CSE471 : System Analysis and Design
Lab Assignment on API Development and Postman

Project Title : CampusConnect- A Student Community Platform

Group Number : 01

Backend Framework : MERN

Submitted by : Simanta Hasan Kollol

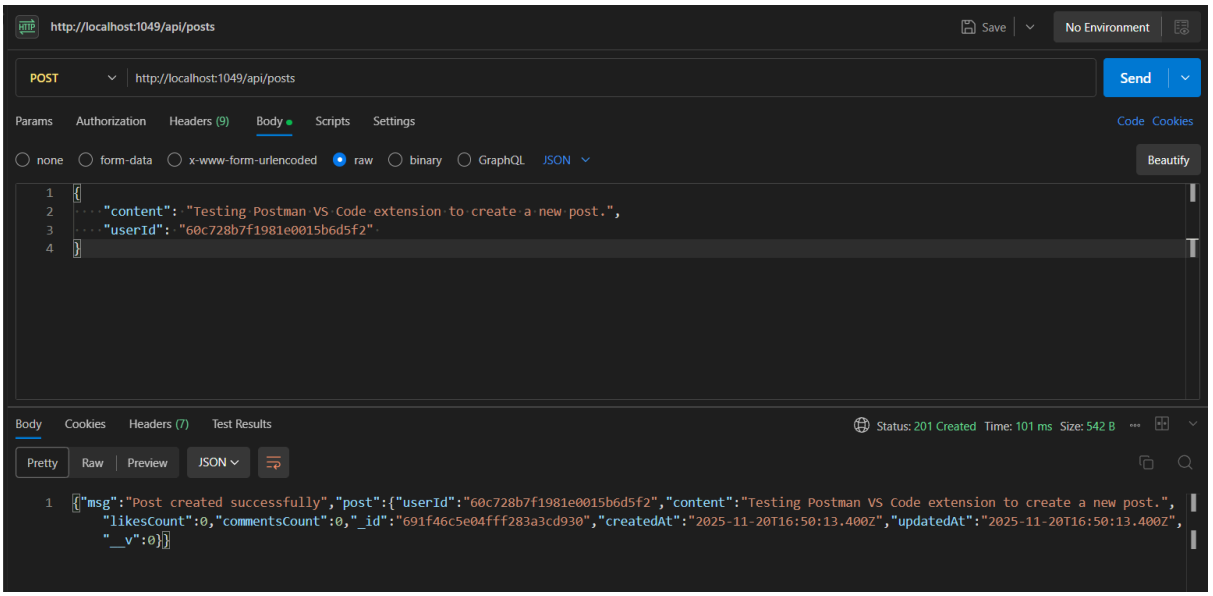
Student ID : 21201049

Section: 07

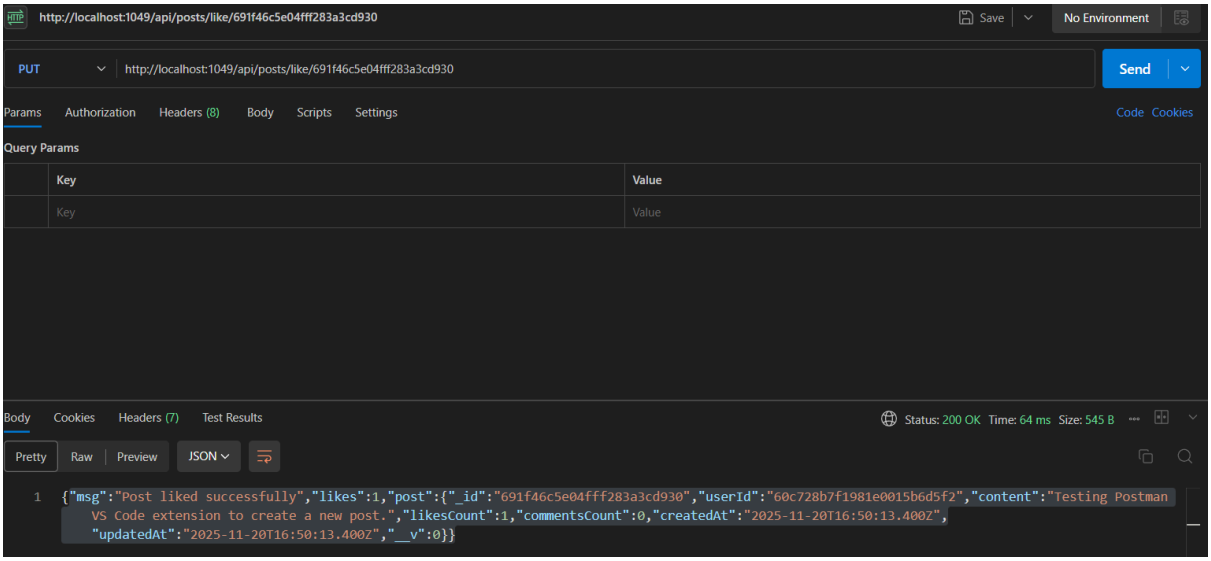
Date : 20 November 2025

APIs of Feature - 01 (Sorting Posts on Latest/Popularity)

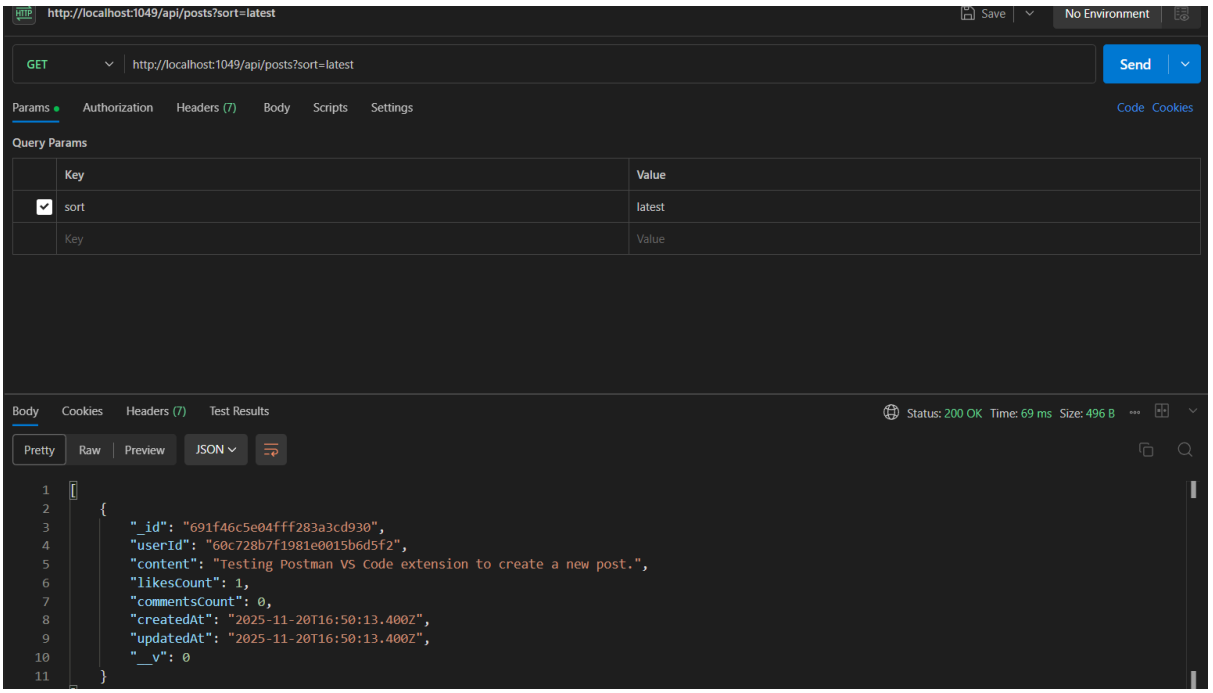
```
// 1. CREATE Post (POST /api/posts)
router.post('/', async (req, res) => {
    try {
        const { userId, content } = req.body;
        const newPost = new Post({
            userId: userId || '60c728b7f1981e0015b6d5f2', // Placeholder User ID
            content
        });
        const post = await newPost.save();
        res.status(201).json({ msg: 'Post created successfully', post });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});
```

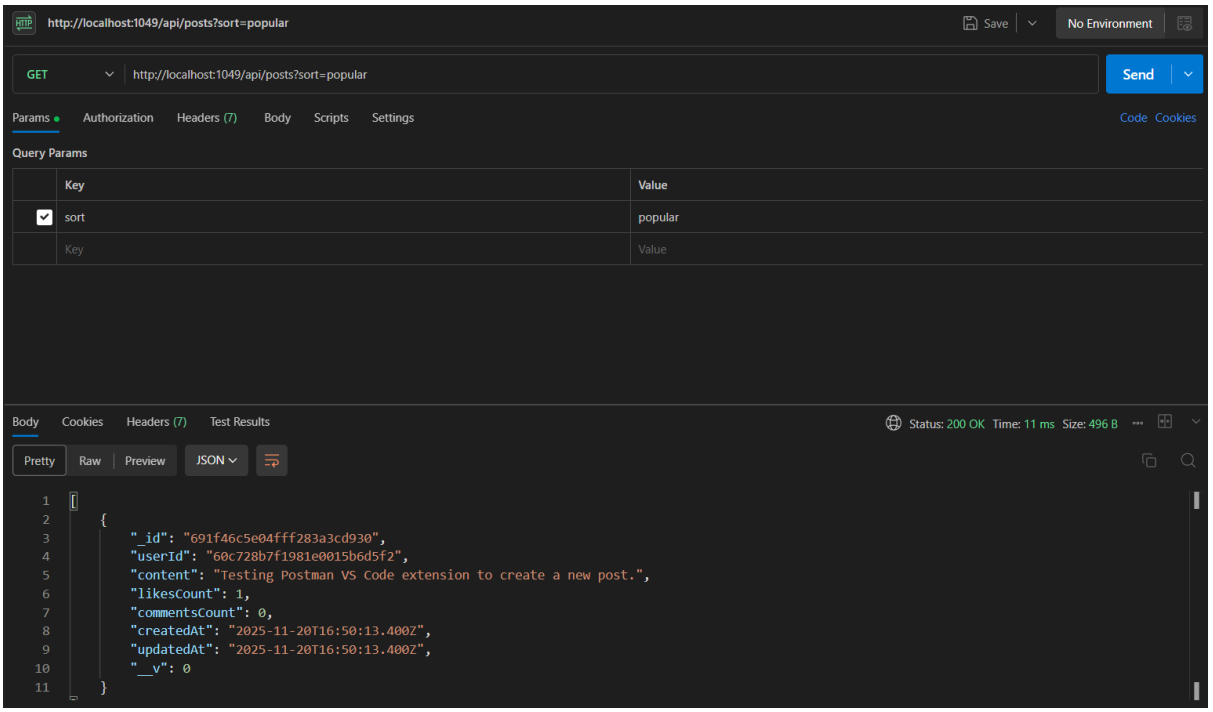


```
// 3. LIKE/UNLIKE a Post (PUT /api/posts/like/:id)
router.put('/like/:id', async (req, res) => {
    try {
        const post = await Post.findByIdAndUpdate(
            req.params.id,
            { $inc: { likesCount: 1 } }, // Atomic Increment operation
            { new: true } // Returns the updated document
        );

        if (!post) {
            return res.status(404).json({ msg: 'Post not found' });
        }

        res.status(200).json({ msg: 'Post liked successfully', likes: post.likesCount, post });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});
```



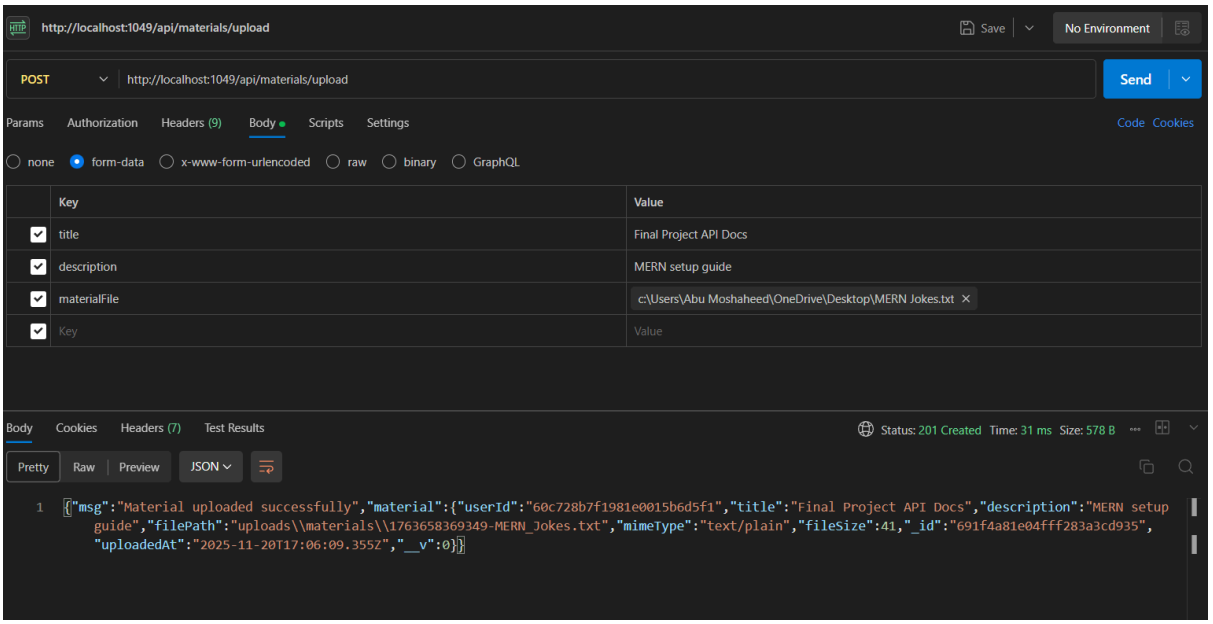

```
// 2. GET and SORT Posts (GET /api/posts?sort=...)
// Available sorts: 'latest' (default) or 'popular'
router.get('/', async (req, res) => {
    try {
        // Get 'sort' parameter from the query string, defaulting to 'latest'
        const { sort = 'latest' } = req.query;
        let posts;

        if (sort.toLowerCase() === 'popular') {
            // Sort by likesCount (descending) then by creation date
            posts = await Post.find().sort({ likesCount: -1, createdAt: -1 });
        } else {
            // Default (latest): Sort by the last updated time (updatedAt)
            posts = await Post.find().sort({ updatedAt: -1, createdAt: -1 });
        }

        res.status(200).json(posts);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});
```


http://localhost:1049/api/posts?sort=latest
Save
No Environment
GET
http://localhost:1049/api/posts?sort=latest
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Code
Cookies
Query Params
Key
Value
sort
latest
Key
Value
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 69 ms
Size: 496 B
Pretty
Raw
Preview
JSON
[
{
"_id": "691f46c5e04fff283a3cd930",
"userId": "60c728b7f1981e0015b6d5f2",
"content": "Testing Postman VS Code extension to create a new post.",
"likesCount": 1,
"commentsCount": 0,
"createdAt": "2025-11-20T16:50:13.400Z",
"updatedAt": "2025-11-20T16:50:13.400Z",
"__v": 0
}


http://localhost:1049/api/posts?sort=popular
Save
No Environment
GET
http://localhost:1049/api/posts?sort=popular
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Code
Cookies
Query Params
Key
Value
sort
popular
Key
Value
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 11 ms
Size: 496 B
Pretty
Raw
Preview
JSON
[
{
"_id": "691f46c5e04fff283a3cd930",
"userId": "60c728b7f1981e0015b6d5f2",
"content": "Testing Postman VS Code extension to create a new post.",
"likesCount": 1,
"commentsCount": 0,
"createdAt": "2025-11-20T16:50:13.400Z",
"updatedAt": "2025-11-20T16:50:13.400Z",
"__v": 0
}

APIs of Feature - 02 (Upload or Download Study Materials)

```
// 1. UPLOAD Study Material (POST /api/materials/upload)
router.post('/upload', upload.single('materialFile'), async (req, res) => {
    try {
        if (!req.file) {
            return res.status(400).json({ msg: 'No file uploaded' });
        }

        // Placeholder User ID (Replace with ID from Authentication/Token)
        const userId = req.body.userId || '60c728b7f1981e0015b6d5f1';

        const newMaterial = new Material({
            userId: userId,
            title: req.body.title,
            description: req.body.description,
            filePath: req.file.path,
            mimeType: req.file.mimetype,
            fileSize: req.file.size
        });
```



```
// 3. DOWNLOAD Study Material (GET /api/materials/download/:id)
router.get('/download/:id', async (req, res) => {
    try {
        const material = await Material.findById(req.params.id);

        if (!material) {
            return res.status(404).json({ msg: 'Material not found' });
        }

        // Use Express's res.download to initiate file download
        const absolutePath = path.resolve(material.filePath);
        res.download(absolutePath, material.title, (err) => {
            if (err) {
                // If the file is not found on disk, or other download error
                console.error("Download Error:", err);
                res.status(500).send({ msg: "Could not download the file, or file not found on server." });
            }
        });

    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});
```