# Experiment 4

**Aim -** REST API Design with  MongoDB + Mongoose Integration.

**Code -**

**db.js -**

social-media-mern > backend > config > JS db.js > ...

```
1   import mongoose from "mongoose";
2
3   const connectDB = async () => {
4     try {
5       await mongoose.connect(process.env.MONGO_URI);
6       console.log("MongoDB Connected");
7     } catch (error) {
8       console.error("MongoDB connection error:", error.message);
9       process.exit(1);
10    }
11  };
12
13  export default connectDB;
14
```

**models/posts.js -**

social-media-mern > backend > models > JS Post.js > ...

```
1   import mongoose from "mongoose";
2
3   const postSchema = new mongoose.Schema({
4     username: { type: String, required: true },
5     caption: { type: String, required: true },
6     imageUrl: { type: String },
7     likes: { type: Number, default: 0 },
8     createdAt: { type: Date, default: Date.now },
9   });
10
11  export default mongoose.model("Post", postSchema);
12
```

**postRoutes.js** -

social-media-mern > backend > routes > JS postRoutes.js > ...

```
import express from "express";
import Post from "../models/Post.js";
const router = express.Router();

// GET all posts
router.get("/", async (req, res) => {
  const posts = await Post.find().sort({ createdAt: -1 });
  res.json(posts);
});

// POST new post
router.post("/", async (req, res) => {
  const { username, caption, imageUrl } = req.body;
  const newPost = new Post({ username, caption, imageUrl });
  await newPost.save();
  res.json({ message: "✅ Post created", post: newPost });
});

// PUT (like post)
router.put("/:id/like", async (req, res) => {
  const post = await Post.findById(req.params.id);
  if (!post) return res.status(404).json({ message: "Post not found" });
  post.likes += 1;
  await post.save();
  res.json({ message: "❤️ Post liked", post });
});

// PUT (edit caption)
router.put("/:id", async (req, res) => {
  const { caption } = req.body;
  const updated = await Post.findByIdAndUpdate(req.params.id, { caption }, { new: true });
  res.json({ message: "🖍️ Post updated", post: updated });
});

// DELETE post
router.delete("/:id", async (req, res) => {
  await Post.findByIdAndDelete(req.params.id);
  res.json({ message: "🗑️ Post deleted" });
});
export default router;
```

**server.js** -

social-media-mern > backend > JS server.js > ...

```
1   import express from "express";
2   import dotenv from "dotenv";
3   import cors from "cors";
4   import connectDB from "./config/db.js";
5   import postRoutes from "./routes/postRoutes.js";
6
7   dotenv.config();
8   connectDB();
9
10  const app = express();
11  app.use(cors());
12  app.use(express.json());
13
14  app.get("/", (req, res) => res.send("🌐 SocialConnect API Running"));
15  app.use("/api/posts", postRoutes);
16
17  app.listen(process.env.PORT, () =>
18    console.log(`🚀 Server running on port ${process.env.PORT}`)
19  );
20
```

## Home.jsx -

social-media-mern > frontend > src > pages > Home.jsx > ...

```jsx
import React, { useState, useEffect } from "react";
import axios from "axios";
import PostCard from "../components/PostCard";

export default function Home() {
  const [posts, setPosts] = useState([]);
  const [form, setForm] = useState({ username: "", caption: "", imageUrl: "" });

  const fetchPosts = async () => {
    const res = await axios.get("http://localhost:5000/api/posts");
    setPosts(res.data);
  };

  const addPost = async (e) => {
    e.preventDefault();
    await axios.post("http://localhost:5000/api/posts", form);
    setForm({ username: "", caption: "", imageUrl: "" });
    fetchPosts();
  };

  useEffect(() => {
    fetchPosts();
  }, []);

  return (
    <div className="max-w-3xl mx-auto p-4">
      <form onSubmit={addPost} className="bg-white dark:bg-gray-800 shadow rounded-lg p-4 mb-6">
        <h2 className="text-xl font-semibold mb-3 text-blue-700 dark:text-blue-400">Create a Post</h2>
        <input
          type="text"···
        />
        <textarea
          placeholder="Write something..."···
        />
        <input
          type="text"···
        />
        <button className="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600">···
        </button>
      </form>

      <div className="grid gap-4">
        {posts.map((post) => (
          <PostCard key={post._id} post={post} refreshPosts={fetchPosts} />
        ))}
      </div>
    </div>
  );
}
```

**components/PostCard.jsx -**

social-media-mern > frontend > src > components > PostCard.jsx > ...

```jsx
export default function PostCard({ post, refreshPosts }) {
  const [editing, setEditing] = useState(false);
  const [newCaption, setNewCaption] = useState(post.caption);

  const likePost = async () => {
    await axios.put(`http://localhost:5000/api/posts/${post._id}/like`);
    refreshPosts();
  };

  const deletePost = async () => {
    await axios.delete(`http://localhost:5000/api/posts/${post._id}`);
    refreshPosts();
  };

  const saveEdit = async () => {
    await axios.put(`http://localhost:5000/api/posts/${post._id}`, { caption: newCaption });
    setEditing(false);
    refreshPosts();
  };

  return (
    <div className="bg-white dark:bg-gray-800 shadow-md rounded-lg p-4 space-y-2">
      <div className="font-semibold text-blue-700 dark:text-blue-400">@{post.username}</div>
      {post.imageUrl && <img src={post.imageUrl} className="rounded-md mb-2" />}
      {editing ? (
        <textarea
          value={newCaption}
          onChange={(e) => setNewCaption(e.target.value)}
          className="border p-2 w-full rounded dark:bg-gray-700"
        />
      ) : (
        <p>{post.caption}</p>
      )}
      <div className="flex gap-3 mt-2 text-sm text-gray-600 dark:text-gray-300">
        <button onClick={likePost}>❤️ {post.likes}</button>
        <button onClick={() => setEditing(!editing)}>{editing ? "💾 Save" : "🖍️ Edit"}</button>
        {editing && <button onClick={saveEdit}>✅ Update</button>}
        <button onClick={deletePost}>🗑️ Delete</button>
      </div>
      <div className="text-xs text-gray-400">{new Date(post.createdAt).toLocaleString()}</div>
```

**Output -**

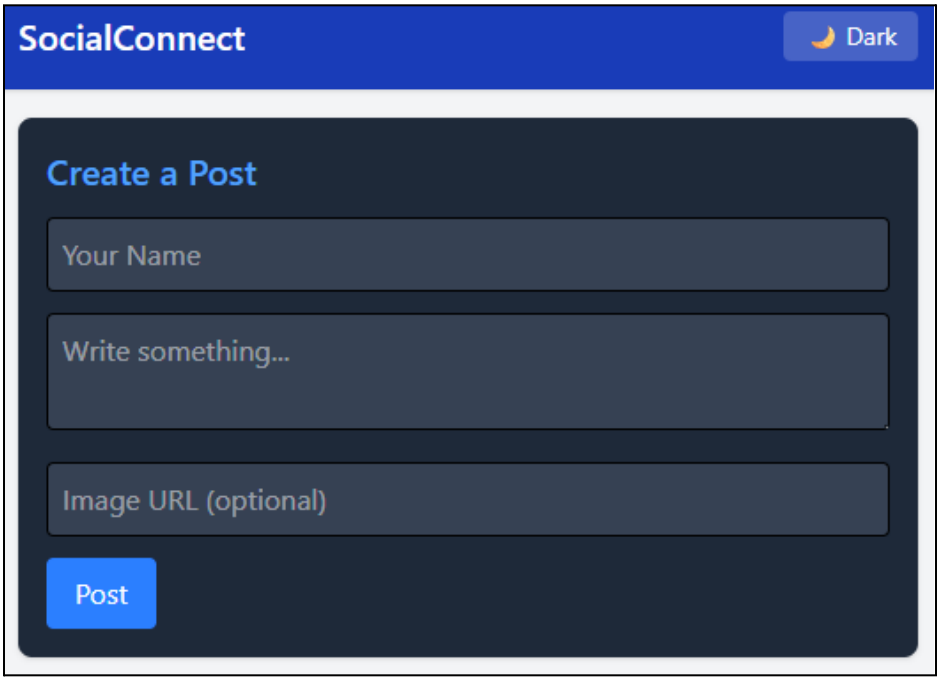


Fig 1.1 - Homepage for adding a post

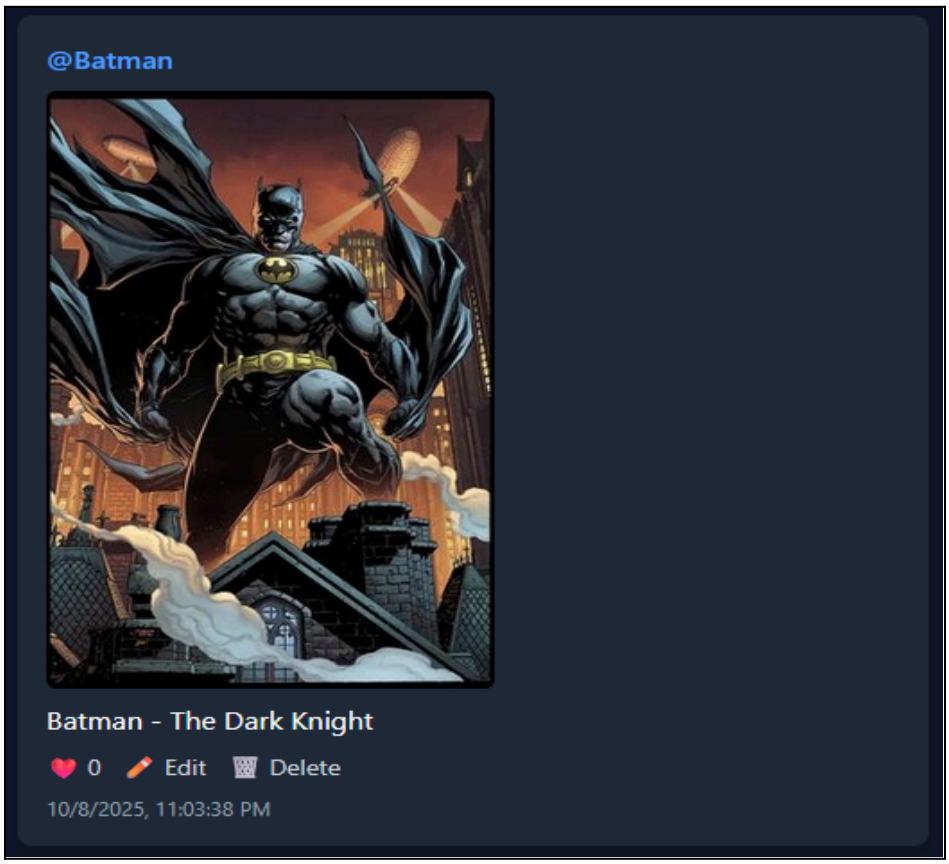


Fig 2.1 - Post


Fig 2.2 - Edit post using REST API methods

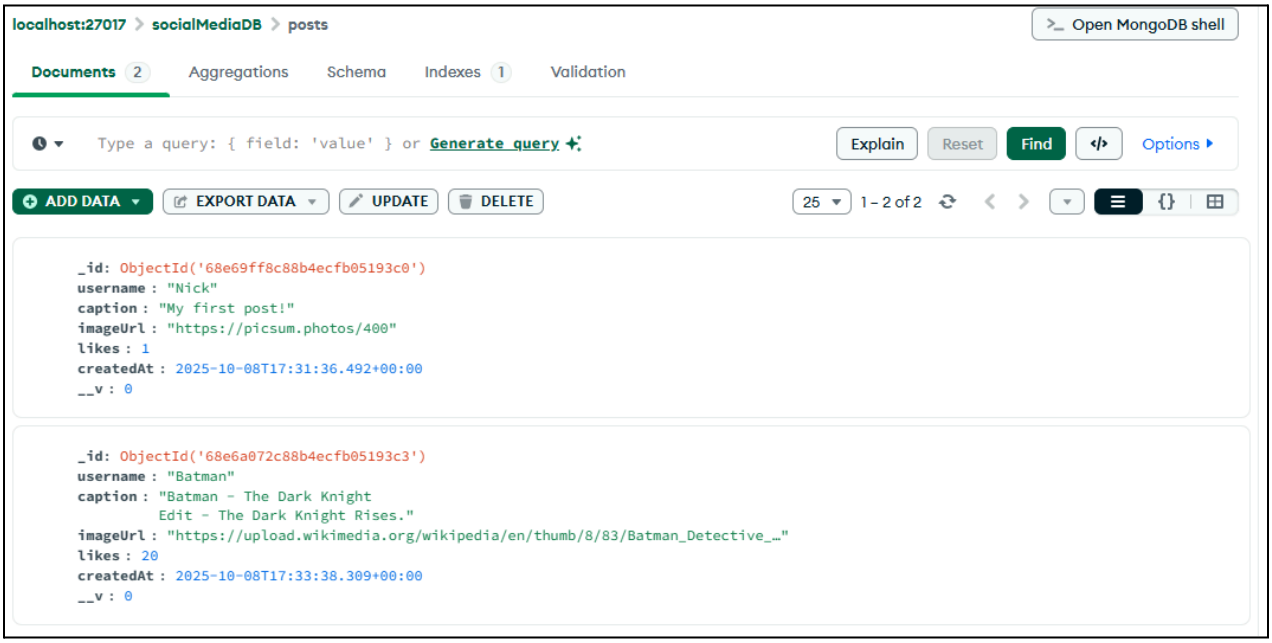


Fig 3 - MongoDB + Mongoose Integration

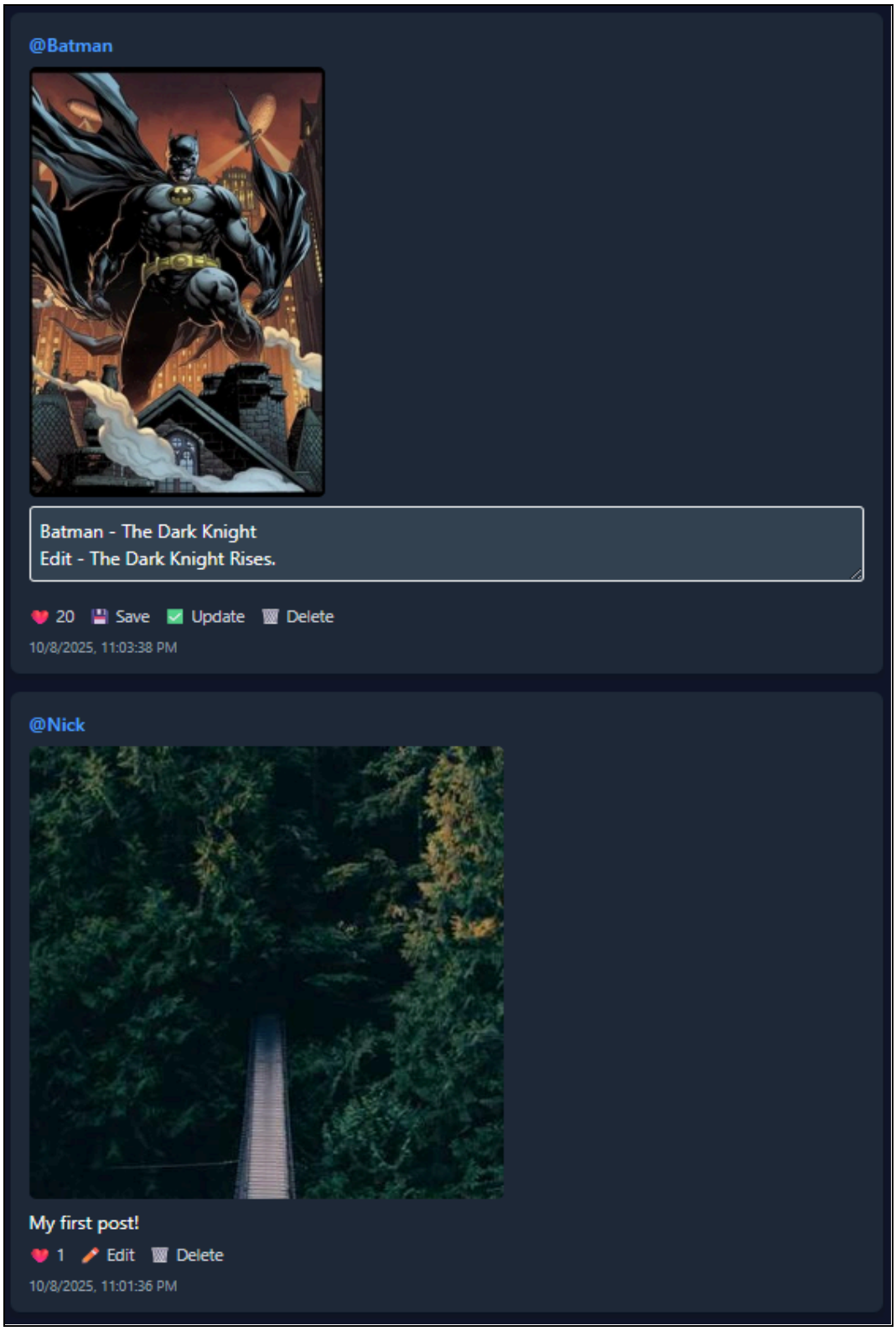


Fig 4 - Homepage for all posts