Angular Certification Training

Certification Project Statement

edureka!

Problem Statement: Build an online friends book web application using Angular

1. Background

Friends Book is a popular social networking web application that allows registered users to connect with friends. Users can send requests, write posts and share pictures among friends.

This web application is a regular stop for millions of users. Since it is designed using traditional web development methodologies, navigating to different page of the application resulted in reloading the entire page. With the increase in web traffic the website load time increased significantly which in turn, impacted the speed and performance of the website. Also, the website could not adapt to different resolutions and device screens.

All these challenges led to unsatisfied user experience. So, the company decided to upgrade their website using Angular Framework to focus on building responsive and customer-oriented single page web application.

2. Goal

The objective is to develop a RESTful web application that can be easily adopted by users to engage them in social activities with a faster and safer web environment. The framework will be built in a way that ensures maximum re-usability.

3. Use Cases:

The application is designed to provide user-specific functionality. We will have two users for this application:

1. **Admin:**

   An admin will be able to:

- ✓ Block the account of any user
- ✓ Change and reset the password
- ✓ Post any message or advertisement
- ✓ Manage profile details
- ✓ Hide the post of any user

2. **User:**

   User's will be able to:

- ✓ Register themselves as a user
- ✓ Change and reset their password
- ✓ Post any message, article or upload picture
- ✓ Send, accept or reject friend requests
- ✓ Manage their profile details
- ✓ Hide their own post
- ✓ See posts from all the users

Other than the above functionality, application will have authorization/authentication based on JSON web token (JWT).

4. Web Application Requirement:

Angular framework, HTML, CSS, VS Code, NodeJS and MongoDB (you will be provided with a Node.js API which will fetch the data stored in MongoDB database.

5. Web Application Implementation:

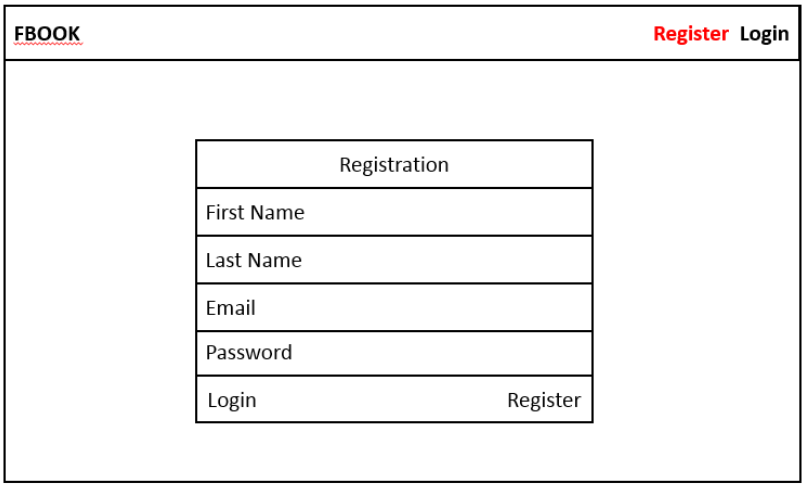
The Web Application will include following aspects:

1) Registration page
2) Login page
3) Forgot password page
4) Home Page
5) Network Page, where all the registered users will be available for friendship
6) Friends List Page
7) Settings page
8) Users list (only for admin)

All the above sections are explained below with the block diagram for better understanding (Active links are shown in red color)

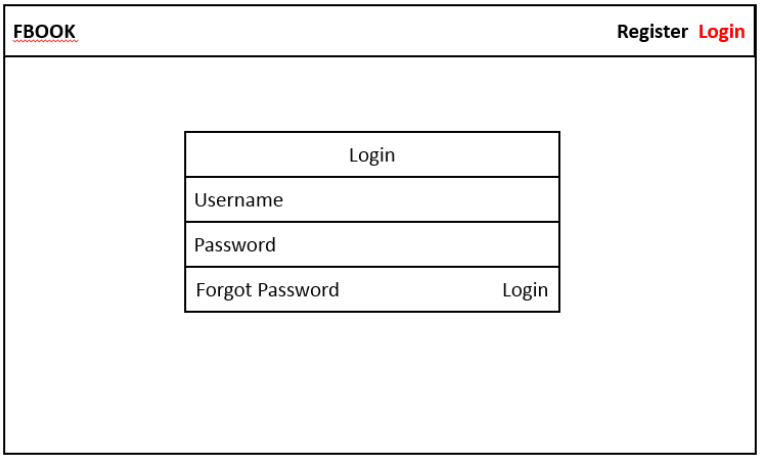
1) **Registration page**

This section contains a page for new users to register themselves on the application by providing few personal details



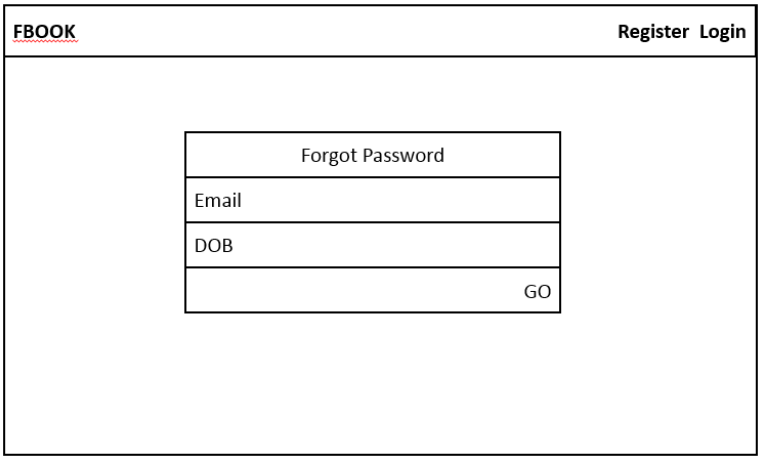
2) **Login page**

This section contains a login page where registered user can log in to the application

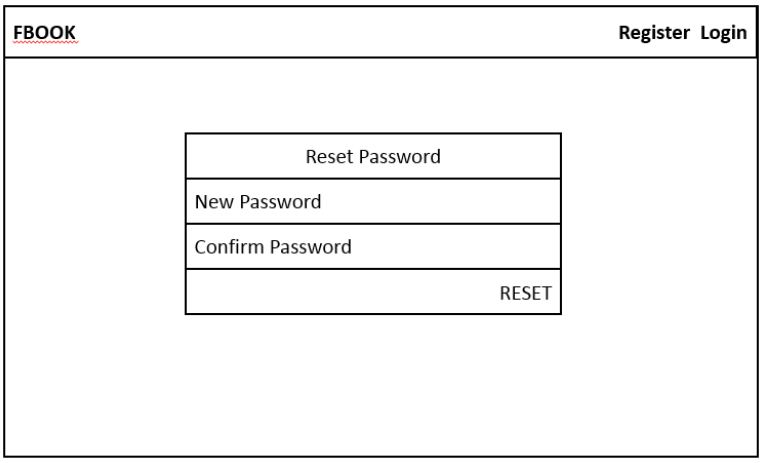


3) **Forgot password page**

This page will allow user to reset their account password. They will first have to authenticate themselves by providing few details

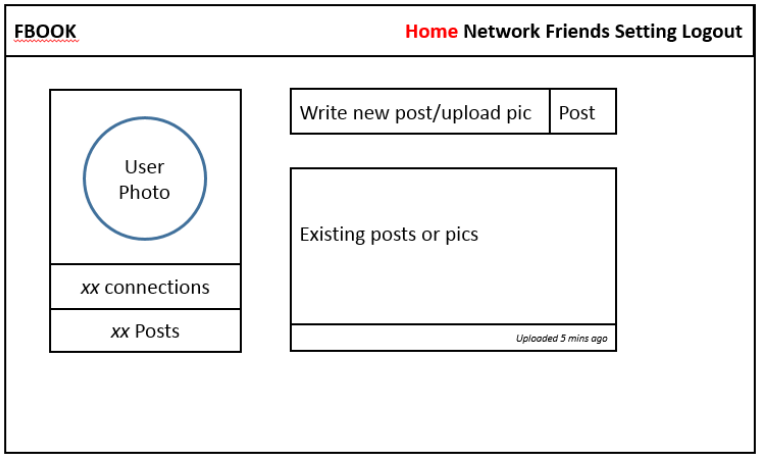


Once you provide the correct information, you will be asked to reset a new password



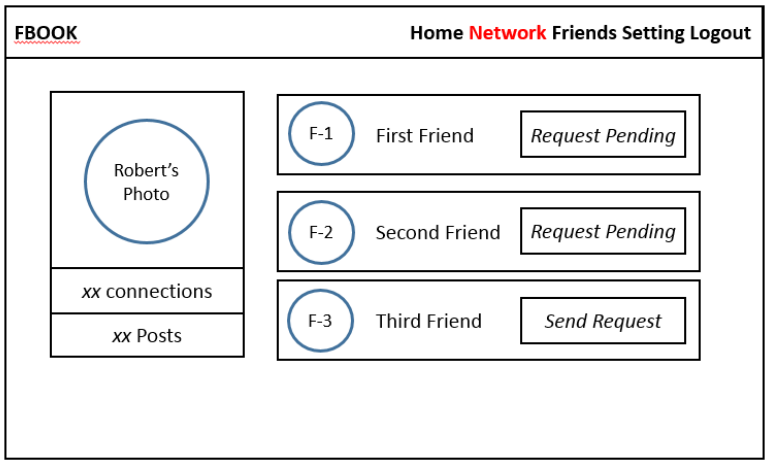
4) Home page

This is the first page that will be rendered when user's login to the application. Users can write posts or upload pictures here. All the posts available will be shown on this page.



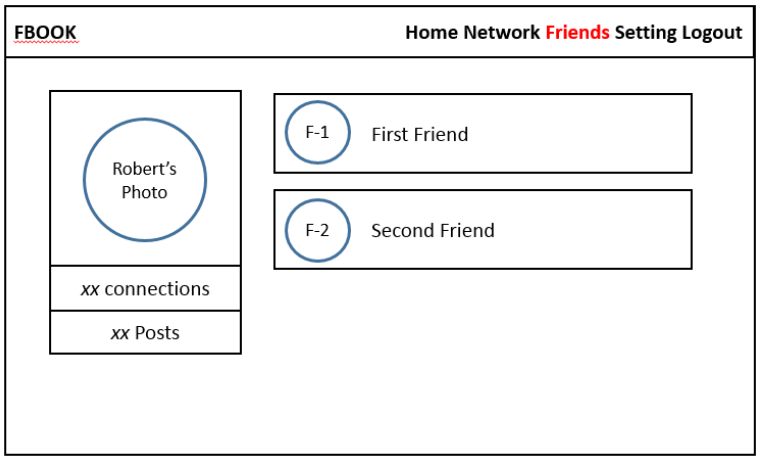
5) Network page

All the registered users will be shown here, so that logged in user can send them a friend request and track the status of the requests



6) Friends List page

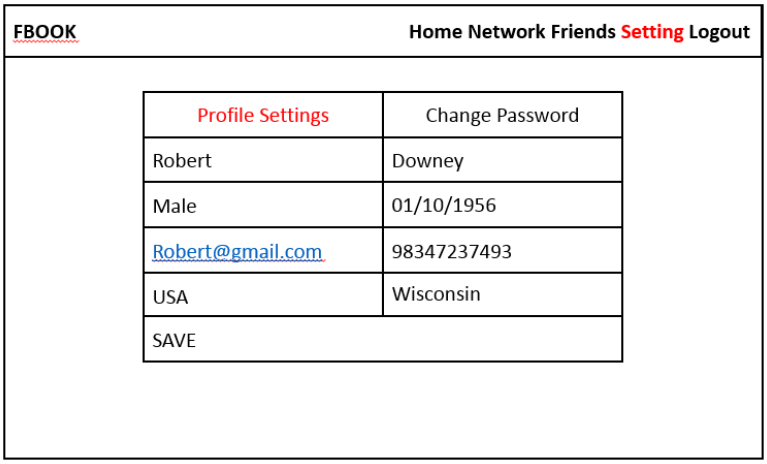
All the friends of the logged-in user will be shown on this page



7) Settings page:

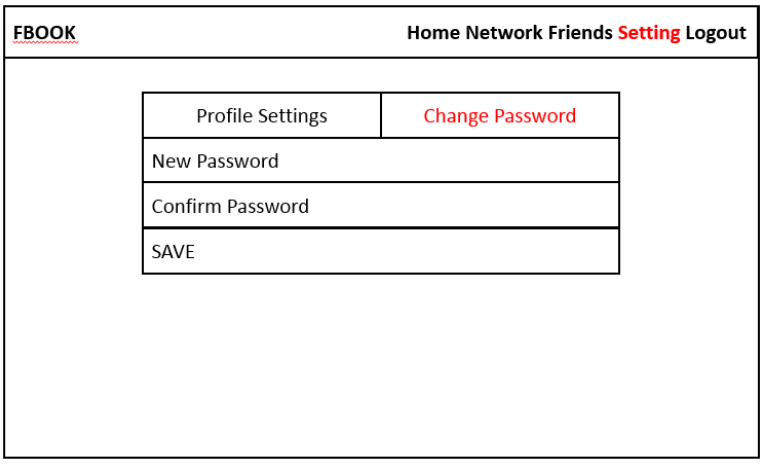
a.) Profile Settings:

This page will allow user to change or update their personal details



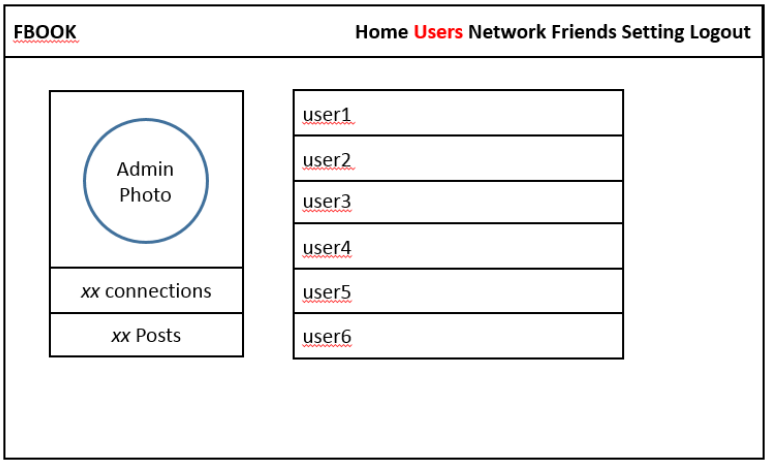
b.) Change password page

This page will allow user to change the password



8) **Users List page (only for Admin)**

This page will only be visible to the admin. The admin can see the registered users and can also block the account of any user



6. List of REST APIs created to develop this project:

This section explains the REST APIs created to develop this project. It displays the final APIs implemented with routes which will be used by front-end application directly:

The baseURL that we will be using to connect to our backend is:

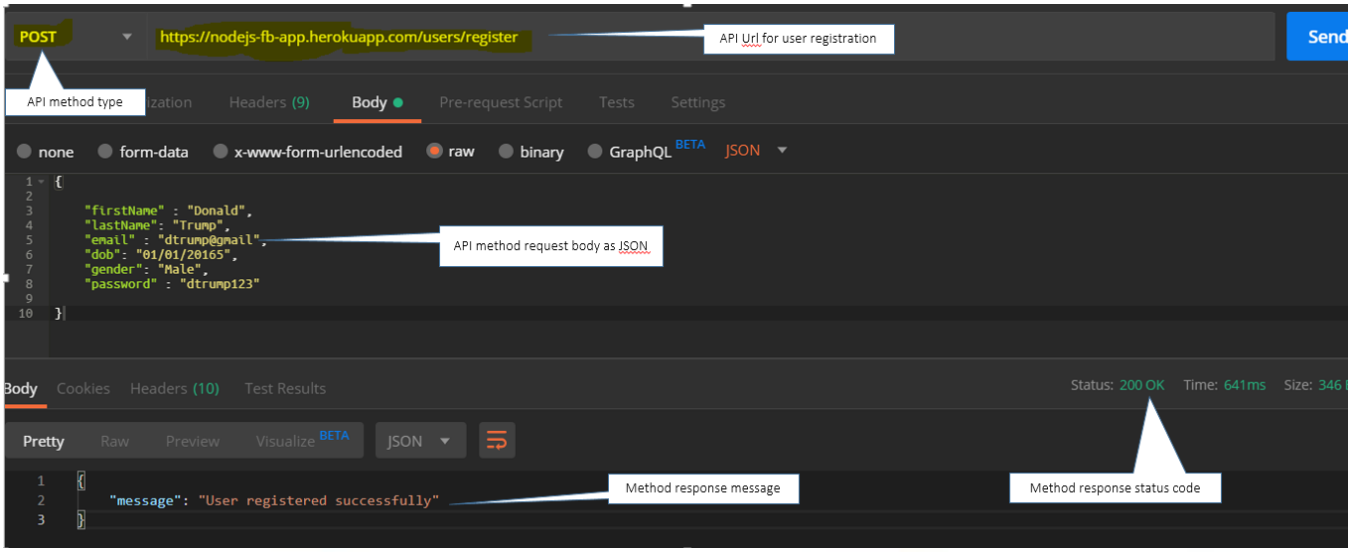
https://nodejs-fb-app.herokuapp.com/

API Routes

1) **User Service**

   a. **User Registration: 'users/register'**

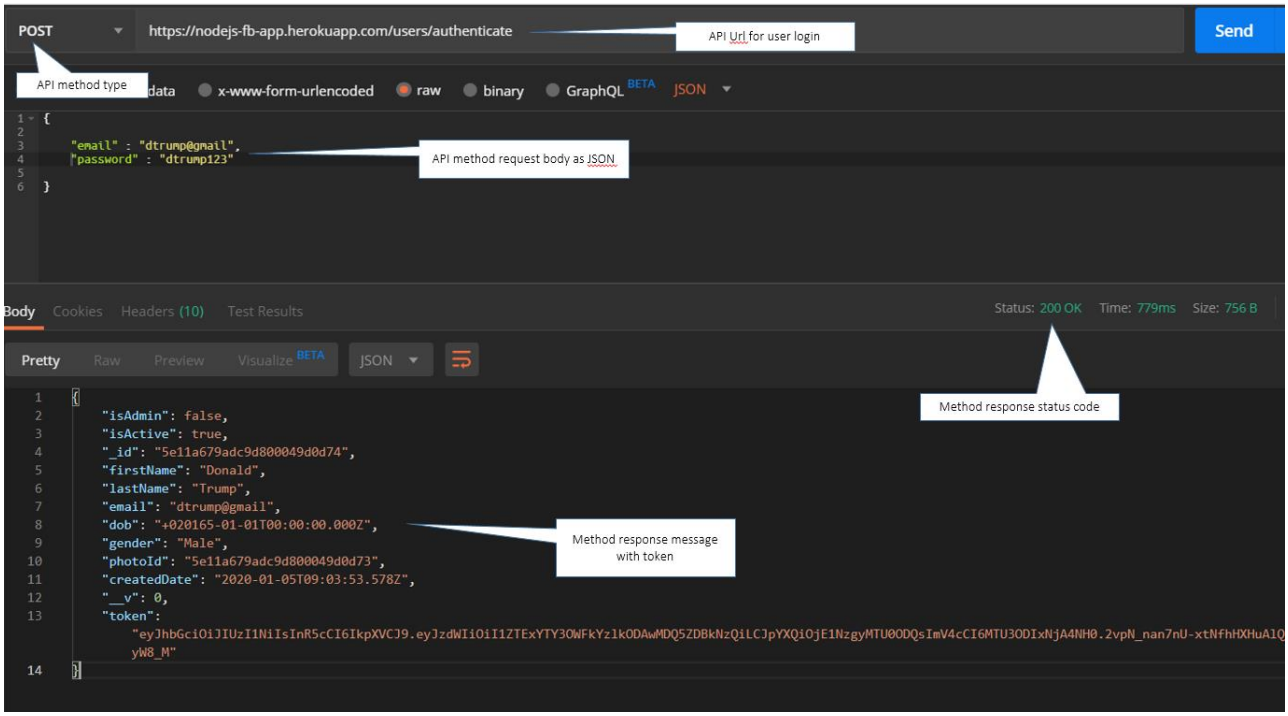
```
return this.http.post<User>(this.apiBaseURL + 'users/register', newUser
```



b. User Authentication/Login: 'users/authenticate'

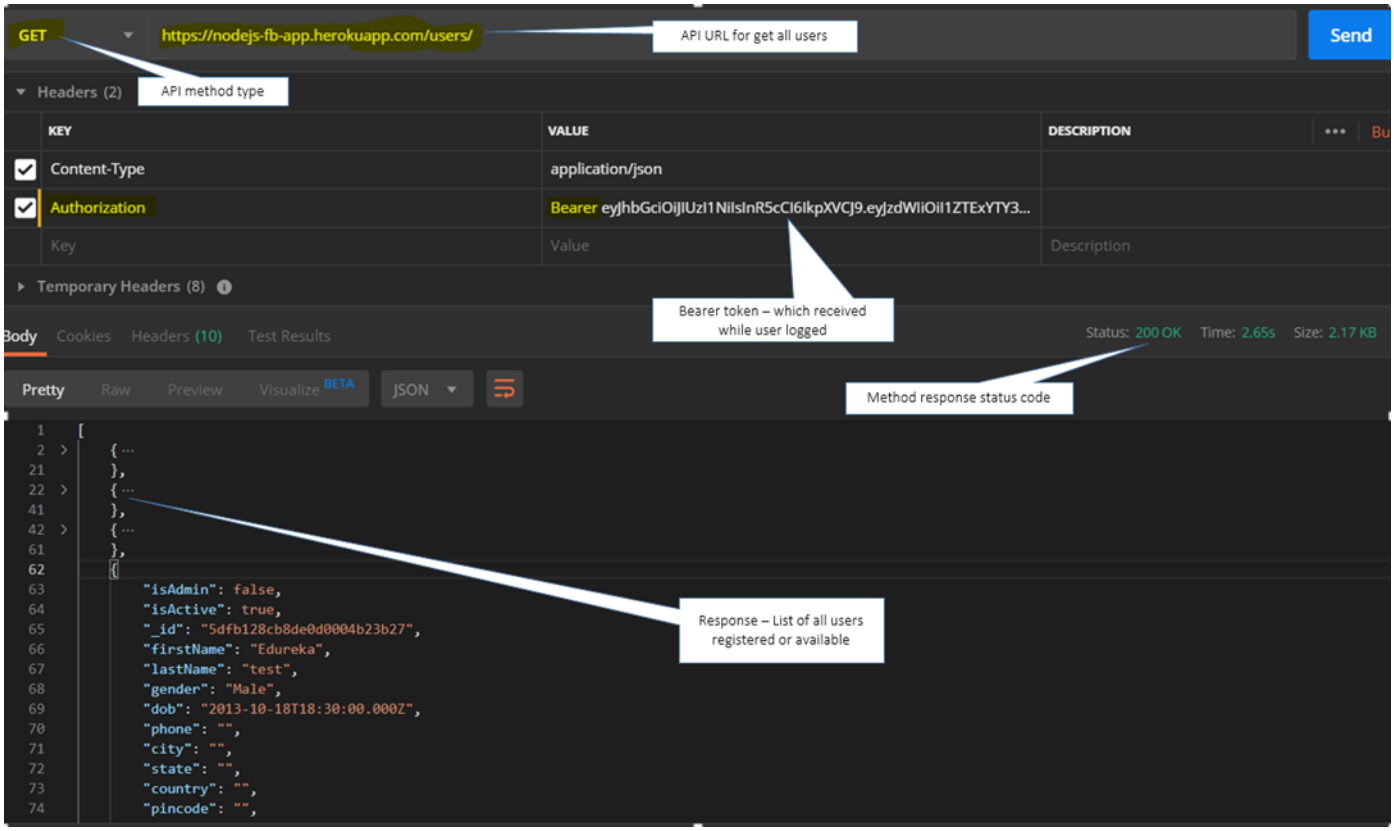
Token received from this will be used for further communication with the other APIs

```
return this.http.post<any>(this.apiBaseURL + 'users/authenticate', { email: email, password: password })
```



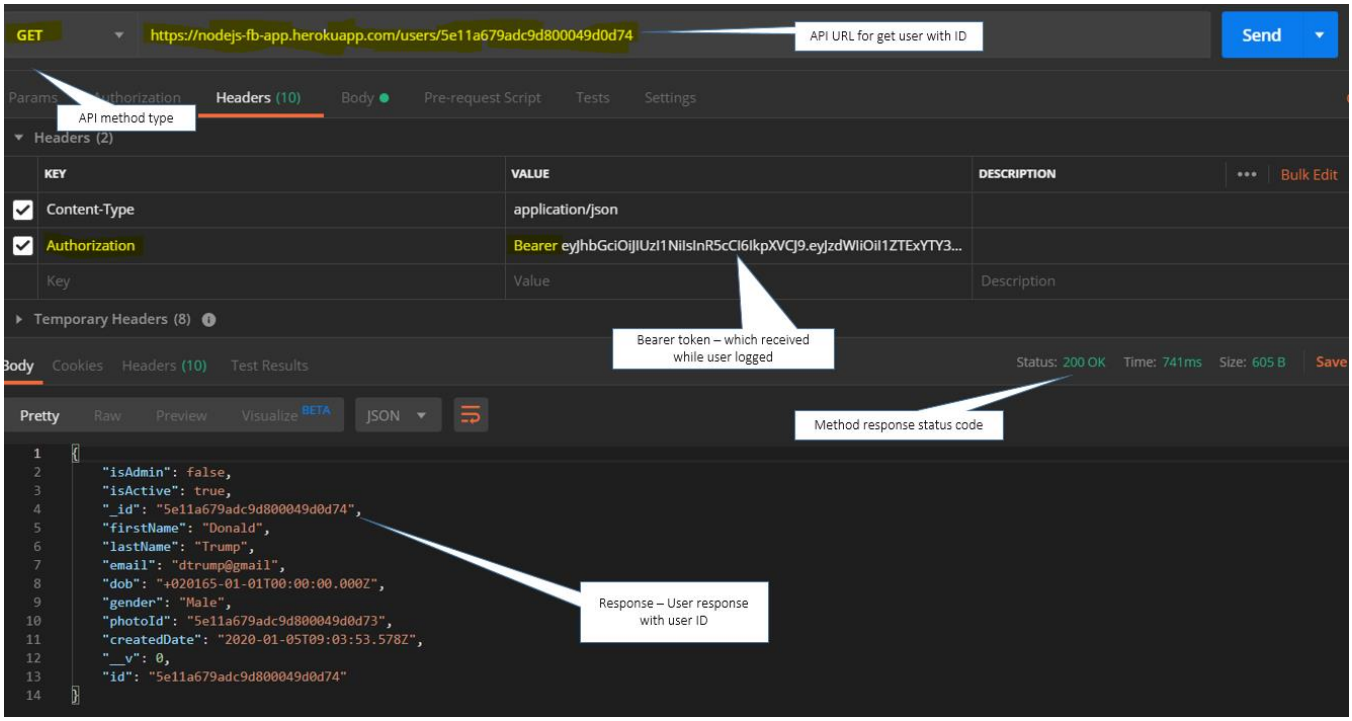
c. **Find All Users:** Retrieve all the users registered in the system. It is used only by admin **'users/'**

```
return this.http.get<User[]>(this.apiBaseURL + 'users/')
```



d. **Find User by ID –** Retrieve any user registered in the system by their unique ID **'users/' + userId**

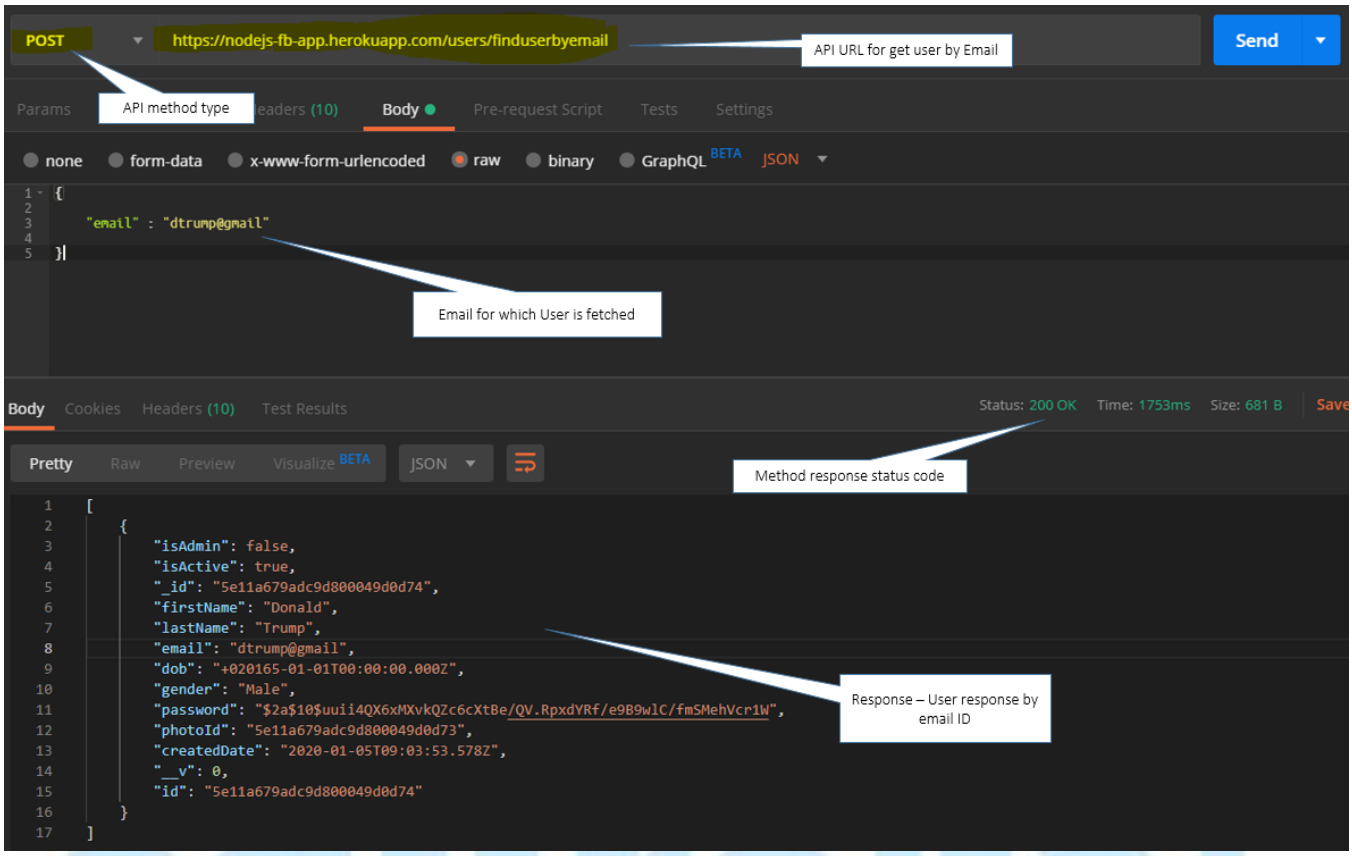
```
return this.http.get(this.apiBaseURL + 'users/' + userId)
```



e. **Find User by Email** – Retrieve any user registered in the system by their email ID.

'users/finduserbyemail'

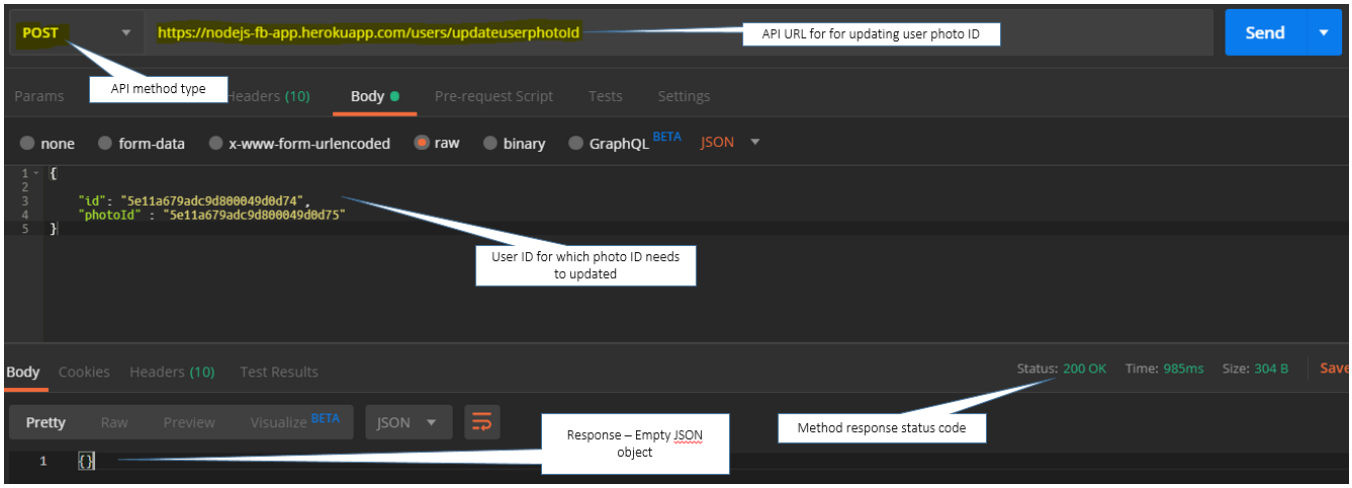
```
return this.http.post(this.apiBaseURL + 'users/finduserbyemail', { email: email }
```



f. **<u>Update user photo ID</u>** – Update user's photo ID using user ID

'users/updateuserphotoId'

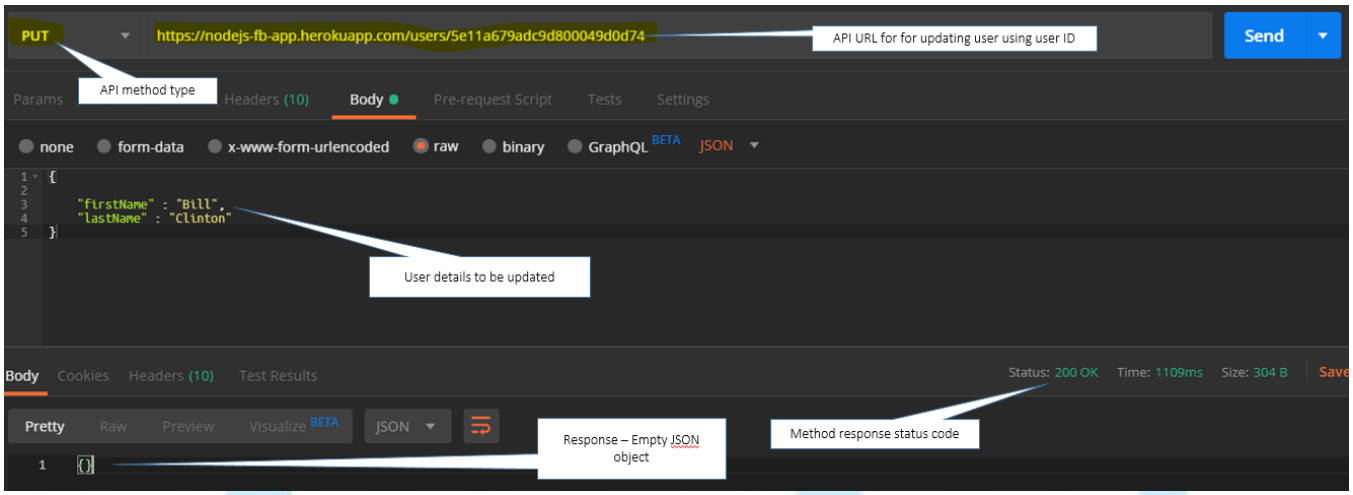
```
return this.http.post(this.apiBaseURL + 'users/updateuserphotoId', updatedUser,
```



g. **Update User** – Update user information mainly used for settings page

'users/' + updatedUser.id

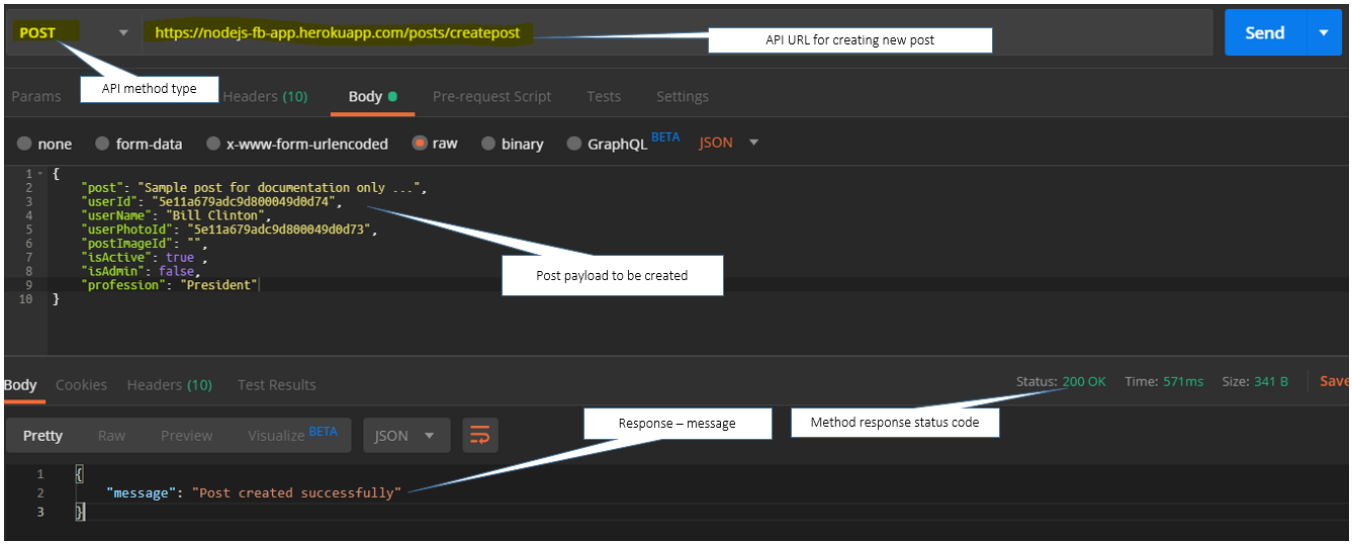
```
return this.http.put(this.apiBaseURL + 'users/' + updatedUser.id, updatedUser)
```



2) Post service:

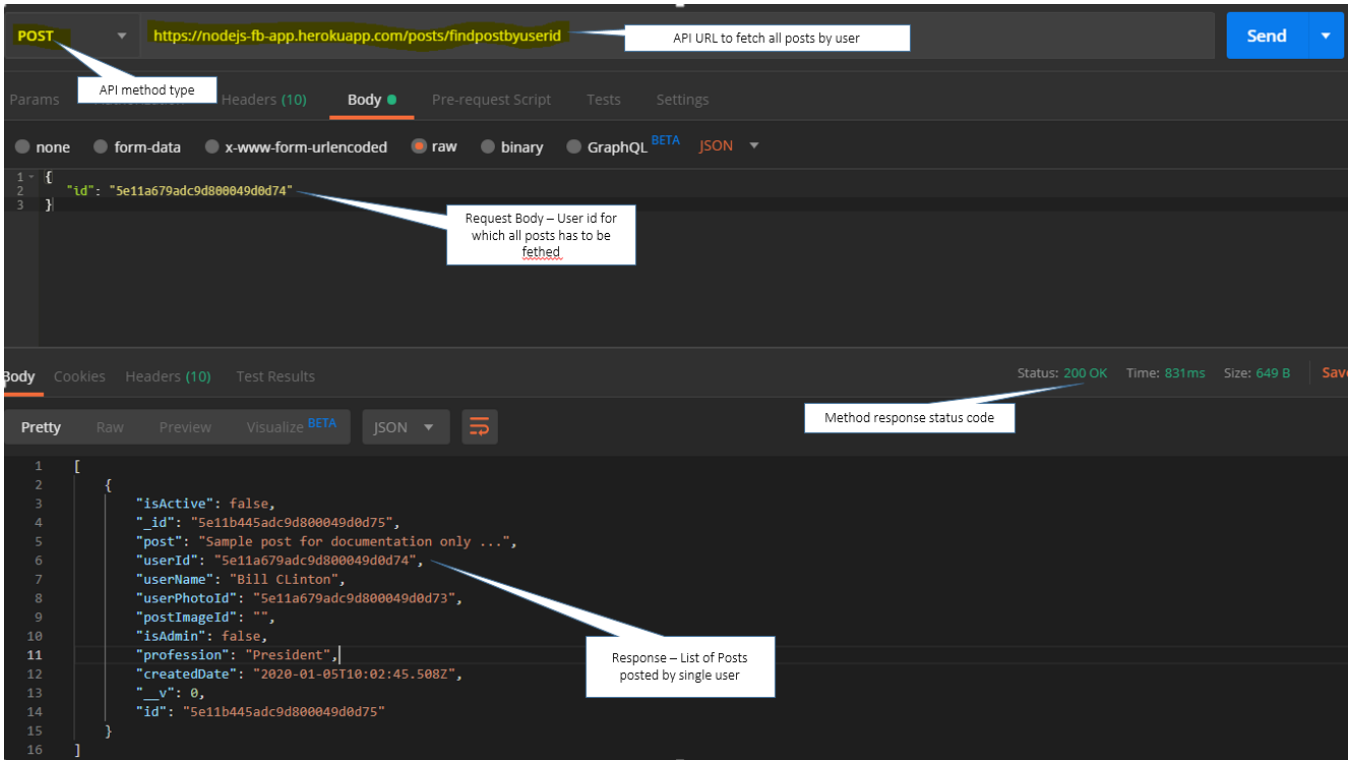
a. Create Post – Create new post or upload new image for post
'posts/createpost'

```
return this.http.post<Post>(this.apiBaseURL + 'posts/createpost',
```



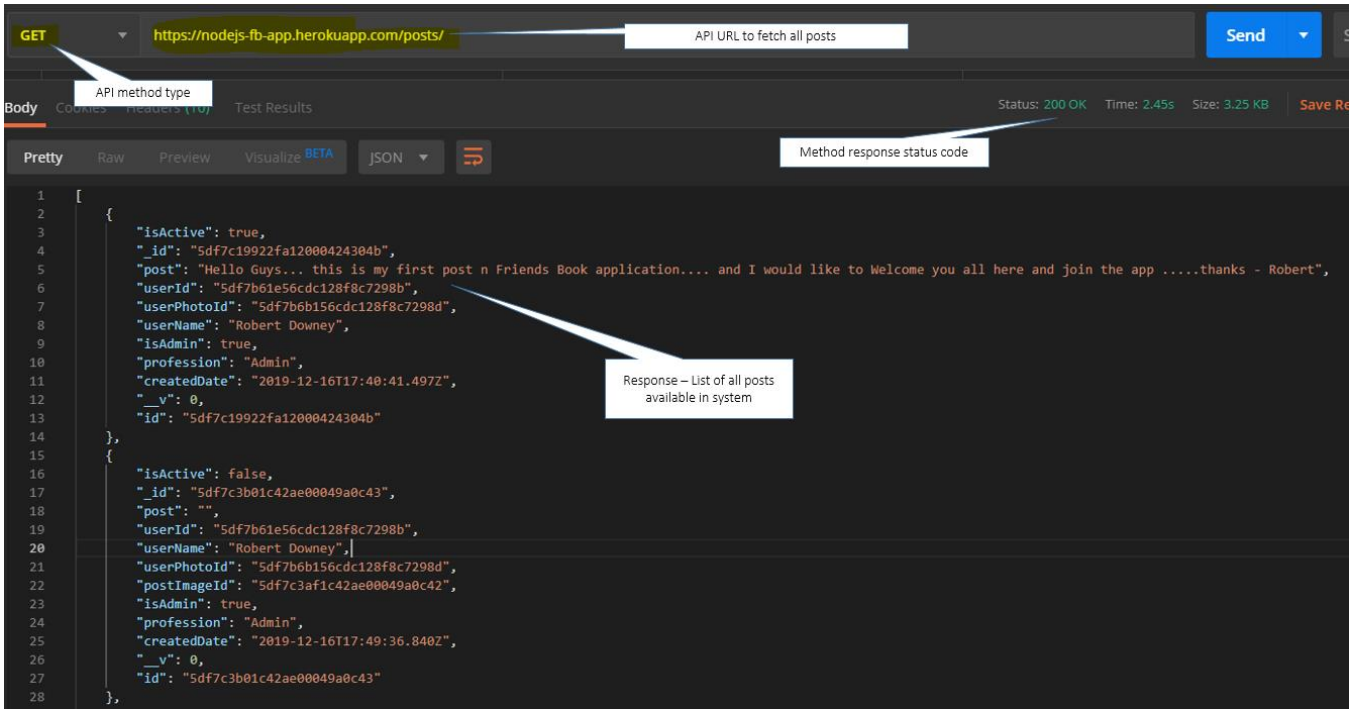
b. **Get Posts by User ID –** Fetch all posts posted by a specific user using user ID **'posts/' + postId**

```
return this.http.get(this.apiBaseURL + 'posts/' + postId)
```



c. **Get All Post** – Fetch all active posts created/available in the system

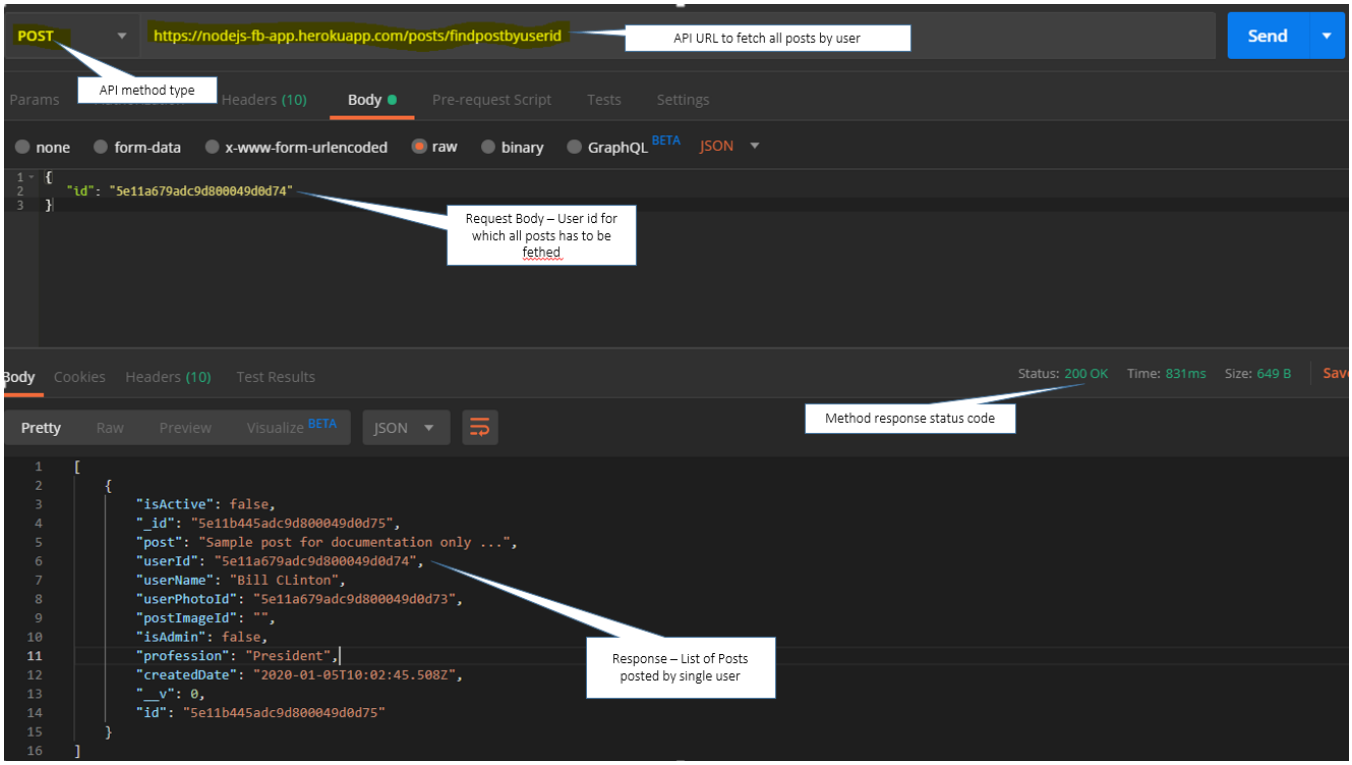
```
return this.http.get<Post[]>(this.apiBaseURL + 'posts/');
```




```
[
    {
        "isActive": true,
        "_id": "5df7c19922fa12000424304b",
        "post": "Hello Guys... this is my first post n Friends Book application.... and I would like to Welcome you all here and join the app .....thanks - Robert",
        "userId": "5df7b61e56cdc128f8c7298b",
        "userPhotoId": "5df7b6b156cdc128f8c7298d",
        "userName": "Robert Downey",
        "isAdmin": true,
        "profession": "Admin",
        "createdDate": "2019-12-16T17:40:41.497Z",
        "__v": 0,
        "id": "5df7c19922fa12000424304b"
    },
    {
        "isActive": false,
        "_id": "5df7c3b01c42ae00049a0c43",
        "post": "",
        "userId": "5df7b61e56cdc128f8c7298b",
        "userName": "Robert Downey",
        "userPhotoId": "5df7b6b156cdc128f8c7298d",
        "postImageId": "5df7c3af1c42ae00049a0c42",
        "isAdmin": true,
        "profession": "Admin",
        "createdDate": "2019-12-16T17:49:36.840Z",
        "__v": 0,
        "id": "5df7c3b01c42ae00049a0c43"
    },
```

d. **Get Posts by User ID** – Fetch all posts posted by a specific user using user ID **'posts/findpostbyuserid'**

```
return this.http.post(this.apiBaseURL + 'posts/findpostbyuserid', { id: userId }
```

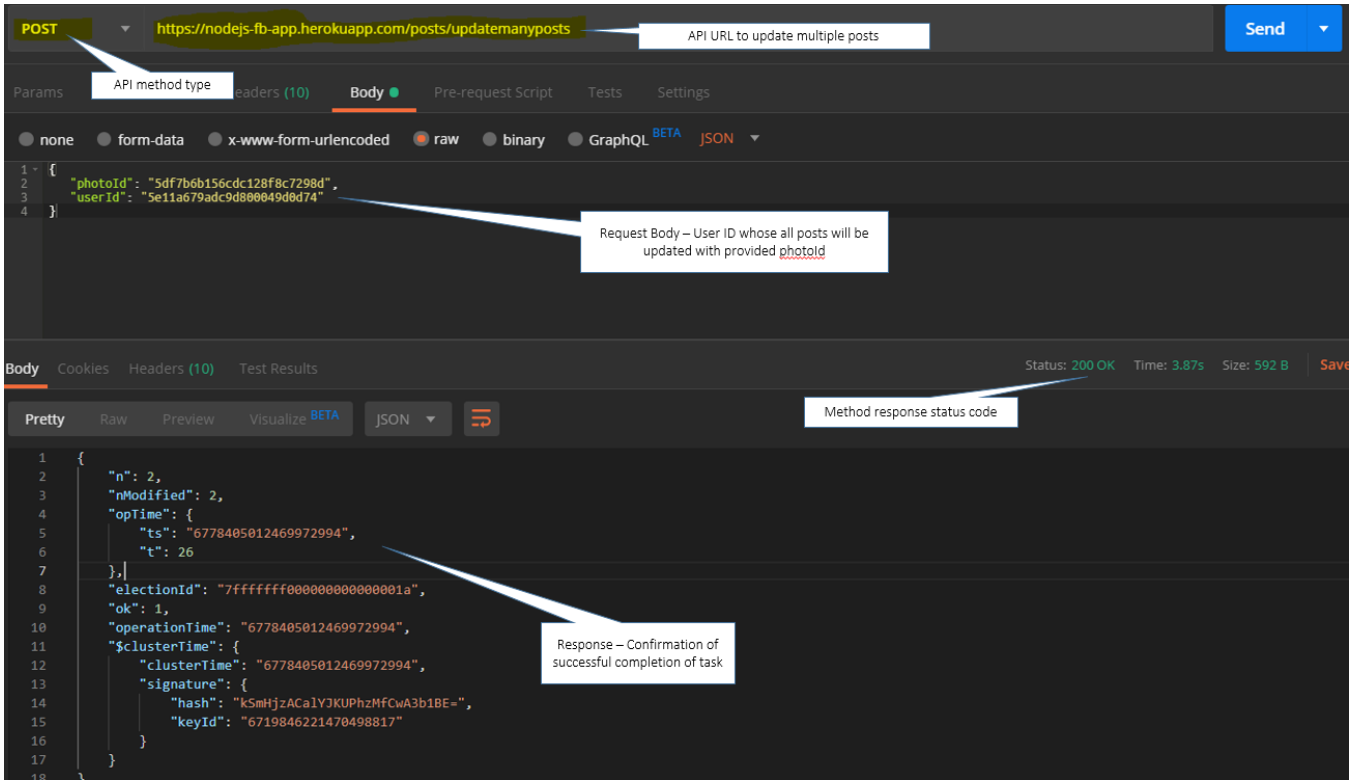



```
{
  "id": "5e11a679adc9d800049d0d74"
}
```

```
[
    {
        "isActive": false,
        "_id": "5e11b445adc9d800049d0d75",
        "post": "Sample post for documentation only ...",
        "userId": "5e11a679adc9d800049d0d74",
        "userName": "Bill CLinton",
        "userPhotoId": "5e11a679adc9d800049d0d73",
        "postImageId": "",
        "isAdmin": false,
        "profession": "President",
        "createdDate": "2020-01-05T10:02:45.508Z",
        "__v": 0,
        "id": "5e11b445adc9d800049d0d75"
    }
]
```

e. **<u>Update Bulk Posts</u>** – Update multiple post record in singe request. Used to update user profile photo for each post record when the photo of user changed who has posted the posts
'posts/updatemanyposts'

```
return this.http.post<Post>(this.apiBaseURL + 'posts/updatemanyposts', updatePayload)
```



f. **<u>Update Post</u>** – Update a single post content using post Id
'posts/'

```
return this.http.put<Post>(this.apiBaseURL + 'posts/' + updatedPost.id, updatedPost)
```

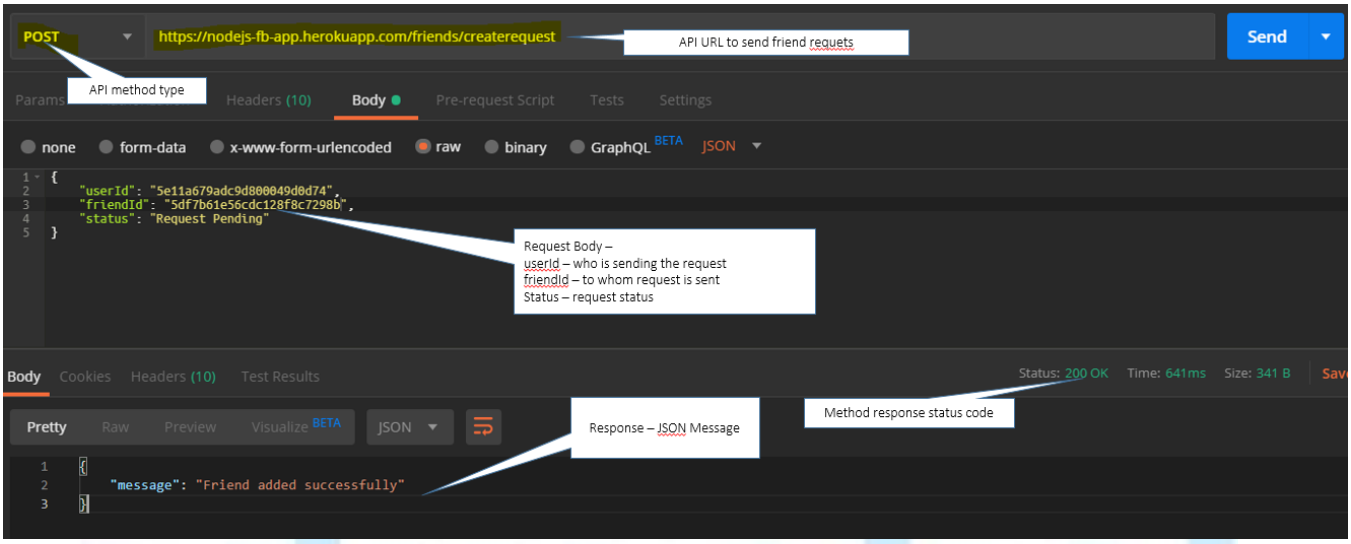
g. **Delete Post –**

```
return this.http.delete<Post>(this.apiBaseURL + 'posts/' + deletedPost.id)
```

3) Friends Service:

a. Create request: 'friends/createrequest'

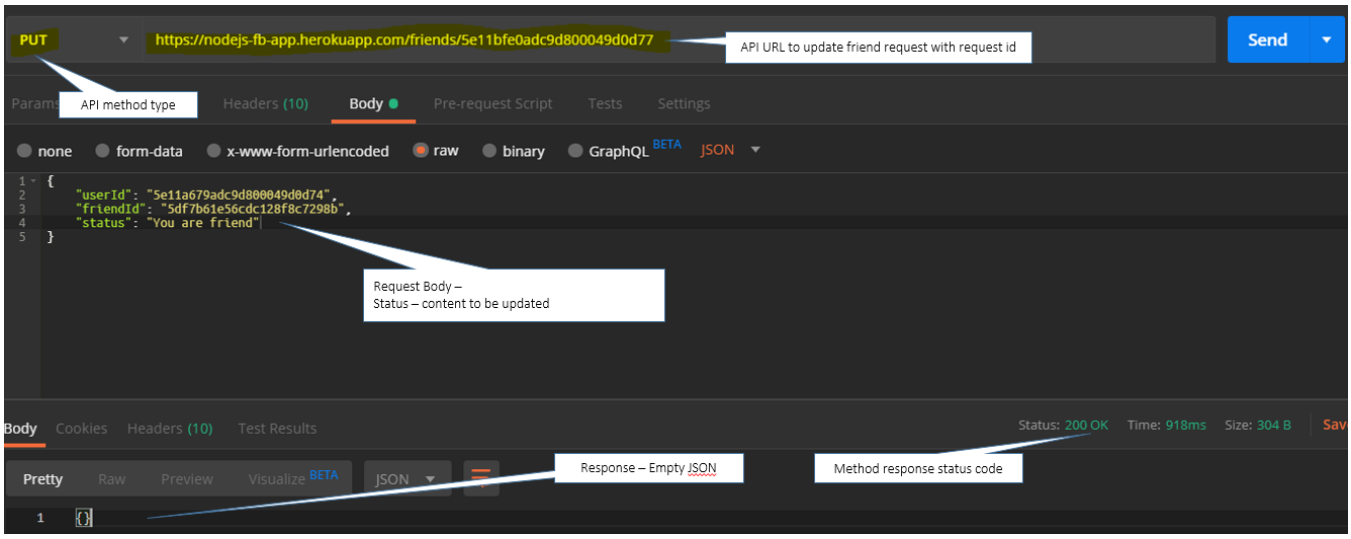
```
return this.http.post<Friend>(this.apiBaseURL + 'friends/createrequest'
```



b. Update Friend Request by ID – Update any friend request by unique request ID

'friends/' + updateRequest.id, updatedRequest

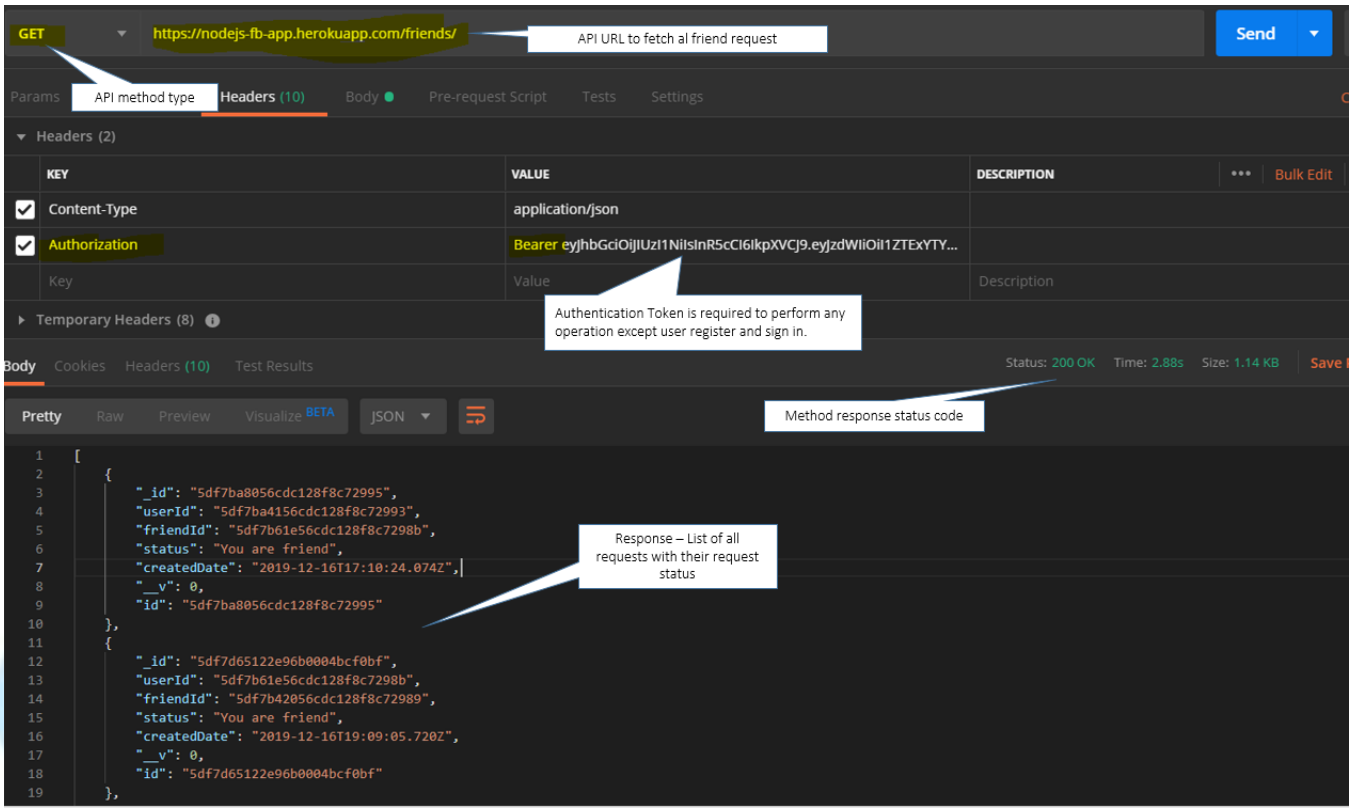
```
return this.http.put(this.apiBaseURL + 'friends/' + updatedRequest.id, updatedRequest).
```



c. **Get All Friend Request** – Retrieve all friend requests available in the system

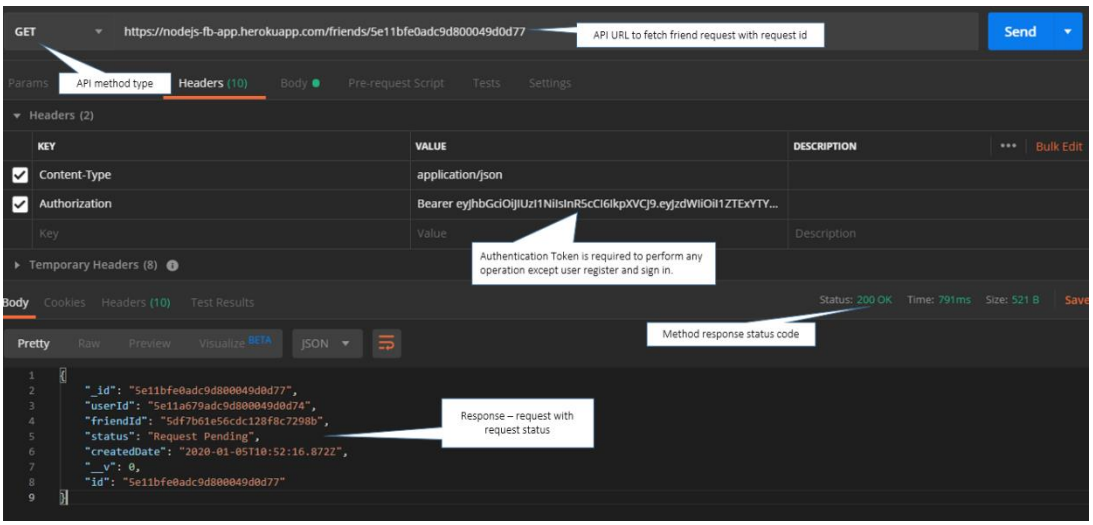
'friends/'

```
return this.http.get<any[]>(this.apiBaseURL + 'friends/')
```



d. **<u>Get Friend Request by ID</u>** – Simply retrieve any friend request by unique request ID

```
return this.http.get<Friend>(this.apiBaseURL + 'friends/' + id)
```



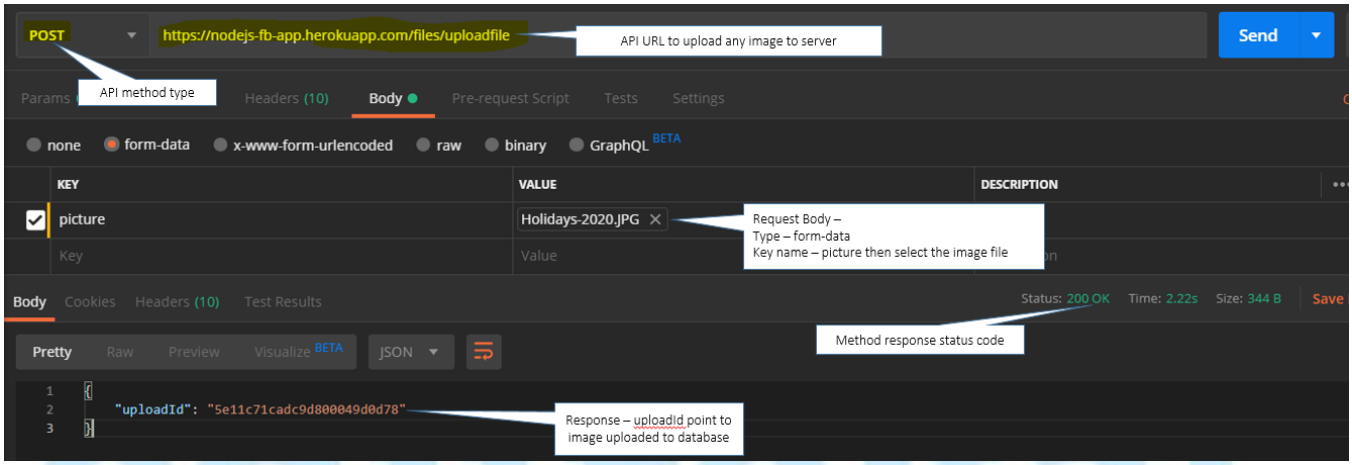
4) File Upload Service:

This module is responsible for uploading and downloading the user profile image and post images from server/database

a. **Upload File** – Upload any new user profile photo or post image

'files/uploadfile'

```
return this.http.post<any>(this.apiBaseURL + 'files/uploadfile', formData)
```



b. **Get File/Photo by ID** – Download/Fetch any user profile photo or post image using photo ID

'files/' + photoId

```
return this.http.get(this.apiBaseURL + 'files/' + photoId, { responseType: "blob" })
```