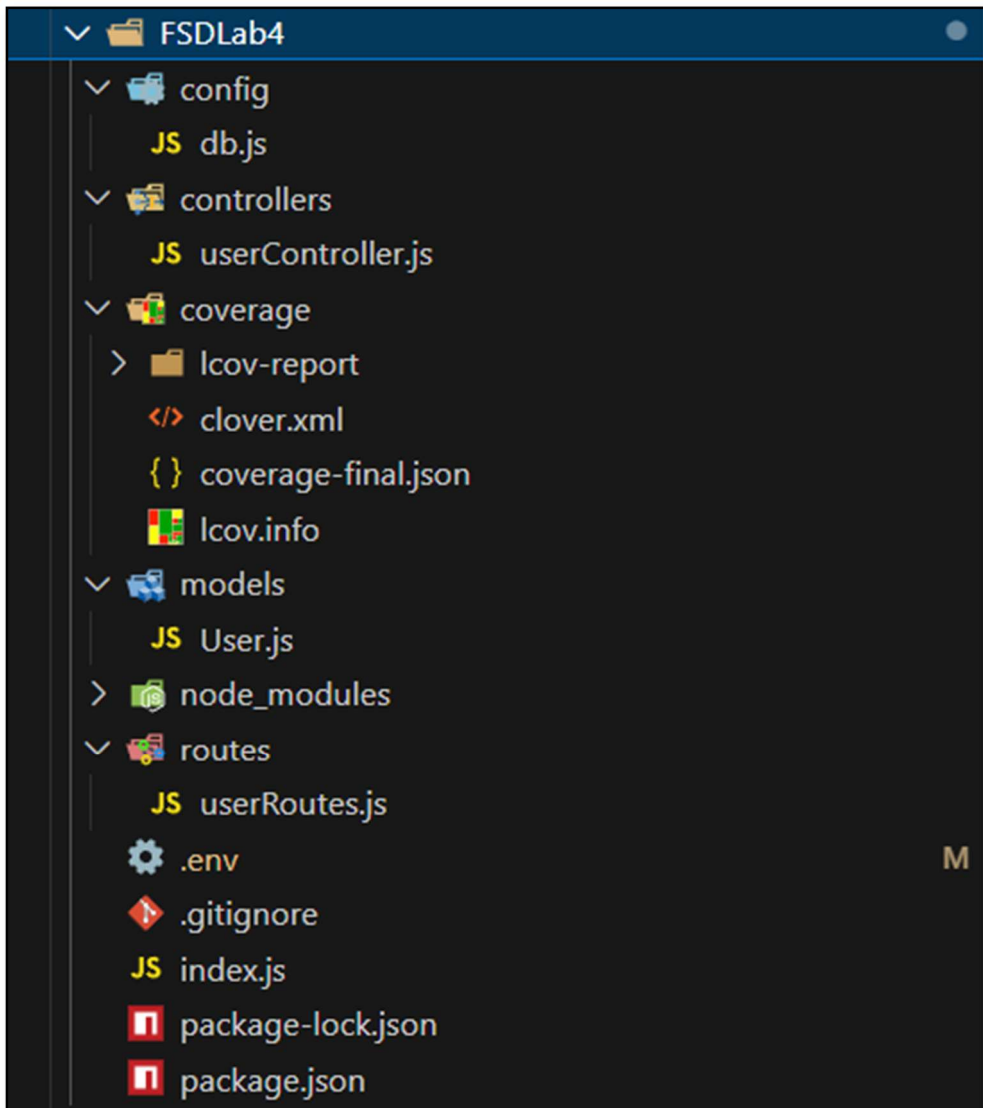


## Experiment No. – 4

Roll No.	21
Name	Prajwal Kasbale
Class	D15A
Subject	Full Stack Development
Lab Outcome	REST API Design with MongoDB + Mongoose Integration
Date of Performance/ Submission	
Signature & Grades	

## Process :-

Directory Structure :-



## Code:

### Index.js

```
1  require("dotenv").config();
2  const express = require("express");
3  const cors = require("cors");
4  const connectDB = require("../config/db");
5  const userRoutes = require("../routes/userRoutes");
6
7  const app = express();
8
9  // Middleware
10 app.use(cors());
11 app.use(express.json());
12
13 // Connect DB only if not testing
14 if (process.env.NODE_ENV !== "test") {
15   connectDB();
16 }
17
18 // Routes
19 app.use("/users", userRoutes);
20
21 // Start server only if not testing
22 if (process.env.NODE_ENV !== "test") {
23   const PORT = process.env.PORT || 5000;
24   app.listen(PORT, () =>
25     console.log(`🚀 Server running at http://localhost:${PORT}`)
26   );
27 }
28
29 module.exports = app; // ✅ export app for supertest
30
```

### routes/userRoutes.js

```
1  const express = require("express");
2  const {
3    createUser,
4    getUsers,
5    getUserById,
6    updateUser,
7    deleteUser,
8  } = require("../controllers/userController");
9
10 const router = express.Router();
11
12 router.post("/", createUser);
13 router.get("/", getUsers);
14 router.get("/:id", getUserById);
15 router.put("/:id", updateUser);
16 router.delete("/:id", deleteUser);
17
18 module.exports = router;
19
```

## Models/User.js

```
1  const mongoose = require("mongoose");
2
3  const userSchema = new mongoose.Schema(
4    {
5      name: { type: String, required: true },
6      email: { type: String, required: true, unique: true },
7      age: { type: Number, required: true },
8    },
9    { timestamps: true }
10 );
11
12 module.exports = mongoose.model("User", userSchema);
13
```

## Config/db.js

```
1  const mongoose = require("mongoose");
2
3  const connectDB = async () => {
4    console.log(process.env.MONGO_URI);
5    try {
6      await mongoose.connect(process.env.MONGO_URI);
7      console.log("✅ MongoDB Connected");
8    } catch (error) {
9      console.error("❌ MongoDB Connection Failed:", error.message);
10     process.exit(1);
11   }
12 };
13
14 module.exports = connectDB;
```

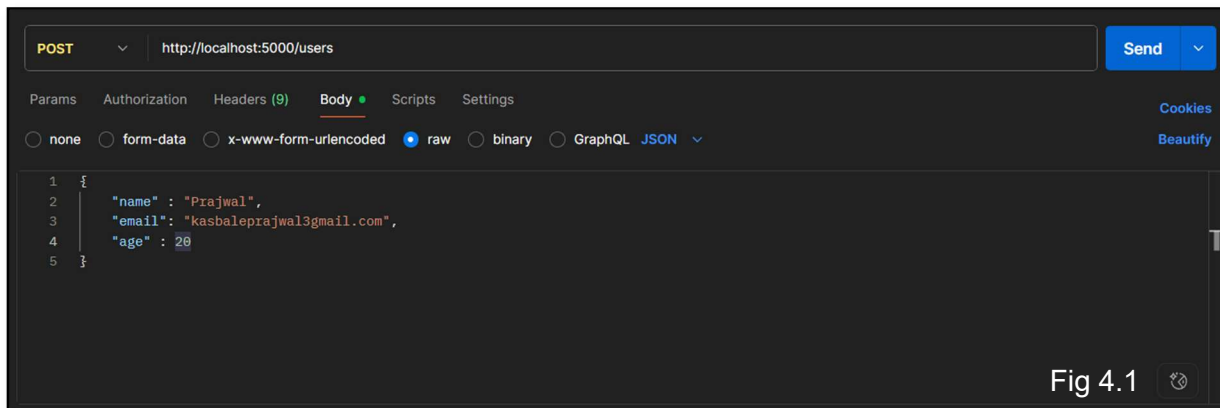
## Controllers/userController.js

```
1  const User = require("../models/User");
2
3  exports.createUser = async (req, res) => {
4    try {
5      const user = await User.create(req.body);
6      res.status(201).json({ success: true, data: user });
7    } catch (err) {
8      res.status(400).json({ success: false, message: err.message });
9    }
10  };
11
12  exports getUsers = async (req, res) => {
13    try {
14      const users = await User.find();
15      res.json({ success: true, data: users });
16    } catch (err) {
17      res.status(500).json({ success: false, message: err.message });
18    }
19  };
20
21  exports.getUserById = async (req, res) => {
22    try {
23      const user = await User.findById(req.params.id);
24      if (!user) return res.status(404).json({ success: false, message: "User not found" });
25      res.json({ success: true, data: user });
26    } catch (err) {
27      res.status(500).json({ success: false, message: err.message });
28    }
29  };
30
31  exports.updateUser = async (req, res) => {
32    try {
33      const user = await User.findByIdAndUpdate(req.params.id, req.body, {
34        new: true,
35        runValidators: true,
36      });
37      if (!user) return res.status(404).json({ success: false, message: "User not found" });
38      res.json({ success: true, data: user });
39    } catch (err) {
40      res.status(400).json({ success: false, message: err.message });
41    }
42  };
43
44  // ♦ DELETE
45  exports.deleteUser = async (req, res) => {
46    try {
47      const user = await User.findByIdAndDelete(req.params.id);
48      if (!user) return res.status(404).json({ success: false, message: "User not found" });
49      res.json({ success: true, message: "User deleted successfully" });
50    } catch (err) {
51      res.status(500).json({ success: false, message: err.message });
52    }
53  };
54
```

## Output:

### Create User (POST)

Request :-

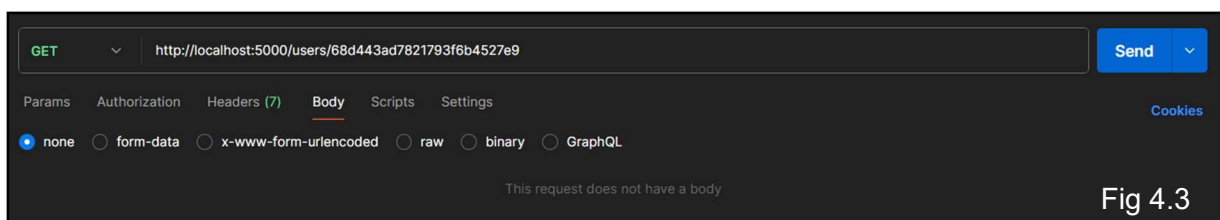


Response :-

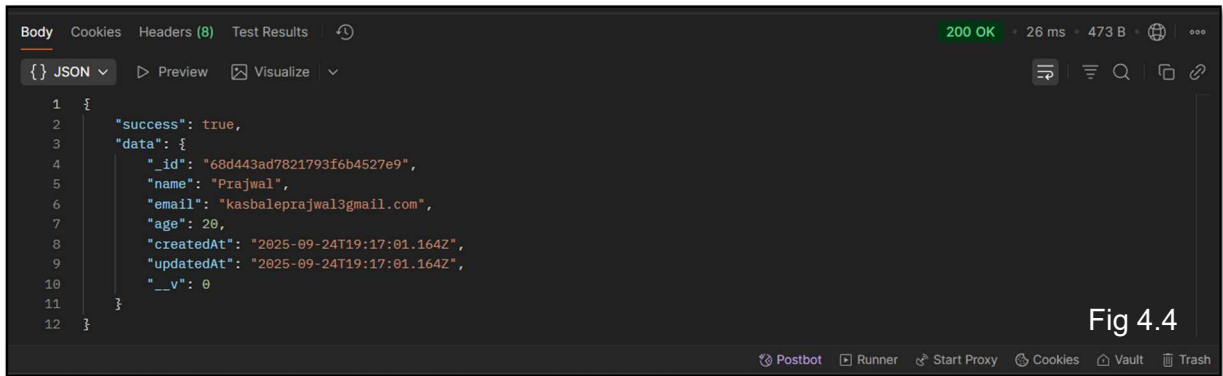


### Fetch User Data (GET user by ID)

Request :-

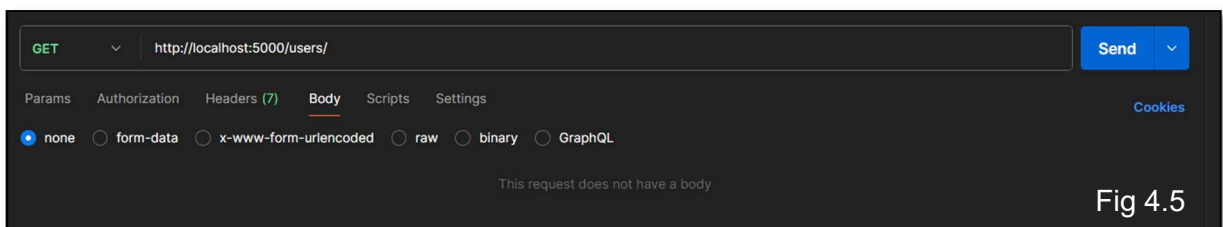


Response :-



## Fetch All Users (Get)

Request :-

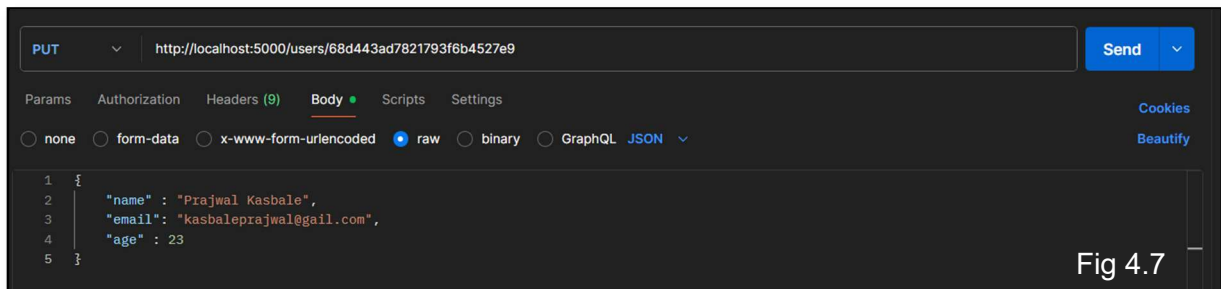


Response :-

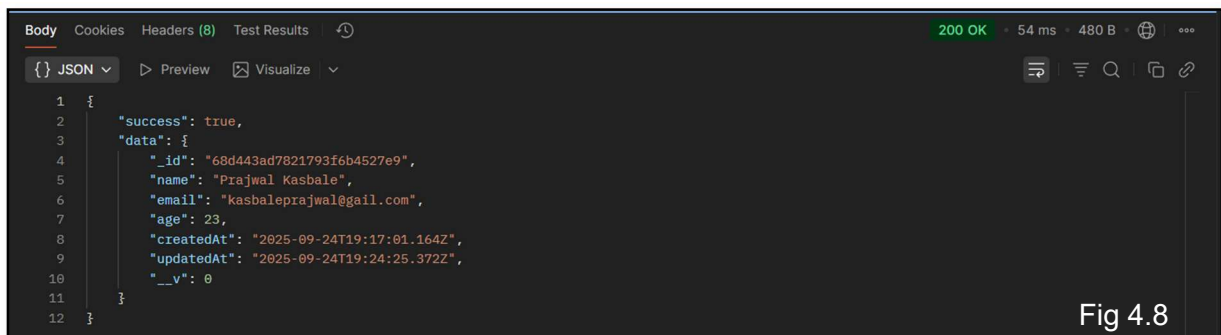


## Update User (PUT)

Request :-

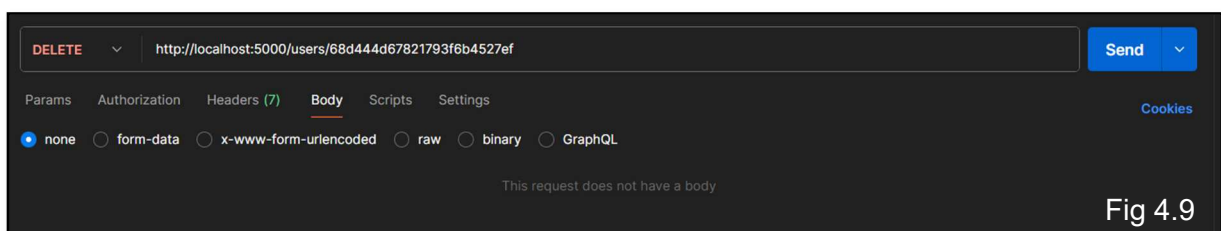


Response :-



## Delete User (Delete)

Request :-



Response :-





## Mongo Atlas Server :-

The screenshot displays the MongoDB Atlas web interface. At the top, the navigation bar shows the Organization (Prajwal's Org - 2024-...), Project (Project 0), and Cluster (FSD). The left sidebar shows the database structure: sample\_mflix, test, and users. The main panel shows the 'test.users' collection with a storage size of 36KB, logical data size of 139B, total documents of 1, and indexes total size of 72KB. The 'Find' tab is active, showing a query filter bar with the text 'Type a query: { field: 'value' }'. Below the filter bar, the query results are displayed as a single document:

```
{
  "_id": ObjectId('68d443ad7821793f6b4527e9'),
  "name": "Prajwal Kasbale",
  "email": "kasbaleprajwal@gmail.com",
  "age": 23,
  "createdAt": 2025-09-24T19:17:01.164+00:00,
  "updatedAt": 2025-09-24T19:24:25.372+00:00,
  "__v": 0
}
```

The interface also includes buttons for '+ Create Database', 'Search Namespaces', 'Visualize Your Data', and 'Refresh'.

Fig 4.11