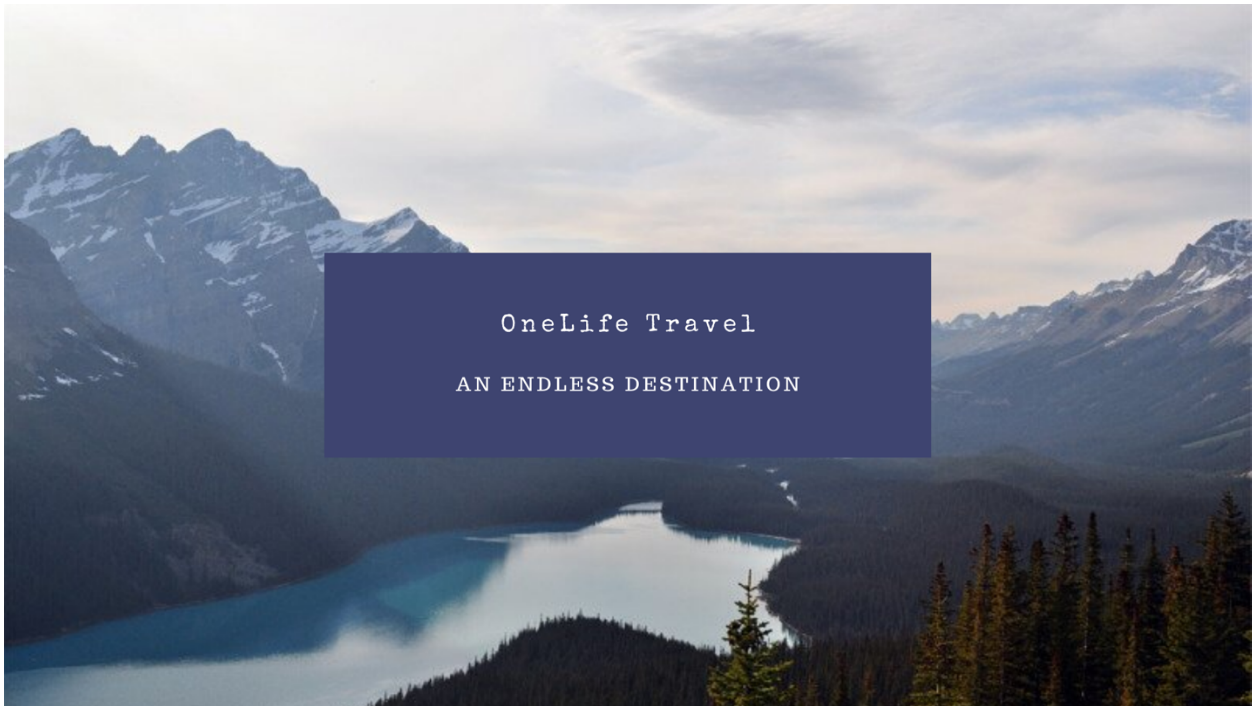


**Service Layer**
Garrison Holt
10/30/2022

**Changes to Service Layers**
For my project, I will not be redesigning any portion of the service layers based on feedback. I believe all endpoints fit the MVP of my application.

**Overview of Service Layers**

For the backend service for my web application, I will be building it using the Express and mongoose libraries from NodeJS. It will be a simple REST API which the frontend will call in order to create and manipulate the data that is being stored in the application's MongoDB database. This backend will have mongoose schemas for the data models.

The backed service will be broken down into the following three layers:

1. Routes: These routes will define specific paths that are available and can be used to get/post/put/delete specific data to/from the database.

2. Controllers: The routes will point to specific controllers that are available. These controllers will handle the incoming HTTP request and return the corresponding response(s) back to the user.

3. Services: Each controller will call specific service methods that apply the logic to handle the data that is being passed in or being requested. Methods on this layer will talk directly with the database and manipulate the data.

**Specifications**
Below I have listed all the routes and endpoints my service layer for OneLife Travel will consist of. I am using a placeholder URL for the time being. The commands listed below will not work, they are just there for example purposes.

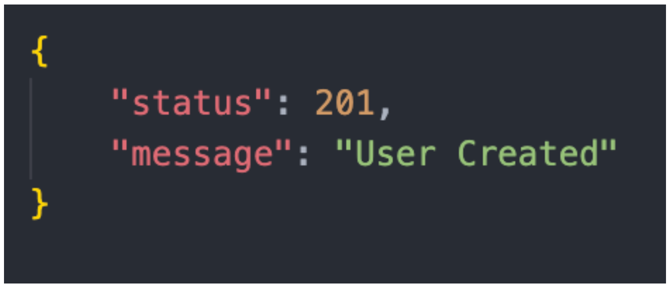
**Auth Route**

i. Create User:
   Method: POST
   URL: https://onelifetravel.com/auth/register

Purpose: When a user creates an account, this endpoint will be called from the front end and the data will be sent to the database.

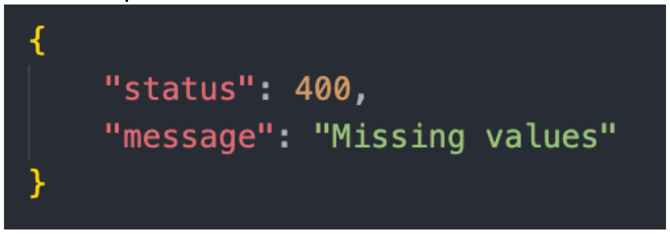
Example requests:
curl --request POST --url https://onelifetravel.com/auth/register

Success 201:



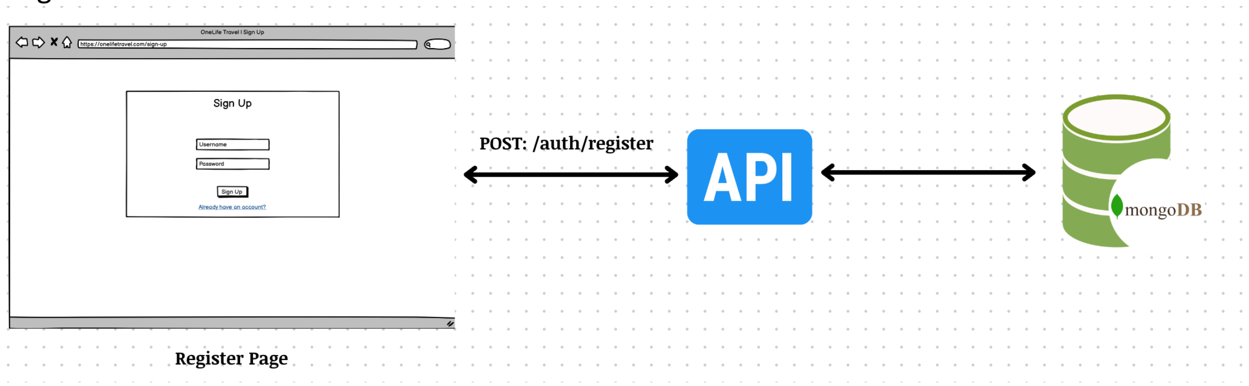
```
{
    "status": 201,
    "message": "User Created"
}
```

Error Response:



```
{
    "status": 400,
    "message": "Missing values"
}
```

Diagram:



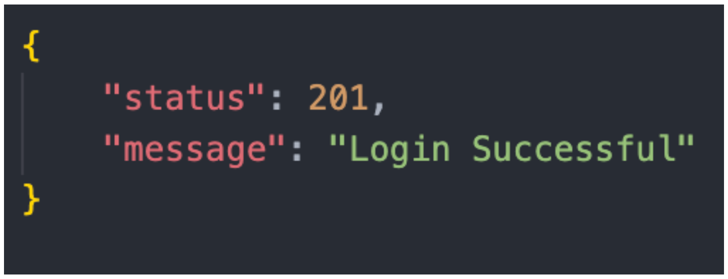
ii. User Login
Method: POST

URL: https://onelifetravel.com/auth/login

Purpose: When a user logs into their account, this endpoint will be called from the front end and the data will be sent to the database. Then the user will be redirected to the posts page.

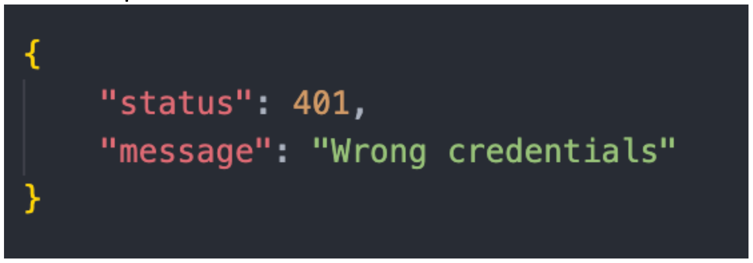
Example requests:
curl --request POST --url https://onelifetravel.com/login

Success 201:



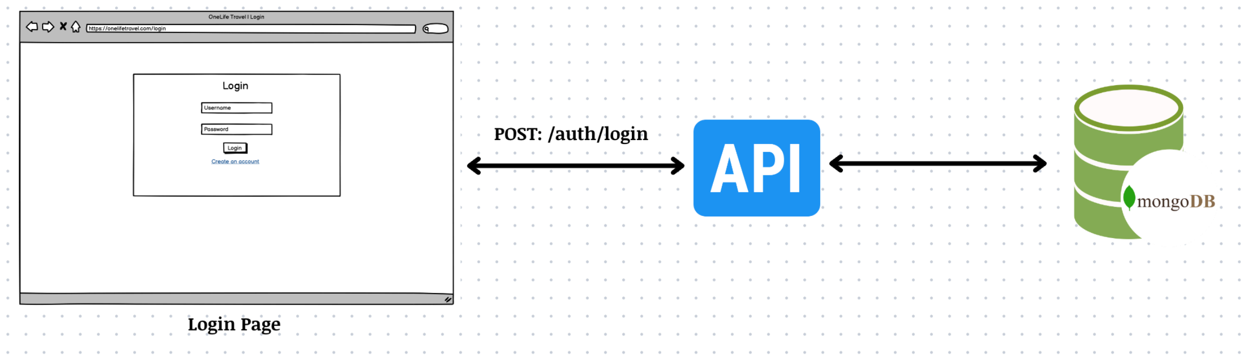
```
{
    "status": 201,
    "message": "Login Successful"
}
```

Error Response:



```
{
    "status": 401,
    "message": "Wrong credentials"
}
```

Diagram:



**Post Route**

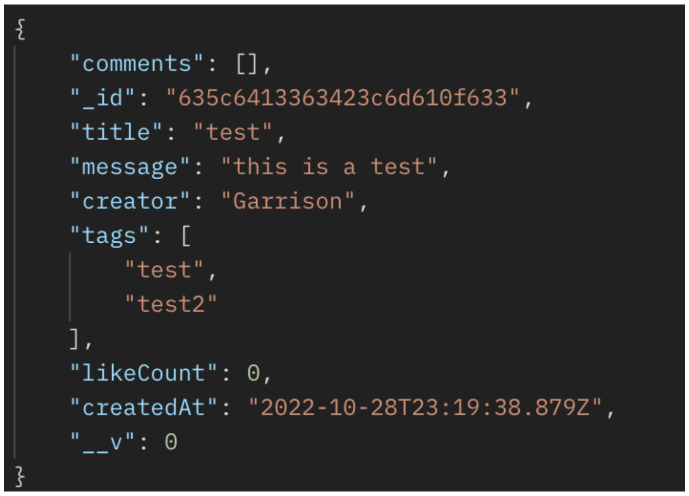
i. Get Posts:
Method: GET
URL: https://onelifetravel.com/api/posts/

Purpose: When a user logs in successfully, this endpoint will be called from the front end and the data returned will be used to render the posts page.

Example requests:
curl --request GET --url https://onelifetravel.com/api/posts/

Success response:

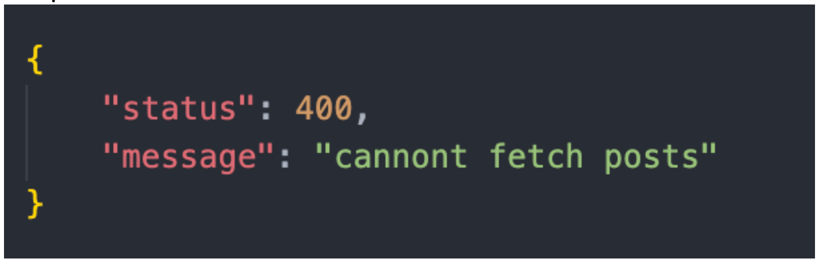
Response: 200



```
{
    "comments": [],
    "_id": "635c6413363423c6d610f633",
    "title": "test",
    "message": "this is a test",
    "creator": "Garrison",
    "tags": [
        "test",
        "test2"
    ],
    "likeCount": 0,
    "createdAt": "2022-10-28T23:19:38.879Z",
    "__v": 0
}
```

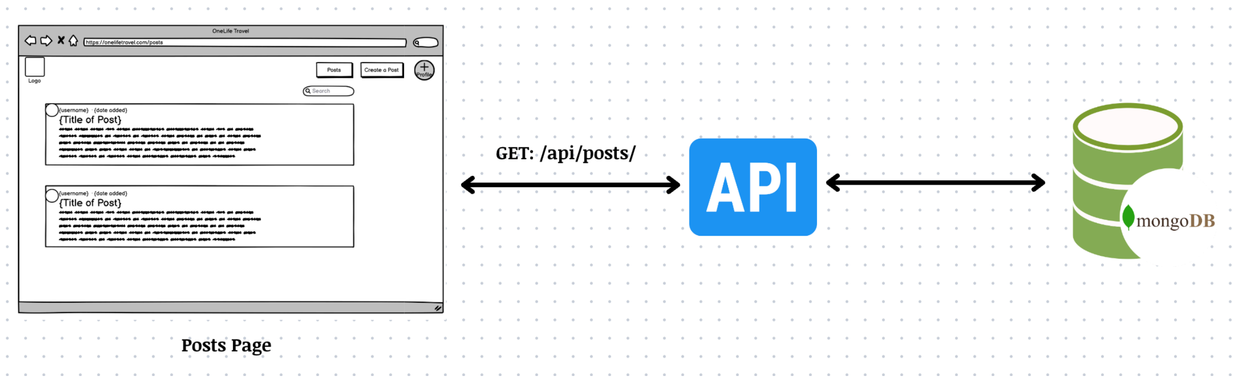
Error Response:

Response: 400



```
{
    "status": 400,
    "message": "cannont fetch posts"
}
```

Diagram:



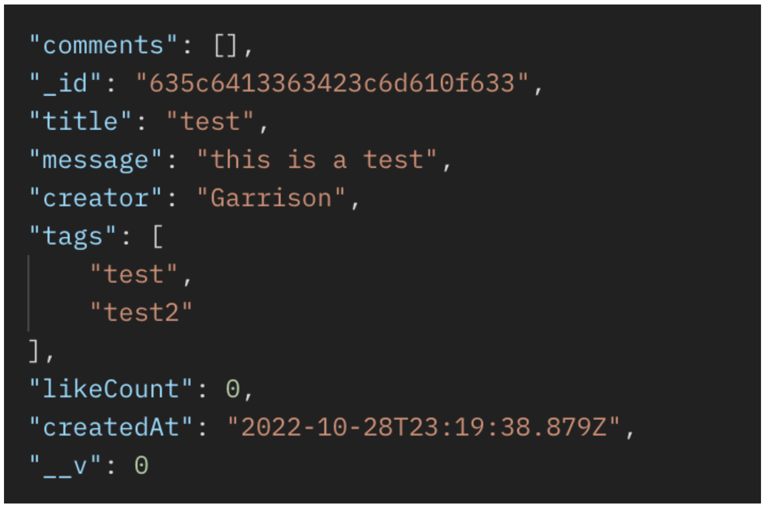
ii. Get Posts by id:
Method: GET
URL: https://onelifetravel.com/api/posts/:id

Purpose: When a user selects an individual post, this endpoint will be called from the front end and the data will be fetched from the database and rendered onto the page.

Example requests:

```
curl --request GET --url https://onelifetravel.com/api/posts/ 635c6413363423c6d610f633
```

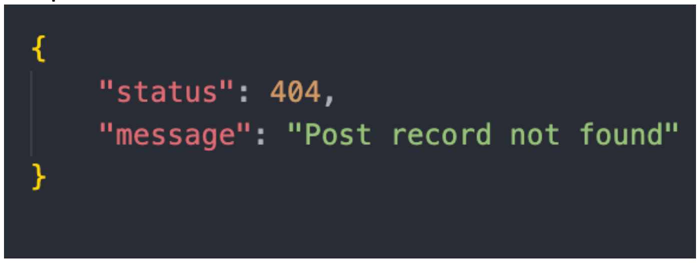
Response: 200



```
"comments": [],
"_id": "635c6413363423c6d610f633",
"title": "test",
"message": "this is a test",
"creator": "Garrison",
"tags": [
    "test",
    "test2"
],
"likeCount": 0,
"createdAt": "2022-10-28T23:19:38.879Z",
"__v": 0
```

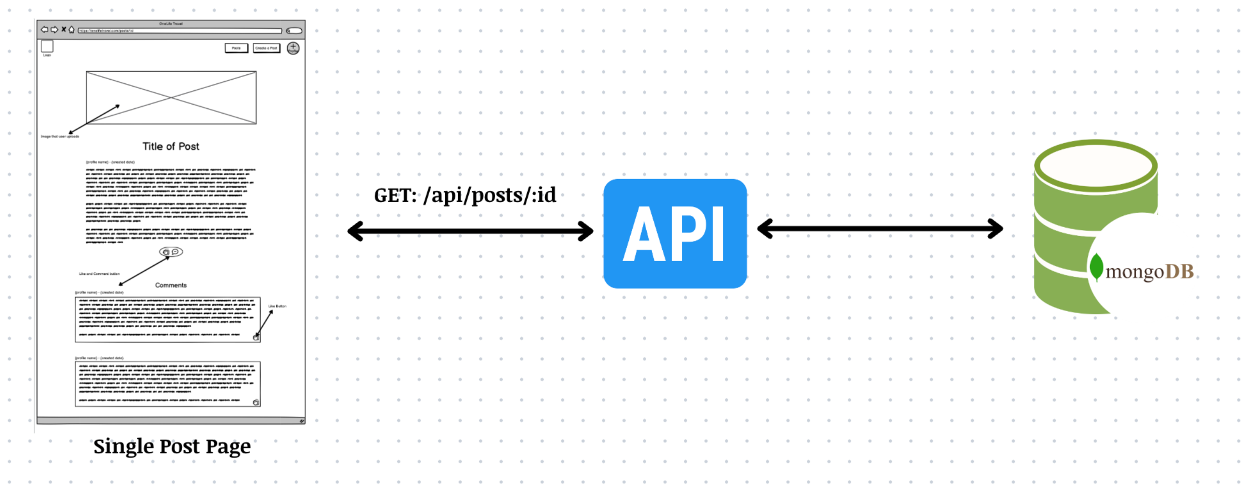
Error Response:

Response: 404



```
{
    "status": 404,
    "message": "Post record not found"
}
```

Diagram:



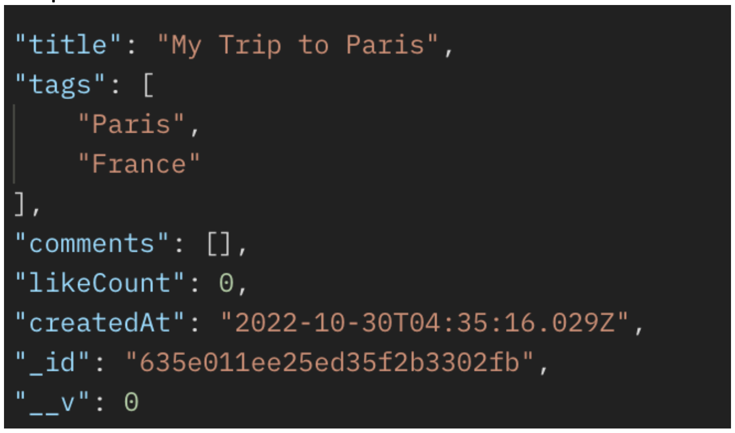
iii. Create Post Record:
Method: POST
URL: https://onelifetravel.com/api/posts/

Purpose: When a user creates a post, this endpoint will be called from the front end and the data will be sent to the database.

Example requests:

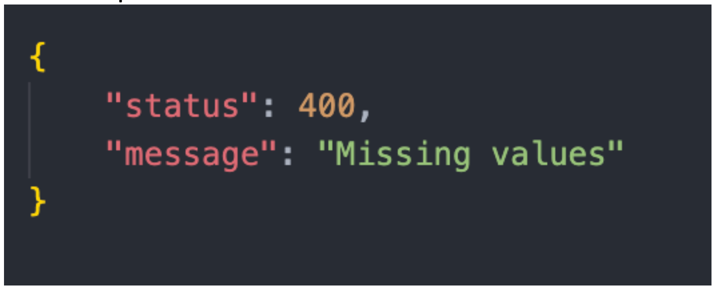
```
curl --request POST \
--url https://onelifetravel.com/posts/\ --header 'Content-Type: application/json' \
--data '{
"title": "My trip to Paris",
“username: “garrison”
“content”: “My trip to Paris was very eventful…”,
"tags": [“Paris”, “France”]
}
```

Response 201:



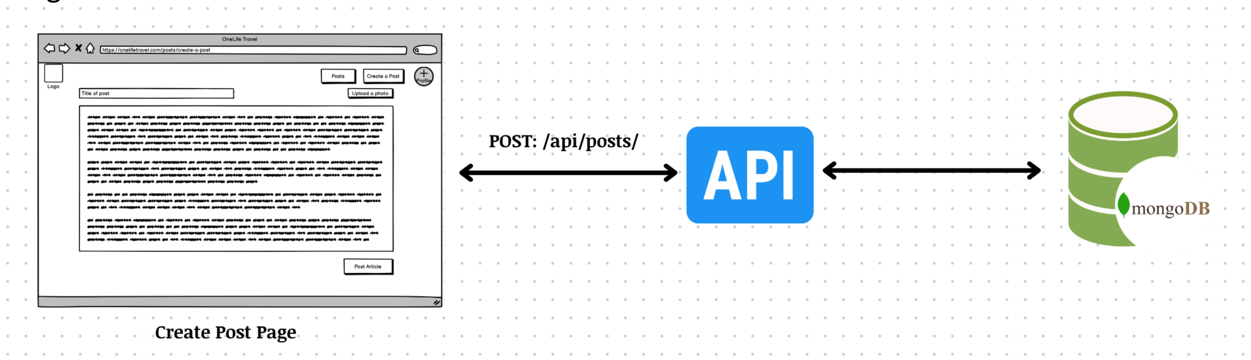
```
"title": "My Trip to Paris",
"tags": [
    "Paris",
    "France"
],
"comments": [],
"likeCount": 0,
"createdAt": "2022-10-30T04:35:16.029Z",
"_id": "635e011ee25ed35f2b3302fb",
"__v": 0
```

Error response:



```
{
    "status": 400,
    "message": "Missing values"
}
```

Diagram:

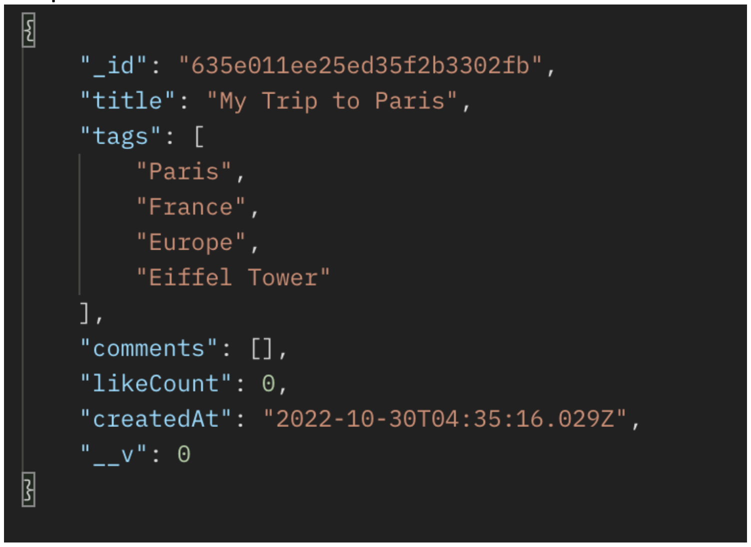


V. Update Post Record:
Method: PUT
URL: https://onelifetravel.com/api/posts/:id

Purpose: When a user wants to update a post, this endpoint will be called from the front end and the data that the user inputs will be sent to the database.

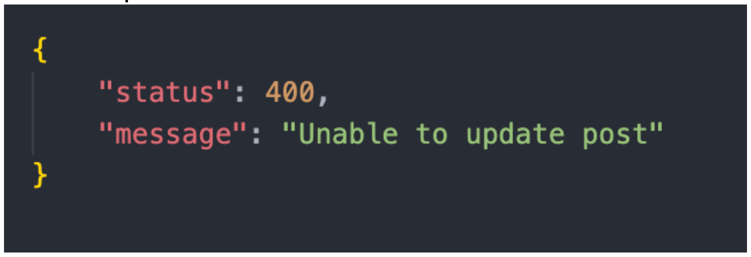
Example requests:
curl --request POST \
--url https://onelifetravel.com/api/posts/:id \ --header 'Content-Type: application/json' \
--data '{
"title": "My trip to Paris",
“content”: “My trip to Paris was very eventful…”,
"tags": [“Paris”, “France”, “Europe”, “Eiffel Tower”]
}

Response 200:



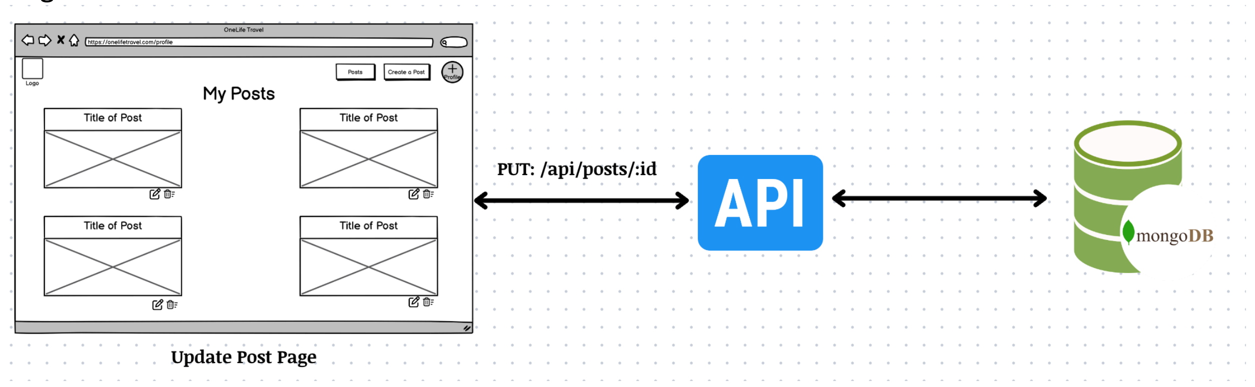
```
{
    "_id": "635e011ee25ed35f2b3302fb",
    "title": "My Trip to Paris",
    "tags": [
        "Paris",
        "France",
        "Europe",
        "Eiffel Tower"
    ],
    "comments": [],
    "likeCount": 0,
    "createdAt": "2022-10-30T04:35:16.029Z",
    "__v": 0
}
```

Error Response:



```
{
    "status": 400,
    "message": "Unable to update post"
}
```

Diagram:

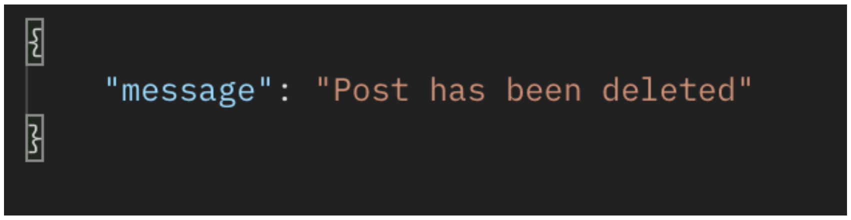


V. Delete Post Record:
Method: DELETE
URL: https://onelifetravel.com/api/posts/:id

Purpose: When a user wants to delete a post, this endpoint will be called from the front end and the data will be deleted from the database.

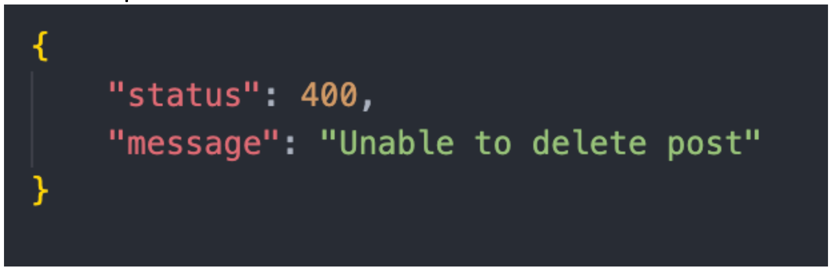
Example requests:
curl --request DELETE--url https://onelifetravel.com/api/posts/:id

Success 200:



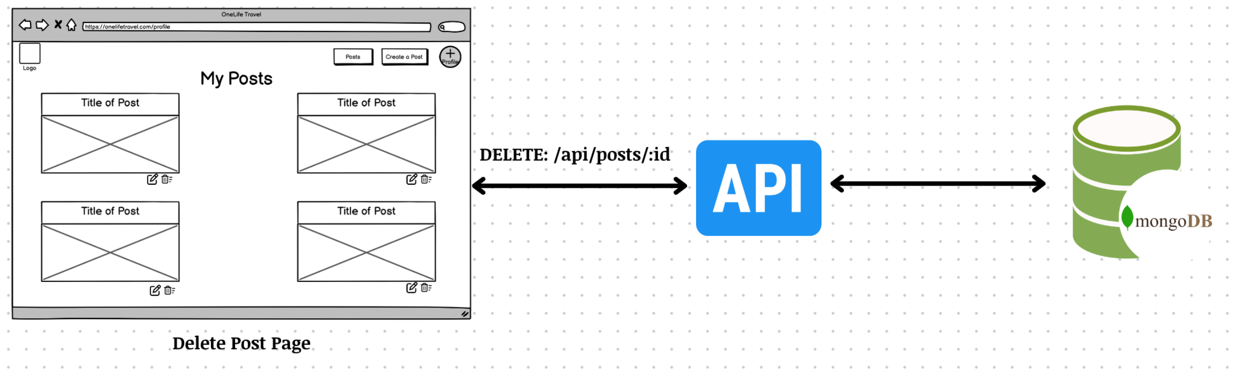
```
{
    "message": "Post has been deleted"
}
```

Error Response:



```
{
    "status": 400,
    "message": "Unable to delete post"
}
```

Diagram:

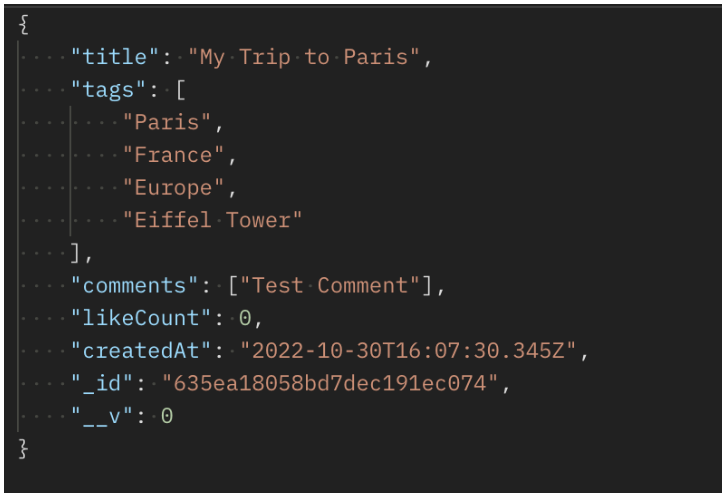


vi. Create Comment Record:
Method: POST
URL: [https://onelifetravel.com/api/posts/:id/comment](https://onelifetravel.com/api/posts/:id/comment)

Purpose: When a user creates a comment, this endpoint will be called from the front end and the data will be sent to the database.

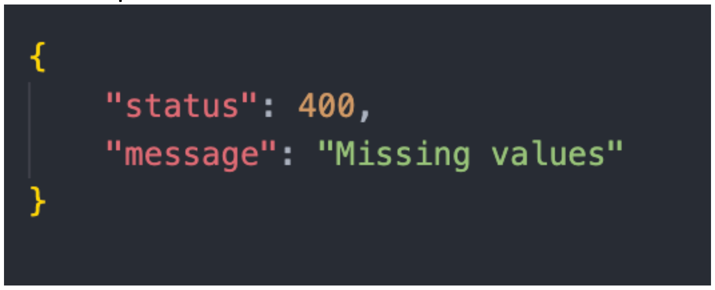
Example requests:
curl --request POST \
--url [https://onelifetravel.com/api/posts/:id/comment](https://onelifetravel.com/api/posts/:id/comment) \ --header 'Content-Type: application/json' \
--data '{
"username": "garrison",
“content”: “Test comment”
}

Response 201:



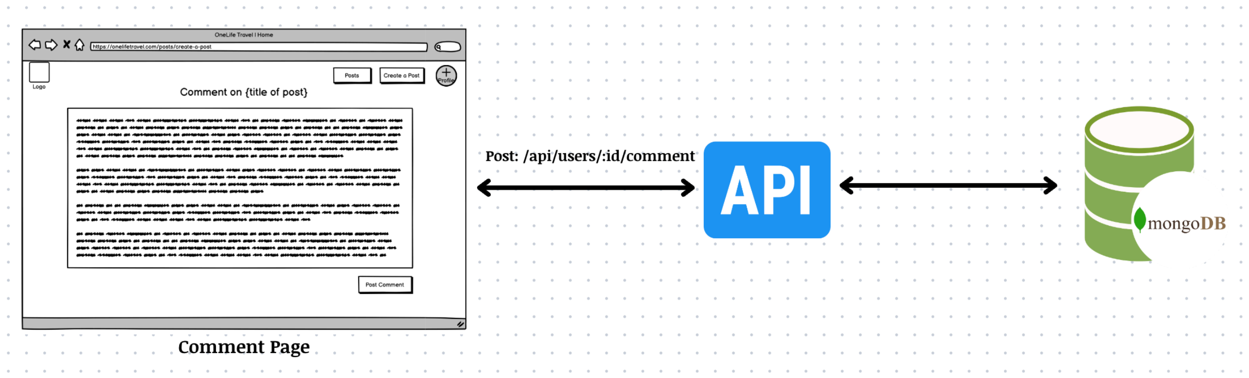
```
{
    "title": "My Trip to Paris",
    "tags": [
        "Paris",
        "France",
        "Europe",
        "Eiffel Tower"
    ],
    "comments": ["Test Comment"],
    "likeCount": 0,
    "createdAt": "2022-10-30T16:07:30.345Z",
    "_id": "635ea18058bd7dec191ec074",
    "__v": 0
}
```

Error response:



```
{
    "status": 400,
    "message": "Missing values"
}
```

Diagram:


OneLife Travel | Home
https://onelifetravel.com/posts/create-a-post
Logo
Posts
Create a Post
Profile
Comment on {title of post}
Post Comment
Comment Page
Post: /api/users/:id/comment
API
mongoDB

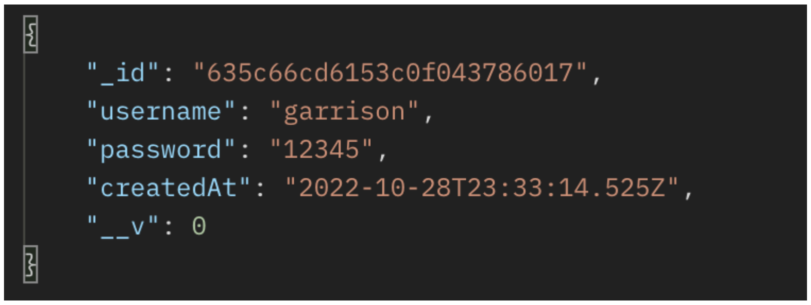
## User Route

i. Get User by id:
   Method: GET
   URL: https://onelifetravel.com/api/user/:id

Purpose: When a user wants to access their profile, this endpoint will be called from the front end and the data will be fetched to the database.

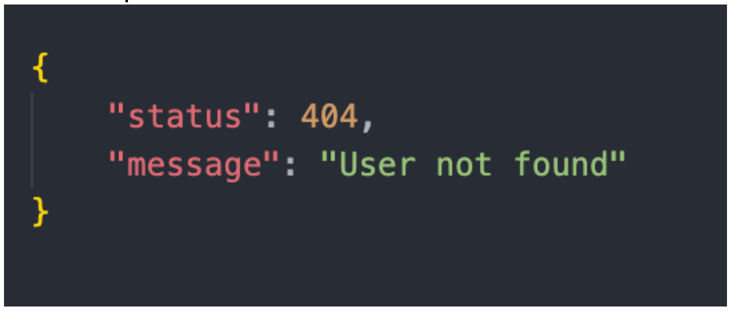
Example requests:
curl --request GET --url https://onelifetravel.com/api/user/90adkpfaf0l01

Success 200:



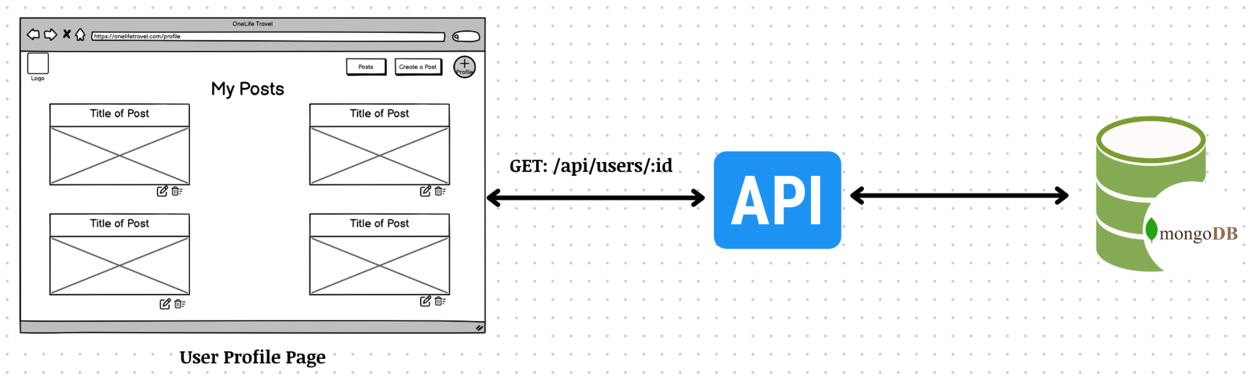
```
{
    "_id": "635c66cd6153c0f043786017",
    "username": "garrison",
    "password": "12345",
    "createdAt": "2022-10-28T23:33:14.525Z",
    "__v": 0
}
```

Error Response:



```
{
    "status": 404,
    "message": "User not found"
}
```

Diagram:



## Stretch Features after MVP Completion

**Saved Posts Route**

i. Get Saved Posts:
Method: GET
URL: https://onelifetravel.com/api/user/:id/saved-posts

Purpose: When a user logs in successfully, this endpoint will be called from the front end and the data returned will be used to render the posts page.

Example requests:
```
curl --request GET --url https://onelifetravel.com/api/user/:id/saved-posts
```

ii. Get Saved Post by Id:
Method: GET
URL: https://onelifetravel.com/api/user/:id/saved-posts/:id

Purpose: When a user logs in successfully, this endpoint will be called from the front end and the data returned will be used to render the posts page.

Example requests:
```
curl --request GET --url https://onelifetravel.com/api/user/:id/saved-posts/:id
```

iii. Remove Saved Post by Id:
Method: DELETE
URL: https://onelifetravel.com/api/user/:id/saved-posts/:id

Purpose: When a user logs in successfully, this endpoint will be called from the front end and the data returned will be used to render the posts page.

Example requests:

```
curl --request DELETE --url https://onelifetravel.com/api/user/:id/saved-posts/:id
```