Deepa Karthick
Candidate Num: Y106847

Contents

Introduction 3
Resources 3
Database 4
Post Schema 4
User Schema 5
Database Structure 5
File Structure 7
Files and folders directory structure of the application is given: 8
File Details 8
Design 10
Implementation 10
Installing packages and libraries 10
Environment Setup 11
Basic Setup to run the node.js code. 11
Libraries 13
Authentication and Authorization using JWT 13
RESTful API Implementation 14
CRUD Functionalities 14
Testing the application. 15
TC1.Olga, Nick, Mary, and Nestor register and are ready to access the Piazza API. 15
Users in the DB 17
TC2. Olga, Nick, Mary, and Nestor use the oAuth v2 authorisation service to register and get their tokens. 18
TC3 Olga makes a call to the API without using her token. This call should be unsuccessful as the user is unauthorised. 20
T4.Olga posts a message in the Tech topic with an expiration time (e.g. 5 minutes) using her token. After the end of the expiration time, the message will not accept any further user interactions (likes, dislikes, or comments). 21
Posted a message with 5 min expiration time 21
Msg in the DB 21
Like a post that is expired 22
Dislike a post that is expired 22
Add comment to expired post 23

T5.Nick posts a message in the Tech topic with an expiration time using his token. 23

T6.Mary posts a message in the Tech topic with an expiration time using her token. 24

TC7. Nick and Olga browse all the available posts in the Tech topic; three posts should be available with zero likes, zero dislikes and no comments. 25

Nick browses Messages in Tech Topic using his login 26

Olga browses Messages in Tech Topic 27

TC8. Nick and Olga "like" Mary's post on the Tech topic. 28

Nick likes Mary's post on the tech topic. 28

Olga like's Mary's post on the tech topic 28

TC9.Nestor "likes" Nick's post and "dislikes" Mary's on the Tech topic 29

Nestor 'likes' Nick's post 29

Nestor dislikes 'Mary's' post 30

TC10. Nick browses all the available posts on the Tech topic; at this stage, he can see the number of likes and dislikes for each post (Mary has two likes and one dislike, and Nick has one like). There are no comments made yet. 31

TC11. Mary likes her post on the Tech topic. This call should be unsuccessful; in Piazza, a post owner cannot like their messages. 32

TC12. Nick and Olga comment on Mary's post on the Tech topic in a round-robin fashion (one after the other, adding at least two comments each) 33

Nick comment on Mary's post 34

Olga comment on Mary's post 35

TC13.Nick browses all the available posts in the Tech topic; at this stage, he can see the number of likes and dislikes of each post and the comments made. 35

TC14. Nestor posts a message in the Health topic with an expiration time using her token 37

TC15. Mary browses all the available posts on the health topic; at this stage, she can see only Nestor's post. 37

TC16. Mary posts a comment in Nestor's message on the Health topic. 38

TC17. Mary dislikes Nestor's message on the health topic after the end of post-expiration time. This should fail 39

TC18. Nestor browses all the messages on the Health topic. There should be only one post (his own) with one comment (Mary's) 40

TC19. Nick browses all the expired messages on the Sports topic. These should be empty 41

TC20. Nestor queries for an active post with the highest interest (maximum number of likes and dislikes) in the Tech topic. This should be Mary's post. 42

Deploy your Piazza project into a VM using Docker 43

6.Create a docker file named dockerfile 45

Contents of the Dockerfile 45

7.Docker image created using the command 45

Deploy Piazza application in Kubernetes 46

Introduction

Computing resources are costly for single organizational usage, hence using offered computing. services results in flexible, speedy and relatively fewer operating costs. The services include software, database, network, servers, analysis on the internet aka cloud. Software-as-a-Service is a method that offers access to software to its remote users from the cloud. The purchase and installation of a software is costlier than subscribing to it.

Piazza is a SaaS based cloud application that allows authenticated users to post their thoughts for other users authorised to the software. The users like and comment on each other's posts which builds a community altogether. This report concentrates on implementation, development and testing of a Piazza application.

The report starts with a brief introduction about the domain. Next section focuses on the resources such as tools and platforms used in implementation. Next, database model and file structure are described followed by UML modelling of the application. The later sections cover the installation of libraries, the authentication and validation using JWT (Copes, 2023), development of RESTful API (including CRUD operations), test cases generation and deployment.

Functionalities offered by Piazza are as listed below:

1. oAuth v2.0 protocol authorises the users

2. Authorised users can share their thoughts via text.

3. All posts are available to read to the valid users

4. Users can comment, like, dislike on the posts

5. The Post owner can set expiry time after which the post cannot be liked, disliked or commented.

6.The posts can be retrieved by a particular topic

Resources

Resources The tools and platforms used in implementation of this application are as listed below:

1. GCP – To create a VM instance for deployment

2. Node JS – Writing code in JS (Chris, 2021)

3. GitHub – hosting code (Kaelin,2022)

4. MongoDB – Data storage (MongoDB tutorial, 2022)

5. Postman – design and test the generated test cases (Node.js, 2022)

Database

The database is designed using MongoDB for piazza application. The schema design is as described:

Post Schema

models > JS PPOST > PPOSTSchema > post_owner > type

```
const mongoose = require('mongoose')

const PPOSTSchema = mongoose.Schema({
    post_title:{
        type:String
    },
    post_topic:{
        type:String,
        enum:['Politics', 'Health', 'Sport', 'Tech']
    },

    likes_count:{
        type:Number,
        default:0
    },
    dislikes_count:{
        type:Number,
        default:0
    },

    comments:{
        type:Array
    },

    post_date:{
        type:Date,
        default:Date.now
    },

    post_owner:{
        type:String
    },

    post_time:{
        type:Date
    },

    post_owner_id:{
        type:String
    },

    post_status:{
```

User Schema

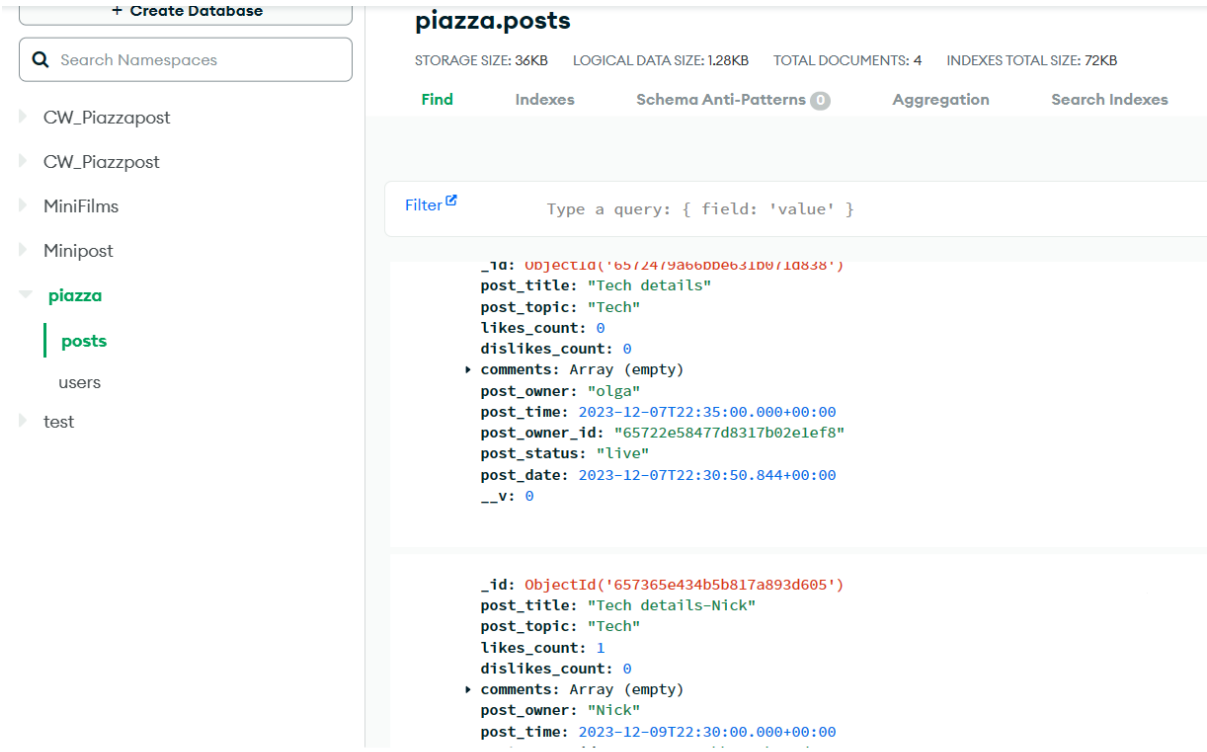
models > User.js > userSchema

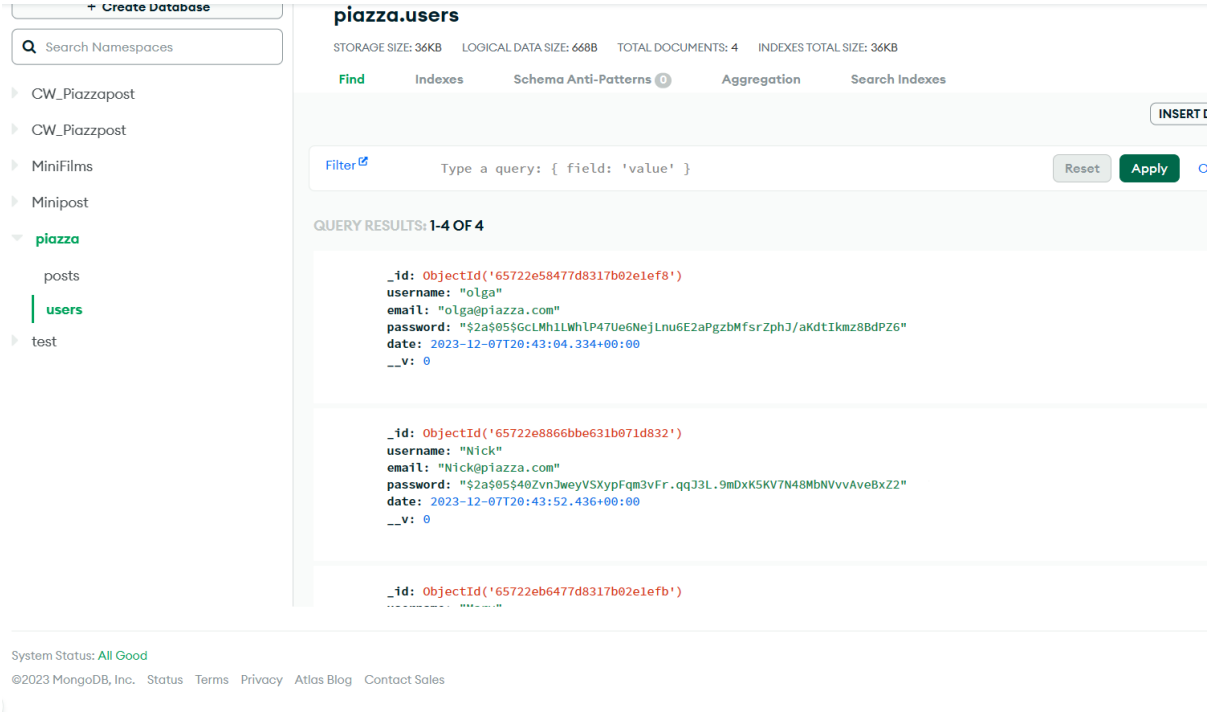
```
const mongoose = require('mongoose')

const userSchema = mongoose.Schema({
    username:{
        type:String,
        require:true,
        min:3,
        max:256
    },
    email:{
        type:String,
        require:true,
        min:6,
        max:256
    },
    password:{
        type:String,
        require:true,
        min:6,
        max:1024
    },
    date:{
        type:Date,
        default:Date.now
    }
})
module.exports=mongoose.model('users',userSchema)
```

Database Structure

Mongo DB is used to manage the data. New database 'piazza' is created, and two collections posts and users are created. Posts holds all the messages and users hold the user details.

Deepa Karthick
Candidate Num: Y106847

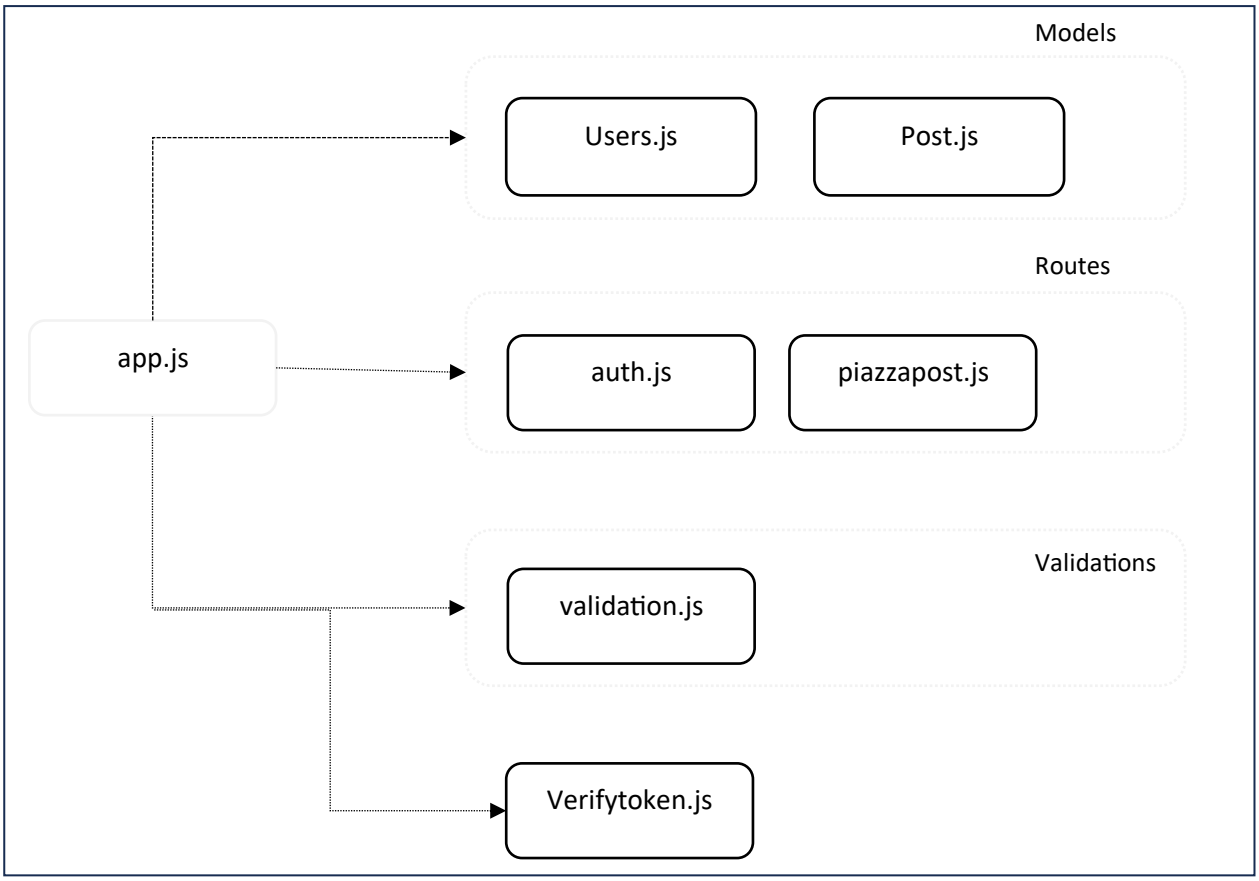




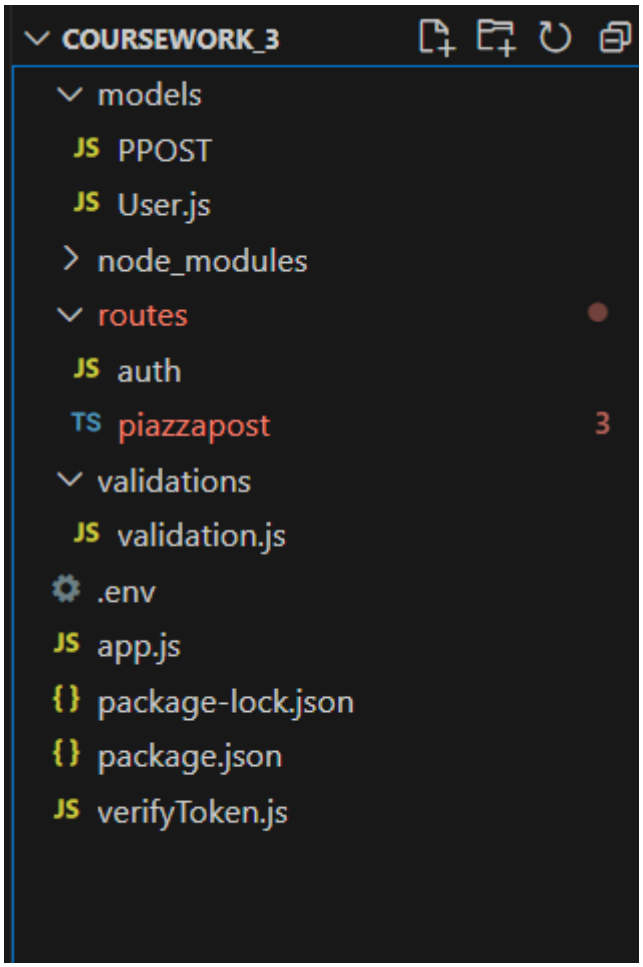
Deepa Karthick
Candidate Num: Y106847

File Structure

Files in the application are organized in MVC fashion; where models are stored in folder Models, View in the folder Routes and Controller in the Validations folder. The figure below describes detailed file structure formed. The app has three pages, home: app.js; auth.js, piazza.js that has all the feeds and validations.js to validate the input from the user at registration and login sessions. verifyToken.js contains code in relation to authentication discussed in later sections.



Files and folders directory structure of the application is given:



File Details

1.***app.js file:*** This is the entry point to the application where middleware is created, API endpoints are mapped, connection to MongoDB is established and finally the server is set on port 3000.

2. ***models folder***: Includes the files to create the model to define each database structure

a. Post.js: A schema to define the posts database table and export the model.

- Post_title – String – given by user
- Post_topic – Enum and holds four values 'Politics' , 'Sport', 'Health' and 'Tech'
- Likes_count – Every time a post is liked, the counter is increased.
- Dislikes_count - Every time a post is disliked, the counter is increased.
- Comments – creates an array of comments when input by the user.
- Post_date – created by the system for every post
- Post_time – expiry date time set for each post.
- Post_owner_id – created from the login authentication token to identify post owner.
- Post_status – Enum can hold 'live' and 'expired' calculated by the system. Defaults live and sets it to expired if current date > post_time.

a. User.js: A schema to define the user's database table and export the model.
   - Username – String and needs to be minimum 3 char and max 256, mandatory
   - Email – email address and min 6 characters, mandatory
   - password – string, min 6 characters, mandatory.

3. **routes folder:** Files to create the routes to the api endpoints

a. posts.js: Imports the Post schema model and do the following operations with user authentication.

i. post- for posting a new post,

ii. get - to get **all the posts** and get **posts by id**, to get **posts by topic**, to get posts by **topic and status.**

iii. post-to like a post by incrementing the likes_count

iv. Post – to dislike a post by incrementing the dislikes_count

v. post-to comment a particular post

b. user-auth.js: Imports the User schema model and do the validations from validation.js file and after successful login give the auth-token to the user.

4.**validations folder**: a. validation.js: This file contains all the validations for user registration and login.

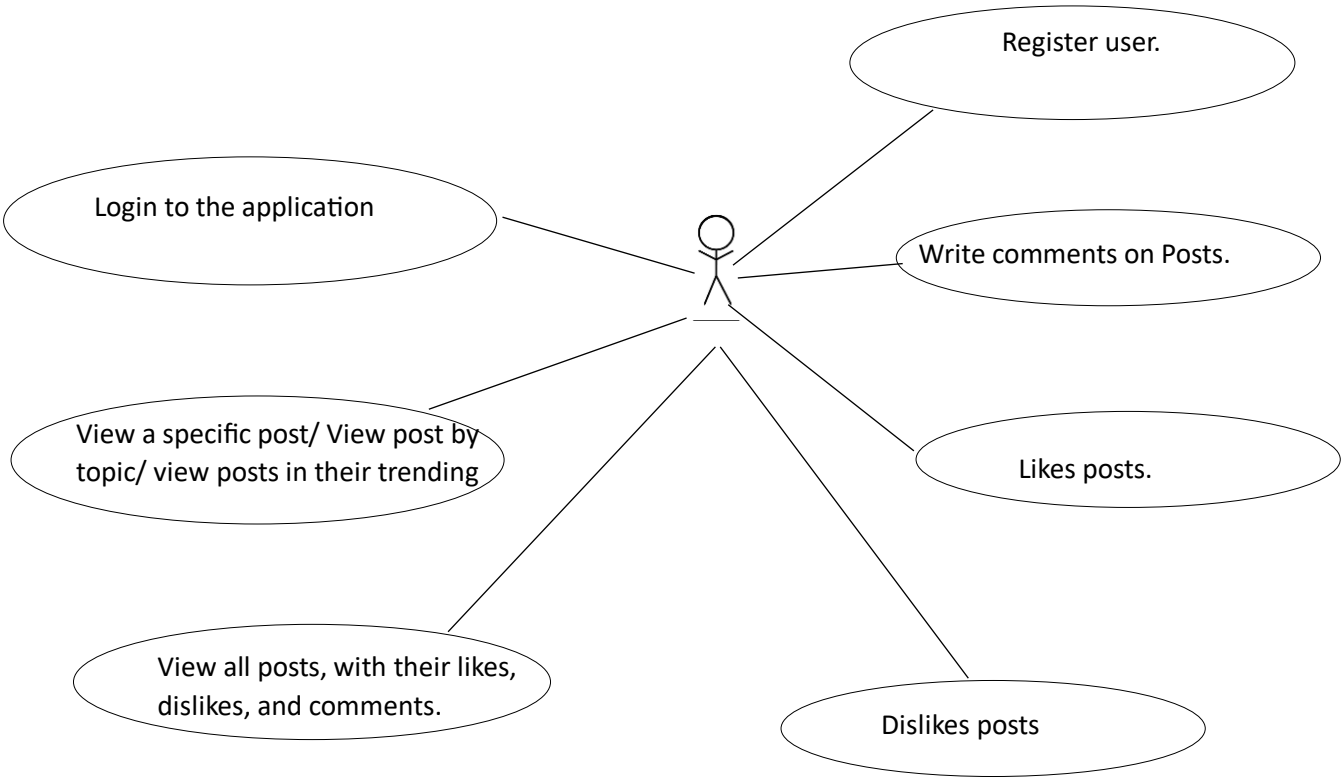
5. **verifyToken.js file**: This file contains the code for Token verification once it is provided by the user for CRUD operations after login

6. **env file:** This file contains the path DB_CONNECTOR to connect the application to the MongoDB and TOKEN_SECRET to generate token for authentication.

Deepa Karthick
Candidate Num: Y106847

Design

Use case Diagram



Implementation

Installing packages and libraries

1.npm init: This command generates the package. Json file in the root folder "Coursework_3", which contains all the project details, scripts and dependencies used for the project.

2. npm install: The following are the dependencies installed:

a. express: framework is required to develop web RESTful APIs

b. nodemon: very helpful in restarting the application as soon as the code files are changed and saved.

c. mongoose: this package is useful when MongoDB holds application data. It models, validates and manipulates data.

d. body-parser :helps in parsing the body that includes information when operations like POST, PATCH or PUT are involved.

e. dotenv: – newly added feature that loads environment variables added into the .env file created by the developer.

f. joi: A package for object schema descriptions and validations.

g. jsonwebtoken: provides strong authentication by generating authentication tokens

h. bcryptjs: To encrypt and decrypt the passwords.

3. npm start: To start the server listening at port 3000, written in file app.js

Environment Setup

1. The packages are first installed using the terminal of Node JS. The package.json file contains the starting point of the application (figure 6). In Scripts, test:"echo…" is changed with start:"nodemon app.js". This confirms that the starting point of this application is app.js. Every time start command is invoked; it asks nodemon to run app.js app.

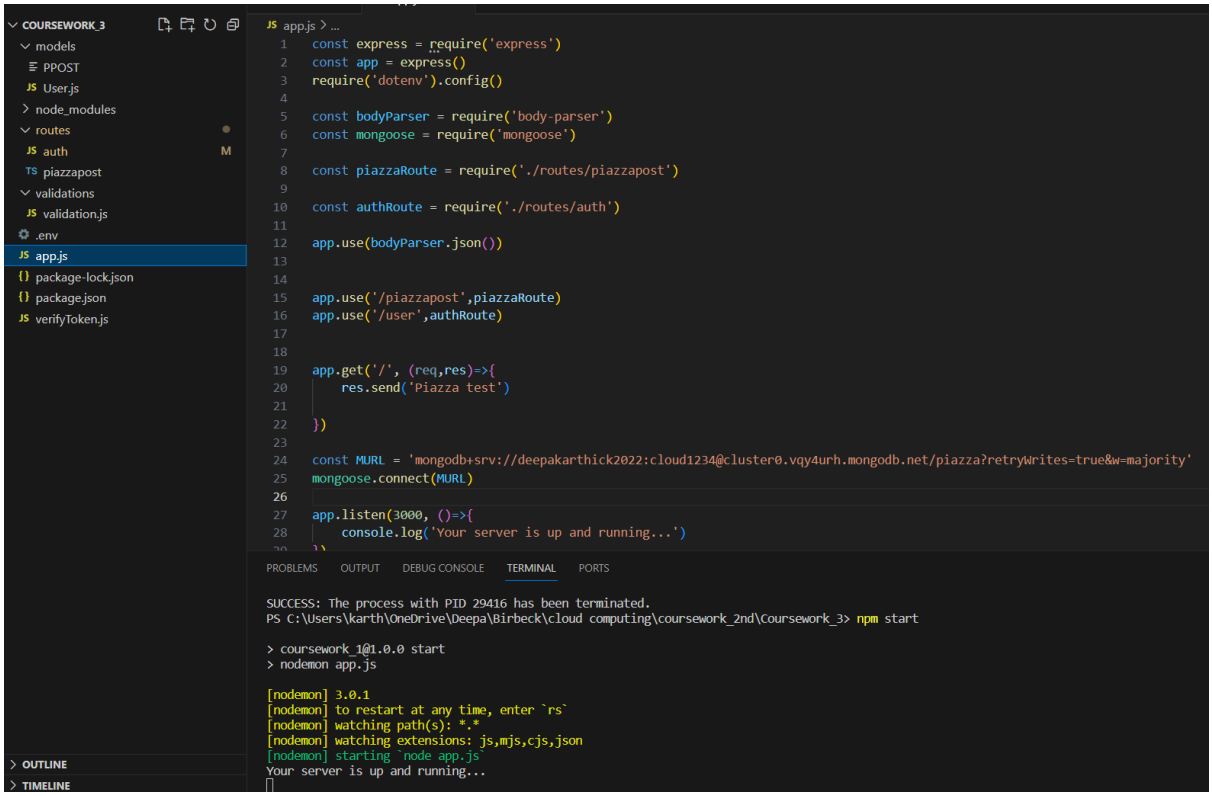
Basic Setup to run the node.js code.

These are the basic commands used:

```
{
  "name": "coursework_1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "start": "nodemon app.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.20.2",
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "express-validator": "^7.0.1",
    "joi": "^17.11.0",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^8.0.1",
    "nodemon": "^3.0.1"
  },
  "devDependencies": {
    "@types/express": "^4.17.21"
  }
}
```

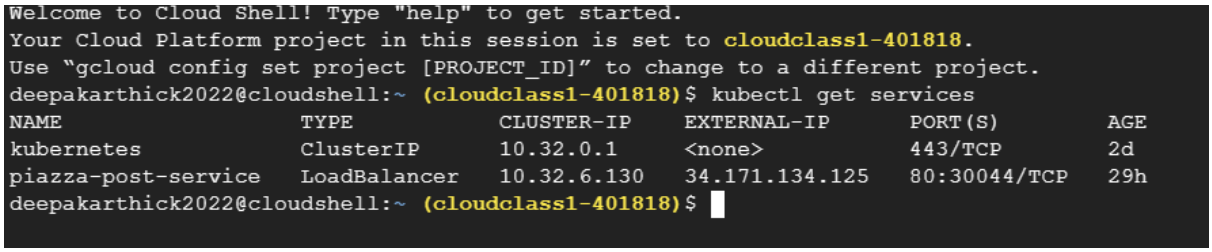
specification of app.js as starting point

2. npm start – command starts the server and the web application is ready for testing as shown in figure.



npm start

3. The web application is deployed in virtual machine and has five replicas of the service implemented in Kubernetes using the deployment.yaml and service.yaml so the application can be accessed with url: 34.171.134.125 given by the 'get services' in Kubernetes, that invokes app.js file as shown in figure.



To get the External IP to run the application

4. Endpoints

API endpoints	Description
POST - 34.171.134.125/user/register	To register a new user
POST- 34.171.134.125/user/login	To login as a registered user
POST - 34.171.134.125/piazzapost	To create a new post
POST - 34.171.134.125/piazzapost/postId/likes	To like a particular post
POST - 34.171.134.125/piazzapost/postId/dislikes	To dislike a particular post
POST - 34.171.134.125/piazzapost/postId/comments	To comment on a post
GET - 34.171.134.125/piazzapost	To retrieve all the posts

GET - 34.171.134.125/piazzapost/postId	To retrieve post by specific ID
GET - 34.171.134.125/piazzapost?topic=Sport	To get the posts corresponding to Sport Topic
GET- 34.171.134.125/ piazzapost?topic=Sport&status=expired	To get posts for Sport topic and expired status
GET- 34.171.134.125/piazzapost?topic=Sport&status=live	To get posts for Sport topic and live status
GET- 34.171.134.125/piazzapost/trending	To get the post that has most number of likes and dislikes

Libraries

The libraries express and nodemon are imported at the beginning before anything else in the application.

Authentication and Authorization using JWT

Authentication

Using email and password, user is allowed to login to the system. If the username does not exist in the database, it prompts 'User doesn't exist'. Similarly, for incorrect password, it displays the message to the user. Once the user is allowed to login with correct credentials, JWT creates an authentication token for the logged in user. This token is important to pass as parameter in the header to access the Piazza application further. The users in the application are authenticated using oAuth 2.0 protocol as shown in figure 9.

Authorization

The users are expected to register by providing their username, email and password. The application checks for the existing email and username, if either exists, it prompts that user that email or username exists and hence duplicate registration is not accepted. The user is not allowed to comment or like own posts. If the attempt is made the application sends a message that the operation is not allowed. No other user can edit or delete posts of the other users.

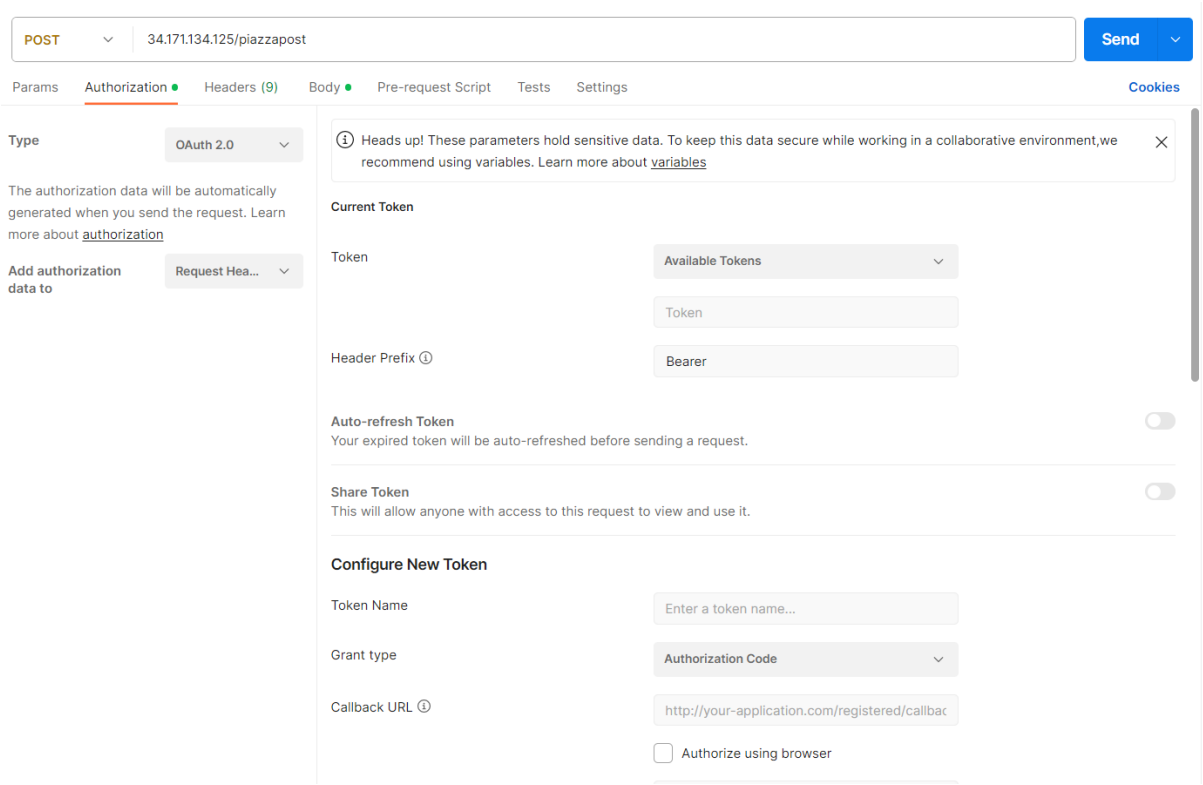


Figure 9 Authentication with oAuth 2.0

RESTful API Implementation

The implementation of RESTful API involves following basic and necessary steps:

1. Directory – the directories are created as per the requirements of the application.
2. App Express – the express and nodemon libraries are imported.
3. Schema – The models are created as per requirements.
4. Authentication – JWT authentication is important for secured application.

CRUD Functionalities

The application effectively implemented CRUD functionalities.

Create – creates users, posts and comments

Read – reads data

Update – updates posts

Delete – deletes posts

Deepa Karthick
Candidate Num: Y106847

Testing the application.

TC1.Olga, Nick, Mary, and Nestor register and are ready to access the Piazza API.

POST 34.171.134.125/user/register

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"username": "olga",
"email": "olga@piazza.com",
"password": "olga12"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 458 ms

Pretty Raw Preview Visualize JSON

```
{
    "username": "olga",
    "email": "olga@piazza.com",
    "password": "$2a$05$GcLMh1LWhlP47Ue6NejLnu6E2aPgzbMfsrZphJ/aKdtIkmz8BdPZ6",
    "_id": "65722e58477d8317b02e1ef8",
    "date": "2023-12-07T20:43:04.334Z",
    "__v": 0
}
```

Deepa Karthick
Candidate Num: Y106847

POST 34.171.134.125/user/register

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"username": "Nick",
"email": "Nick@piazza.com",
"password": "Nick12"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 391 ms Size: 430 B

Pretty Raw Preview Visualize JSON

```
{
    "username": "Nick",
    "email": "Nick@piazza.com",
    "password": "$2a$05$40ZvnJweyVSXypFqm3vFr.qqJ3L.9mDxK5KV7N48MbNVvvAveBxZ2",
    "_id": "65722e8866bbe631b071d832",
    "date": "2023-12-07T20:43:52.436Z",
    "__v": 0
}
```

POST 34.171.134.125/user/register

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"username": "Mary",
"email": "Mary@piazza.com",
"password": "Mary12"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 343 ms Size: 430 I

Pretty Raw Preview Visualize JSON

```
{
    "username": "Mary",
    "email": "Mary@piazza.com",
    "password": "$2a$05$KVmn7PEVmapq4gCcx7jhduaGEbcH.Xrqx92dZvMjG55Z2OtSjUJDO",
    "_id": "65722eb6477d8317b02e1efb",
    "date": "2023-12-07T20:44:38.717Z",
    "__v": 0
}
```

Deepa Karthick
Candidate Num: Y106847

POST 34.171.134.125/user/register

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"username": "Nestor",
"email": "Nestor@piazza.com",
"password": "Nestor"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 336 ms Size: 434 B Sav

Pretty Raw Preview Visualize JSON

```
{
    "username": "Nestor",
    "email": "Nestor@piazza.com",
    "password": "$2a$05$gdjTVoPLLSQd/gdLrW5AUOm4du5jOvA56Ix3fN5Mi49yPVgvVILLm",
    "_id": "65722ed466bbe631b071d835",
    "date": "2023-12-07T20:45:08.964Z",
    "__v": 0
}
```

Users in the DB

QUERY RESULTS: 1-4 OF 4

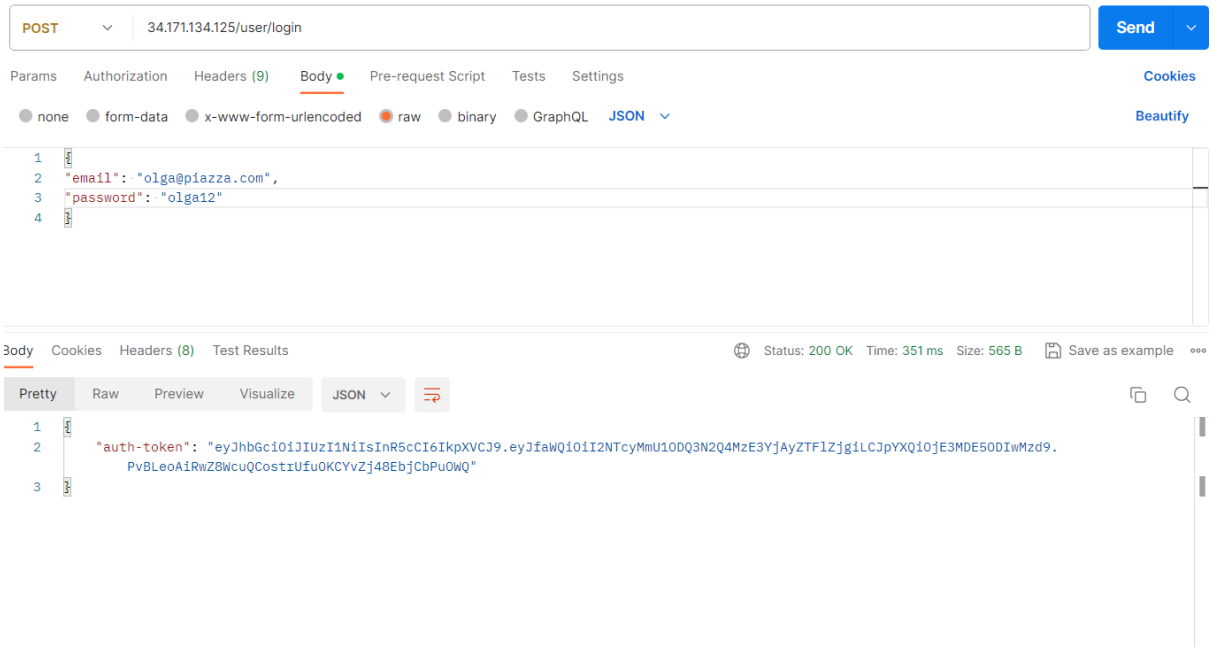
```
_id: ObjectId('65722e58477d8317b02e1ef8')
username: "olga"
email: "olga@piazza.com"
password: "$2a$05$GcLMh1LWhlP47Ue6NejLnu6E2aPgzbMfsrZphJ/aKdtIkmz8BdPZ6"
date: 2023-12-07T20:43:04.334+00:00
__v: 0
```

```
_id: ObjectId('65722e8866bbe631b071d832')
username: "Nick"
email: "Nick@piazza.com"
password: "$2a$05$40ZvnJweyVSXypFqm3vFr.qqJ3L.9mDxK5KV7N48MbNVvvAveBxZ2"
date: 2023-12-07T20:43:52.436+00:00
__v: 0
```

```
_id: ObjectId('65722eb6477d8317b02e1efb')
username: "Mary"
email: "Mary@piazza.com"
password: "$2a$05$KVmn7PEVmapq4gCcx7jhduaGEbcH.Xrqx92dZvMjG55Z2OtSjUJDO"
date: 2023-12-07T20:44:38.717+00:00
__v: 0
```

```
_id: ObjectId('65722ed466bbe631b071d835')
username: "Nestor"
email: "Nestor@piazza.com"
password: "$2a$05$gdjTVoPLLSQd/gdLrW5AUOm4du5jOvA56Ix3fN5Mi49yPVgvVILLm"
date: 2023-12-07T20:45:08.964+00:00
__v: 0
```


TC2. Olga, Nick, Mary, and Nestor use the oAuth v2 authorisation service to register and get their tokens.



Deepa Karthick
Candidate Num: Y106847

POST 34.171.134.125/user/login

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"email": "Nick@piazza.com",
"password": "Nick12"
}
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 317 ms Size: 565 B Sav

Pretty Raw Preview Visualize JSON


```
{
  "auth-token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NTcyMmU4ODY2YmJlNjMxYjA3MWQ4MzIiLCJpYXQiOjE3MDE5ODIwODZ9.
    JcWk-VCWCuo5I12_9Kxht8CJAJ0fEIr7F-wxBogjhRg"
}
```


POST 34.171.134.125/user/login

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"email": "Mary@piazza.com",
"password": "Mary12"
}
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 411 ms Size: 565 B Sav

Pretty Raw Preview Visualize JSON


```
{
  "auth-token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NTcyMmViNjQ3N2Q4MzE3YjAyZTFlZmIiLCJpYXQiOjE3MDE5ODIxMjR9.
    y-f_j_ky3GEd2RDPYbB6kJuvENi2MaZ5waof1fmqAOk"
}
```

POST 34.171.134.125/user/login

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"email": "Nestor@piazza.com",
"password": "Nestor"
}
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 365 ms Size: 565 B Save

Pretty Raw Preview Visualize JSON

```
{
    "auth-token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NTcyMmVkNDY2YmJlNjMxYjA3MWQ4MzUiLCJpYXQiOjE3MDE5ODIxNjB9.
    2yu4TH2c1Po71j0wJHYo3PQ_lFQFFsgYxHfX00PBVQM"
}
```

TC3 Olga makes a call to the API without using her token. This call should be unsuccessful as the user is unauthorised.

GET 34.171.134.125/piazzapost

Params Authorization Headers (9) Body Pre-request Script Tests Settings

☑	Content-Type	application/json	
☑	Content-Length	<calculated when request is sent>	
☑	Host	<calculated when request is sent>	
☑	User-Agent	PostmanRuntime/7.35.0	
☑	Accept	*/*	
☑	Accept-Encoding	gzip, deflate, br	
☑	Connection	keep-alive	
☑	auth-token		
	Key	Value	Description

Body Cookies Headers (7) Test Results Status: 401 Unauthorized Time: 256 ms Size: 272 B Save

Pretty Raw Preview Visualize JSON

```
{
    "message": "Access denied"
}
```

Deepa Karthick
Candidate Num: Y106847

T4.Olga posts a message in the Tech topic with an expiration time (e.g. 5 minutes) using her token. After the end of the expiration time, the message will not accept any further user interactions (likes, dislikes, or comments).

Posted a message with 5 min expiration time

POST 34.171.134.125/piazzapost Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
"post_title": "Tech details",
"post_topic": "Tech",
"post_owner": "olga",
"post_time":"2023-12-07T22:35:00.000Z"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 358 ms Size: 538 B Save as example

Pretty Raw Preview Visualize JSON

```
{
    "post_title": "Tech details",
    "post_topic": "Tech",
    "likes_count": 0,
    "dislikes_count": 0,
    "comments": [],
    "post_owner": "olga",
    "post_time": "2023-12-07T22:35:00.000Z",
    "post_owner_id": "65722e58477d8317b02e1ef8",
    "post_status": "live",
    "_id": "6572479a66bbe631b071d838",
    "post_date": "2023-12-07T22:30:50.844Z",
    "__v": 0
}
```

Postbot Runner Start Proxy Cookies Trash

ENG UK 22:30 07/12/2023

Msg in the DB

piazza
- posts
- users

test

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('6572479a66bbe631b071d838')
post_title: "Tech details"
post_topic: "Tech"
likes_count: 0
dislikes_count: 0
comments: Array (empty)
post_owner: "olga"
post_time: 2023-12-07T22:35:00.000+00:00
post_owner_id: "65722e58477d8317b02e1ef8"
post_status: "live"
post_date: 2023-12-07T22:30:50.844+00:00
__v: 0
```

Deepa Karthick
Candidate Num: Y106847

Like a post that is expired

POST 34.171.134.125/piazzapost/6572479a66bbe631b071d838/like

Params Authorization Headers (8) Body Pre-request Script Tests Settings

User-Agent	PostmanRuntime/7.35.0	
Accept	*/*	
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 403 Forbidden Time: 394 ms Size: 298 B Sav

Pretty Raw Preview Visualize JSON

```
1 {
2     "message": "This post has expired and cannot be liked."
3 }
```

Dislike a post that is expired.

POST 34.171.134.125/piazzapost/6572479a66bbe631b071d838/dislikes

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

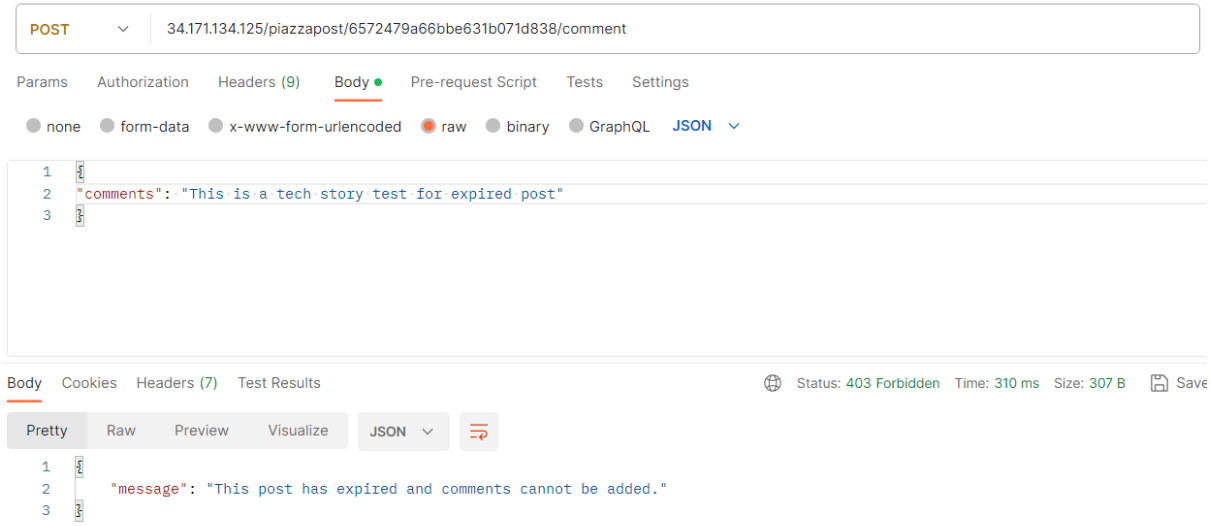
Body Cookies Headers (7) Test Results Status: 403 Forbidden Time: 325 ms Size: 301 B Save

Pretty Raw Preview Visualize JSON

```
1 {
2     "message": "This post has expired and cannot be disliked."
3 }
```

Deepa Karthick
Candidate Num: Y106847

Add comment to expired post



T5.Nick posts a message in the Tech topic with an expiration time using his token.

POST 34.171.134.125/piazzapost

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"post_title": "Tech details-Nick",
"post_topic": "Tech",
"post_owner": "Nick",
"post_time":"2023-12-09T22:30:00.000Z"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 326 ms Size: 543 B Save

Pretty Raw Preview Visualize JSON

```
{
    "post_title": "Tech details-Nick",
    "post_topic": "Tech",
    "likes_count": 0,
    "dislikes_count": 0,
    "comments": [],
    "post_owner": "Nick",
    "post_time": "2023-12-09T22:30:00.000Z",
    "post_owner_id": "65722e8866bbe631b071d832",
    "post_status": "live",
    "_id": "657365e434b5b817a893d605",
    "post_date": "2023-12-08T18:52:20.114Z",
    "__v": 0
}
```

T6.Mary posts a message in the Tech topic with an expiration time using her token.

POST 34.171.134.125/piazzapost

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"post_title": "Tech details-Nick",
"post_topic": "Tech",
"post_owner": "Mary",
"post_time":"2023-12-09T22:30:00.000Z"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 306 ms Size: 543 B Save

Pretty Raw Preview Visualize JSON

```
{
    "post_title": "Tech details-Nick",
    "post_topic": "Tech",
    "likes_count": 0,
    "dislikes_count": 0,
    "comments": [],
    "post_owner": "Mary",
    "post_time": "2023-12-09T22:30:00.000Z",
    "post_owner_id": "65722eb6477d8317b02e1efb",
    "post_status": "live",
    "_id": "6573668734b5b817a893d607",
    "post_date": "2023-12-08T18:55:03.676Z",
    "__v": 0
}
```

TC7. Nick and Olga browse all the available posts in the Tech topic; three posts should be available with zero likes, zero dislikes and no comments.

Deepa Karthick
Candidate Num: Y106847

Nick browses Messages in Tech Topic using his login

GET http://localhost:3000/piazzapost?topic=Tech

Params Authorization Headers (7) Body Pre-request Script Tests Settings

User-Agent	PostmanRuntime/7.35.0	
Accept	*/*	
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 770 ms Size: 1.13 KB Save

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6572479a66bbe631b071d838",
        "post_title": "Tech details",
        "post_topic": "Tech",
        "likes_count": 0,
        "dislikes_count": 0,
        "comments": [],
        "post_owner": "olga",
        "post_time": "2023-12-07T22:35:00.000Z",
        "post_owner_id": "65722e58477d8317b02e1ef8",
        "post_status": "expired",
        "post_date": "2023-12-07T22:30:50.844Z",
        "__v": 0
    },
```

```
        "_id": "657365e434b5b817a893d605",
        "post_title": "Tech details-Nick",
        "post_topic": "Tech",
        "likes_count": 0,
        "dislikes_count": 0,
        "comments": [],
        "post_owner": "Nick",
        "post_time": "2023-12-09T22:30:00.000Z",
        "post_owner_id": "65722e8866bbe631b071d832",
        "post_status": "live",
        "post_date": "2023-12-08T18:52:20.114Z",
        "__v": 0
    },
    {
        "_id": "6573668734b5b817a893d607",
        "post_title": "Tech details-Nick",
        "post_topic": "Tech",
        "likes_count": 0,
        "dislikes_count": 0,
        "comments": [],
        "post_owner": "Mary",
        "post_time": "2023-12-09T22:30:00.000Z",
        "post_owner_id": "65722eb6477d8317b02e1efb",
        "post_status": "live",
        "post_date": "2023-12-08T18:55:03.676Z",
        "__v": 0
    }
```

Deepa Karthick
Candidate Num: Y106847

Olga browses Messages in Tech Topic

GET http://localhost:3000/piazzapost?topic=Tech

Params Authorization Headers (7) Body Pre-request Script Tests Settings

☑	auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...
	Key	Value

Body Cookies Headers (7) Test Results Status: 200 OK Time: 117 ms Size: 1.13 KB Save

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6572479a66bbe631b071d838",
        "post_title": "Tech details",
        "post_topic": "Tech",
        "likes_count": 0,
        "dislikes_count": 0,
        "comments": [],
        "post_owner": "olga",
        "post_time": "2023-12-07T22:35:00.000Z",
        "post_owner_id": "65722e58477d8317b02e1ef8",
        "post_status": "expired",
        "post_date": "2023-12-07T22:30:50.844Z",
        "__v": 0
    },
    {
        "_id": "657365e434b5b817a893d605",
        "post_title": "Tech details-Nick",
        "post_topic": "Tech",
        "likes_count": 0,
        "dislikes_count": 0,
        "comments": [],
        "post_owner": "Nick",
        "post_time": "2023-12-09T22:30:00.000Z",
        "post_owner_id": "65722e8866bbe631b071d832",
        "post_status": "live",
        "post_date": "2023-12-08T18:52:20.114Z",
        "__v": 0
    },
    {
        "_id": "6573668734b5b817a893d607",
        "post_title": "Tech details-Nick",
        "post_topic": "Tech",
        "likes_count": 2,
        "dislikes_count": 0,
        "comments": [],
        "post_owner": "Mary",
        "post_time": "2023-12-09T22:30:00.000Z",
        "post_owner_id": "65722eb6477d8317b02e1efb",
        "post_status": "live",
        "post_date": "2023-12-08T18:55:03.676Z",
        "__v": 0
```

TC8. Nick and Olga “like” Mary’s post on the Tech topic.

Nick likes Mary’s post on the tech topic.

POST 34.171.134.125/piazzapost/6573668734b5b817a893d607/like

Params Authorization Headers (8) Body Pre-request Script Tests Settings

☑	User-Agent	PostmanRuntime/7.35.0	
☑	Accept	*/*	
☑	Accept-Encoding	gzip, deflate, br	
☑	Connection	keep-alive	
☑	auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
	Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 438 ms Size: 543 B Sav

Pretty Raw Preview Visualize JSON


```
{
    "_id": "6573668734b5b817a893d607",
    "post_title": "Tech details-Nick",
    "post_topic": "Tech",
    "likes_count": 1,
    "dislikes_count": 0,
    "comments": [],
    "post_owner": "Mary",
    "post_time": "2023-12-09T22:30:00.000Z",
    "post_owner_id": "65722eb6477d8317b02e1efb",
    "post_status": "live",
    "post_date": "2023-12-08T18:55:03.676Z",
    "__v": 0
}
```


Olga like’s Mary’s post on the tech topic

POST 34.171.134.125/piazzapost/6573668734b5b817a893d607/like

Params Authorization Headers (8) Body Pre-request Script Tests Settings

☑	User-Agent	PostmanRuntime/7.35.0	
☑	Accept	*/*	
☑	Accept-Encoding	gzip, deflate, br	
☑	Connection	keep-alive	
☑	auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
	Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 352 ms Size: 543 B Save

Pretty Raw Preview Visualize JSON


```
{
    "_id": "6573668734b5b817a893d607",
    "post_title": "Tech details-Nick",
    "post_topic": "Tech",
    "likes_count": 2,
    "dislikes_count": 0,
    "comments": [],
    "post_owner": "Mary",
    "post_time": "2023-12-09T22:30:00.000Z",
    "post_owner_id": "65722eb6477d8317b02e1efb",
    "post_status": "live",
    "post_date": "2023-12-08T18:55:03.676Z",
    "__v": 0
}
```


TC9.Nestor “likes” Nick’s post and “dislikes” Mary’s on the Tech topic

Nestor ‘likes’ Nick’s post

POST 34.171.134.125/piazzapost/657365e434b5b817a893d605/like

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Host	<calculated when request is sent>	
User-Agent	PostmanRuntime/7.35.0	
Accept	*/*	
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 340 ms Size: 543 B Sav

Pretty Raw Preview Visualize JSON


```
{
    "_id": "657365e434b5b817a893d605",
    "post_title": "Tech details-Nick",
    "post_topic": "Tech",
    "likes_count": 1,
    "dislikes_count": 0,
    "comments": [],
    "post_owner": "Nick",
    "post_time": "2023-12-09T22:30:00.000Z",
    "post_owner_id": "65722e8866bbe631b071d832",
    "post_status": "live",
    "post_date": "2023-12-08T18:52:20.114Z",
    "__v": 0
}
```


Nestor dislikes 'Mary's' post

POST 34.171.134.125/piazzapost/6573668734b5b817a893d607/dislikes

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

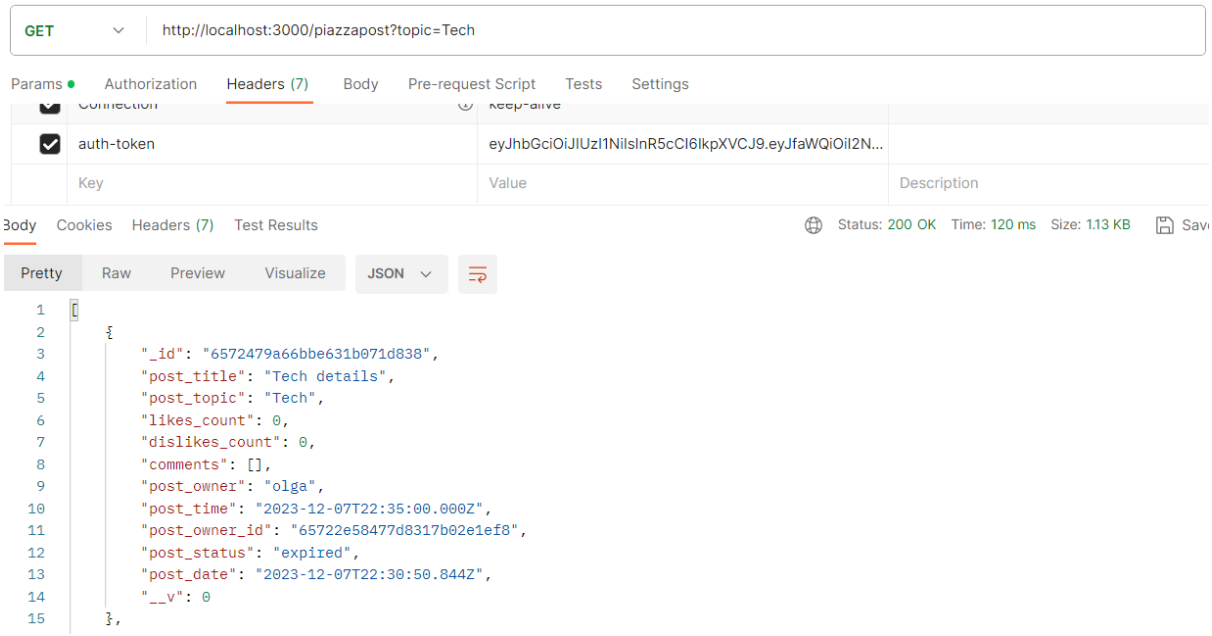
Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 418 ms Size: 543 B Save

Pretty Raw Preview Visualize JSON


```
{
    "_id": "6573668734b5b817a893d607",
    "post_title": "Tech details-Nick",
    "post_topic": "Tech",
    "likes_count": 2,
    "dislikes_count": 1,
    "comments": [],
    "post_owner": "Mary",
    "post_time": "2023-12-09T22:30:00.000Z",
    "post_owner_id": "65722eb6477d8317b02e1efb",
    "post_status": "live",
    "post_date": "2023-12-08T18:55:03.676Z",
    "__v": 0
}
```

TC10. Nick browses all the available posts on the Tech topic; at this stage, he can see the number of likes and dislikes for each post (Mary has two likes and one dislike, and Nick has one like). There are no comments made yet.



GET http://localhost:3000/piazzapost?topic=Tech

Params Authorization Headers (7) Body Pre-request Script Tests Settings

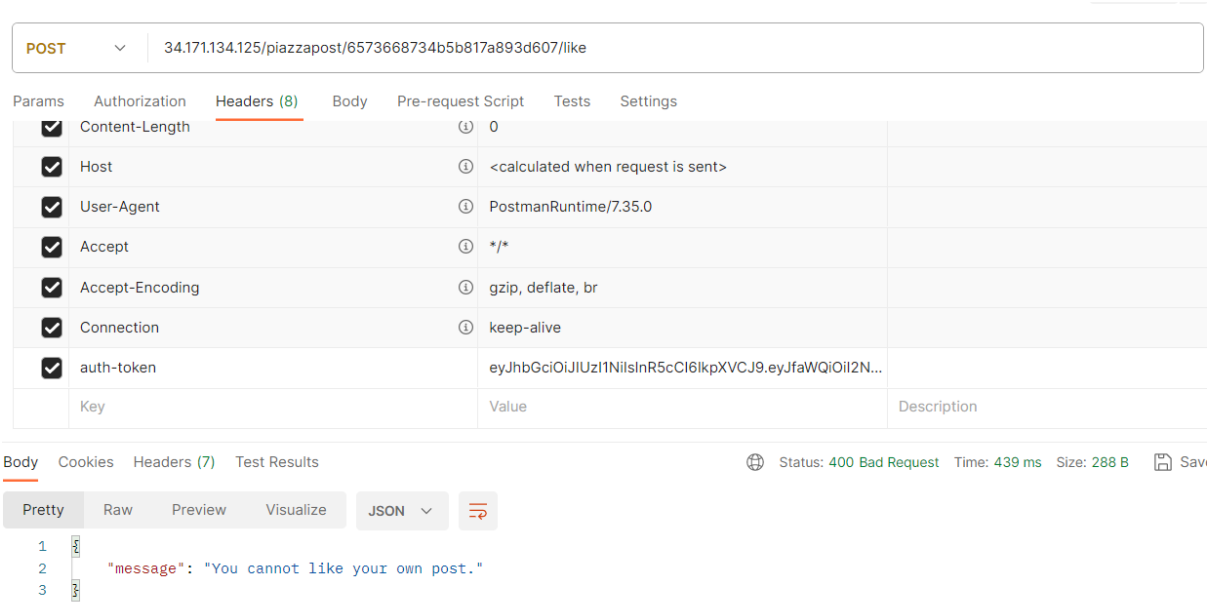
Connection	keep-alive	
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 120 ms Size: 1.13 KB Sav

Pretty Raw Preview Visualize JSON


```
17        "_id": "657365e434b5b817a893d605",
18        "post_title": "Tech details-Nick",
19        "post_topic": "Tech",
20        "likes_count": 1,
21        "dislikes_count": 0,
22        "comments": [],
23        "post_owner": "Nick",
24        "post_time": "2023-12-09T22:30:00.000Z",
25        "post_owner_id": "65722e8866bbe631b071d832",
26        "post_status": "live",
27        "post_date": "2023-12-08T18:52:20.114Z",
28        "__v": 0
29    },
30    {
31        "_id": "6573668734b5b817a893d607",
32        "post_title": "Tech details-Nick",
33        "post_topic": "Tech",
34        "likes_count": 2,
35        "dislikes_count": 1,
36        "comments": [],
37        "post_owner": "Mary",
38        "post_time": "2023-12-09T22:30:00.000Z",
39        "post_owner_id": "65722eb6477d8317b02e1efb",
40        "post_status": "live",
41        "post_date": "2023-12-08T18:55:03.676Z",
42        "__v": 0
43    }
```


TC11. Mary likes her post on the Tech topic. This call should be unsuccessful; in Piazza, a post owner cannot like their messages.



TC12. Nick and Olga comment on Mary's post on the Tech topic in a round-robin fashion (one after the other, adding at least two comments each)

Nick comment on Mary's post

POST 34.171.134.125/piazzapost/6573668734b5b817a893d607/comment

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"comments": "This is Nick's second's comment on Mary post"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 350 ms Size: 631 B Sav

Pretty Raw Preview Visualize JSON


```
{
    "_id": "6573668734b5b817a893d607",
    "post_title": "Tech details-Nick",
    "post_topic": "Tech",
    "likes_count": 2,
    "dislikes_count": 1,
    "comments": [
        "This is Nick first comment on Mary post",
        "This is Nick's second's comment on Mary post"
    ],
    "post_owner": "Mary",
    "post_time": "2023-12-09T22:30:00.000Z",
    "post_owner_id": "65722eb6477d8317b02e1efb",
    "post_status": "live",
    "post_date": "2023-12-08T18:55:03.676Z",
    "__v": 2
}
```

Olga comment on Mary's post

POST 34.171.134.125/piazzapost/6573668734b5b817a893d607/comment

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"comments": "This is Olga's first comment on Mary post"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 320 ms Size: 722 B Save

Pretty Raw Preview Visualize JSON


```
{
    "_id": "6573668734b5b817a893d607",
    "post_title": "Tech details-Nick",
    "post_topic": "Tech",
    "likes_count": 2,
    "dislikes_count": 1,
    "comments": [
        "This is Nick first comment on Mary post",
        "This is Nick's second's comment on Mary post",
        "This is Olga's second comment on Mary's post",
        "This is Olga's first comment on Mary post"
    ],
    "post_owner": "Mary",
    "post_time": "2023-12-09T22:30:00.000Z",
    "post_owner_id": "65722eb6477d8317b02e1efb",
    "post_status": "live",
    "post_date": "2023-12-08T18:55:03.676Z",
    "__v": 4
}
```


TC13.Nick browses all the available posts in the Tech topic; at this stage, he can see the number of likes and dislikes of each post and the comments made.

Deepa Karthick
Candidate Num: Y106847

GET http://localhost:3000/piazzapost?topic=Tech

Params Authorization Headers (7) Body Pre-request Script Tests Settings

auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 122 ms Size: 1.31 KB Save

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6572479a66bbe631b071d838",
        "post_title": "Tech details",
        "post_topic": "Tech",
        "likes_count": 0,
        "dislikes_count": 0,
        "comments": [],
        "post_owner": "olga",
        "post_time": "2023-12-07T22:35:00.000Z",
        "post_owner_id": "65722e58477d8317b02e1ef8",
        "post_status": "expired",
        "post_date": "2023-12-07T22:30:50.844Z",
        "__v": 0
    },
    {
```


GET http://localhost:3000/piazzapost?topic=Tech

Params Authorization Headers (7) Body Pre-request Script Tests Settings

auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 122 ms Size: 1.31 KB Save

Pretty Raw Preview Visualize JSON


```
    {
        "_id": "657365e434b5b817a893d605",
        "post_title": "Tech details-Nick",
        "post_topic": "Tech",
        "likes_count": 1,
        "dislikes_count": 0,
        "comments": [],
        "post_owner": "Nick",
        "post_time": "2023-12-09T22:30:00.000Z",
        "post_owner_id": "65722e8866bbe631b071d832",
        "post_status": "live",
        "post_date": "2023-12-08T18:52:20.114Z",
        "__v": 0
    },
    {
        "_id": "6573668734b5b817a893d607",
        "post_title": "Tech details-Nick",
        "post_topic": "Tech",
        "likes_count": 2,
        "dislikes_count": 1,
        "comments": [
            "This is Nick first comment on Mary post",
            "This is Nick's second's comment on Mary post",
            "This is Olga's second comment on Mary's post",
            "This is Olga's first comment on Mary post"
        ],
        "post_owner": "Mary",
```

TC14. Nestor posts a message in the Health topic with an expiration time using her token

POST 34.171.134.125/piazzapost

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"post_title": "Health details-Nestor",
"post_topic": "Health",
"post_owner": "Nestor",
"post_time":"2023-12-08T20:10:00.000Z"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 363 ms Size: 551 B Save

Pretty Raw Preview Visualize JSON

```
{
    "post_title": "Health details-Nestor",
    "post_topic": "Health",
    "likes_count": 0,
    "dislikes_count": 0,
    "comments": [],
    "post_owner": "Nestor",
    "post_time": "2023-12-08T20:10:00.000Z",
    "post_owner_id": "65722ed466bbe631b071d835",
    "post_status": "live",
    "_id": "6573765ffa095d6cfca60151",
    "post_date": "2023-12-08T20:02:39.120Z",
    "__v": 0
}
```

TC15. Mary browses all the available posts on the health topic; at this stage, she can see only Nestor's post.

GET http://localhost:3000/piazzapost?topic=Health

Params Authorization Headers (7) Body Pre-request Script Tests Settings

☑ auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 117 ms Size: 553 B Sav

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6573765ffa095d6cfca60151",
        "post_title": "Health details-Nestor",
        "post_topic": "Health",
        "likes_count": 0,
        "dislikes_count": 0,
        "comments": [],
        "post_owner": "Nestor",
        "post_time": "2023-12-08T20:10:00.000Z",
        "post_owner_id": "65722ed466bbe631b071d835",
        "post_status": "live",
        "post_date": "2023-12-08T20:02:39.120Z",
        "__v": 0
    }
]
```


TC16. Mary posts a comment in Nestor's message on the Health topic.

POST 34.171.134.125/piazzapost/6573765ffa095d6cfca60151/comment

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON


```
{
"comments": "This is Mary's first comment on Nestor health post"
}
```


Body Cookies Headers (7) Test Results Status: 200 OK Time: 321 ms Size: 603 B Save

Pretty Raw Preview Visualize JSON


```
{
    "_id": "6573765ffa095d6cfca60151",
    "post_title": "Health details-Nestor",
    "post_topic": "Health",
    "likes_count": 0,
    "dislikes_count": 0,
    "comments": [
        "This is Mary's first comment on Nestor health post"
    ],
    "post_owner": "Nestor",
    "post_time": "2023-12-08T20:10:00.000Z",
    "post_owner_id": "65722ed466bbe631b071d835",
    "post_status": "live",
    "post_date": "2023-12-08T20:02:39.120Z",
    "__v": 1
}
```

TC17. Mary dislikes Nestor's message on the health topic after the end of post-expiration time. This should fail.

POST 34.171.134.125/piazzapost/6573765ffa095d6cfca60151/dislikes

Params Authorization Headers (9) Body Pre-request Script Tests Settings

☑	User-Agent	PostmanRuntime/7.35.0	
☑	Accept	*/*	
☑	Accept-Encoding	gzip, deflate, br	
☑	Connection	keep-alive	
☑	auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
	Key	Value	Description

Body Cookies Headers (7) Test Results Status: 403 Forbidden Time: 332 ms Size: 301 B Sav

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "This post has expired and cannot be disliked."
3 }
```

TC18. Nestor browses all the messages on the Health topic. There should be only one post (his own) with one comment (Mary's).

GET http://localhost:3000/piazzapost?topic=Health

Params Authorization Headers (7) Body Pre-request Script Tests Settings

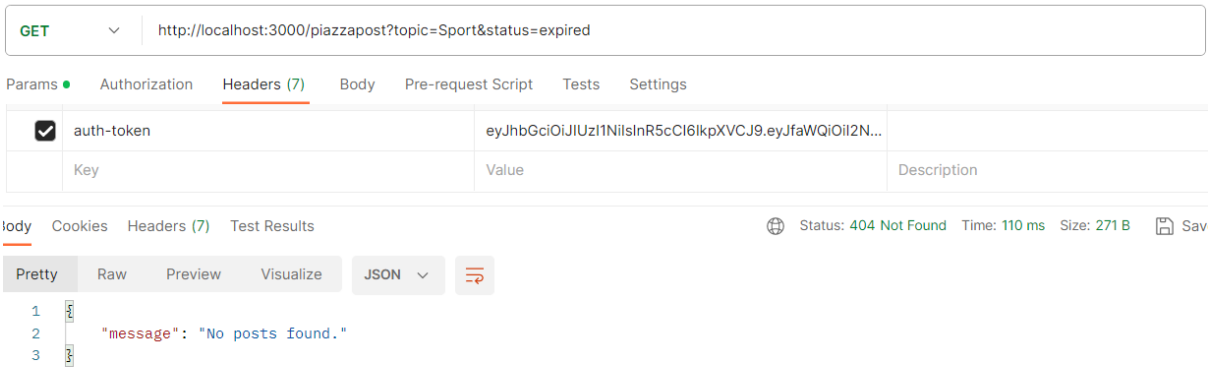
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 112 ms Size: 608 B Sav

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6573765ffa095d6cfca60151",
        "post_title": "Health details-Nestor",
        "post_topic": "Health",
        "likes_count": 0,
        "dislikes_count": 0,
        "comments": [
            "This is Mary's first comment on Nestor health post"
        ],
        "post_owner": "Nestor",
        "post_time": "2023-12-08T20:10:00.000Z",
        "post_owner_id": "65722ed466bbe631b071d835",
        "post_status": "expired",
        "post_date": "2023-12-08T20:02:39.120Z",
        "__v": 1
    }
]
```

TC19. Nick browses all the expired messages on the Sports topic. These should be empty



TC20. Nestor queries for an active post with the highest interest (maximum number of likes and dislikes) in the Tech topic. This should be Mary's post.

GET 34.171.134.125/piazzapost/trending

Params Authorization Headers (9) Body Pre-request Script Tests Settings

☑ auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 372 ms Size: 1.75 KB Sav

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6573668734b5b817a893d607",
        "post_title": "Tech details-Nick",
        "post_topic": "Tech",
        "likes_count": 2,
        "dislikes_count": 1,
        "comments": [
            "This is Nick first comment on Mary post",
            "This is Nick's second's comment on Mary post",
            "This is Olga's second comment on Mary's post",
            "This is Olga's first comment on Mary post"
        ],
        "post_owner": "Mary",
        "post_time": "2023-12-09T22:30:00.000Z",
        "post_owner_id": "65722eb6477d8317b02e1efb",
        "post_status": "live",
        "post_date": "2023-12-08T18:55:03.676Z",
        "__v": 4,
        "likesAndDislikesSum": 3
    },
```

GET 34.171.134.125/piazzapost/trending

Params Authorization Headers (9) Body Pre-request Script Tests Settings

☑ auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2N...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 372 ms Size: 1.75 KB Sav

Pretty Raw Preview Visualize JSON


```
23        "_id": "657365e434b5b817a893d605",
24        "post_title": "Tech details-Nick",
25        "post_topic": "Tech",
26        "likes_count": 1,
27        "dislikes_count": 0,
28        "comments": [],
29        "post_owner": "Nick",
30        "post_time": "2023-12-09T22:30:00.000Z",
31        "post_owner_id": "65722e8866bbe631b071d832",
32        "post_status": "live",
33        "post_date": "2023-12-08T18:52:20.114Z",
34        "__v": 0,
35        "likesAndDislikesSum": 1
36    },
37    {
38        "_id": "6572479a66bbe631b071d838",
39        "post_title": "Tech details",
40        "post_topic": "Tech",
41        "likes_count": 0,
42        "dislikes_count": 0,
43        "comments": [],
44        "post_owner": "olga",
45        "post_time": "2023-12-07T22:35:00.000Z",
46        "post_owner_id": "65722e58477d8317b02e1ef8",
47        "post_status": "live",
48        "post_date": "2023-12-07T22:30:50.844Z",
49        "__v": 0,
50        "likesAndDislikesSum": 0
```


Deploy your Piazza project into a VM using Docker

1.Code moved to Github

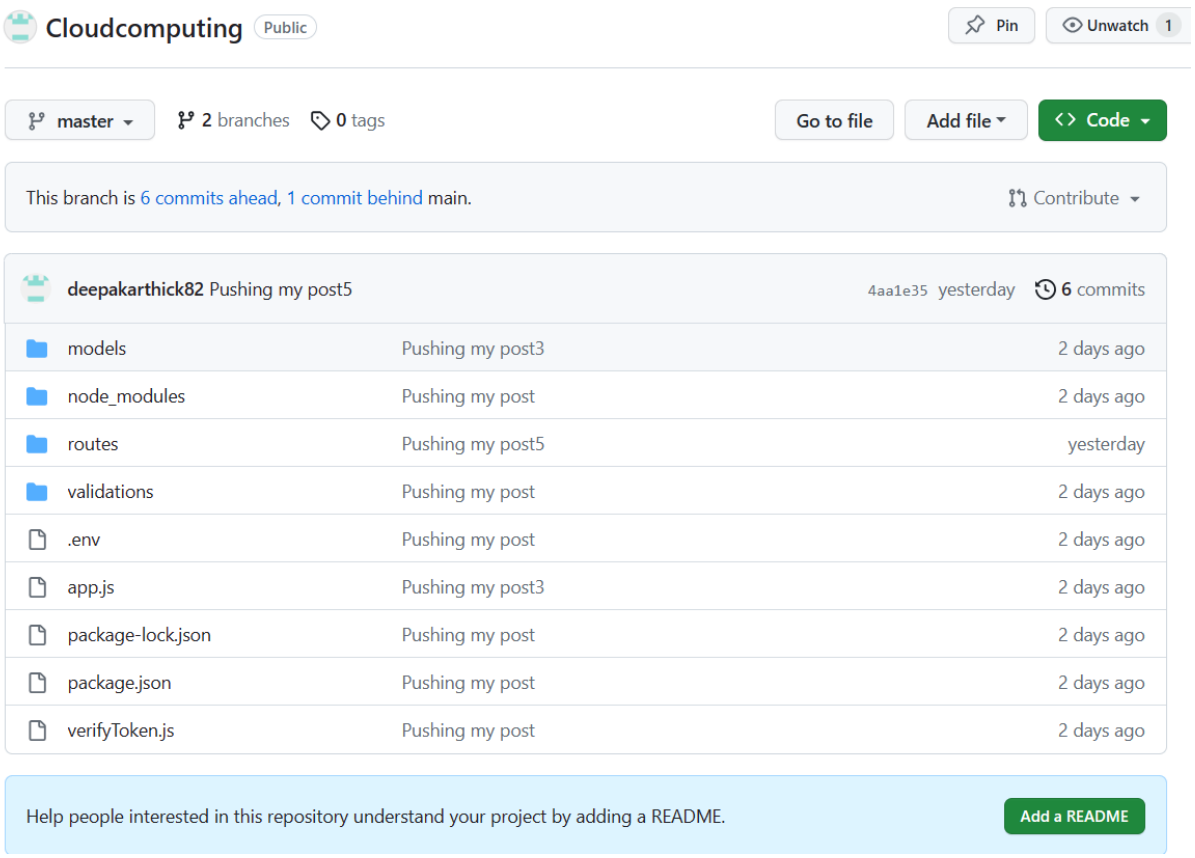
1. Git Init
2. git remote add origin https://github.com/deepakarthick82/Cloudcomputing.git
3. git add .
4. git commit -m "Pushing my post"
5. git push -f origin master
6. git push
7. Git hub repository link

2.Github repo

The following screenshot shows the git commands used to upload the project into github repo.

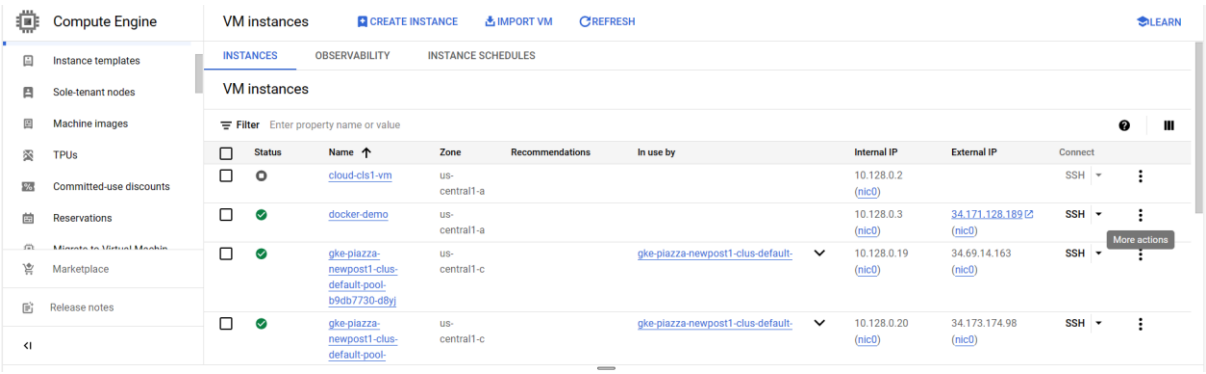
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[master 4aa1e35] Pushing my post5
 1 file changed, 3 insertions(+), 2 deletions(-)

karth@Diya MINGW64 ~/OneDrive/Deepa/Birbeck/cloud computing/coursework_2nd/Coursework_3 (master)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 413 bytes | 413.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/deepakarthick82/Cloudcomputing.git
```



Code pushed to github

3.Created a VM in GCP virtual machine called docker-vm is created on google cloud.



Created VM named docker-demo in GCP

4.Command to push github code to VM:

git clone git@github.com:deepakarthick82/Cloudcomputing.git

5.Code is pushed from Git to the VM and the code is checked out.

```
docker-user@docker-demo:~/Cloudcomputing$ ls
'Cloud Computing Coursework 2023 Piazza.pdf'   app.js   node_modules        package.json   validations
 Dockerfile                                    models   package-lock.json   routes         verifyToken.js
docker-user@docker-demo:~/Cloudcomputing$
```

6.Create a docker file named dockerfile using ***pico*** command

```
docker-user@docker-demo:~$ cat Dockerfile
FROM alpine
RUN apk add --update nodejs npm
COPY . /src
WORKDIR /src
EXPOSE 3000
ENTRYPOINT ["node", "./app.js"]
docker-user@docker-demo:~$ ^C
```

Contents of the Dockerfile

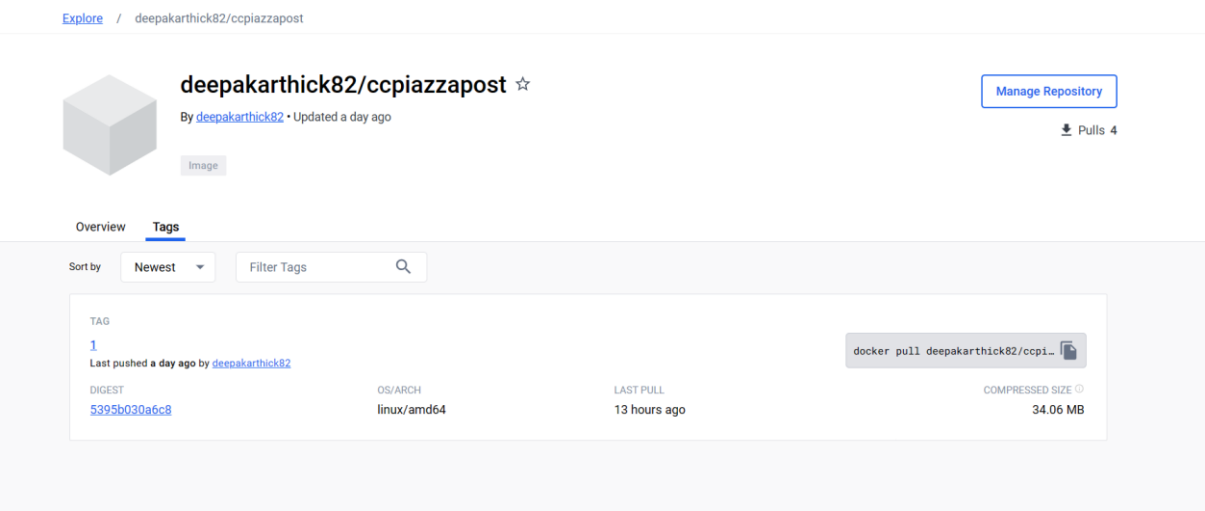
7.Docker image created using the command docker image build -t cloudcomputing-image .

8.Create a container and run it docker container run -d --name deepa-web --publish 80:3000 cloudcomputing-image

9.View created Docker images

```
docker-user@docker-demo:~$ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED        SIZE
deepakarthick82/ccpiazzapost  1         b94a7d9403f4   29 hours ago   94.3MB
mini-python3-image            latest    e6ccdb57cb85   5 days ago     253MB
ubuntu                        latest    b6548eacb063   7 days ago     77.8MB
alpine                        latest    b541f2080109   7 days ago     7.34MB
python                        latest    afb69f3af77f   6 weeks ago    1.02GB
```

10.Pushed the code to docker hub: docker push deepakarthick82/ccpiazzapost:1



Deepa Karthick
Candidate Num: Y106847

Deploy Piazza application in Kubernetes

1.Create Kubernetes Cluster

OVERVIEW OBSERVABILITY COST OPTIMISATION

Filter Enter property name or value

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
✓	piazza-newpost1-cluster	us-central1-c	3	6	12 GB		–

Piazza-newpost1-cluster is created

2.Create deployment.yaml using ***pico*** command in the cloud shell

```
      app: piazza-post
  template:
    metadata:
      labels:
        app: piazza-post
    spec:
      containers:
      - name: piazza-post
        image: deepakarthick82/ccpiazzapost:1
        imagePullPolicy: Always
        ports:
        - containerPort: 3000
deepakarthick2022@cloudshell:~ (cloudclass1-401818)$
```

3.Apply the deployment file using command ***kubectl apply -f deployment.yaml***

PODS details:

```
deepakarthick2022@cloudshell:~ (cloudclass1-401818)$ kubectl get pods
NAME                                      READY   STATUS    RESTARTS   AGE
piazza-post-deployment-5d4549756d-468x4   1/1     Running   0          23h
piazza-post-deployment-5d4549756d-78t2l   1/1     Running   0          23h
piazza-post-deployment-5d4549756d-7w57r   1/1     Running   0          23h
piazza-post-deployment-5d4549756d-djmzh   1/1     Running   0          23h
piazza-post-deployment-5d4549756d-ljgj5   1/1     Running   0          23h
deepakarthick2022@cloudshell:~ (cloudclass1-401818)$
```

4.Create service.yaml using ***pico*** command and apply it using ***kubectl apply -f service.yaml***

```
  labels:
    app: piazza-post-service
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 3000
  selector:
    app: piazza-post
  sessionAffinity: None
deepakarthick2022@cloudshell:~ (cloudclass1-401818)$
```

5.Get the API end points under virtual machine IP address

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cloudclass1-401818.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
deepakarthick2022@cloudshell:~ (cloudclass1-401818)$ kubectl get services
NAME                  TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)        AGE
kubernetes            ClusterIP      10.32.0.1     <none>           443/TCP        2d
piazza-post-service   LoadBalancer   10.32.6.130   34.171.134.125   80:30044/TCP   29h
deepakarthick2022@cloudshell:~ (cloudclass1-401818)$
```

Test cases are triggered in Post man using the External IP

POST 34.171.134.125/user/register

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2 "username": "Nestor",
3 "email": "Nestor@piazza.com",
4 "password": "Nestor"
5 }
```

Body Cookies Headers (7) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2     "username": "Nestor",
3     "email": "Nestor@piazza.com",
4     "password": "$2a$05$gdjTVoPLLSQd/gdLrW5AUOm4du5jOvA56Ix3fN5Mi49yPVgvVILLm",
5     "_id": "65722ed466bbe631b071d835",
6     "date": "2023-12-07T20:45:08.964Z",
7     "__v": 0
8 }
```

POST 34.171.134.125/piazzapost

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"post_title": "Health details-Nestor",
"post_topic": "Health",
"post_owner": "Nestor",
"post_time":"2023-12-08T20:10:00.000Z"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 363 ms Size: 551 B

Pretty Raw Preview Visualize JSON

```
{
    "post_title": "Health details-Nestor",
    "post_topic": "Health",
    "likes_count": 0,
    "dislikes_count": 0,
    "comments": [],
    "post_owner": "Nestor",
    "post_time": "2023-12-08T20:10:00.000Z",
    "post_owner_id": "65722ed466bbe631b071d835",
    "post_status": "live",
    "_id": "6573765ffa095d6cfca60151",
    "post_date": "2023-12-08T20:02:39.120Z",
    "__v": 0
}
```

References

Chris, K. (2021) Rest API best practices – rest endpoint design examples, freeCodeCamp.org. freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/rest-api-best- practices-rest-endpoint-design-examples/ (Accessed: December 11, 2022).

Copes, F. (2022) JWT authentication: Best practices and when to use it, LogRocket Blog. Available at: https://blog.logrocket.com/jwt-authentication-best-practices/ (Accessed: December 11, 2022).

Kaelin, M.W. et al. (2022) How to create a virtual machine in Google Cloud Platform, TechRepublic. Available at: https://www.techrepublic.com/article/how-to-create-a-virtual- machine-in-google-cloud-platform/ (Accessed: December 11, 2022).

MongoDB tutorial: What it is and features - javatpoint (no date) www.javatpoint.com. Available at: https://www.javatpoint.com/mongodb-tutorial (Accessed: December 11, 2022).

Node.js (no date) Node.js. Available at: https://nodejs.org/en/ (Accessed: December 11, 2022).