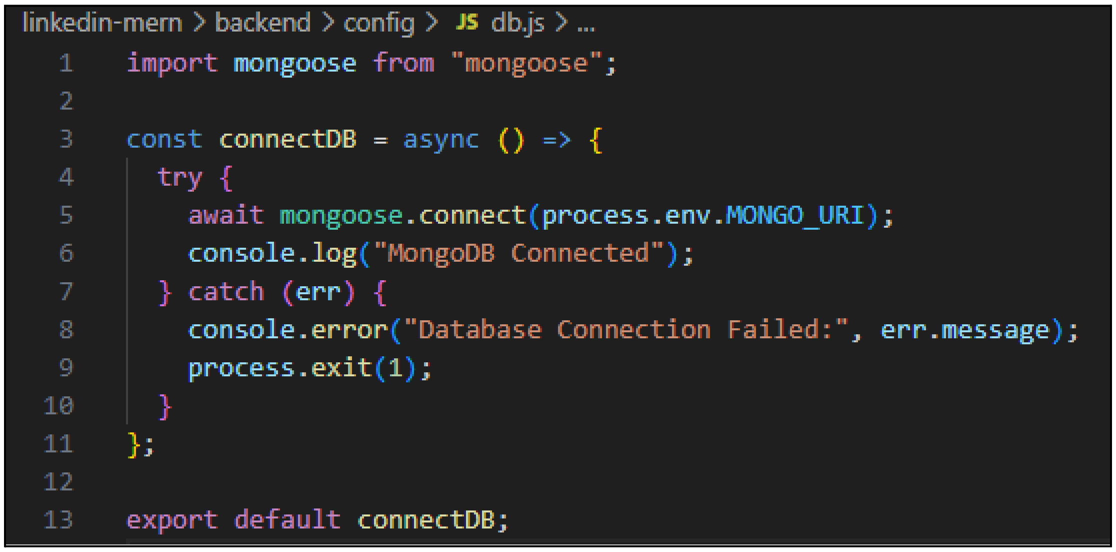
Experiment 7

Aim - Validating RESTful APIs using Postman.

Code -

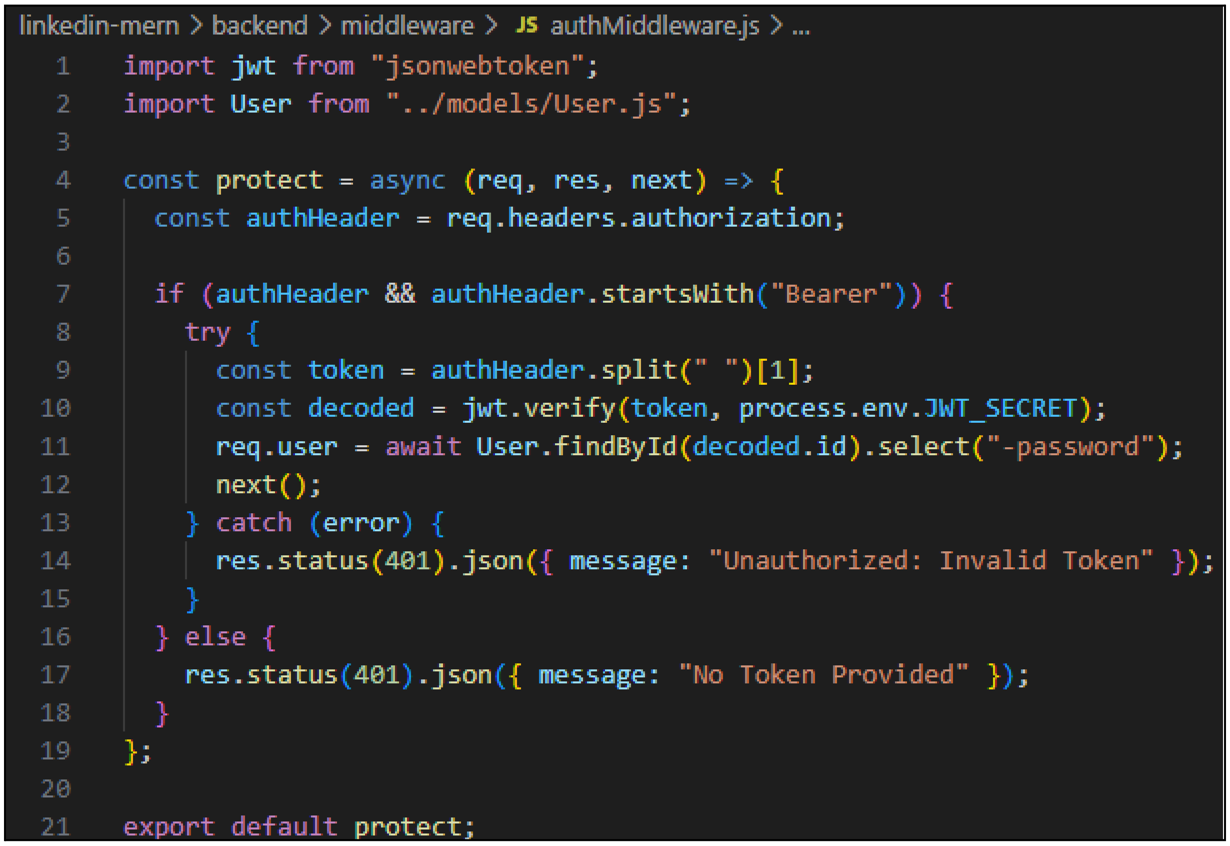
config/db.js -



linkedin-mern > backend > config > JS db.js > ...

```
import mongoose from "mongoose";

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URI);
    console.log("MongoDB Connected");
  } catch (err) {
    console.error("Database Connection Failed:", err.message);
    process.exit(1);
  }
};

export default connectDB;
```

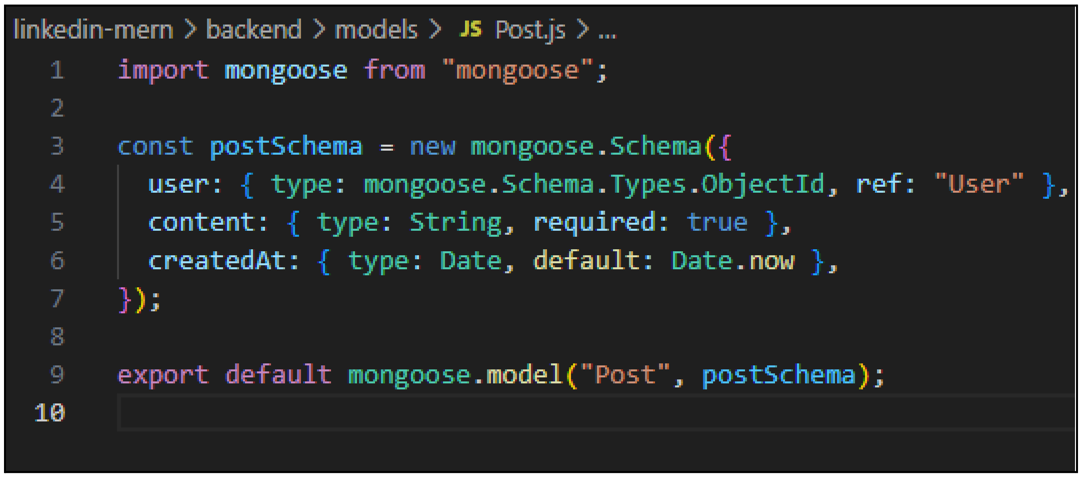
authMiddleware.js -



linkedin-mern > backend > middleware > JS authMiddleware.js > ...

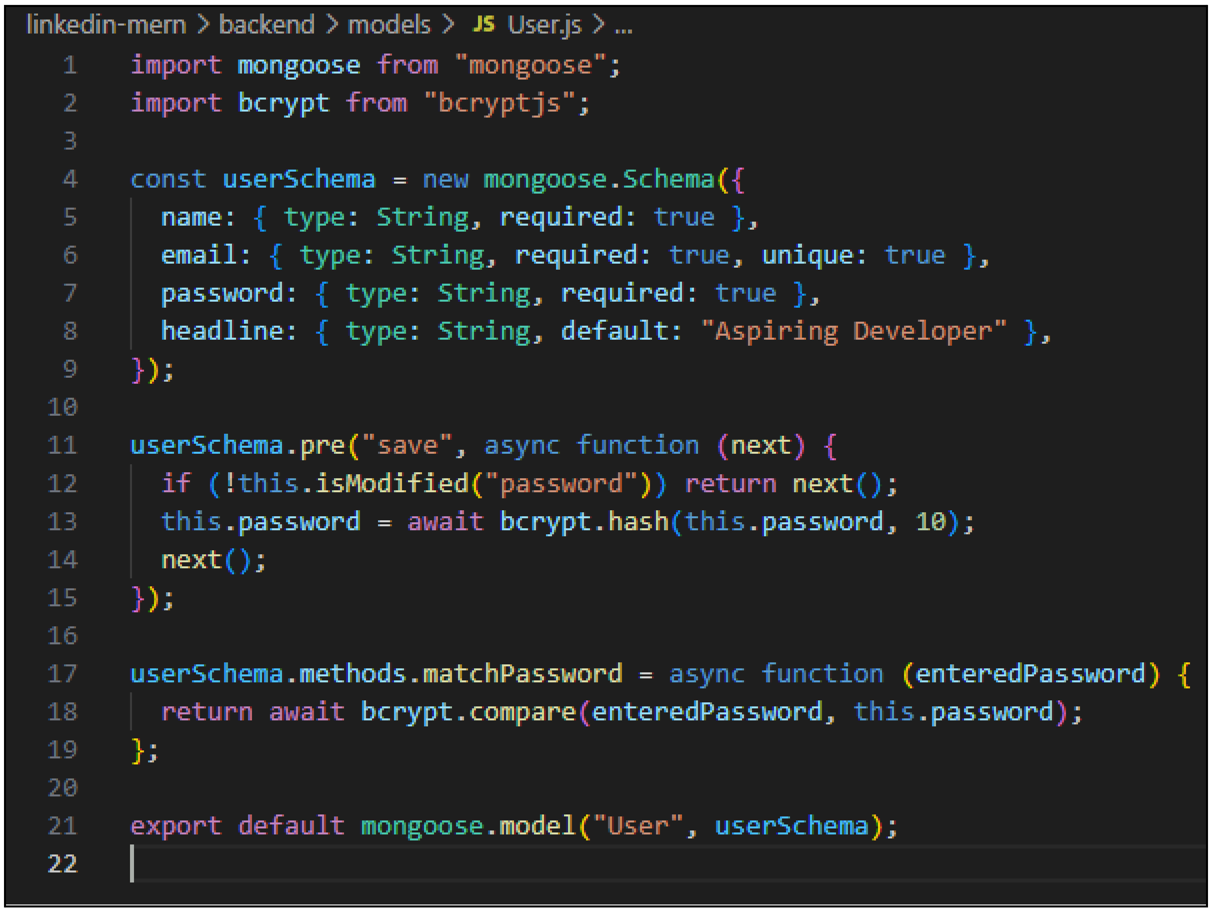
```
import jwt from "jsonwebtoken";
import User from "../models/User.js";

const protect = async (req, res, next) => {
  const authHeader = req.headers.authorization;

  if (authHeader && authHeader.startsWith("Bearer")) {
    try {
      const token = authHeader.split(" ")[1];
      const decoded = jwt.verify(token, process.env.JWT_SECRET);
      req.user = await User.findById(decoded.id).select("-password");
      next();
    } catch (error) {
      res.status(401).json({ message: "Unauthorized: Invalid Token" });
    }
  } else {
    res.status(401).json({ message: "No Token Provided" });
  }
};

export default protect;
```

models/post.js -


linkedin-mern > backend > models > JS Post.js > ...

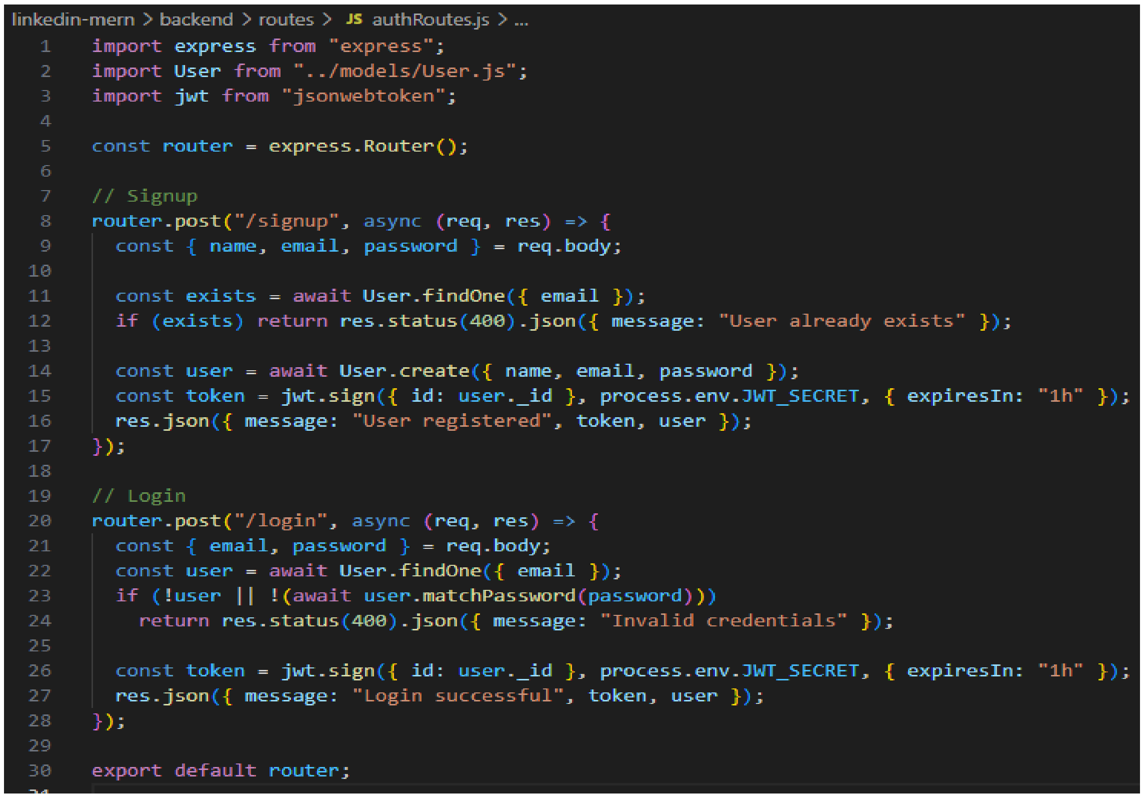
```
import mongoose from "mongoose";

const postSchema = new mongoose.Schema({
  user: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
  content: { type: String, required: true },
  createdAt: { type: Date, default: Date.now },
});

export default mongoose.model("Post", postSchema);
```

models/user.js -


linkedin-mern > backend > models > JS User.js > ...

```
import mongoose from "mongoose";
import bcrypt from "bcryptjs";

const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  headline: { type: String, default: "Aspiring Developer" },
});

userSchema.pre("save", async function (next) {
  if (!this.isModified("password")) return next();
  this.password = await bcrypt.hash(this.password, 10);
  next();
});

userSchema.methods.matchPassword = async function (enteredPassword) {
  return await bcrypt.compare(enteredPassword, this.password);
};

export default mongoose.model("User", userSchema);
```

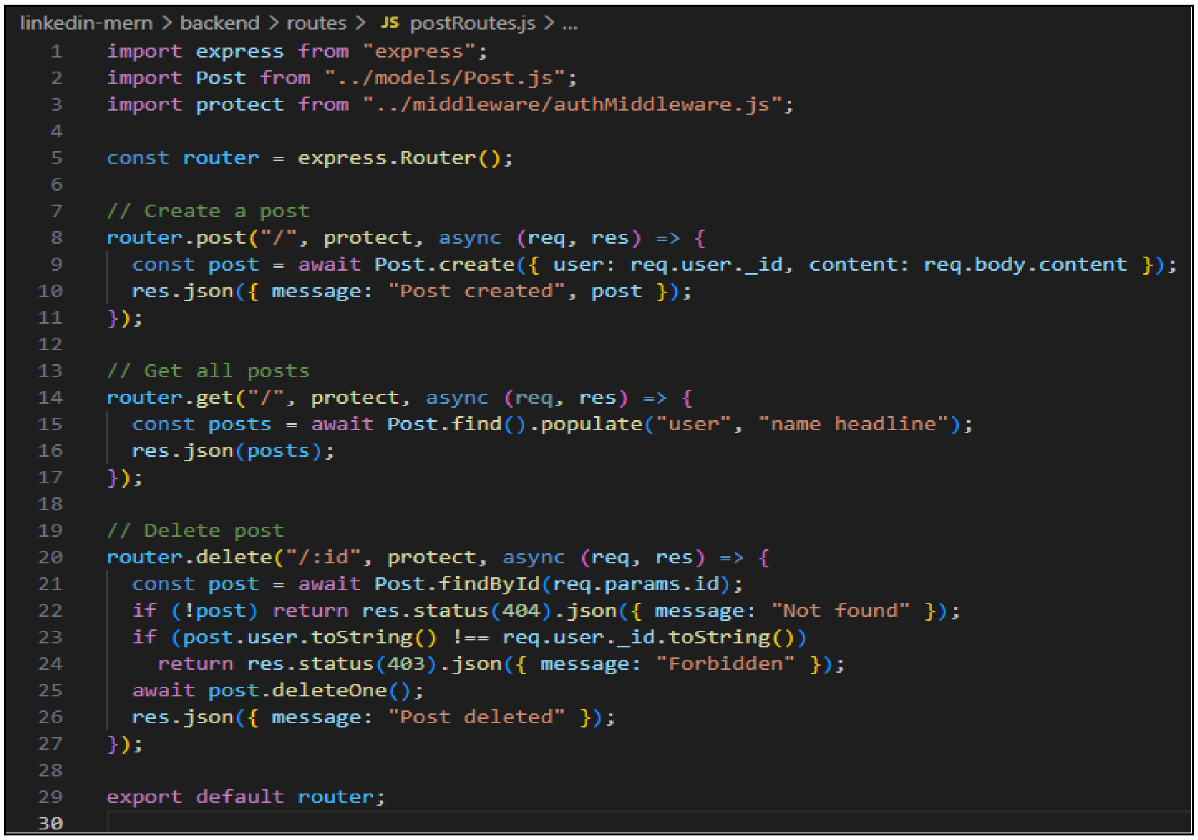
authRoutes.js -



linkedin-mern > backend > routes > JS authRoutes.js > ...

```js
import express from "express";
import User from "../models/User.js";
import jwt from "jsonwebtoken";

const router = express.Router();

// Signup
router.post("/signup", async (req, res) => {
  const { name, email, password } = req.body;

  const exists = await User.findOne({ email });
  if (exists) return res.status(400).json({ message: "User already exists" });

  const user = await User.create({ name, email, password });
  const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: "1h" });
  res.json({ message: "User registered", token, user });
});

// Login
router.post("/login", async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user || !(await user.matchPassword(password)))
    return res.status(400).json({ message: "Invalid credentials" });

  const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: "1h" });
  res.json({ message: "Login successful", token, user });
});

export default router;
```

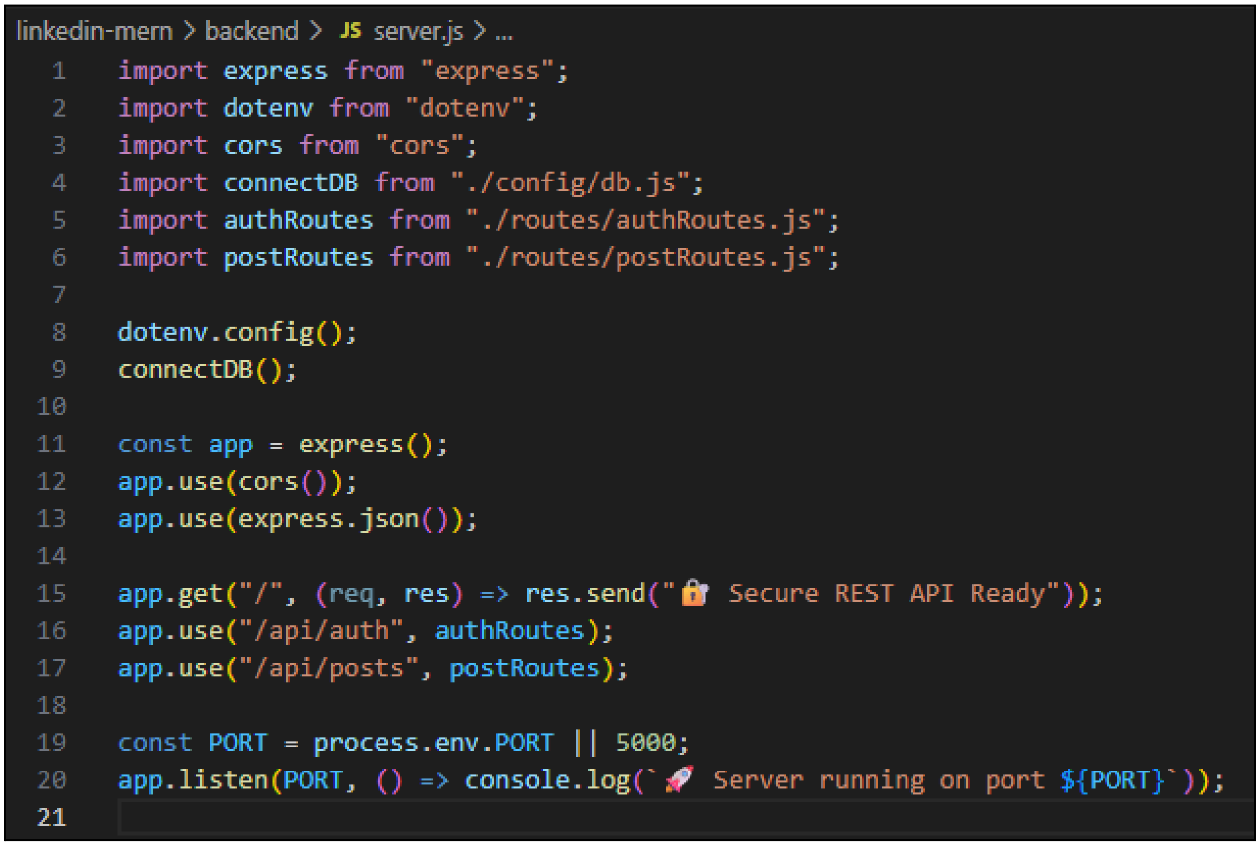
postRoutes.js -



linkedin-mern > backend > routes > JS postRoutes.js > ...

```js
import express from "express";
import Post from "../models/Post.js";
import protect from "../middleware/authMiddleware.js";

const router = express.Router();

// Create a post
router.post("/", protect, async (req, res) => {
  const post = await Post.create({ user: req.user._id, content: req.body.content });
  res.json({ message: "Post created", post });
});

// Get all posts
router.get("/", protect, async (req, res) => {
  const posts = await Post.find().populate("user", "name headline");
  res.json(posts);
});

// Delete post
router.delete("/:id", protect, async (req, res) => {
  const post = await Post.findById(req.params.id);
  if (!post) return res.status(404).json({ message: "Not found" });
  if (post.user.toString() !== req.user._id.toString())
    return res.status(403).json({ message: "Forbidden" });
  await post.deleteOne();
  res.json({ message: "Post deleted" });
});

export default router;

```

server.js -

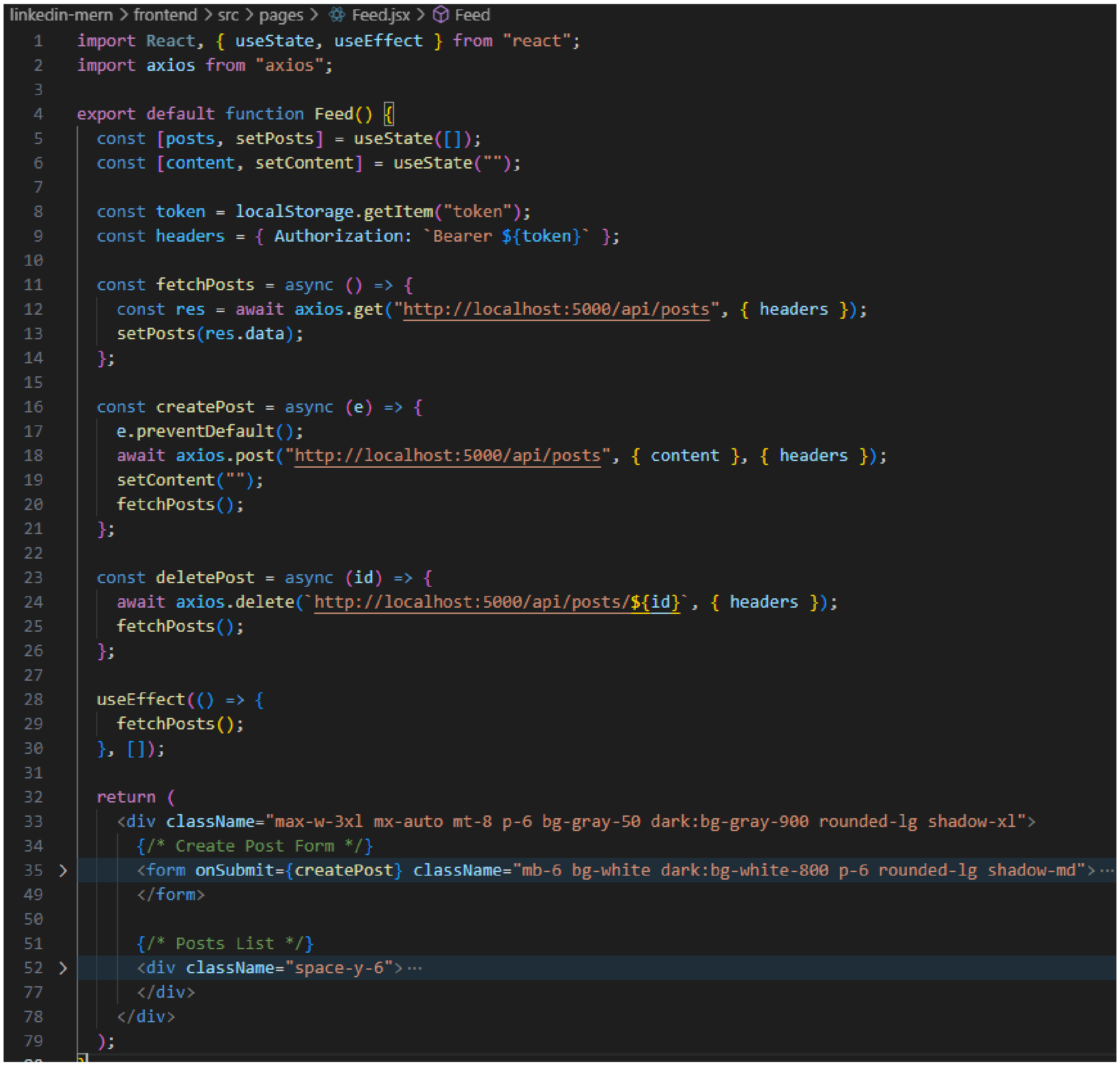

linkedin-mern > backend > JS server.js > ...

```
import express from "express";
import dotenv from "dotenv";
import cors from "cors";
import connectDB from "./config/db.js";
import authRoutes from "./routes/authRoutes.js";
import postRoutes from "./routes/postRoutes.js";

dotenv.config();
connectDB();

const app = express();
app.use(cors());
app.use(express.json());

app.get("/", (req, res) => res.send("🔐 Secure REST API Ready"));
app.use("/api/auth", authRoutes);
app.use("/api/posts", postRoutes);

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`🚀 Server running on port ${PORT}`));
```

Features -

- Secure login and signup using **JWT**
- Passwords hashed using **bcrypt**
- Protected API routes using **middleware**
- CRUD operations for posts
- React frontend (LinkedIn-like UI)
- Tailwind-based **light/dark mode toggle**

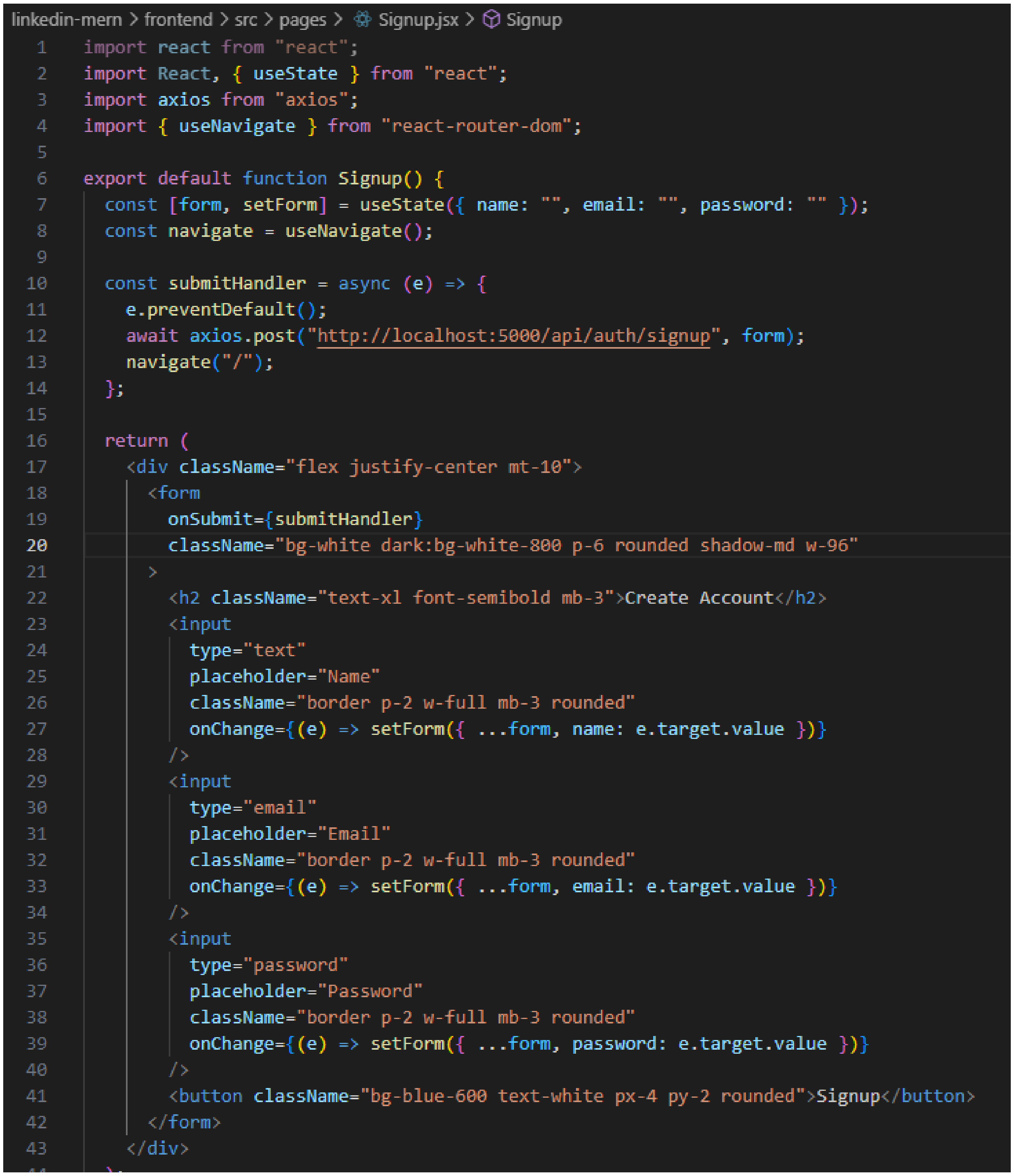
Feed.jsx -



linkedin-mern > frontend > src > pages > Feed.jsx > Feed

```jsx
import React, { useState, useEffect } from "react";
import axios from "axios";

export default function Feed() {
  const [posts, setPosts] = useState([]);
  const [content, setContent] = useState("");

  const token = localStorage.getItem("token");
  const headers = { Authorization: `Bearer ${token}` };

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

  const createPost = async (e) => {
    e.preventDefault();
    await axios.post("http://localhost:5000/api/posts", { content }, { headers });
    setContent("");
    fetchPosts();
  };

  const deletePost = async (id) => {
    await axios.delete(`http://localhost:5000/api/posts/${id}`, { headers });
    fetchPosts();
  };

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

  return (
    <div className="max-w-3xl mx-auto mt-8 p-6 bg-gray-50 dark:bg-gray-900 rounded-lg shadow-xl">
      {/* Create Post Form */}
      <form onSubmit={createPost} className="mb-6 bg-white dark:bg-white-800 p-6 rounded-lg shadow-md">…
      </form>

      {/* Posts List */}
      <div className="space-y-6">…
      </div>
    </div>
  );
```

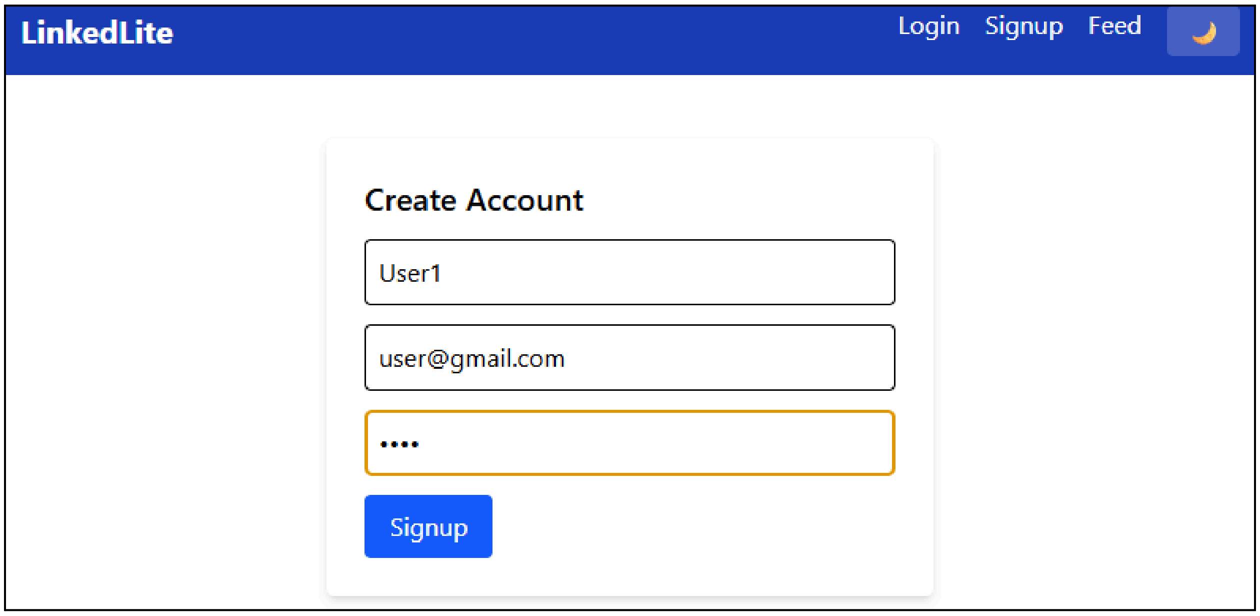
Signup.jsx -



linkedin-mern > frontend > src > pages > Signup.jsx > Signup

```jsx
import react from "react";
import React, { useState } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom";

export default function Signup() {
  const [form, setForm] = useState({ name: "", email: "", password: "" });
  const navigate = useNavigate();

  const submitHandler = async (e) => {
    e.preventDefault();
    await axios.post("http://localhost:5000/api/auth/signup", form);
    navigate("/");
  };

  return (
    <div className="flex justify-center mt-10">
      <form
        onSubmit={submitHandler}
        className="bg-white dark:bg-white-800 p-6 rounded shadow-md w-96"
      >
        <h2 className="text-xl font-semibold mb-3">Create Account</h2>
        <input
          type="text"
          placeholder="Name"
          className="border p-2 w-full mb-3 rounded"
          onChange={(e) => setForm({ ...form, name: e.target.value })}
        />
        <input
          type="email"
          placeholder="Email"
          className="border p-2 w-full mb-3 rounded"
          onChange={(e) => setForm({ ...form, email: e.target.value })}
        />
        <input
          type="password"
          placeholder="Password"
          className="border p-2 w-full mb-3 rounded"
          onChange={(e) => setForm({ ...form, password: e.target.value })}
        />
        <button className="bg-blue-600 text-white px-4 py-2 rounded">Signup</button>
      </form>
    </div>
```

Output -



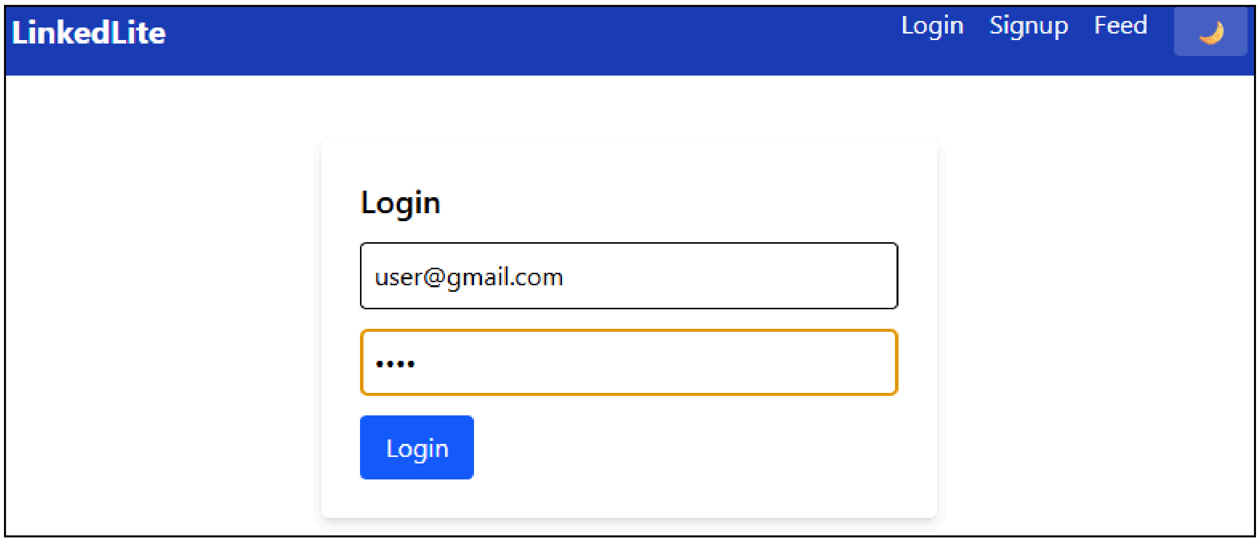
Create Account

User1

user@gmail.com

••••

Fig 1.1 - Secure Signup

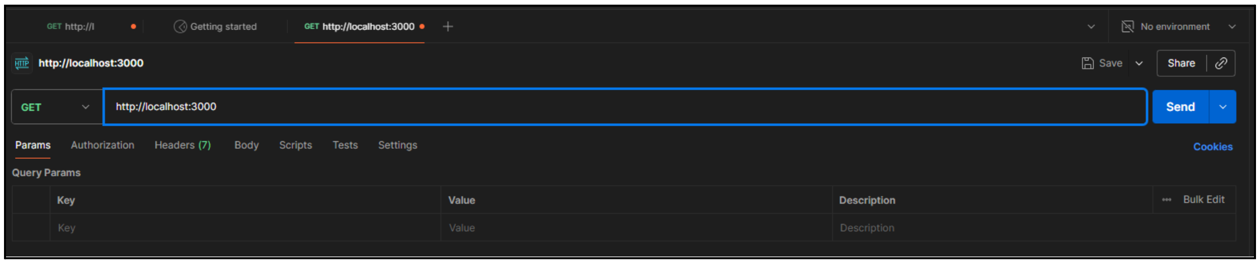


Login

user@gmail.com

••••

Fig 1.2 Secure Login



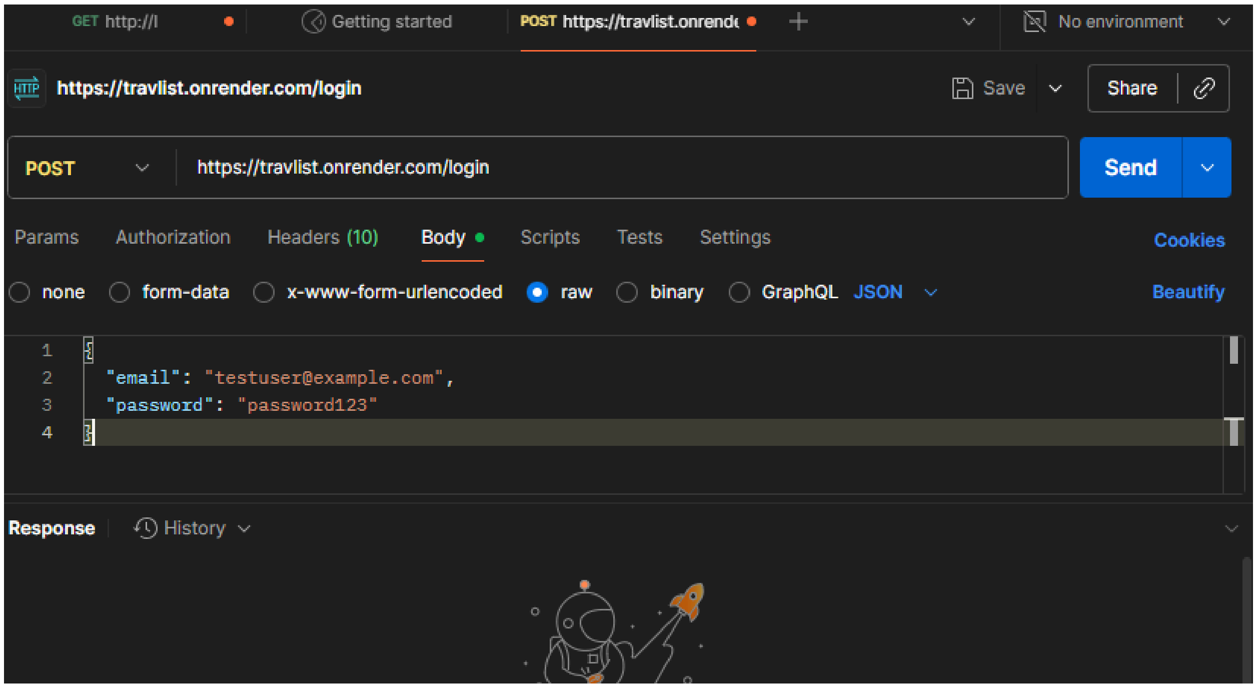
http://localhost:3000

GET http://localhost:3000

Query Params

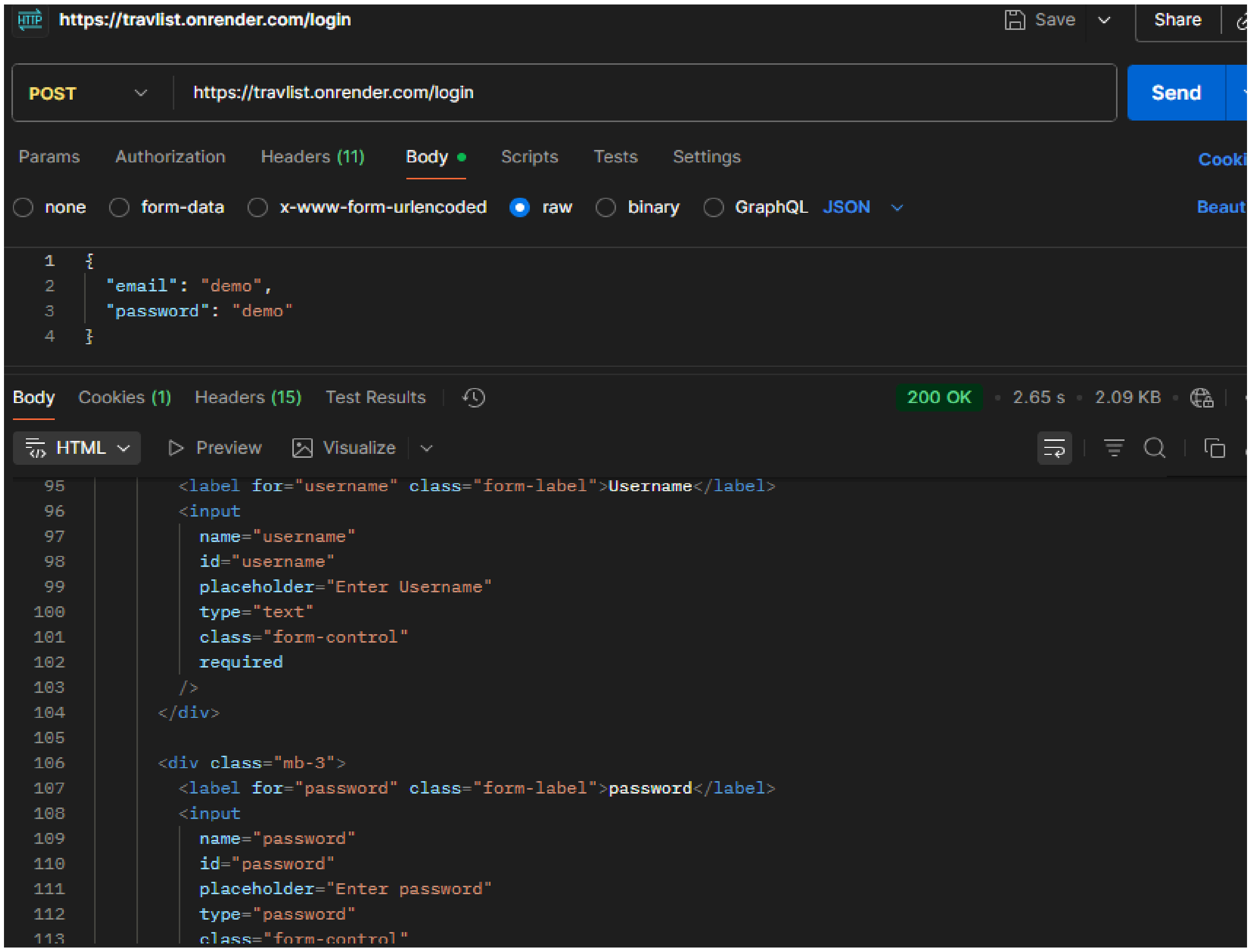
Key	Value	Description
Key	Value	Description

Fig 2.1 - Postman Validation



```
{
  "email": "testuser@example.com",
  "password": "password123"
}
```

Fig 2.2 - POST request



```
{
  "email": "demo",
  "password": "demo"
}
```

```
        <label for="username" class="form-label">Username</label>
        <input
          name="username"
          id="username"
          placeholder="Enter Username"
          type="text"
          class="form-control"
          required
        />
      </div>

      <div class="mb-3">
        <label for="password" class="form-label">password</label>
        <input
          name="password"
          id="password"
          placeholder="Enter password"
          type="password"
          class="form-control"
```

Fig 2.3 - Response Status