

Angular Certification Training

Certification Project Statement

edureka!

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

FBOOK	Register Login
Registration	
First Name	
Last Name	
Email	
Password	
Login	Register

2) Login page

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

FBOOK	Register Login
Login	
Username	
Password	
Forgot Password	Login

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

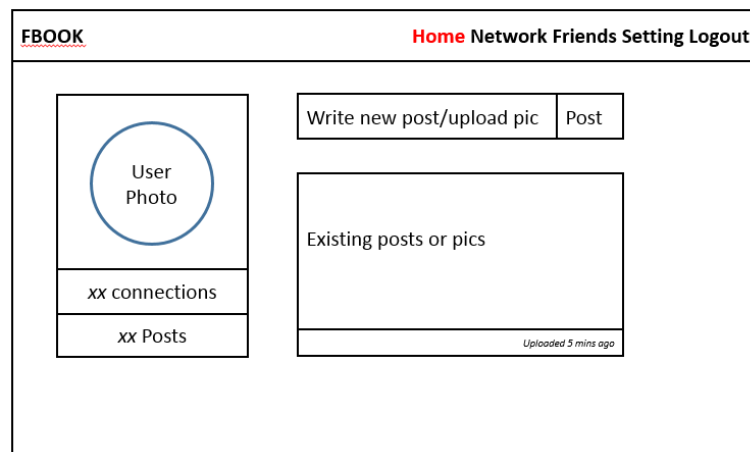
FBOOK	Register Login
<div><div>Forgot Password</div><div>Email</div><div>DOB</div><div>GO</div></div>	

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

FBOOK	Register Login
<div><div>Reset Password</div><div>New Password</div><div>Confirm Password</div><div>RESET</div></div>	

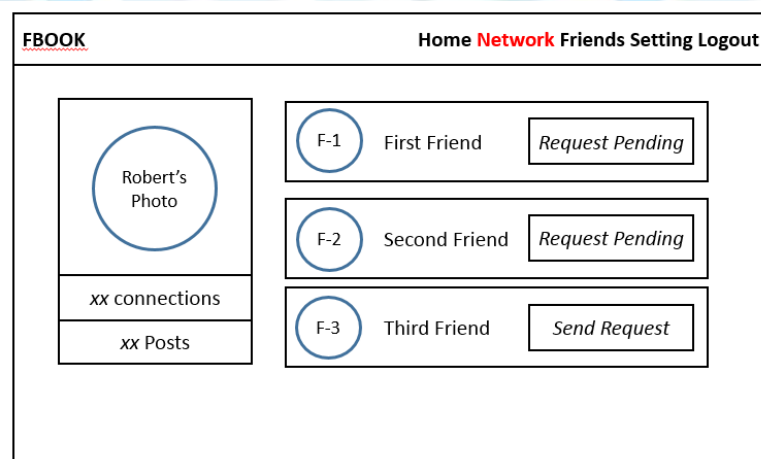
4) Home page

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

FBOOK	Home Network Friends Setting Logout
<div>Robert's Photo</div> <div>xx connections</div> <div>xx Posts</div>	<div>F-1 First Friend</div> <div>F-2 Second Friend</div>

7) Settings page:

a.) Profile Settings:

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

FBOOK	Home Network Friends Setting Logout												
<table border="1"> <thead> <tr> <th>Profile Settings</th> <th>Change Password</th> </tr> </thead> <tbody> <tr> <td>Robert</td> <td>Downey</td> </tr> <tr> <td>Male</td> <td>01/10/1956</td> </tr> <tr> <td>Robert@gmail.com</td> <td>98347237493</td> </tr> <tr> <td>USA</td> <td>Wisconsin</td> </tr> <tr> <td colspan="2">SAVE</td> </tr> </tbody> </table>	Profile Settings	Change Password	Robert	Downey	Male	01/10/1956	Robert@gmail.com	98347237493	USA	Wisconsin	SAVE		
Profile Settings	Change Password												
Robert	Downey												
Male	01/10/1956												
Robert@gmail.com	98347237493												
USA	Wisconsin												
SAVE													

b.) Change password page

This page will allow user to change the password

FBOOK		Home Network Friends Setting Logout
Profile Settings	Change Password	
New Password		
Confirm Password		
SAVE		

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

FBOOK		Home Users Network Friends Setting Logout
<div>Admin Photo</div> <div>xx connections</div> <div>xx Posts</div>	<div>user1</div> <div>user2</div> <div>user3</div> <div>user4</div> <div>user5</div> <div>user6</div>	

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:

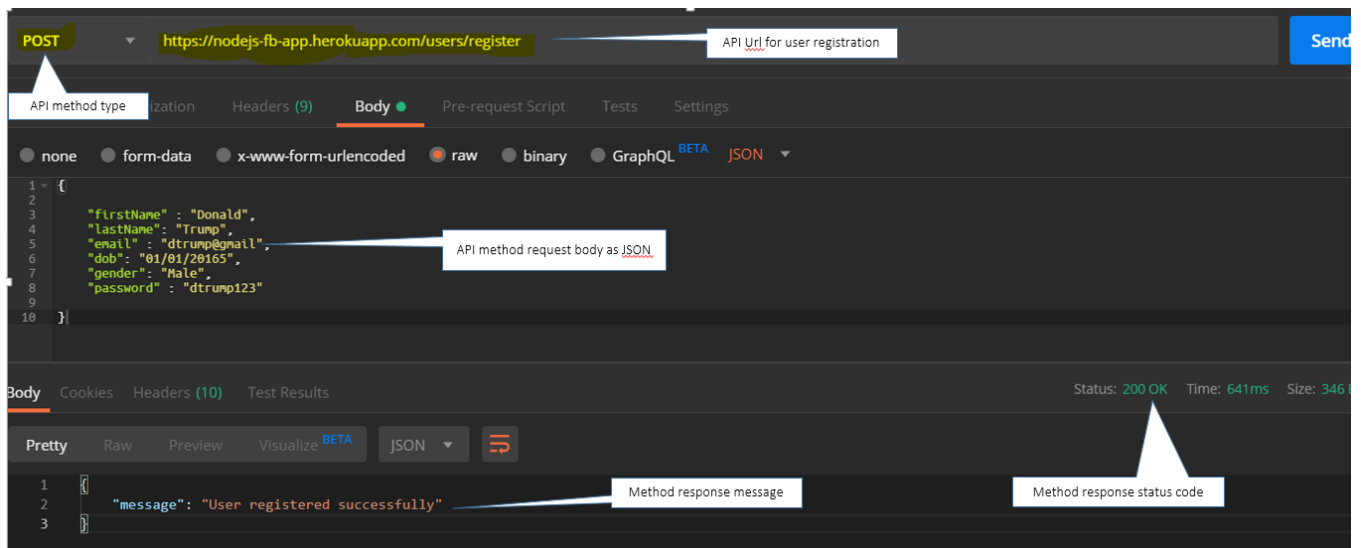
<https://nodejs-fb-app.herokuapp.com/>

API Routes

1) User Service

a. User Registration: 'users/register'

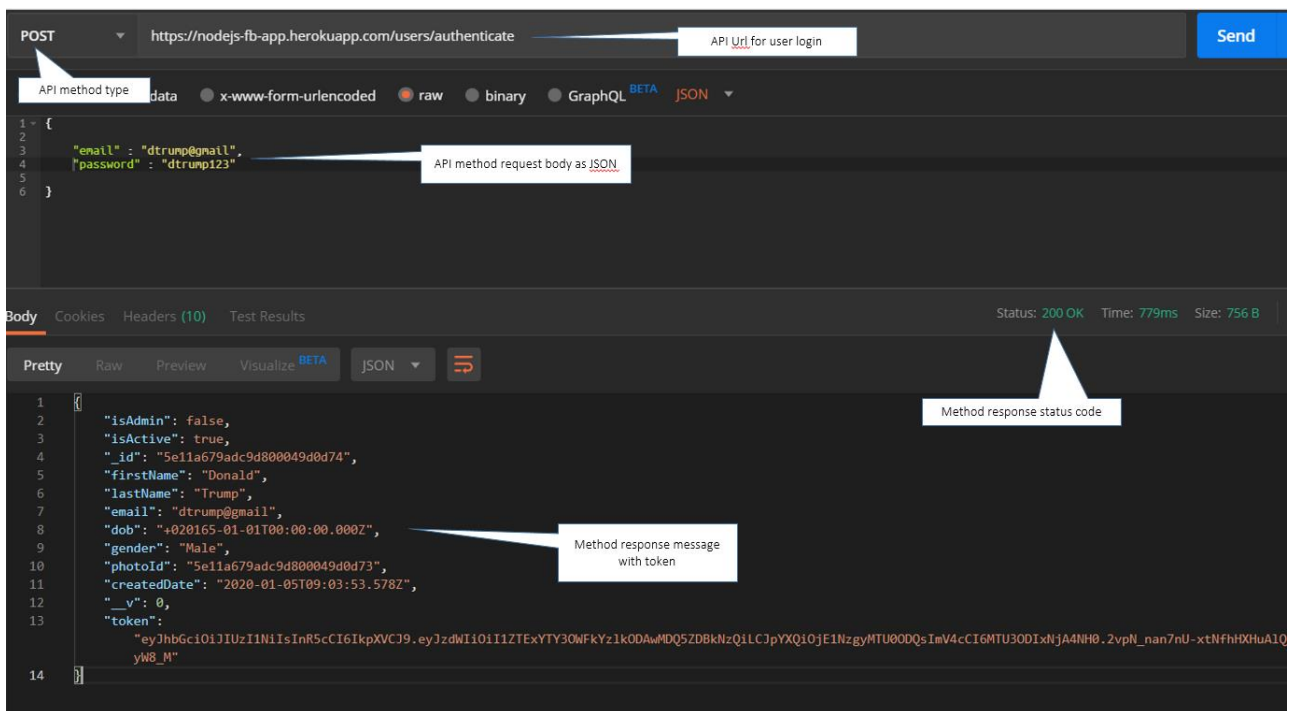

```
return this.http.post<User>(this.apiBaseUrl + 'users/register', newUser);
```



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/')
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `https://nodejs-fb-app.herokuapp.com/users/` (Annotated: API URL for get all users)
- Headers (2):**
 - Content-Type:** `application/json`
 - Authorization:** `Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxZTEzYTExYT...` (Annotated: Bearer token – which received while user logged)
- Temporary Headers (8):** (Expanded view shows Key and Value columns)
- Status:** 200 OK (Annotated: Method response status code)
- Time:** 2.65s
- Size:** 2.17 KB
- Body:** JSON format showing a list of users. The first user object is:


```
{
  "isAdmin": false,
  "isActive": true,
  "id": "5dfb128cb8de0d0004b23b27",
  "firstName": "Edureka",
  "lastName": "test",
  "gender": "Male",
  "dob": "2013-10-18T18:30:00.000Z",
  "phone": "",
  "city": "",
  "state": "",
  "country": "",
  "pincode": ""
}
```

 (Annotated: Response – List of all users registered or available)

- 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)
```

API method type

API URL for get user with ID

Send

Headers (10)

KEY	VALUE	DESCRIPTION
Content-Type	application/json	
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1ZTEeYTY3...	

Bearer token – which received while user logged

Method response status code

Status: 200 OK Time: 741ms Size: 605 B Save

Response – User response with user ID

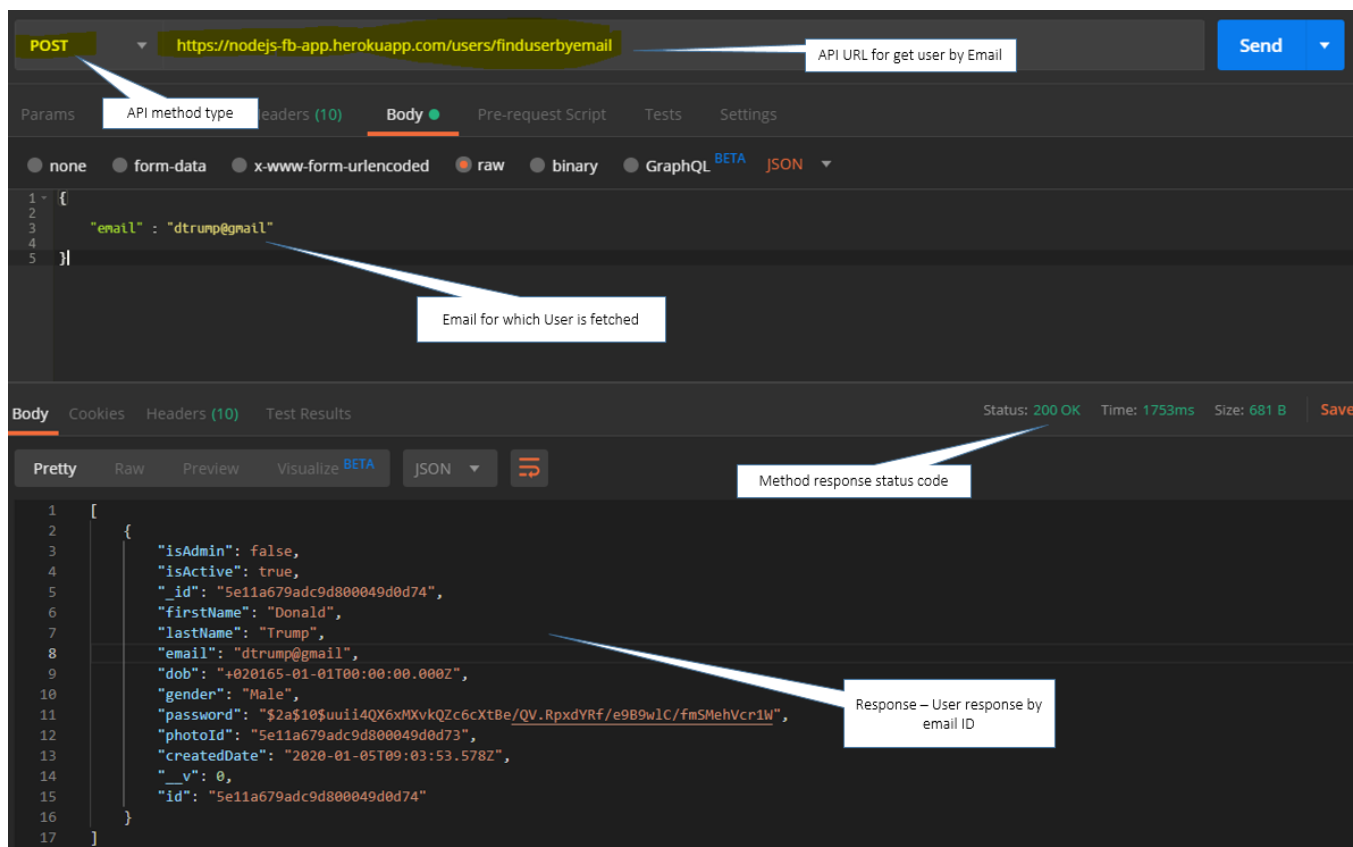
```

1 {
2   "isAdmin": false,
3   "isActive": true,
4   "id": "5e11a679adc9d800049d0d74",
5   "firstName": "Donald",
6   "lastName": "Trump",
7   "email": "dtrump@gmail",
8   "dob": "+020165-01-01T00:00:00.000Z",
9   "gender": "Male",
10  "photoId": "5e11a679adc9d800049d0d73",
11  "createdAt": "2020-01-05T09:03:53.578Z",
12  "__v": 0,
13  "id": "5e11a679adc9d800049d0d74"
14 }
    
```

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

'users/finduserbyemail'

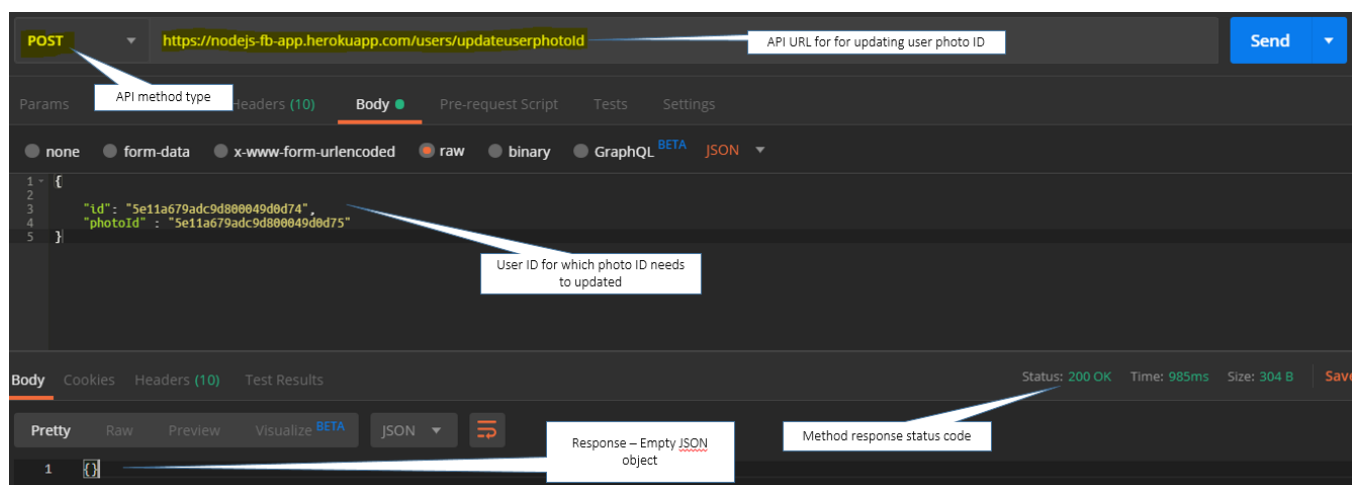
```
return this.http.post(this.apiBaseURL + 'users/finduserbyemail', { email: email })
```



f. **Update user photo ID** – Update user's photo ID using user ID

'users/updateuserphotoId'

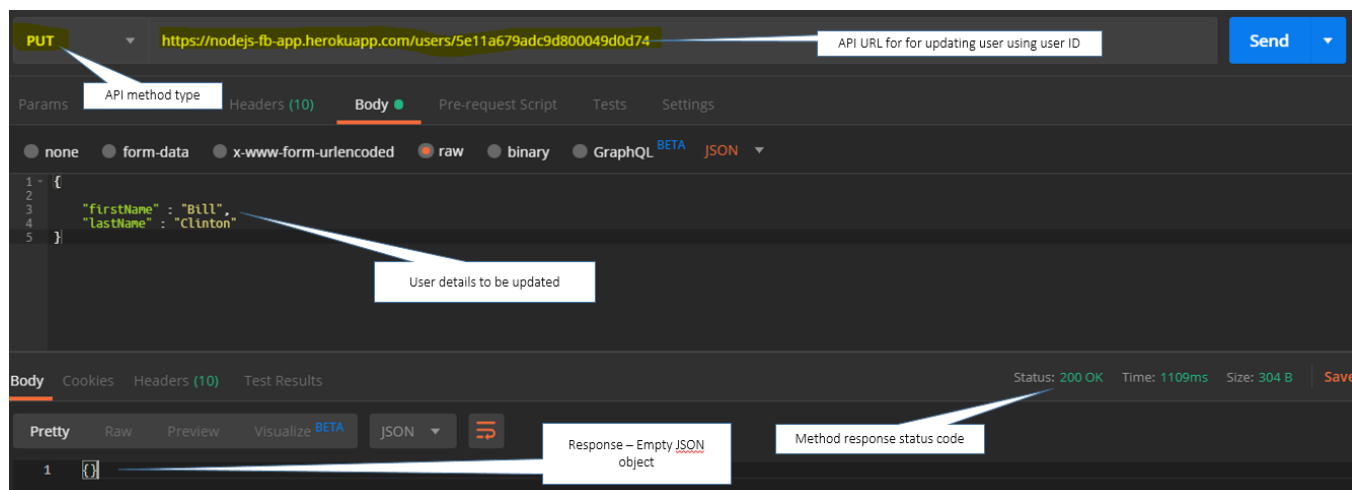
```
updateUserPhoto(updateUser) {
  return this.http.post(this.apiBaseUrl + 'users/updateuserphotoId', updatedUser,
```



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

'users/' + updatedUser.id

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

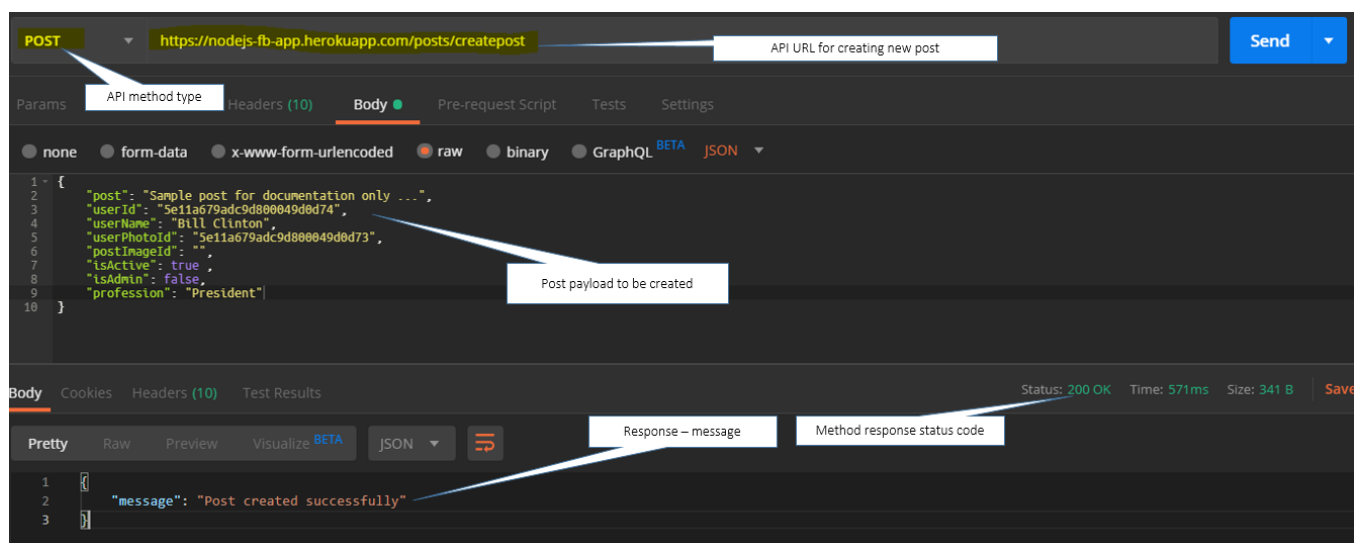


2) Post service:

- 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)
```

The screenshot shows a REST client interface with the following details:

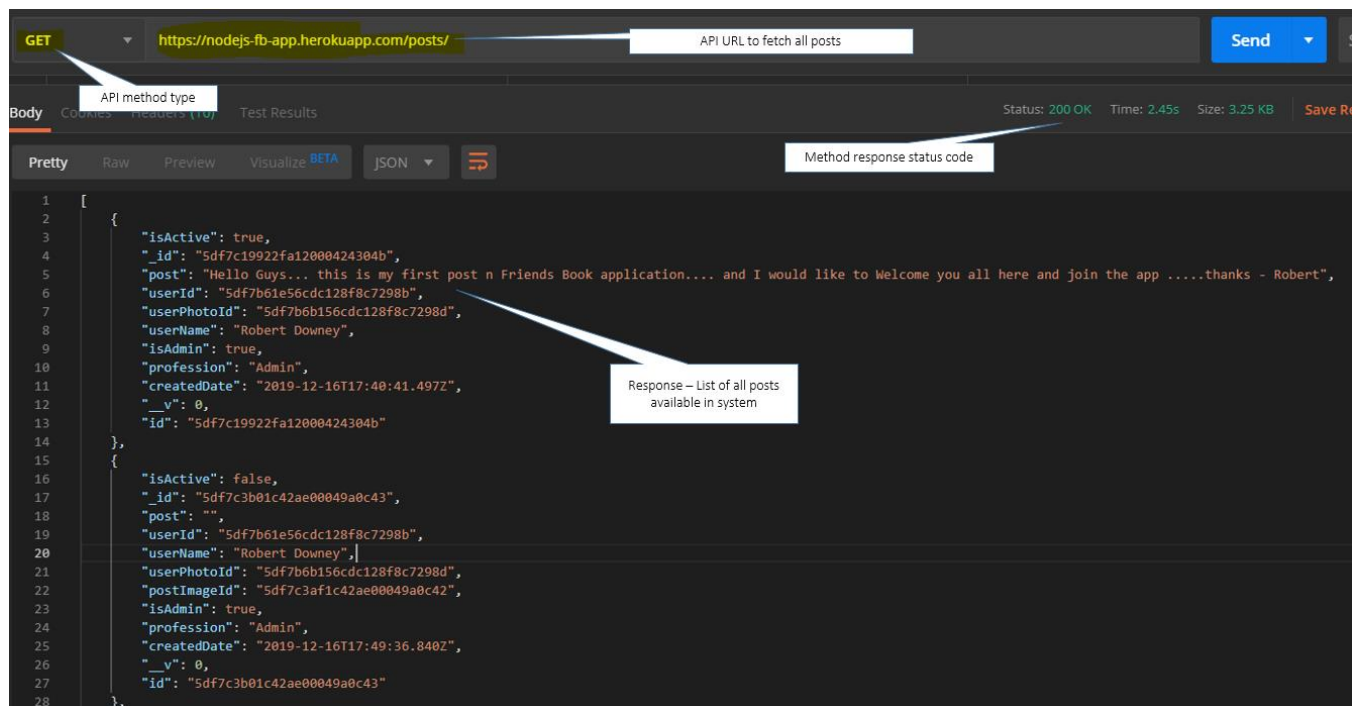
- Method:** POST
- URL:** `https://nodejs-fb-app.herokuapp.com/posts/findpostbyuserid` (Annotated: API URL to fetch all posts by user)
- Body:** `{ "id": "5e11a679adc9d800049d0d74" }` (Annotated: Request Body – User id for which all posts has to be fetched)
- Status:** 200 OK (Annotated: Method response status code)
- Response Body:**

```
[
  {
    "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"
  }
]
```

(Annotated: Response – List of Posts posted by single user)

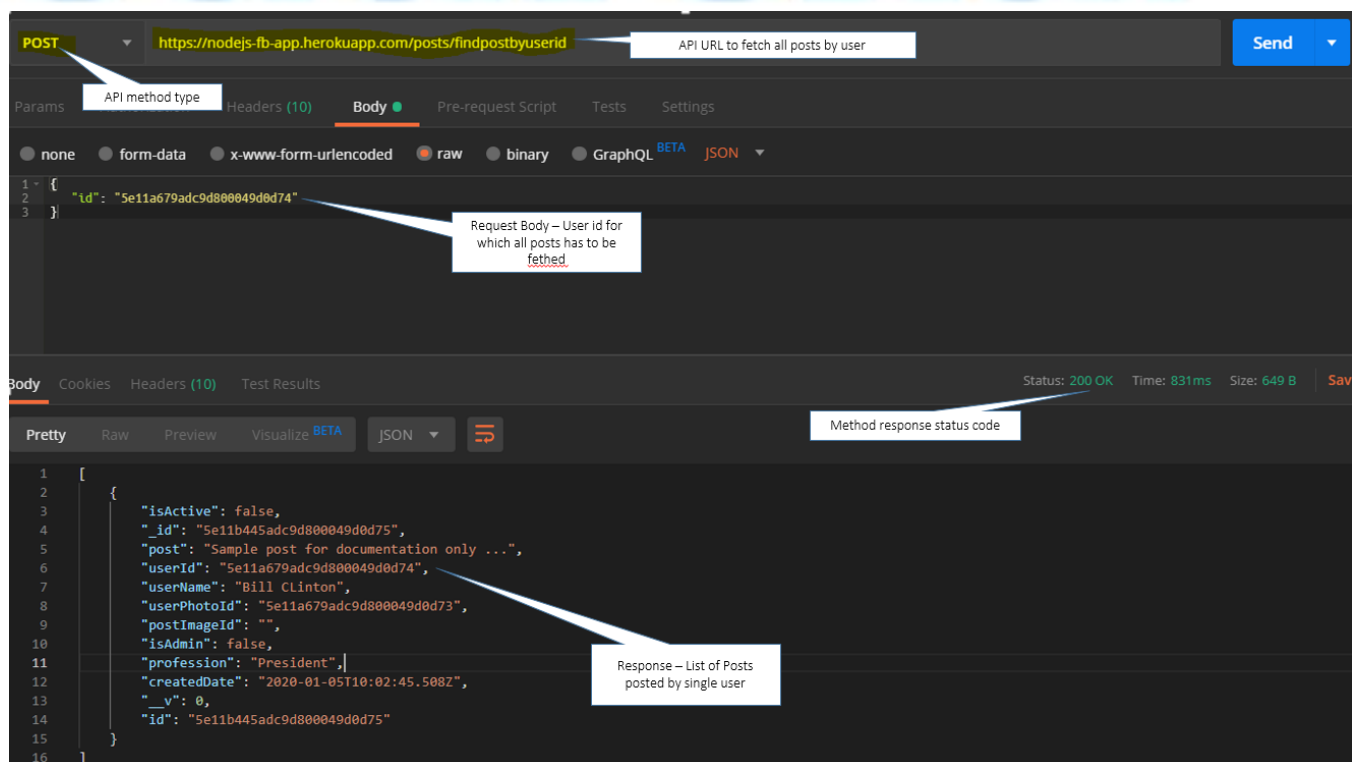
- c. **Get All Post** – Fetch all active posts created/available in the system

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



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 })
```



- e. **Update Bulk Posts** – Update multiple post record in single 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)
```

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** https://nodejs-fb-app.herokuapp.com/posts/updatemanyposts (Annotated: API URL to update multiple posts)
- Body:** JSON


```
{
  "photoId": "5df7b6b156cdc128f8c7298d",
  "userId": "5e11a679adc9d880049d0d74"
}
```

 (Annotated: Request Body – User ID whose all posts will be updated with provided photoId)
- Status:** 200 OK (Annotated: Method response status code)
- Response Body:**

```
{
  "n": 2,
  "nModified": 2,
  "opTime": {
    "ts": "6778405012469972994",
    "t": 26
  },
  "electionId": "7fffffff0000000000000001a",
  "ok": 1,
  "operationTime": "6778405012469972994",
  "$clusterTime": {
    "clusterTime": "6778405012469972994",
    "signature": {
      "hash": "kSmHjzACaIYJKUPhzMfCwA3b1BE=",
      "keyId": "6719846221470498817"
    }
  }
}
```

 (Annotated: Response – Confirmation of successful completion of task)

- f. **Update Post** – 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'
```

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `https://nodejs-fb-app.herokuapp.com/friends/createrequest` (labeled as "API URL to send friend requests")
- Body:**

```
{
  "userId": "5e11a679adc9d800049d0d74",
  "friendId": "5df7b61e56cdc128f8c7298b",
  "status": "Request Pending"
}
```

Request Body –
 userId – who is sending the request
 friendId – to whom request is sent
 Status – request status
- Response:**

```
{
  "message": "Friend added successfully"
}
```

Response – JSON Message
- Status:** 200 OK, Time: 641ms, Size: 341 B

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)
```

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** `https://nodejs-fb-app.herokuapp.com/friends/5e11bfe0adc9d800049d0d77` (labeled as "API URL to update friend request with request id")
- Body:**

```
{
  "userId": "5e11a679adc9d800049d0d74",
  "friendId": "5df7b61e56cdc128f8c7298b",
  "status": "You are friend"
}
```

Request Body –
 Status – content to be updated
- Response:** Empty JSON
- Status:** 200 OK, Time: 918ms, Size: 304 B

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

'friends/'

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

GET <https://nodejs-fb-app.herokuapp.com/friends/> API URL to fetch all friend request

API method type Headers (10) Body Pre-request Script Tests Settings

Headers (2)

KEY	VALUE	DESCRIPTION
Content-Type	application/json	
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1ZTEyYTY...	

Temporary Headers (8)

Body Cookies Headers (10) Test Results

Status: 200 OK Time: 2.88s Size: 1.14 KB Save

Method response status code

```

1 {
2   {
3     "_id": "5df7ba8056cdc128f8c72995",
4     "userId": "5df7ba4156cdc128f8c72993",
5     "friendId": "5df7b61e56cdc128f8c7298b",
6     "status": "You are friend",
7     "createdAt": "2019-12-16T17:10:24.074Z",
8     "_v": 0,
9     "id": "5df7ba8056cdc128f8c72995"
10  },
11  {
12    "_id": "5df7d65122e96b0004bcf0bf",
13    "userId": "5df7b61e56cdc128f8c7298b",
14    "friendId": "5df7b42056cdc128f8c72989",
15    "status": "You are friend",
16    "createdAt": "2019-12-16T19:09:05.720Z",
17    "_v": 0,
18    "id": "5df7d65122e96b0004bcf0bf"
19  },
20 }

```

Response - List of all requests with their request status

d. **Get Friend Request by ID** – Simply retrieve any friend request by unique request ID

```
friendsById(id: string) {
  return this.http.get<Friend>(this.apiBaseUrl + 'friends/' + id)
```

GET <https://nodejs-fb-app.herokuapp.com/friends/5e11bfe0adc9d800049d0d77> API URL to fetch friend request with request id

API method type Headers (10) Body Pre-request Script Tests Settings

Headers (2)

KEY	VALUE	DESCRIPTION
Content-Type	application/json	
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1ZTEyYTY...	

Temporary Headers (8)

Body Cookies Headers (10) Test Results

Status: 200 OK Time: 791ms Size: 521 B Save

Method response status code

```

1 {
2   "_id": "5e11bfe0adc9d800049d0d77",
3   "userId": "5e11a679adc9d800049d0d74",
4   "friendId": "5df7b61e56cdc128f8c7298b",
5   "status": "Request Pending",
6   "createdAt": "2020-01-05T10:52:16.872Z",
7   "_v": 0,
8   "id": "5e11bfe0adc9d800049d0d77"
9 }

```

Response - request with request status

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)
```

The screenshot shows a Postman interface for a POST request. The URL is `https://nodejs-fb-app.herokuapp.com/files/uploadfile`. The API method type is set to POST. The request body is configured as form-data with a key named 'picture' and a value of 'Holidays-2020.JPG'. The response status is 200 OK, and the response body is `"uploadId": "5e11c71cad9d80049d0d78"`.

Annotations:

- API method type
- API URL to upload any image to server
- Request Body – Type – form-data
Key name – picture then select the image file
- Method response status code
- Response – uploadId point to image uploaded to database

- 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" })
```

The screenshot shows a Postman interface for a GET request. The URL is `https://nodejs-fb-app.herokuapp.com/files/5e11c71cad9d80049d0d78`. The API method type is set to GET. The request headers include 'Content-Type: application/json' and 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxZTExYTYy...'. The response status is 200 OK, and the response body is a blob type response for image content.

Annotations:

- API method type
- API URL to upload any image to server
- Response – Blob type response for image content
- Method response status code