



**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 Kollo

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 });
  }
});
```

The screenshot shows the Postman interface with a POST request to `http://localhost:1049/api/posts`. The request body is set to raw JSON:

```
[{"content": "Testing Postman VS Code extension to create a new post.", "userId": "60c728b7f1981e0015b6d5f2"}]
```

The response status is 201 Created, and the response body is:

```
{"msg": "Post created successfully", "post": {"userId": "60c728b7f1981e0015b6d5f2", "content": "Testing Postman VS Code extension to create a new post.", "likeCount": 0, "commentCount": 0, "_id": "691f46c5e04fff283a3cd930", "createdAt": "2025-11-20T16:50:13.400Z", "updatedAt": "2025-11-20T16:50:13.400Z", "__v": 0}}
```

```
// 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 });
  }
});
```

The screenshot shows the Postman interface with the following details:

- Method:** PUT
- URL:** http://localhost:1049/api/posts/like/691f46c5e04fff283a3cd930
- Headers:** Authorization, Headers (8), Body, Scripts, Settings
- Code Cookies**
- Query Params:** Key (Value)
- Body:** JSON (Pretty) - Response body: {"msg": "Post liked successfully", "likes": 1, "post": {"_id": "691f46c5e04fff283a3cd930", "userId": "60c728b7f1981e0015b6dsf2", "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}}
- Headers (7):** Status: 200 OK, Time: 64 ms, Size: 545 B, ...

```
// 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 <http://localhost:1049/api/posts?sort=latest>

GET | http://localhost:1049/api/posts?sort=latest | Send | ↴

Params • Authorization Headers (7) Body Scripts Settings

Query Params

| Key | Value |
|--|--------|
| <input checked="" type="checkbox"/> sort | latest |
| Key | Value |

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 69 ms Size: 496 B

Pretty Raw Preview JSON ↴

```
1 [ ] {  
2     "_id": "691f46c5e04fff283a3cd930",  
3     "userId": "60c728b7f1981e0015b6d5f2",  
4     "content": "testing Postman VS Code extension to create a new post.",  
5     "likesCount": 1,  
6     "commentsCount": 0,  
7     "createdAt": "2025-11-20T16:50:13.400Z",  
8     "updatedAt": "2025-11-20T16:50:13.400Z",  
9     "__v": 0  
10 }  
11 
```

HTTP <http://localhost:1049/api/posts?sort=popular>

GET | http://localhost:1049/api/posts?sort=popular | Send | ↴

Params • Authorization Headers (7) Body Scripts Settings

Query Params

| Key | Value |
|--|---------|
| <input checked="" type="checkbox"/> sort | popular |
| Key | Value |

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 11 ms Size: 496 B

Pretty Raw Preview JSON ↴

```
1 [ ] {  
2     "_id": "691f46c5e04fff283a3cd930",  
3     "userId": "60c728b7f1981e0015b6d5f2",  
4     "content": "testing Postman VS Code extension to create a new post.",  
5     "likesCount": 1,  
6     "commentsCount": 0,  
7     "createdAt": "2025-11-20T16:50:13.400Z",  
8     "updatedAt": "2025-11-20T16:50:13.400Z",  
9     "__v": 0  
10 }  
11 
```

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,
      title: req.body.title,
      description: req.body.description,
      filePath: req.file.path,
      mimeType: req.file.mimetype,
      fileSize: req.file.size
    });
  }
});
```

The screenshot shows a Postman interface with the following details:

- Request URL:** `http://localhost:1049/api/materials/upload`
- Method:** POST
- Body:** `form-data`
- Headers:** `(9)`
- Body (form-data):**

| Key | Value |
|--------------|--|
| title | Final Project API Docs |
| description | MERN setup guide |
| materialFile | c:\Users\Abu Mashaheed\OneDrive\Desktop\MERN Jokes.txt |
| Key | Value |
- Test Results:**

| Body | Cookies | Headers (7) | Test Results |
|---|---|-------------|--------------|
| Pretty | Raw | Preview | JSON |
| 1 [{"msg": "Material uploaded successfully", "material": {"userId": "60c728b7f1981e0015b6d5f1", "title": "Final Project API docs", "description": "MERN setup guide", "filePath": "uploads\\materials\\1763658369349-MERN_Jokes.txt", "mimeType": "text/plain", "fileSize": 41, "id": "691f4a81e04fff283a3cd935", "uploadedAt": "2025-11-20T17:06:09.355Z", "v": 0}}] | Status: 201 Created Time: 31 ms Size: 578 B | ... | |

```
// 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 });
  }
});
```

The screenshot shows the Postman application interface. At the top, the URL is set to `http://localhost:1049/api/materials/download/691f4a81e04fff283a3cd935`. Below the URL, the method is set to `GET`. The `Params` tab is selected, showing a single entry: `Key` (Value: `Value`). In the bottom section, the `Body` tab is active, displaying the response from the API. The response status is `Status: 200 OK`, `Time: 50 ms`, and `Size: 424 B`. The response body is shown in `Pretty` format:

```

1 HAHAAHAAHAAHAAHAAHAA
2 You just got MERNrolled

```