

CST 2120 | Coursework 2

Lavisha Bhagnani

December 13, 2024

—

Social Networking Website

—

M00959112

Screenshots of Postman

1. Registration

The screenshot shows a Postman request configuration. The method is set to POST, and the URL is `http://localhost:8080/M00959112/users`. The Body tab is selected, containing the following JSON payload:

```
1 {  
2   "name": "projectUser",  
3   "email": "projectUser@example.com",  
4   "password": "Password123",  
5   "dob": "2000-01-01",  
6   "gender": "male"  
7 }  
8
```

The response status is 201 Created, with a response time of 123 ms and a size of 488 B. The response body is:

```
1 {  
2   "registration": true,  
3   "username": "projectUser",  
4   "message": "Registration successful."  
5 }
```

- All inputs are required.
- There are validations of each input:
 - Username must be at least 3 letters.
 - Email and password use regex.
 - DOB and gender are must.
- If username or email address is existing in database it will give message that “Username already exists” or “Email already exists”.

This is how it shown in mongo dB

```
_id: ObjectId('6759a3dfed628614f538cc0')  
name: "projectUser"  
email: "projectUser@example.com"  
password: "Password123"  
dob: "2000-01-01"  
gender: "male"  
following: Array (empty)
```

2.Login

The screenshot shows a POST request to `http://localhost:8080/M00959112/login`. The Body tab is selected, containing the following JSON payload:

```
1 {
2   "name": "projectUser",
3   "password": "Password123"
4 }
```

The response status is `200 OK` with a response time of 38 ms, a body size of 413 B, and a global icon. The response body is:

```
1 {
2   "login": true
3 }
```

- If any of the fields are left empty or incorrect. It will give error.

3.Post a blog

The screenshot shows a POST request to `http://localhost:8080/M00959112/contents`. The Body tab is selected, containing the following JSON payload:

```
1 {
2   "text": "Example content by projectUser"
3 }
```

The response status is `200 OK` with a response time of 12 ms, a body size of 292 B, and a global icon. The response body is:

```
1 {
2   "success": true,
3   "message": "Content posted successfully."
4 }
```

- A user must be logged in to post a blog.
- It cannot be posted empty.
- Mongo Db

```
_id: ObjectId('6759a416ed628614f538cccd1')
username : "projectUser"
text : "Example content by projectUser"
createdAt : 2024-12-11T14:39:18.115+00:00
```

4.Follow a User

The screenshot shows a Postman interface with a POST request to `http://localhost:8080/M00959112/follow`. The request body is a JSON object with a single key-value pair: `"usernameToFollow": "marium"`. The response status is 200 OK, with a response body indicating success and a message: `{"success": true, "message": "You are now following marium"}`.

```
1 {
2   "usernameToFollow": "marium"
3 }
4
```

Body Cookies (1) Headers (7) Test Results ⏱

200 OK • 17 ms • 292 B • Save Response ...

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "message": "You are now following marium"
4 }
```

- A user must be logged in to follow someone
- If no username provided to follow it will give error “No username provided to Follow”
- A user cannot follow their self.
- If no user found it give proper error.
- If user is already following the name provided again it will give valid error.

5.Login Status

The screenshot shows a Postman interface with a GET request to `http://localhost:8080/M00959112/login`. The response status is 200 OK, with a response body indicating the user is logged in and their username: `{"login": true, "username": "projectUser"}`.

GET `http://localhost:8080/M00959112/login` Send

Params Authorization Headers (8) Body Scripts Settings Cookies

None form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
1
```

Body Cookies (1) Headers (7) Test Results ⏱

200 OK • 11 ms • 274 B • Save Response ...

Pretty Raw Preview Visualize JSON

```
1 {
2   "login": true,
3   "username": "projectUser"
4 }
```

6.Content of followed Users

The screenshot shows a REST API testing interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/M00959112/contents
- Headers:** (8)
- Body:** (Pretty, Raw, Preview, Visualize, JSON selected)
- Response Status:** 200 OK
- Response Time:** 14 ms
- Response Size:** 512 B
- Content:** A JSON response showing two posts from user "marium".

```
1 {  
2   "content": [  
3     {  
4       "_id": "67532b9b227ed2369a463d16",  
5       "username": "marium",  
6       "text": "hello i am lavisha and this is my first blog",  
7       "createdAt": "2024-12-06T16:51:39.203Z"  
8     },  
9     {  
10       "_id": "675845fd512bff558f821f24",  
11       "username": "marium",  
12       "text": "hellooooooo",  
13       "createdAt": "2024-12-10T13:45:33.176Z"  
14     }  
15   ]  
16 }
```

- A user must be logged in to view the content.
- If logged in user is not following anyone it will give proper error.

7. Search for Users

The screenshot shows a REST API testing interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/M00959112/users/search?q=sonu
- Params:** A table showing query parameters:

Key	Value	Description	Bulk Edit
q	sonu		
Key	Value	Description	
- Response Headers:** 200 OK, 26 ms, 356 B, Save Response, etc.
- Response Body (Pretty JSON):**

```
1 {  
2   "user": [  
3     {  
4       "_id": "6758aa7d8eb47c4fd449169c",  
5       "name": "sonu",  
6       "email": "sonu@gmail.com",  
7       "dob": "2000-11-11",  
8       "gender": "other"  
9     }  
10   ]  
11 }
```

- It will give proper errors if:
 - No query provided, or
 - No user found matching the query

8. Search for Contents

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

- Method:** GET
- URL:** http://localhost:8080/M00959112/contents/search?q=blog
- Params:** Authorization, Headers (8), Body, Scripts, Settings
- Query Params:** q (Value: blog)
- Status:** 200 OK (14 ms, 693 B)
- Body:** JSON response (Pretty printed) showing three blog posts.

```
1 {
2   "content": [
3     {
4       "_id": "6747815f8469398f22358cf8",
5       "username": "testUser",
6       "text": "This is a blog post from testUser.",
7       "createdAt": "2024-11-27T20:30:23.862Z"
8     },
9     {
10       "_id": "6747827c0c5e6ee1cd97183d",
11       "username": "anotherUser",
12       "text": "This is a blog post from anotherUser.",
13       "createdAt": "2024-11-27T20:35:08.501Z"
14     },
15     {
16       "_id": "67532b9b227ed2369a463d16",
17       "username": "marium",
18       "text": "hello i am lavisha and this is my first blog",
19       "createdAt": "2024-12-06T16:51:39.203Z"
20     }
21   ]
}
```

- It will give proper errors if:
 - No query provided, or
 - No content found matching the query

Screenshots of front-end

When user open <http://localhost:8080/> only registration, login buttons and login section will be visible until they login. Just like in 2.login.

1.Registration

The screenshot shows the registration page of a "Recipe Sharing" application. At the top, there is a green header bar with the title "Recipe Sharing". Below the header, there are two buttons: "Register" and "Login". The main content area has a light yellow background and contains a registration form. The form is titled "Welcome to Our Application! Please log in or register." and includes a "Register" button. It has four input fields: "Username" (placeholder: "Enter your username"), "Email Address" (placeholder: "Enter your email"), "Password" (placeholder: "Enter your password"), and "Date of Birth: dd/mm/yyyy" (with a dropdown menu for "Gender" containing "Select Gender"). A large red "Submit" button is at the bottom of the form.

- It validates all the input fields.
- If registration is successful or fails it gives appropriate messages.
- After clicking submit the data is saved in mongo Db.

```
_id: ObjectId('675b96aebb3cbce8d506a8a7')
name : "Jason"
email : "jason@gmail.com"
password : "Jason12345"
dob : "2000-02-01"
gender : "male"
▶ following : Array (empty)
```

```
_id: ObjectId('675b96eabb3cbce8d506a8a8')
name : "Michael"
email : "Michael@gmail.com"
password : "Jason12345"
dob : "2000-02-01"
gender : "male"
▶ following : Array (empty)
```

2.Login

The screenshot shows a web application titled "Recipe Sharing". At the top, there is a dark brown header bar with the title "Recipe Sharing" in white. Below the header, there are two buttons: "Register" and "Login". The main content area has a light beige background. It features a green-bordered box containing a welcome message: "Welcome to Our Application! Please log in or register." Below this message is a section titled "Login". This section includes two input fields: one for "Username" with placeholder text "Enter your username" and another for "Password" with placeholder text "Enter your password". At the bottom of this form is a large, horizontally gradient button with colors transitioning from red to green, labeled "Login".

© 2024 Recipe Sharing

- Proper messages will be displayed on the screen if there are any errors during logging in.
- After logging in
 - Logout
 - Load content
 - Search
 - Follow
 - Unfollow
 - Post a blog
- buttons will be visible.
- A welcome message will keep displaying with the user name who Is logged in

3. Post a blog with file upload

The screenshot shows the Recipe Sharing application interface. At the top, there is a dark brown header bar with the title "Recipe Sharing" in green. Below the header are five buttons: "Logout", "Load Content", "Search", "Follow", and "Unfollow". To the right of these is a button labeled "Post a Blog". The main content area has a light beige background. A white rectangular box contains the text "Welcome, Jason!". Below this is a large input field with a light gray border and a thin green outline, containing the placeholder text "Write your blog...". Underneath the input field are two buttons: "Choose file" and "No file chosen" (disabled), followed by a green rounded rectangle button labeled "Post Blog". At the bottom of the page, a dark brown footer bar displays the copyright information "© 2024 Recipe Sharing".

- When user is log's in its redirect to Post blog section
- User also have option to upload a picture with their blog
- My uploading file function is not working properly but it is being saved in the folder.
- For example, if a user chooses a file and post it with a blog that picture will be saved in a folder called “Uploads”. But unfortunately, it's not displaying in load content.
- After posting a successful message is displayed.

4. Load Content

- If user is not following anyone:

The screenshot shows the Recipe Sharing application's interface. At the top, there is a dark green header bar with the title "Recipe Sharing" in white. Below the header are several buttons: "Logout", "Load Content" (which is highlighted in blue), "Search", "Follow", "Unfollow", and "Post a Blog". The main content area has a light yellow background. It starts with a welcome message "Welcome, Jason!" in a white box. Below that is a section titled "Content from Followed Users" with the sub-instruction "You are not following any users." at the bottom. At the very bottom of the page is a dark brown footer bar containing the copyright notice "© 2024 Recipe Sharing".

- If you are following but no content is posted by the user

This screenshot is similar to the one above, showing the Recipe Sharing application's interface. The dark green header bar at the top features the title "Recipe Sharing" and various buttons: "Logout", "Load Content", "Search", "Follow", "Unfollow", and "Post a Blog". The main content area has a light yellow background. It begins with a welcome message "Welcome, Jason!" in a white box. Below it is a section titled "Content from Followed Users" with the message "No content from followed users." at the bottom. The dark brown footer bar at the bottom contains the copyright notice "© 2024 Recipe Sharing".

If posted

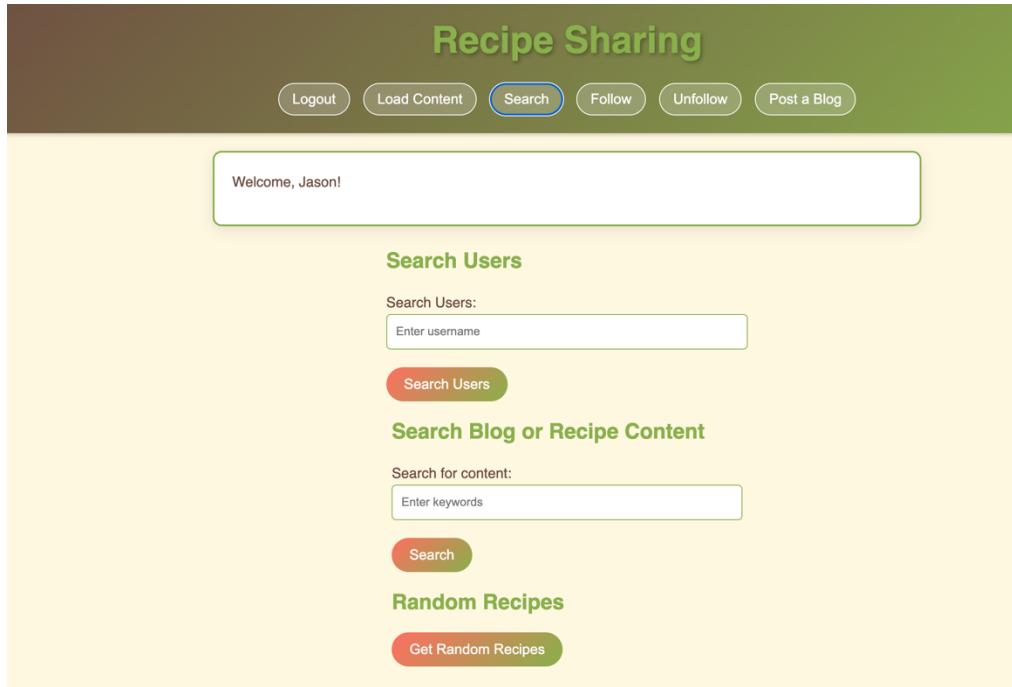
The screenshot shows a user interface for a "Recipe Sharing" application. At the top, there is a dark green header bar with the title "Recipe Sharing" in white. Below the header are several buttons: "Logout", "Load Content", "Search", "Follow", "Unfollow", and "Post a Blog". A welcome message "Welcome, Jason!" is displayed in a light blue box. The main content area has a yellow background and features a section titled "Content from Followed Users". Inside this section, there is a box containing a blog post by a user named Michael. The post text reads: "Michael: Hi I am Michael. This is my First blog with the failed picture file upload." Below the text is a small placeholder image icon labeled "Post image". At the bottom of the post box, it says "Posted on: 13/12/2024, 02:20:34".

- This blog is also being saved in Mongo DB.

```
_id: ObjectId('675b99f2bb3cbce8d506a8a9')
username : "Michael"
text : "Hi I am Michael.
      This is my First blog with the failed picture file u..."
image : "/uploads/1734056434027-content.png"
createdAt : 2024-12-13T02:20:34.035+00:00
```

5. Search

- User can:
 - Search for other users
 - Search for content from all users even if they are not being followed.
- Can also get random recipes.



6.Follow

- User can just go to follow navigation bar and write the name of the user they want to follow



7.Unfollow

- Same for the unfollow as follow.



8.Logout

- In the end when user clicks logout their session is destroyed and they go back to the login page.