Rushiraj Herma rherma@stevens.edu

 # the URL of your public GitHub repo
https://github.com/Rushiraj0608

 # an estimate of how many hours you spent on the project
30 hours

 # a description of how you tested your code

I tested my code while I was implementing the project. I tested my projects starting from small set of code to large. After testing individual functions, after completing all the project I tested whether or not each functions are working together.
After that I used postman to check all my routes, it was fun checking all my routes and getting all the valid expected outputs.

 # bugs and issues
no known bugs or issues

 # resolved issue
In the ****app.py****, Initially I was having an error while deleting invalid post - as in whenever you try to delete post with invalid Id, I was getting an error that says invalid keyword 'user_key' which I was using to delete post with user key. that caused trouble passing all the autograder test cases. After some time I was able to solve that error.

 # Implemented extensions
 # 1) Users and user keys
route : @app.post("/user")

Users and user keys: This extension adds a notion of a user with a private user key, which associates each post with a user. Users can create posts, and it is sufficient to provide the user's key to delete a post. This extension requires modifying existing endpoints and creating new ones. The new endpoints allow creating users and associating a user with a post. When returning information about a post, the user's ID should also be included along with other data.

 # 2) User profiles (needs user)
route : @app.patch("/user/<user_id>")

User profiles (needs user): This extension can only be implemented if the user's notion is already implemented. It adds metadata to users, like a username or an email address. The metadata can be unique or non-unique. The API endpoint lets users retrieve and edit their metadata. When returning information about a post associated with a user, the user's unique metadata should also be included.

 # 3) Date- and time-based range queries
route : @app.post("/post/search")

Date- and time-based range queries: This extension allows users to search for posts based on a date/time. The API endpoint should accept starting date and time, ending date and time, or both, and return a list of post information, including the user, ID, message, and timestamp.

 # 4) User-based range queries (needs user)
route : @app.get("/post/user/<string:id>")

User-based range queries (needs user): This extension can only be implemented if the user's notion is already implemented. It lets users search for posts by a given user. The API endpoint should return a list of post information, including the user, ID, message, and timestamp.

 # 5) Fulltext search
route : @app.get("/post/searchbymsg/<string:msg>")

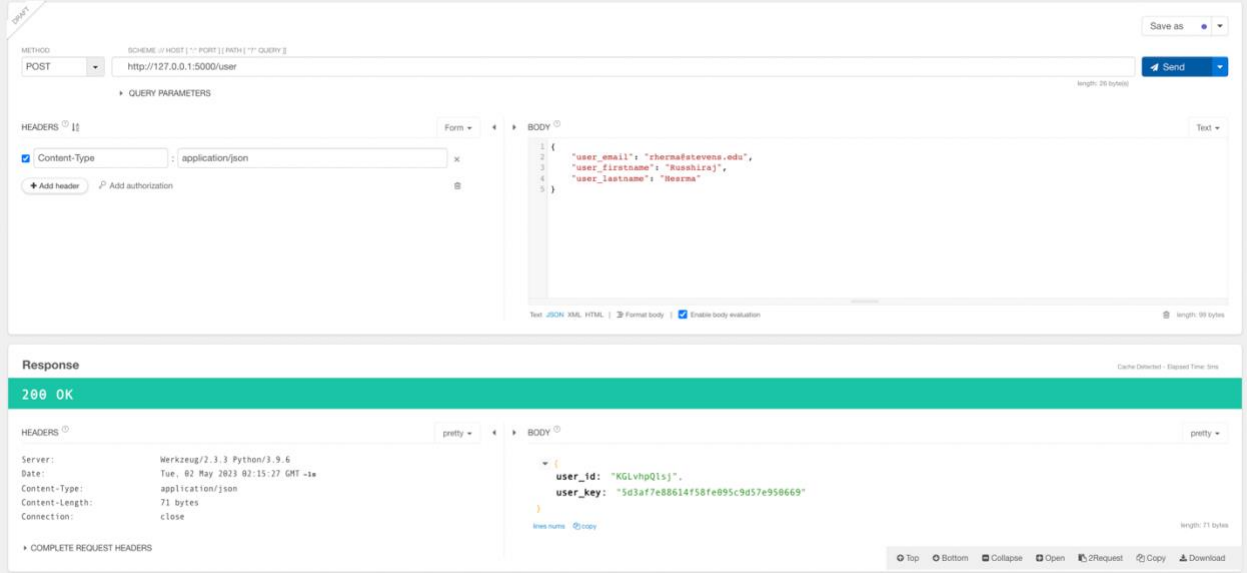
Fulltext search: This extension allows users to search for posts using a full-text search on the contents of the post. The API endpoint should return a list of post information, including the user, ID, message, and timestamp, matching the search query.

 # 5) Detailed summaries of your tests for each of your extensions.

1) Create user:

   To create a user you need to enter user_email, user_firstname, user_lastname as shown in image below. If user has successfully created it will return user_id and user_key, while storing all the details in users json file variable that I loaded as a dictionary.
   Method : Post
   URL : http:/127.0.0.1:5000/user
   Body :
   {
   "user_email": "rherma@stevens.edu",
   "user_firstname": " Russhiraj",
   "user_lastname": "Hesrma"
   }

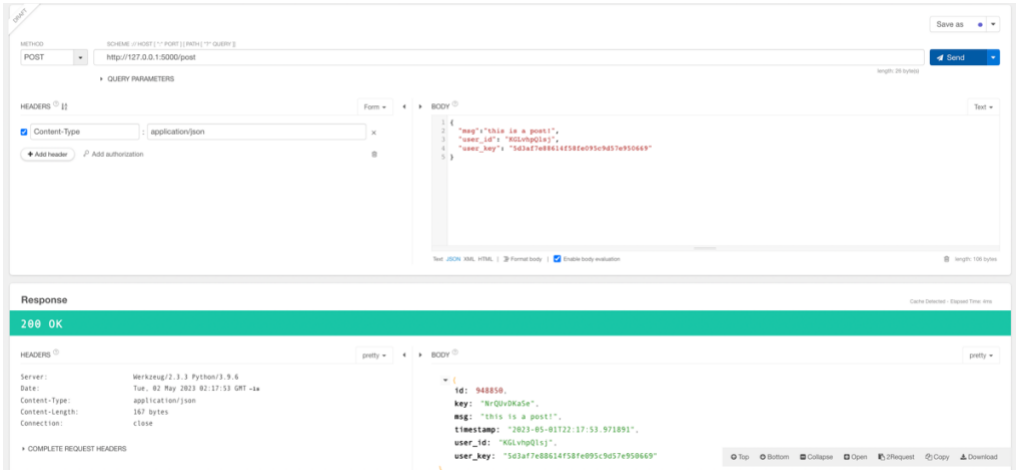
   OUTPUT :
   {
   user_id : “KGLvhpQlsj”
   user_key: “5d3af7e88614f58fe095c9d57e950669”
   }

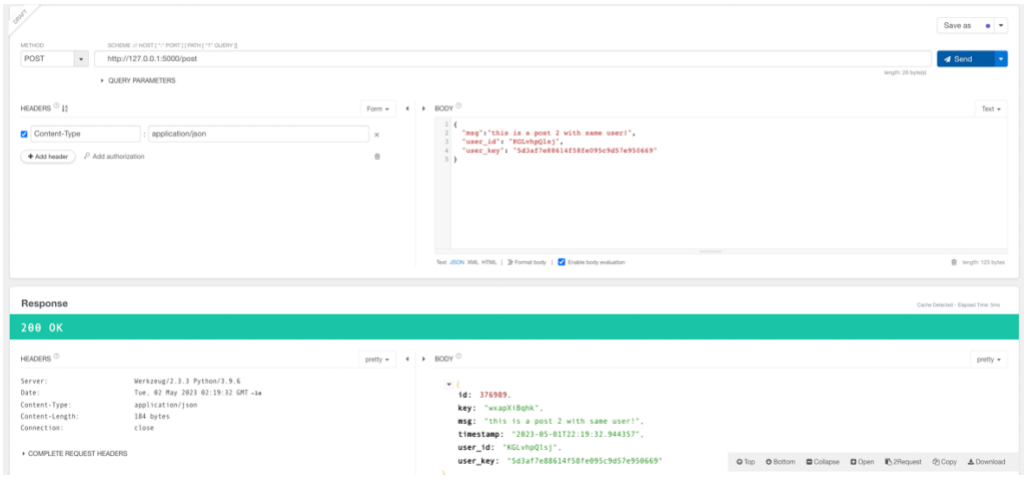


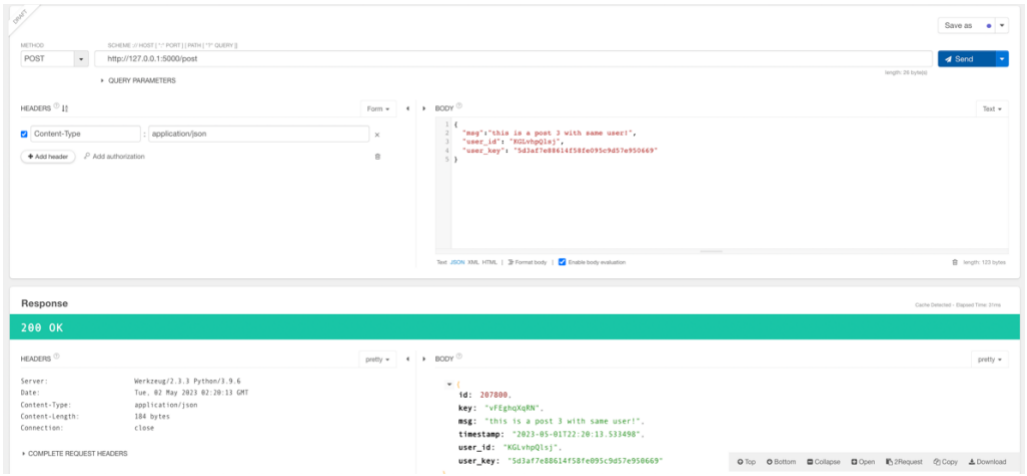
2) Update details of user using patch:
To update user details we are using below function with patch
We need to provide user_email which must be the same, and along with that we are giving the changes that we want to change in the body.
Method : Post
URL : http:/127.0.0.1:5000/user/KGLvhpQlsj
Body :
{
"user_email": "rherma@stevens.edu",
"user_firstname": "F_Name : Russhiraj",
"user_lastname": "L_Name : Hesrma"
}
OUTPUT :
{
"**user_email**": "rherma@stevens.edu",
"**user_firstname**": "F_Name : Russhiraj",
"**user_id**": "KGLvhpQlsj",
"**user_key**": "5d3af7e88614f58fe095c9d57e950669",
"**user_lastname**": "L_Name : Hesrma"
}

3) Create a post with user details:
To create a post with user details, we are using POST request, in body we need to provide the msg for post alongside user_id and user_key. We are getting all the details in return. We can have multiple posts with one user.
Method : Post
URL : http:/127.0.0.1:5000/post
Body :
{
“msg” : “this is a post”,
“user_id” : “KGLvhpQlsj”,
“user_key”: “5d3af7e88614f58fe095c9d57e950669”
}

OUTPUT :
{
“id” : 948850,
“key”:”NrQUvDKaSe”
“msg” : “this is a post”,
“timestamp”:”2023-05-01T22:17:53.971891”
“user_id” : “KGLvhpQlsj”,
“user_key”: “5d3af7e88614f58fe095c9d57e950669”
}



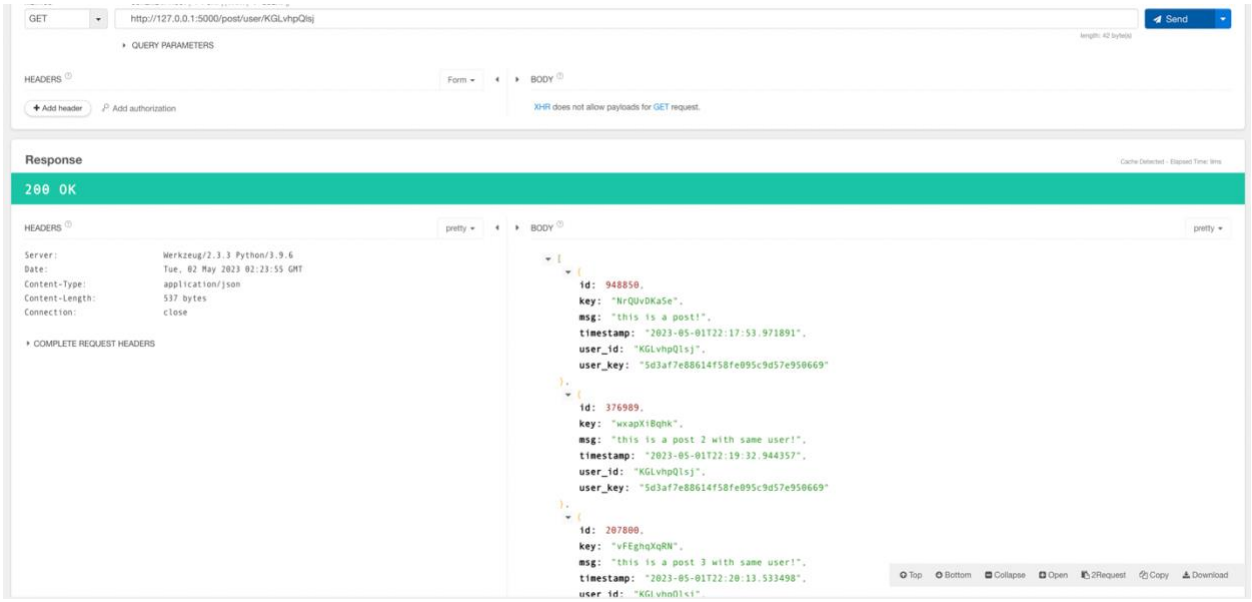




4) Get all posts by the user using user_key:
This is used to get all the posts that are generated by supplied user_id.
Method : GET
URL : http:/127.0.0.1:5000/user/KGLvhpQlsj

OUTPUT : (similar)
{
“id” : 948850,
“key”:”NrQUvDKaSe”
“msg” : “this is a post”,
“timestamp”:”2023-05-01T22:17:53.971891”
“user_id” : “KGLvhpQlsj”,
“user_key”: “5d3af7e88614f58fe095c9d57e950669”

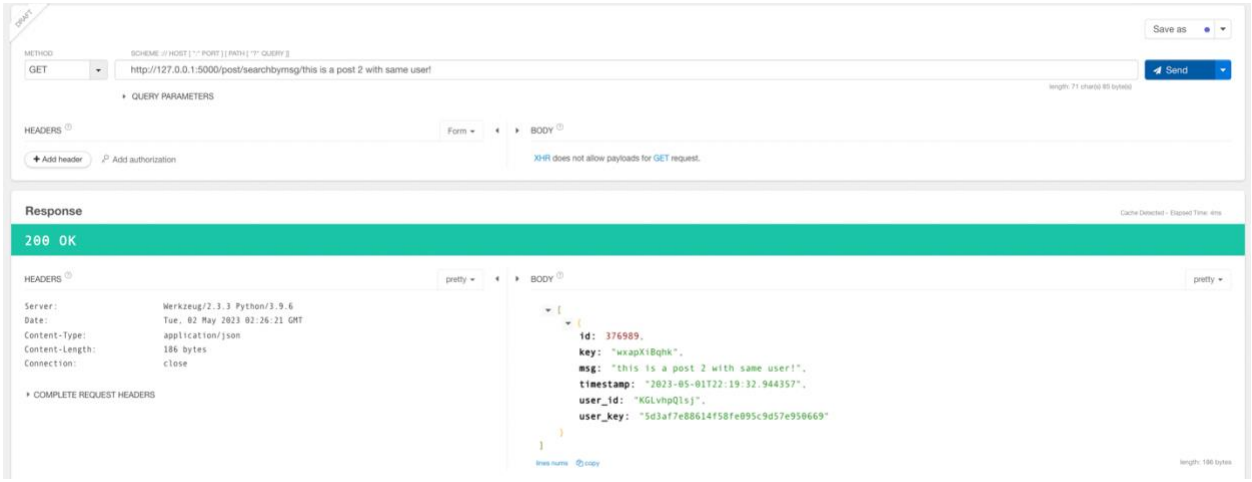
}
{
“id” : 948850,
“key”:”NrQUvDKaSe”
“msg” : “this is a post 2 with same user”,
“timestamp”:”2023-05-01T22:17:53.971891”
“user_id” : “KGLvhpQlsj”,
“user_key”: “5d3af7e88614f58fe095c9d57e950669”
}
{
“id” : 948850,
“key”:”NrQUvDKaSe”
“msg” : “this is a post 3 with same user”,
“timestamp”:”2023-05-01T22:17:53.971891”
“user_id” : “KGLvhpQlsj”,
“user_key”: “5d3af7e88614f58fe095c9d57e950669”
}



5) Search post using message:
We are using this route to search post using the msg string:

Method : GET
URL : http://127.0.0.1:5000/post/searchbymsg/this is a post 2 with same user!

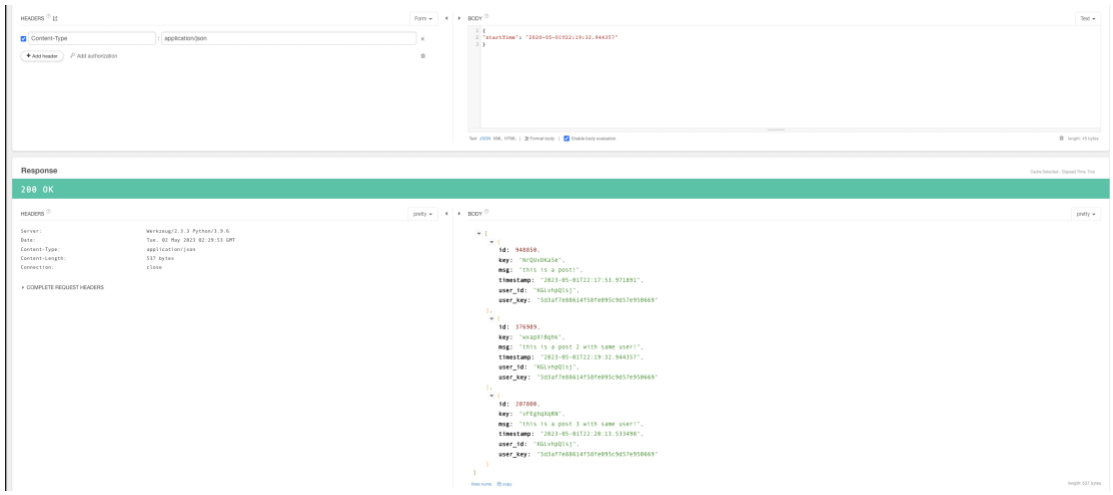
OUTPUT :
[
{
"**id**": 376989,
"**key**": "wxapXiBqhk",
"**msg**": "this is a post 2 with same user!",
"**timestamp**": "2023-05-01T22:19:32.944357",
"**user_id**": "KGLvhpQlsj",
"**user_key**": "5d3af7e88614f58fe095c9d57e950669"
}
]



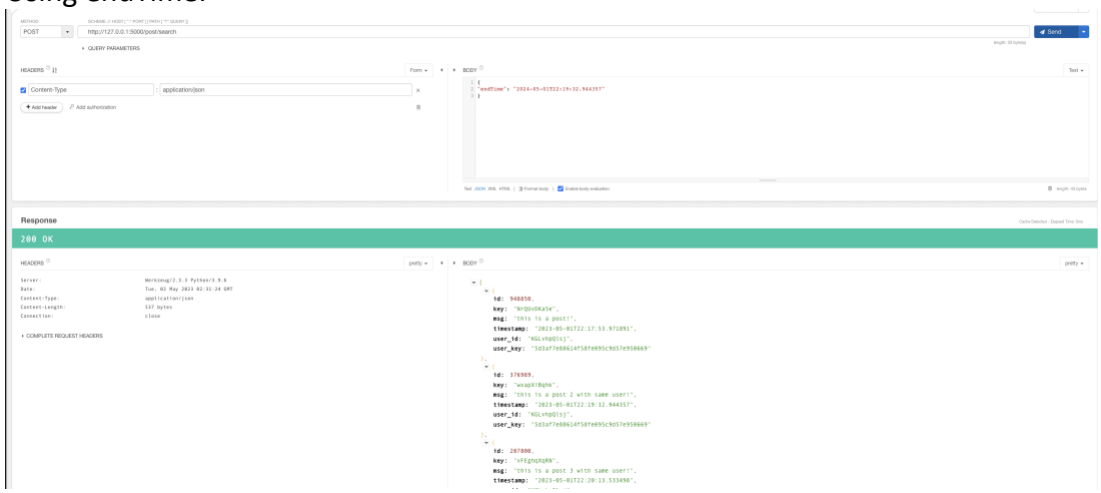
6) Search using start time or end time or both.
We are using this route to search post using startTime or endTime or using both.
Method : Post
URL : http://127.0.0.1:5000/post/search
Body :
{
"startTime": "2020-05-01T22:17:53.971891"

}
Method : Post
URL : http://127.0.0.1:5000/post/search
Body :
{
"endTime": "2024-05-01T22:17:53.971891"
}
Method : Post

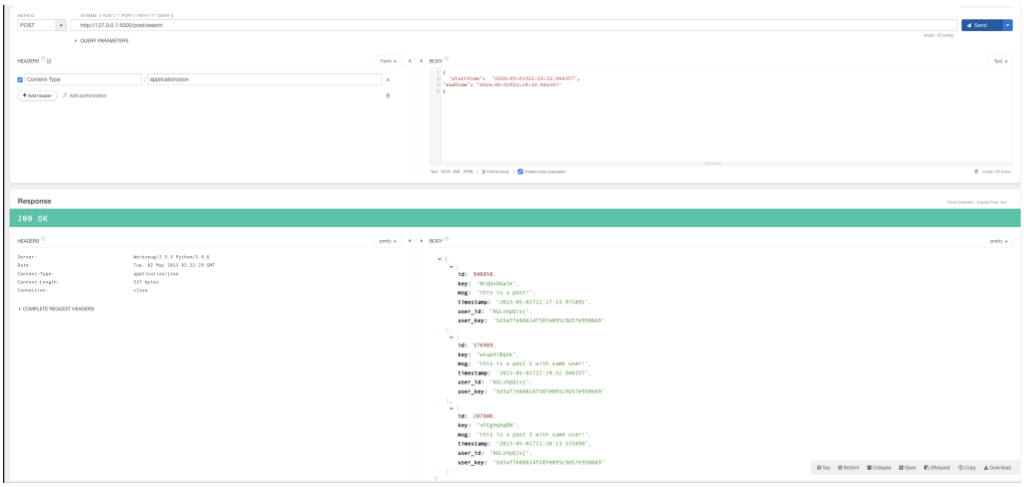
URL : http://127.0.0.1:5000/post/search
Body :
{
“startTime”: ”2020-05-01T22:17:53.971891”
“endTime”: ”2024-05-01T22:17:53.971891”
}



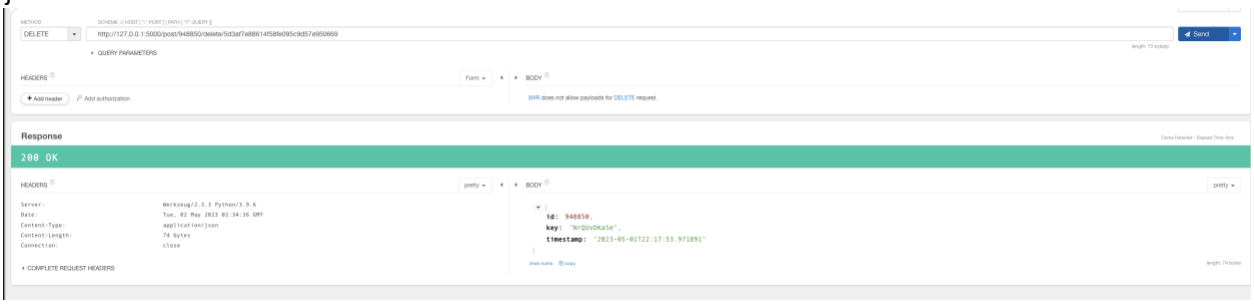
Using endTime:



Using both:



7) Delete post using user_key:
We are using this route to delete the post using user_key instead of post key. And we are also using this route to delete key using post key. But we will get the corresponding message in console, that which key is used.
Method : DELETE
URL: http://127.0.0.1:5000/post/948850/delete/5d3af7e88614f58fe095c9d57e950669
Output:
{
Id: 948850,
Key:”NrQUvDKaSe”
“timestamp”:”2023-05-01T22:17:53.971891”
}



If we delete post using user_key we are printing message “Deleted by user key” in console.