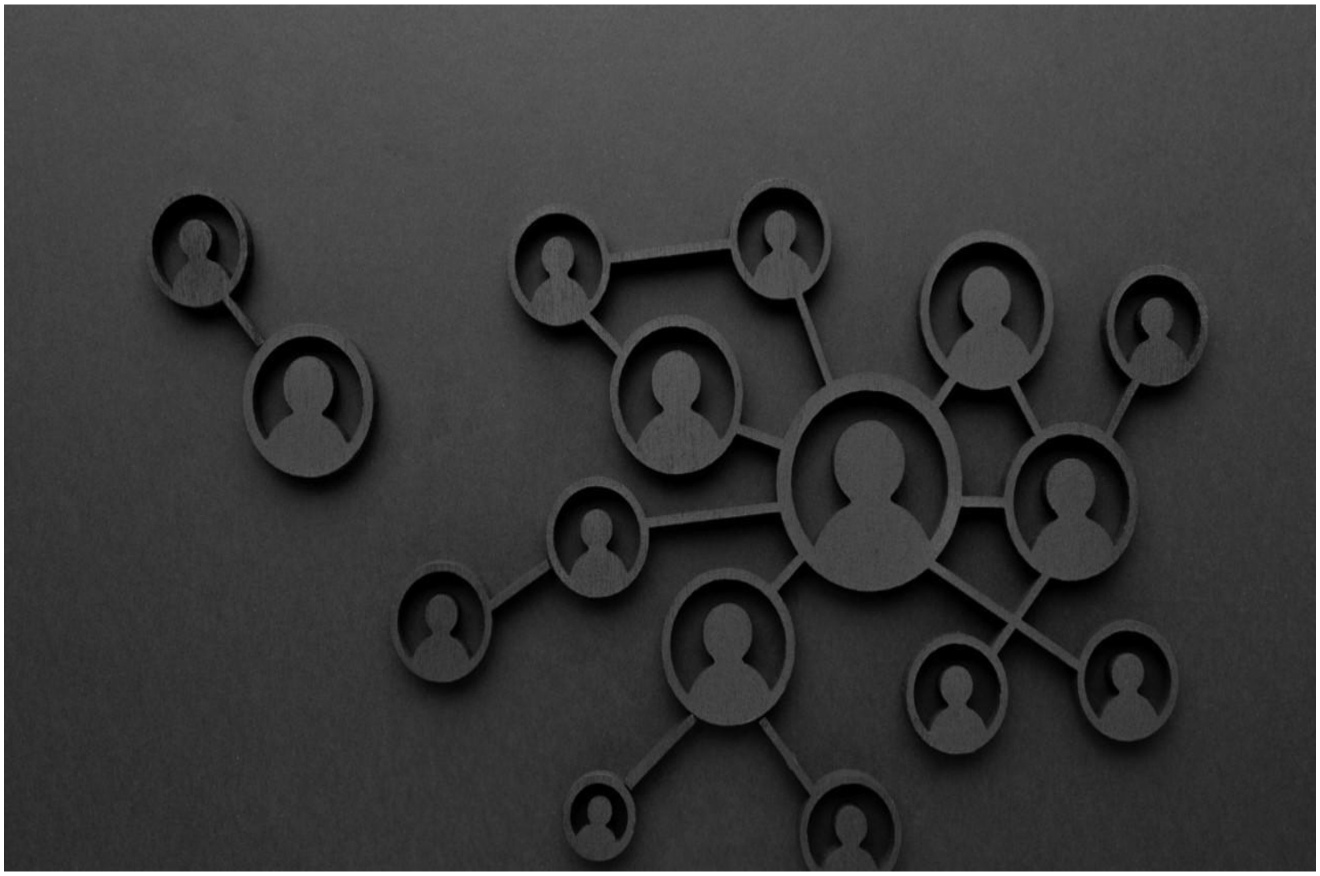
Marlon Pryor

# Club Media Service Layer



## Service Layer Overview and specification

For this project I will use Express and Mongoose to build a REST API for the application.  The service layer will be built using controllers, models, and routes.

### Model

Here is the model that will be used for the database data.

```
const postSchema = mongoose.Schema({
    title: String,
    message: String,
    name: String,
    creator: String,
    tags: [String],
    selectedFile: String,
   likes: { type: [String], default: [] },
   comments: { type: [String], default: [] },
    createdAt: {
        type: Date,
        default: new Date(),
    },
})

var PostMessage = mongoose.model('PostMessage', postSchema);

export default PostMessage;
```

### Routes

The routes listed below will forward the requests to the appropriate controller

**ADDED: route for 'likes' option**

```
import express from 'express';

import { getPosts, getPostsBySearch, getPost, createPost, updatePost, deletePost } from '../controllers/posts.js';
```

```
const router = express.Router();
import auth from "../middleware/auth.js";
router.get('/search', getPostsBySearch);
router.get('/', getPosts);
router.get('/:id', getPost);

router.post('/', auth,  createPost);
router.patch('/:id', auth, updatePost);
router.delete('/:id', auth, deletePost);
router.patch('/:id/likePost', auth, likePost);
router.post('/:id/commentPost', commentPost);


export default router;
```

## GET Post(s) Endpoint function

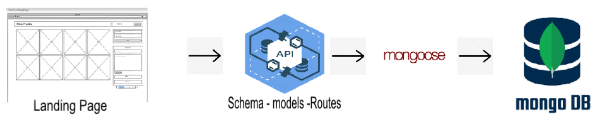
```
export const getPosts = async (req, res) => {
    try {
        const postMessages = await PostMessage.find();

        res.status(200).json(postMessages);
    } catch (error) {
        res.status(404).json({ message: error.message });
    }
}

export const getPost = async (req, res) => {
    const { id } = req.params;

    try {
        const post = await PostMessage.findById(id);

        res.status(200).json(post);
    } catch (error) {
        res.status(404).json({ message: error.message });
    }
}
```



Searching for posts are done from landing page

## CREATE Post Endpoint function

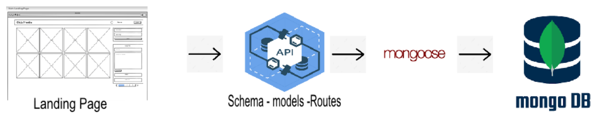
```
export const createPost = async (req, res) => {
    const { title, message, selectedFile, creator, tags } = req.body;

    const newPostMessage = new PostMessage({ title, message, selectedFile, creator, tags })

    try {
        await newPostMessage.save();

        res.status(201).json(newPostMessage );
    } catch (error) {
        res.status(409).json({ message: error.message });

    }
}
```



Creating a new post is done from landing page.

## UPDATE Post Endpoint function

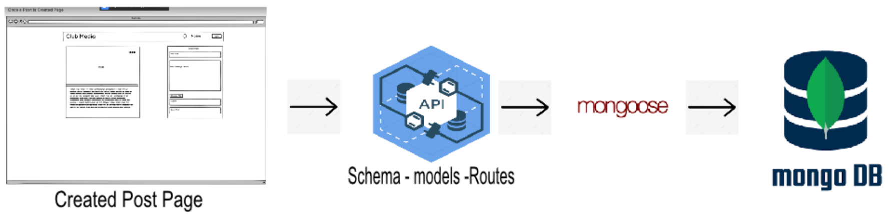
```
export const updatePost = async (req, res) => {
    const { id } = req.params;
    const { title, message, creator, selectedFile, tags } = req.body;

    if (!mongoose.Types.ObjectId.isValid(id)) return res.status(404).send(`No post with id: ${id}`);

    const updatedPost = { creator, title, message, tags, selectedFile, _id: id };

    await PostMessage.findByIdAndUpdate(id, updatedPost, { new: true });

    res.json(updatedPost);
}
```



Updating a post is done from Created Post page

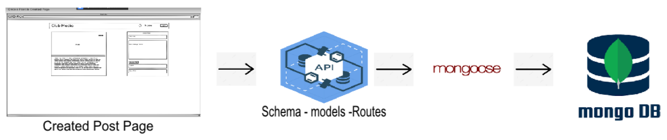
## DELETE  Post Endpoint function

```
export const deletePost = async (req, res) => {
    const { id } = req.params;
    if (!mongoose.Types.ObjectId.isValid(id)) return res.status(404).send(`No post with id: ${id}`);

    await PostMessage.findByIdAndRemove(id);

    res.json({ message: "Post deleted successfully." });
}


export default router;
```



Deleting posts are done from the created post page by clicking on three dots in the post.

## Comment Post Endpoint function

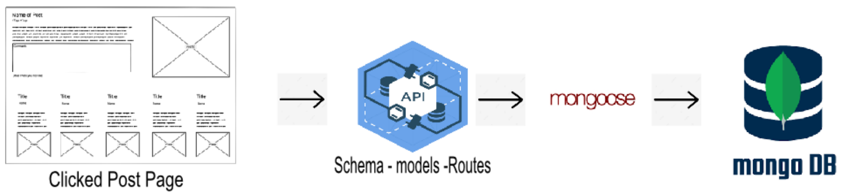
```
export const commentPost = async (req, res) => {
    const { id } = req.params;
    const { value } = req.body;

    const post = await PostMessage.findById(id);

    post.comments.push(value);

    const updatedPost = await PostMessage.findByIdAndUpdate(id, post, { new: true });

    res.json(updatedPost);
};
```



Comments are created from the clicked post page

All Endpoints will use the standard HTTP error codes as stated in the endpoints except in the UPDATE and DELETE functions.