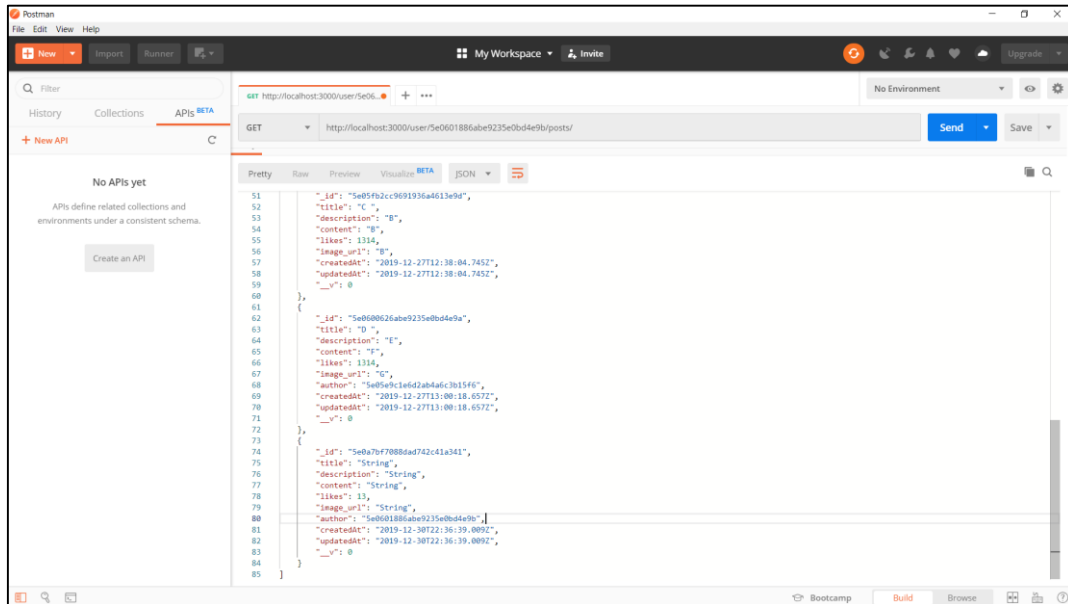


2. GET /user/:userId/posts/

Fetches all the posts/blogs by a particular user.

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

Output :

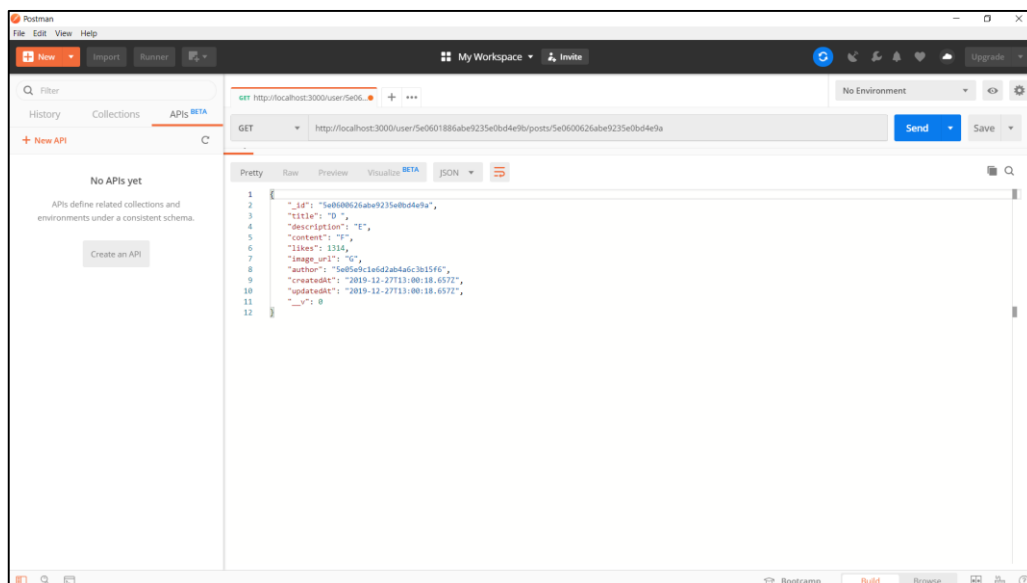


3. GET /user/:userId/posts/:postId

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

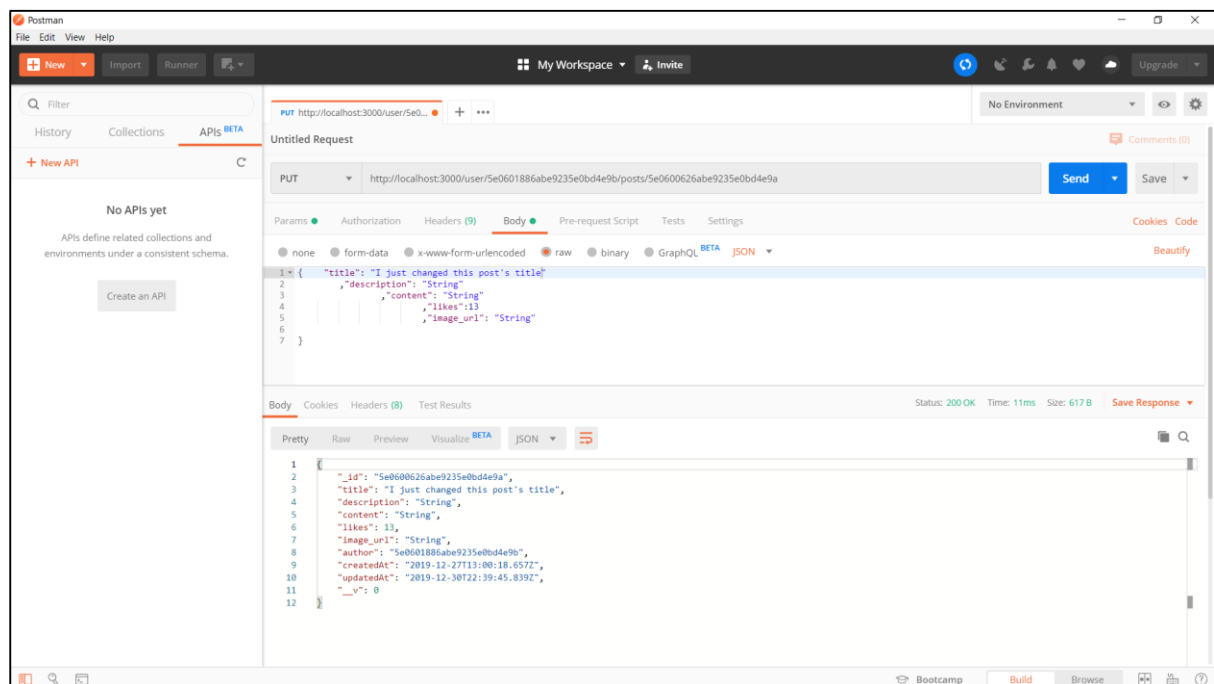
content: String

likes: Number //sent over after aggregating all the likes through MongoDB

image_url: String

author: String //userId of the creator.

Output :

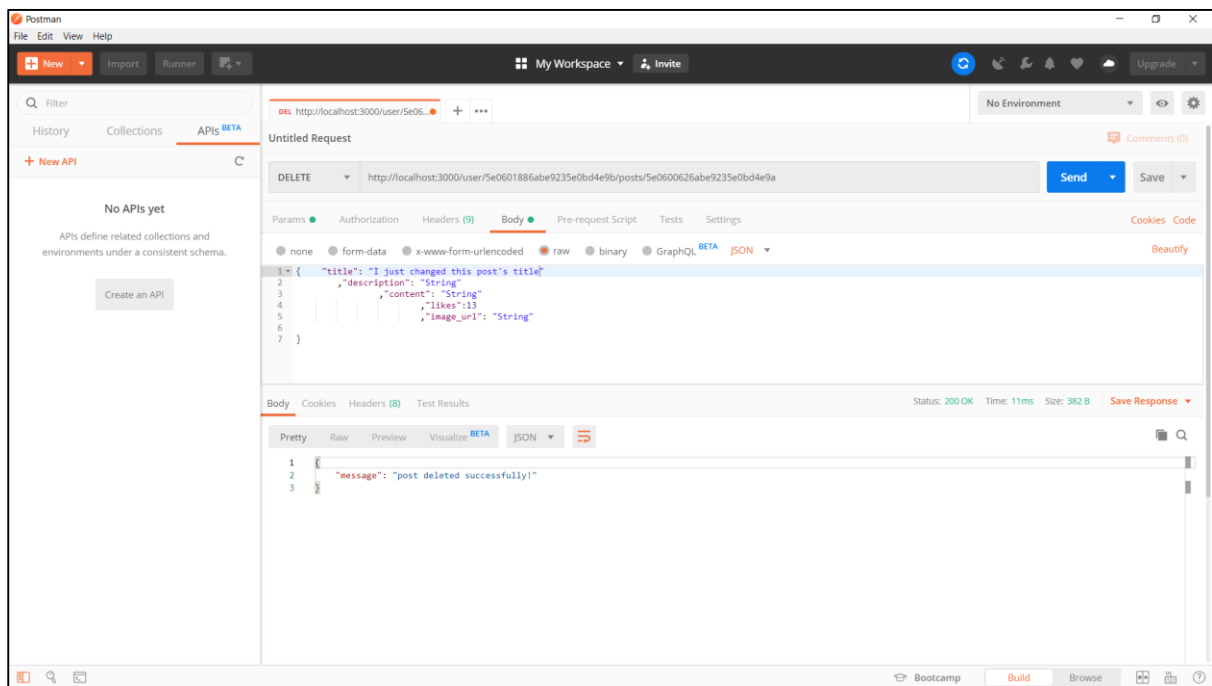


5. DELETE /user/:userId/posts/:postId

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/

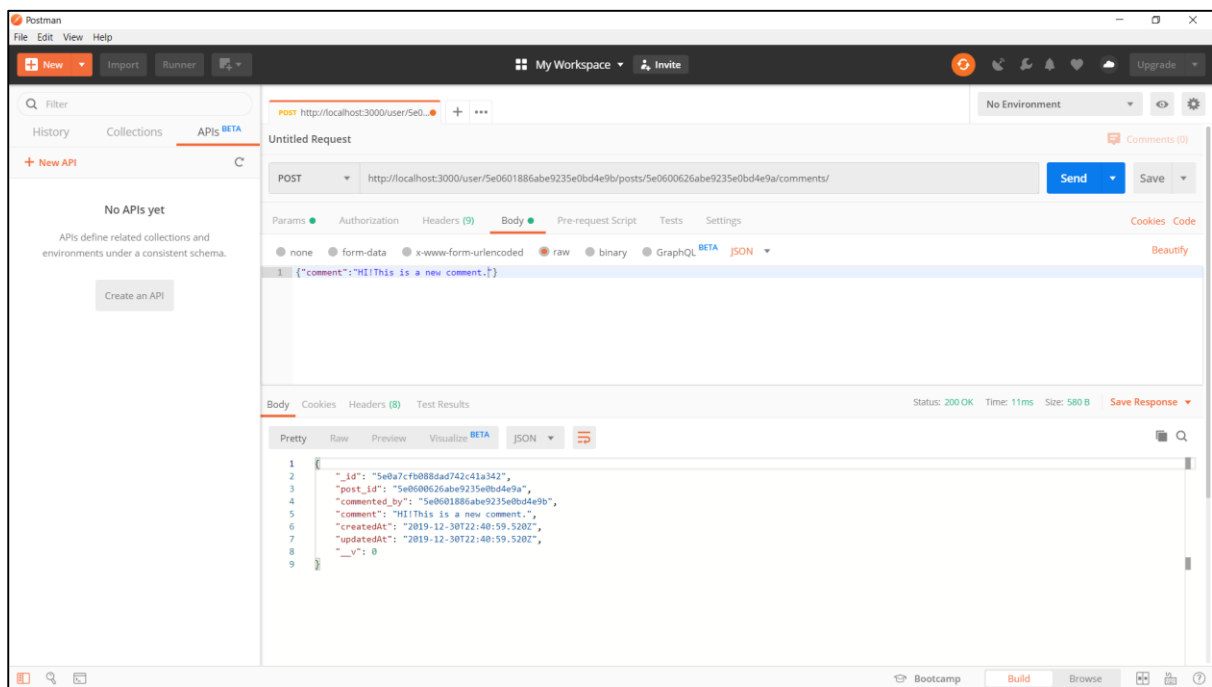
Creates a comment for a post.

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 :

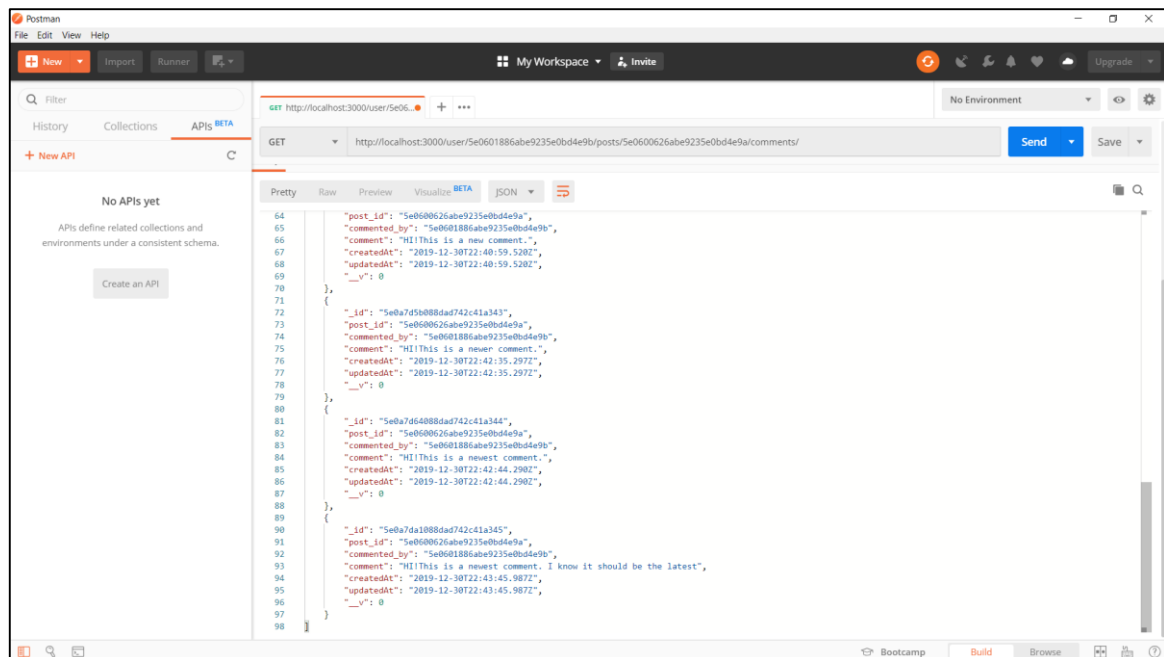


2. GET /user/:userId/posts/:postId/comments

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

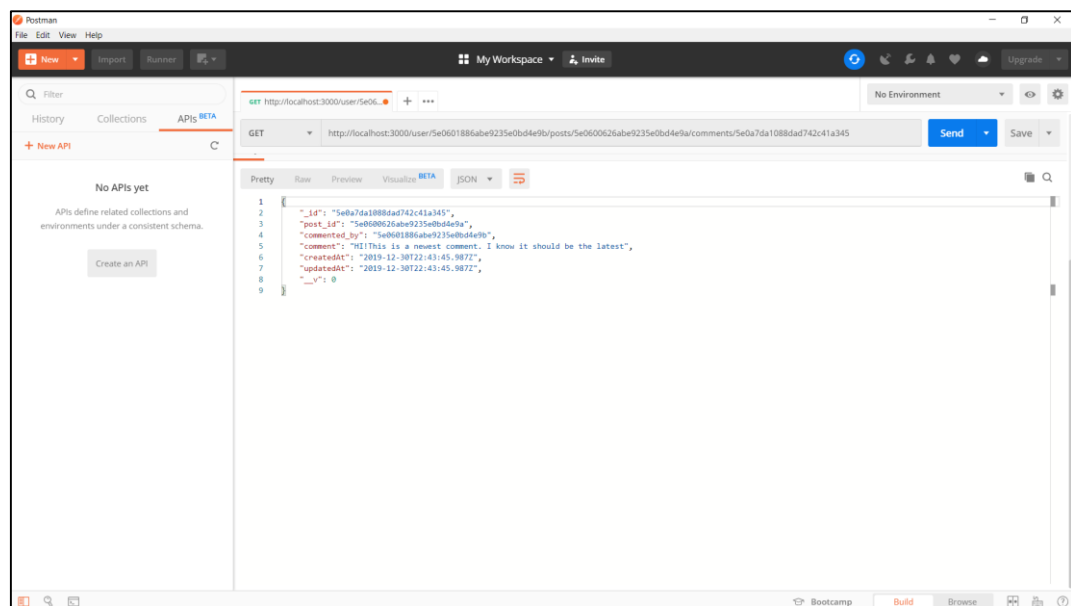
Fetches a particular comment.

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

`postId` //the post's id generated by MongoDB.

`commentId` //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.

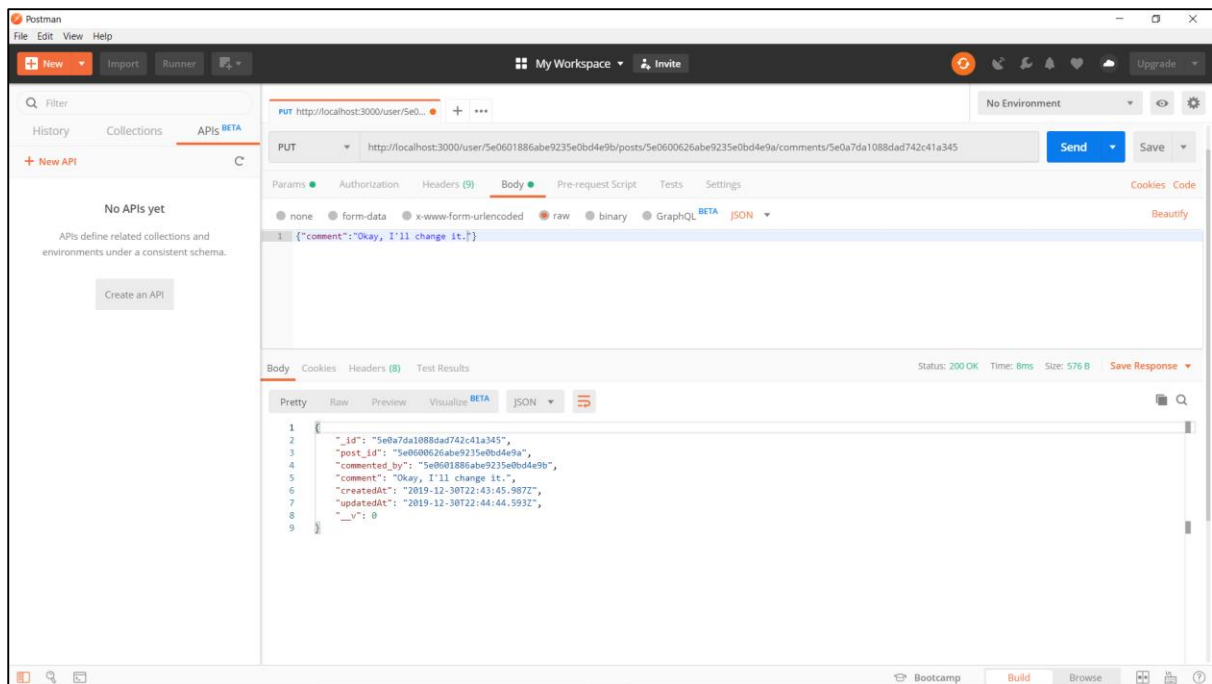
Model Input :

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 :

