

## Experiment 5

**Aim** - Create secure, production-ready **RESTful APIs**

**Code** -

config/[db.js](#) -

```
linkedin-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 (err) {
8      console.error("Database Connection Failed:", err.message);
9      process.exit(1);
10   }
11 };
12
13 export default connectDB;
```

[authMiddleware.js](#) -

```
linkedin-mern > backend > middleware > JS authMiddleware.js > ...
1  import jwt from "jsonwebtoken";
2  import User from "../models/User.js";
3
4  const protect = async (req, res, next) => {
5    const authHeader = req.headers.authorization;
6
7    if (authHeader && authHeader.startsWith("Bearer ")) {
8      try {
9        const token = authHeader.split(" ")[1];
10       const decoded = jwt.verify(token, process.env.JWT_SECRET);
11       req.user = await User.findById(decoded.id).select("-password");
12       next();
13     } catch (error) {
14       res.status(401).json({ message: "Unauthorized: Invalid Token" });
15     }
16   } else {
17     res.status(401).json({ message: "No Token Provided" });
18   }
19 };
20
21 export default protect;
```

models/[post.js](#) -

```
linkedin-mern > backend > models > JS Post.js > ...
1  import mongoose from "mongoose";
2
3  const postSchema = new mongoose.Schema({
4    user: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
5    content: { type: String, required: true },
6    createdAt: { type: Date, default: Date.now },
7  });
8
9  export default mongoose.model("Post", postSchema);
10
```

models/[user.js](#) -

```
linkedin-mern > backend > models > JS User.js > ...
1  import mongoose from "mongoose";
2  import bcrypt from "bcryptjs";
3
4  const userSchema = new mongoose.Schema({
5    name: { type: String, required: true },
6    email: { type: String, required: true, unique: true },
7    password: { type: String, required: true },
8    headline: { type: String, default: "Aspiring Developer" },
9  });
10
11  userSchema.pre("save", async function (next) {
12    if (!this.isModified("password")) return next();
13    this.password = await bcrypt.hash(this.password, 10);
14    next();
15  });
16
17  userSchema.methods.matchPassword = async function (enteredPassword) {
18    return await bcrypt.compare(enteredPassword, this.password);
19  };
20
21  export default mongoose.model("User", userSchema);
22
```

## [authRoutes.js](#) -

```
linkedin-mern > backend > routes > JS authRoutes.js > ...
1  import express from "express";
2  import User from "../models/User.js";
3  import jwt from "jsonwebtoken";
4
5  const router = express.Router();
6
7  // Signup
8  router.post("/signup", async (req, res) => {
9    const { name, email, password } = req.body;
10
11    const exists = await User.findOne({ email });
12    if (exists) return res.status(400).json({ message: "User already exists" });
13
14    const user = await User.create({ name, email, password });
15    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: "1h" });
16    res.json({ message: "User registered", token, user });
17  });
18
19  // Login
20  router.post("/login", async (req, res) => {
21    const { email, password } = req.body;
22    const user = await User.findOne({ email });
23    if (!user || !(await user.matchPassword(password)))
24      return res.status(400).json({ message: "Invalid credentials" });
25
26    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: "1h" });
27    res.json({ message: "Login successful", token, user });
28  });
29
30  export default router;
```

## [postRoutes.js](#) -

```
linkedin-mern > backend > routes > JS postRoutes.js > ...
1  import express from "express";
2  import Post from "../models/Post.js";
3  import protect from "../middleware/authMiddleware.js";
4
5  const router = express.Router();
6
7  // Create a post
8  router.post("/", protect, async (req, res) => {
9    const post = await Post.create({ user: req.user._id, content: req.body.content });
10    res.json({ message: "Post created", post });
11  });
12
13  // Get all posts
14  router.get("/", protect, async (req, res) => {
15    const posts = await Post.find().populate("user", "name headline");
16    res.json(posts);
17  });
18
19  // Delete post
20  router.delete("/:id", protect, async (req, res) => {
21    const post = await Post.findById(req.params.id);
22    if (!post) return res.status(404).json({ message: "Not found" });
23    if (post.user.toString() !== req.user._id.toString())
24      return res.status(403).json({ message: "Forbidden" });
25    await post.deleteOne();
26    res.json({ message: "Post deleted" });
27  });
28
29  export default router;
```

[server.js](#) -

```
linkedin-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 authRoutes from "../routes/authRoutes.js";
 6  import postRoutes from "../routes/postRoutes.js";
 7
 8  dotenv.config();
 9  connectDB();
10
11  const app = express();
12  app.use(cors());
13  app.use(express.json());
14
15  app.get("/", (req, res) => res.send("🔒 Secure REST API Ready"));
16  app.use("/api/auth", authRoutes);
17  app.use("/api/posts", postRoutes);
18
19  const PORT = process.env.PORT || 5000;
20  app.listen(PORT, () => console.log(`🚀 Server running on port ${PORT}`));
21
```

[Feed.jsx](#) -

```

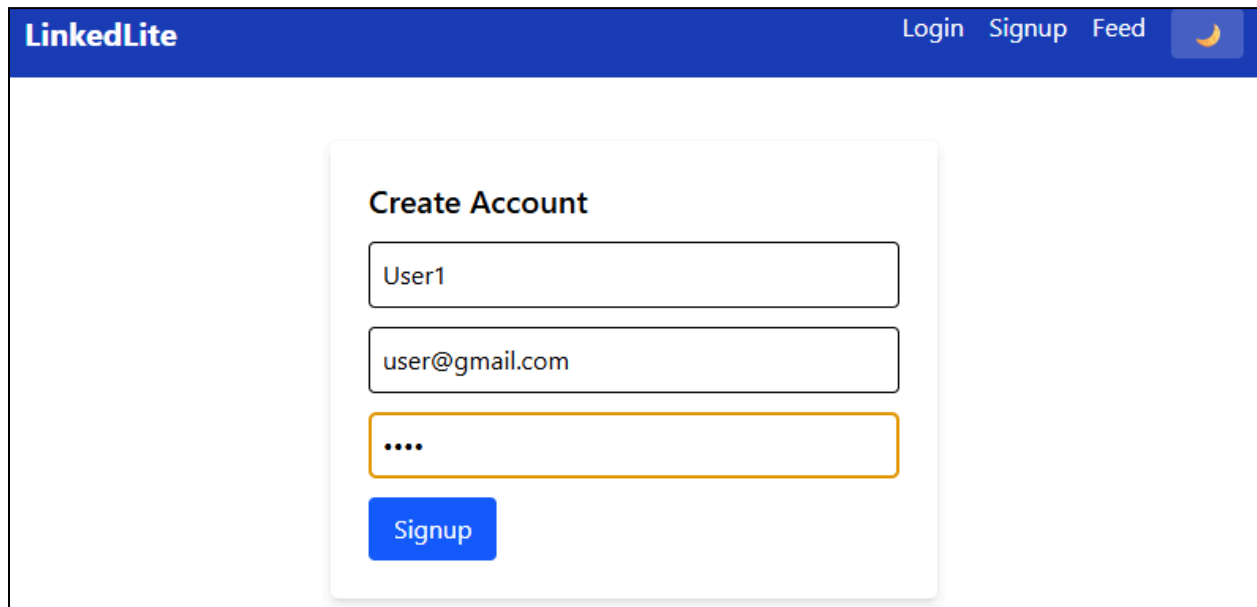
1 import React, { useState, useEffect } from "react";
2 import axios from "axios";
3
4 export default function Feed() {
5   const [posts, setPosts] = useState([]);
6   const [content, setContent] = useState("");
7
8   const token = localStorage.getItem("token");
9   const headers = { Authorization: `Bearer ${token}` };
10
11   const fetchPosts = async () => {
12     const res = await axios.get("http://localhost:5000/api/posts", { headers });
13     setPosts(res.data);
14   };
15
16   const createPost = async (e) => {
17     e.preventDefault();
18     await axios.post("http://localhost:5000/api/posts", { content }, { headers });
19     setContent("");
20     fetchPosts();
21   };
22
23   const deletePost = async (id) => {
24     await axios.delete(`http://localhost:5000/api/posts/${id}`, { headers });
25     fetchPosts();
26   };
27
28   useEffect(() => {
29     fetchPosts();
30   }, []);
31
32   return (
33     <div className="max-w-3xl mx-auto mt-8 p-6 bg-gray-50 dark:bg-gray-900 rounded-lg shadow-xl">
34       <div>
35         <form onSubmit={createPost} className="mb-6 bg-white dark:bg-white-800 p-6 rounded-lg shadow-md">...
36       </div>
37       <div>
38         <div>
39           <div>
40             <div>
41               <div>
42                 <div>
43                 </div>
44               </div>
45             </div>
46           </div>
47         </div>
48       </div>
49     </div>
50   );
51 }

```


## Signup.jsx -

```
linkedin-mern > frontend > src > pages > Signup.jsx > Signup
1  import react from "react";
2  import React, { useState } from "react";
3  import axios from "axios";
4  import { useNavigate } from "react-router-dom";
5
6  export default function Signup() {
7    const [form, setForm] = useState({ name: "", email: "", password: "" });
8    const navigate = useNavigate();
9
10   const submitHandler = async (e) => {
11     e.preventDefault();
12     await axios.post("http://localhost:5000/api/auth/signup", form);
13     navigate("/");
14   };
15
16   return (
17     <div className="flex justify-center mt-10">
18       <form
19         onSubmit={submitHandler}
20         className="bg-white dark:bg-white-800 p-6 rounded shadow-md w-96"
21       >
22         <h2 className="text-xl font-semibold mb-3">Create Account</h2>
23         <input
24           type="text"
25           placeholder="Name"
26           className="border p-2 w-full mb-3 rounded"
27           onChange={(e) => setForm({ ...form, name: e.target.value })}
28         />
29         <input
30           type="email"
31           placeholder="Email"
32           className="border p-2 w-full mb-3 rounded"
33           onChange={(e) => setForm({ ...form, email: e.target.value })}
34         />
35         <input
36           type="password"
37           placeholder="Password"
38           className="border p-2 w-full mb-3 rounded"
39           onChange={(e) => setForm({ ...form, password: e.target.value })}
40         />
41         <button className="bg-blue-600 text-white px-4 py-2 rounded">Signup</button>
42       </form>
43     </div>
44   );
}
```

## Output -

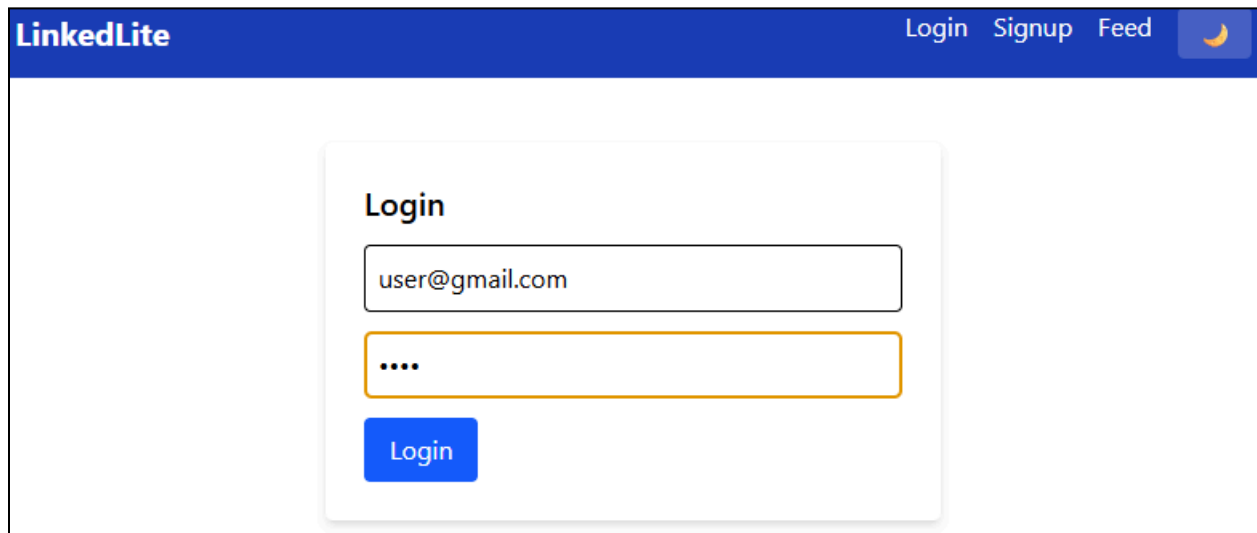


The screenshot shows the 'Create Account' form on the LinkedLite website. The header is a dark blue bar with the 'LinkedLite' logo on the left and 'Login', 'Signup', 'Feed', and a moon icon on the right. The form itself is a white box with a light gray shadow. It has a title 'Create Account' in bold. Below the title are three input fields: the first contains 'User1', the second contains 'user@gmail.com', and the third is a password field with four dots and an orange border. A blue 'Signup' button is at the bottom of the form.


LinkedLite Login Signup Feed 

### Create Account

Fig 1.1 - Secure Signup



The screenshot shows the 'Login' form on the LinkedLite website. The header is identical to the previous one. The form is a white box with a light gray shadow. It has a title 'Login' in bold. Below the title are two input fields: the first contains 'user@gmail.com' and the second is a password field with four dots and an orange border. A blue 'Login' button is at the bottom of the form.

LinkedLite Login Signup Feed 

### Login

Fig 1.2 Secure Login

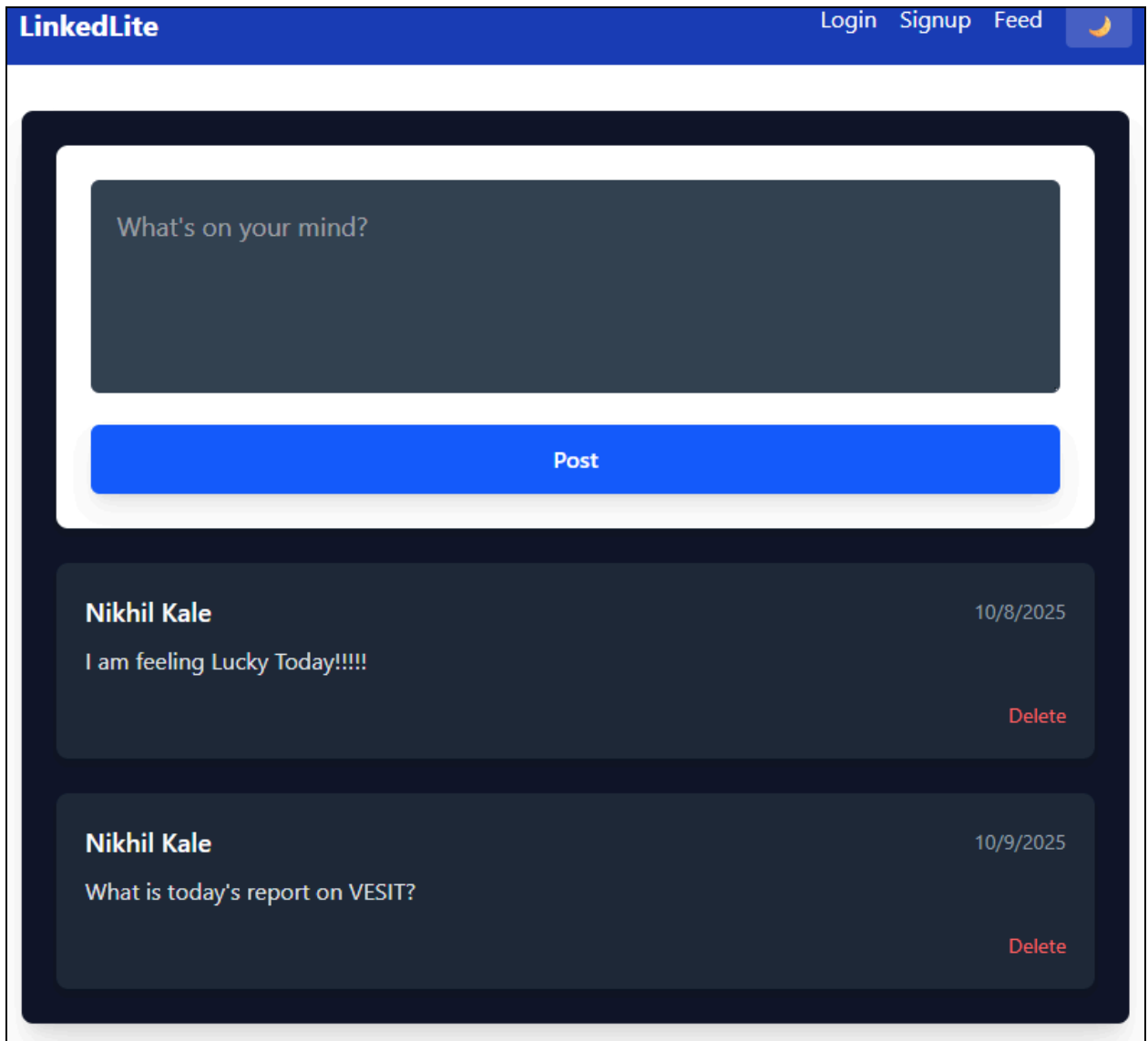


Fig 2.1 - Feed (Add Posts)



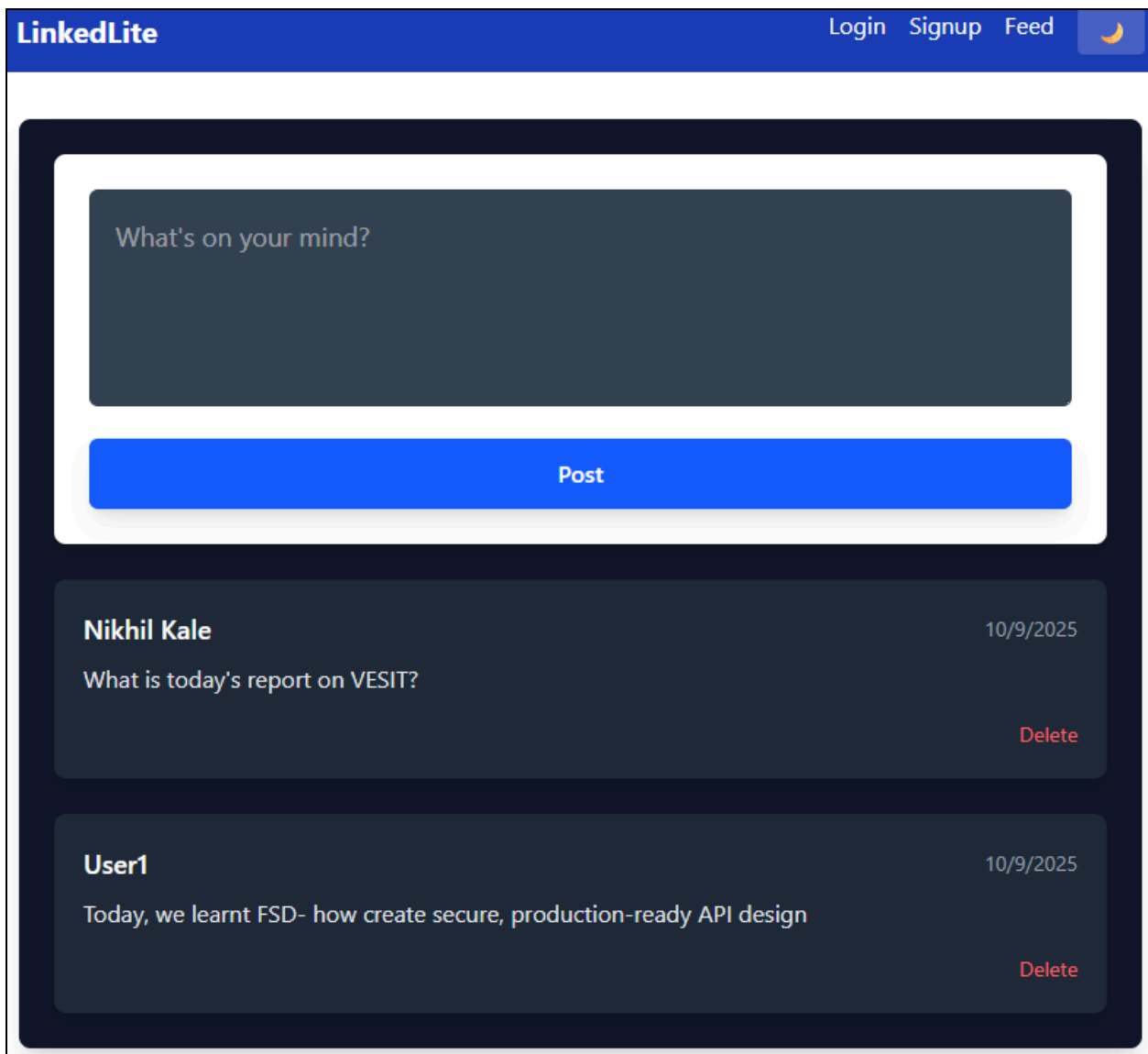


Fig 2.2 - One user cannot delete other user's post

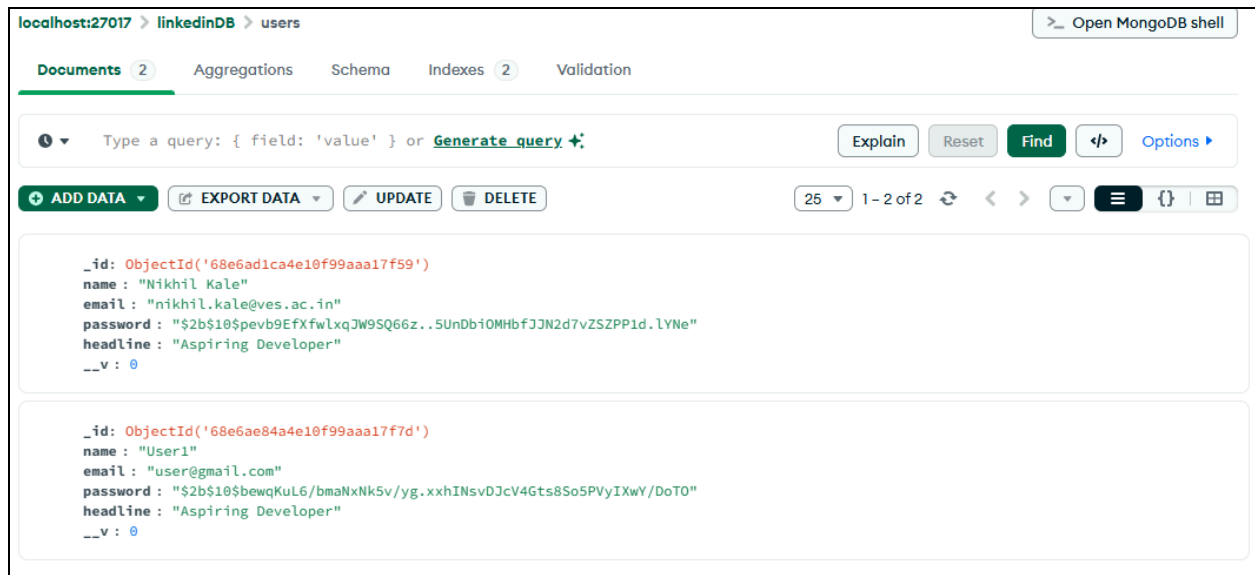


Fig 3.1 - MongoDB Schema for user

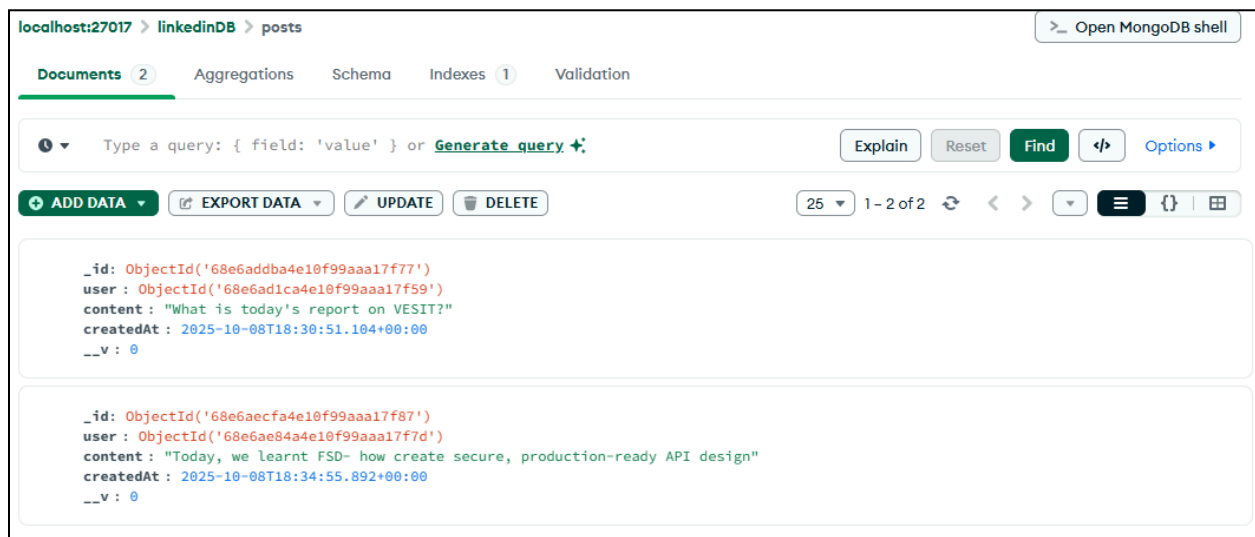


Fig 3.2 - Mongo Schema for posts