

API Explanation:

To set up the backend:

Navigate to the backend directory via your terminal

Run the following command:

```
npm install
```

The required dependencies will be installed.

Now in the root of the backend directory create a file .env

The .env file will contain the following environment variables:

PORT, MONGO_URI, JWT_SECRET.

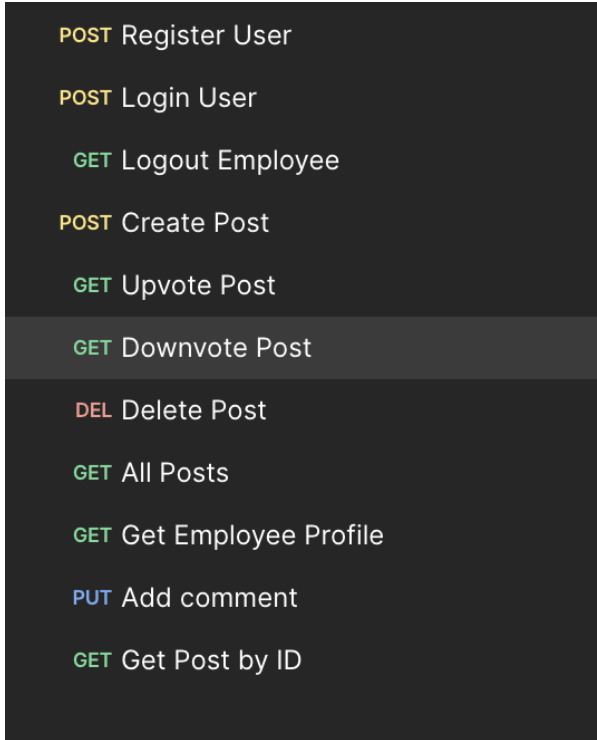
You can use any valid values for them, I used the following:

PORT = 3000

MONGODB_URI = "mongodb://localhost:27017/employeedb"

JWT_SECRET = "super-secret-jwt-generation-key-new-delhi-666"

The API contains the following endpoints:



- POST Register User
- POST Login User
- GET Logout Employee
- POST Create Post
- GET Upvote Post
- GET Downvote Post
- DEL Delete Post
- GET All Posts
- GET Get Employee Profile
- PUT Add comment
- GET Get Post by ID

[08]

Register User:

It is a POST request at the following endpoint:

http://localhost:3000/api/register

The request object contains the fields, name, username, email and password.

An example request object

```
{
  "name": "Priyanshu Dhariwal",
  "username": "pd2425",
  "email": "preed@gmail.com",
  "password": "thisismypassword"
}
```

On successful registration, the response from the server will be:

```
{
  success: true,
  message: "registration successful"
}
```

Login user:

It is a POST request with the following end point:

http://localhost:3000/api/login

The request object contains the fields email and password.

An example request object

```
{
  "email": "pried@gmail.com",
  "password": "thisismypassword"
}
```

On successful logging in, the response from the server will be:

```
{
  success: true,
  employee,
  token
}
```

The employee object in the response will contain the name and username and email of the employee and the token will contain the JWT authentication token of the employee

The remaining endpoints can only be accessed by an authorised user, that is after successful registration and login

Logout:

It is a get request with the following endpoint:

http://localhost:3000/api/logout

On successful log out the response from the server will be:

```
{
  success: true,
  message: "Logged out successfully"
}
```

Create Post:

It is a post request at the following endpoint

`http://localhost:3000/api/post/new`

The request object contains the following fields, content, title and tags.

An example request object:

```
{
  "content": "This is Jane's new post",
  "title": "another good job job",
  "tags": ["test post", "request"]
}
```

On successful creation of post the response from the server will be:

```
{
  success: true,
  newPost
}
```

The newPost object in the response will contain the new post that was created (essentially the request object)

Upvote and Downvote post:

Both of these are get request at these respective endpoints:

`http://localhost:3000/api/post/upvote/:id`

`http://localhost:3000/api/post/downvote/:id`

:id is the unique id of the post that is saved in the post object.

On a successful upvote/downvote the response from the server will be:

```
{
  success: true,
  message: "upvoted"
}
```

Or the message will be downvoted in the case of a downvote

Delete Post:

It is a delete request at the following endpoint:

`http://localhost:3000/api/post/delete/:id`

:id is the unique id of the post that is saved in the post object.

On successful deletion the response from the server will be:

```
{
  success: true,
  message: "post deleted"
}
```

All Posts:

It is a get request that will fetch all the posts that have been made so far

<http://localhost:3000/api/post/all>

On successful request the response from the server will be:

```
{
  success: true,
  posts
}
```

Posts will be an array of all the posts that have been made

Get Employee Profile:

It is a get request that will fetch a particular employee's profile

<http://localhost:3000/api/profile/:id>

On a successful request the response from the server will be:

```
{
  success: true,
  employee
}
```

The employee object will contain the employee details of the requested employee

Add Comment:

It is a put request at the following end point

<http://localhost:3000/api/post/comment/:id>

The request object contains a single field, commentContent.

An example request object:

```
{
  commentContent: "nice post"
}
```

On successful request the response from the server will be:

```
{
  success: true,
  message: "comment added"
}
```

Get Post by id:

It is a get request to fetch a single post based on the id

`http://localhost:3000/api/post/:id`

:id is the unique of the post

On a successful request the response from the server will be

```
{  
  success: true,  
  post  
}
```

Post will contain the post object with the properties of the requested post.

This video is a good tutorial for interfacing:

<https://www.youtube.com/watch?v=nf6sD6KA21E>