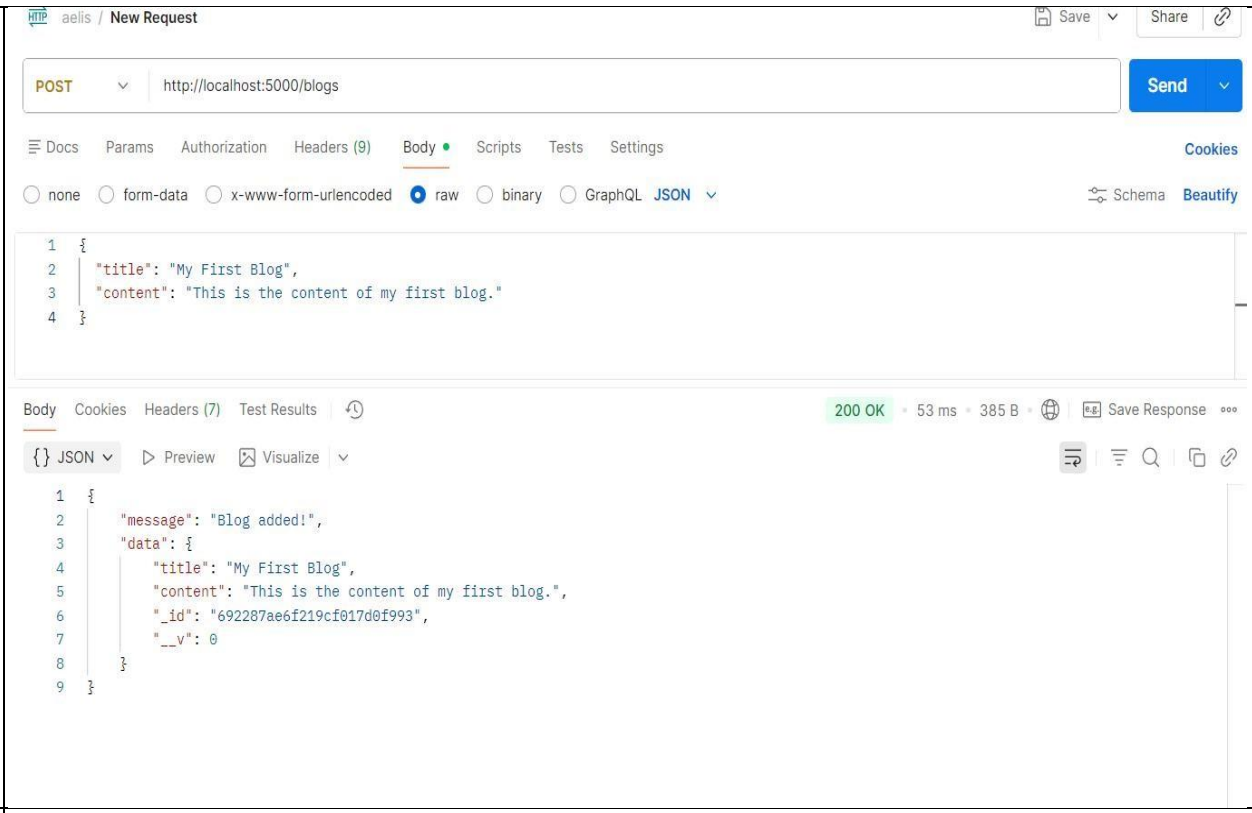
1	**Whilebuilding ablogging application,usersshouldbeabletoaddnewblogpoststo the database. Sometimes, they might forget to include the title. Create a Mongoose model with validation to ensure that each blog post has both title and content. Additionally,configureyourapplicationtohandleinvalidsubmissionsbyreturninga properJSONerror response.**

Code	// server.js
	const express = require("express");
	const mongoose = require("mongoose");
	const blogRoutes = require("./routes/blogRoutes");
	const app = express();
	app.use(express.json());
	mongoose
	.connect("mongodb://127.0.0.1:27017/practical5")
	.then(() => console.log("✓Connected to MongoDB"))
	.catch((err) => console.log("✗MongoDB Error:", err.message));
	app.use("/blogs", blogRoutes);
	const PORT = 5000;
	app.listen(PORT, () => console.log(`Q1 Server running at http://localhost:${PORT}`));
	
	// models/Blog.js
	const mongoose = require("mongoose");
	const blogSchema = new mongoose.Schema({
	title: { type: String, required: [true, "Title is required"] },
	content: { type: String, required: [true, "Content is required"] },
	});
	module.exports = mongoose.model("Blog", blogSchema);
	// controllers/blogController.js
	const Blog = require("../models/Blog");
	exports.createBlog = async (req, res) => {
	try {
	const post = await Blog.create(req.body);
	res.json({ message: "Blog added!", data: post });
	} catch (err) {
	res.status(400).json({ error: err.message });
	}
	};
	// routes/blogRoutes.js
	const express = require("express");
	const { createBlog } = require("../controllers/blogController");
	const router = express.Router();
	router.post("/", createBlog);
	module.exports = router;

output

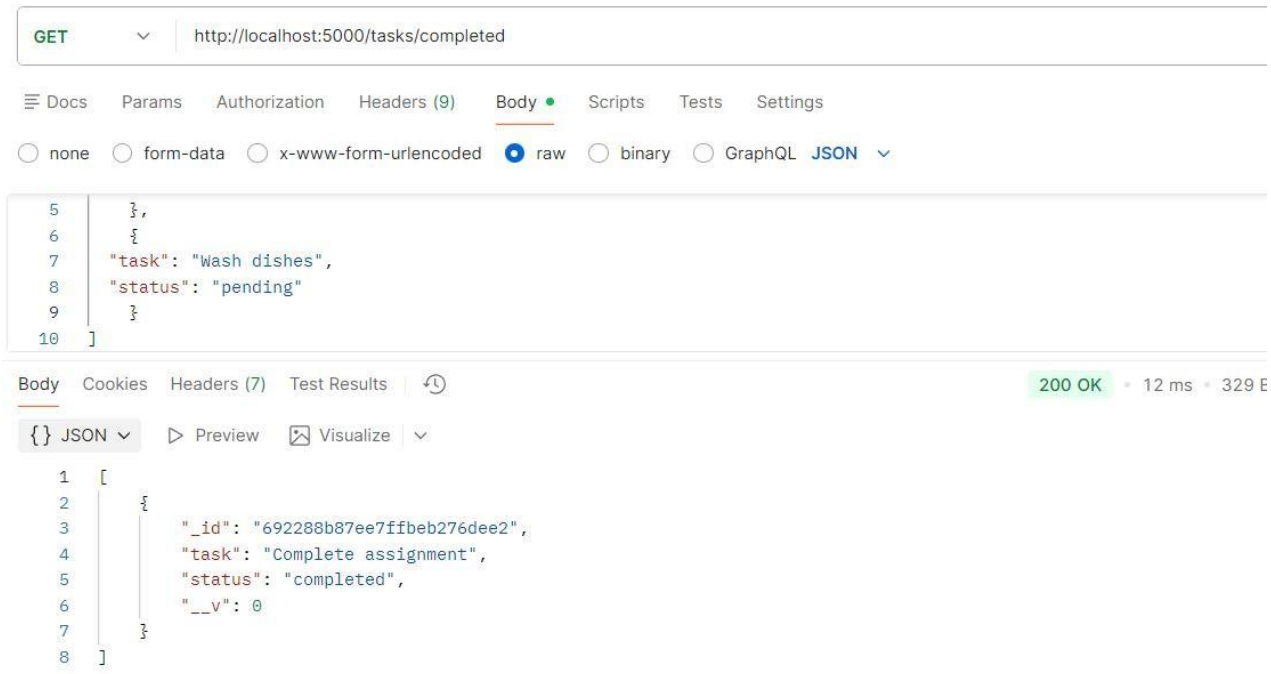


2 **Inataskmanagementsystem,implementa Mongoosequerytofetchanddisplayall taskswherethestatusfieldissetto"completed"**

Code

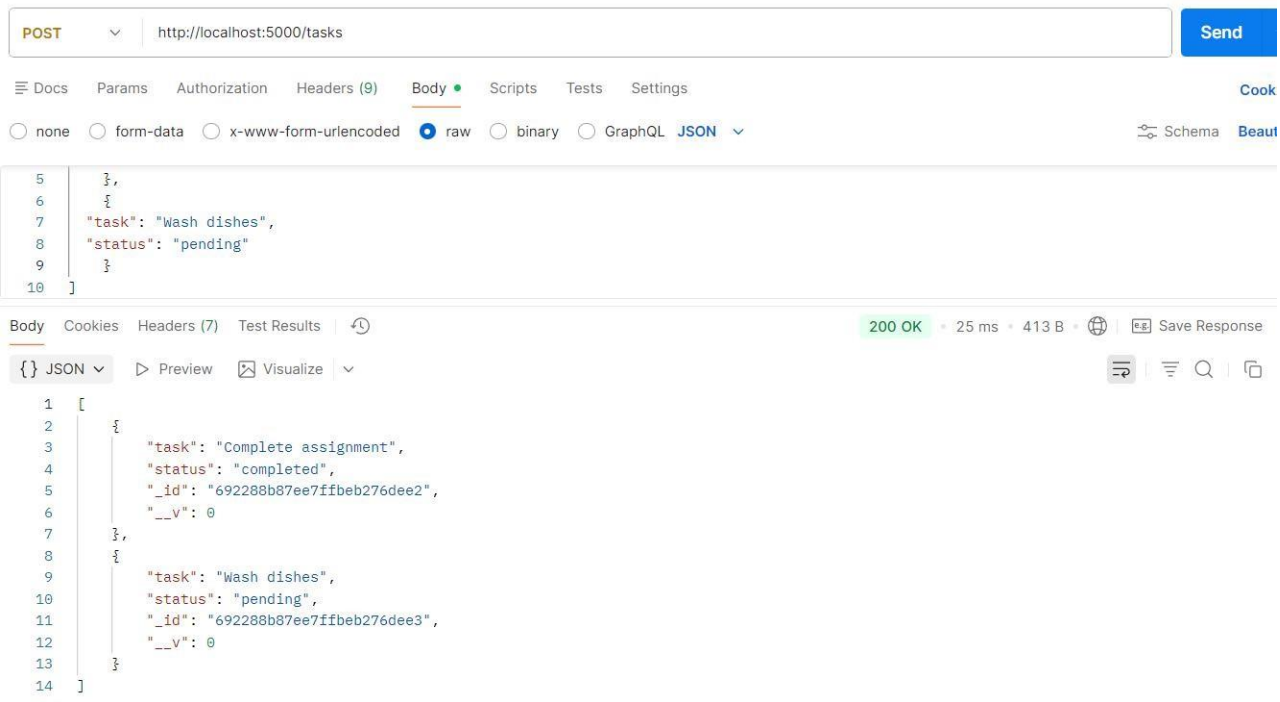
```
// server.js
const express = require("express");
const mongoose = require("mongoose");
const taskRoutes = require("./routes/taskRoutes");
const app = express();
app.use(express.json());
mongoose
.connect("mongodb://127.0.0.1:27017/practical5")
.then(() => console.log("✓Q2 MongoDB connected"))
.catch((err) => console.log("✗", err.message));
app.use("/", taskRoutes);
const PORT = 5002;
app.listen(PORT, () => console.log(`Q2 Server running at http://localhost:${PORT}`));
// models/Task.js
const mongoose = require("mongoose");
const taskSchema = new mongoose.Schema({
task: String,
status: String,
});
module.exports = mongoose.model("Task", taskSchema);
// controllers/taskController.js
const Task = require("../models/Task");
exports.getCompletedTasks = async (req, res) => {
try {
const tasks = await Task.find({ status: "completed" });
res.json(tasks);
```

	} catch (err) {
	res.status(500).json({ error: err.message });
	}
	};
	// routes/taskRoutes.js
	const express = require("express");
	const { getCompletedTasks } = require("../controllers/taskController");
	const router = express.Router();
	router.get("/tasks/completed", getCompletedTasks);
	module.exports = router;

Output	
3	**You are working on a Task Management System where users can create tasks with different statuses, such as "completed", "in-progress", and "pending". You need to implementaMongoosequerytofetchanddisplayalltasksthathavethestatusfield setto"completed".**

Code	// server.js
	const express = require("express");
	const mongoose = require("mongoose");
	const taskRoutes = require("./routes/taskRoutes");
	const app = express();
	app.use(express.json());
	mongoose
	.connect("mongodb://127.0.0.1:27017/practical5")
	.then(() => console.log("✓Q3 MongoDB connected"))
	.catch((err) => console.log("✗", err.message));
	app.use("/", taskRoutes);
	const PORT = 5003;

```
app.listen(PORT, () => console.log(`Q3 Server running at http://localhost:${PORT}`));
// models/Task.js
const mongoose = require("mongoose");
const taskSchema = new mongoose.Schema({
task: String,
status: String,
});
module.exports = mongoose.model("Task", taskSchema);
// controllers/taskController.js
const Task = require("../models/Task");
exports.createTask = async (req, res) => {
try {
const task = await Task.create(req.body);
res.json(task);
} catch (err) {
res.status(400).json({ error: err.message });
}
};
exports.getCompletedTasks = async (req, res) => {
try {
const tasks = await Task.find({ status: "completed" });
res.json(tasks);
} catch (err) {
res.status(500).json({ error: err.message });
}
};
// routes/taskRoutes.js
const express = require("express");
const { createTask, getCompletedTasks } = require("../controllers/taskController");
const router = express.Router();
router.post("/tasks", createTask);
router.get("/tasks/completed", getCompletedTasks);
module.exports = router;
```

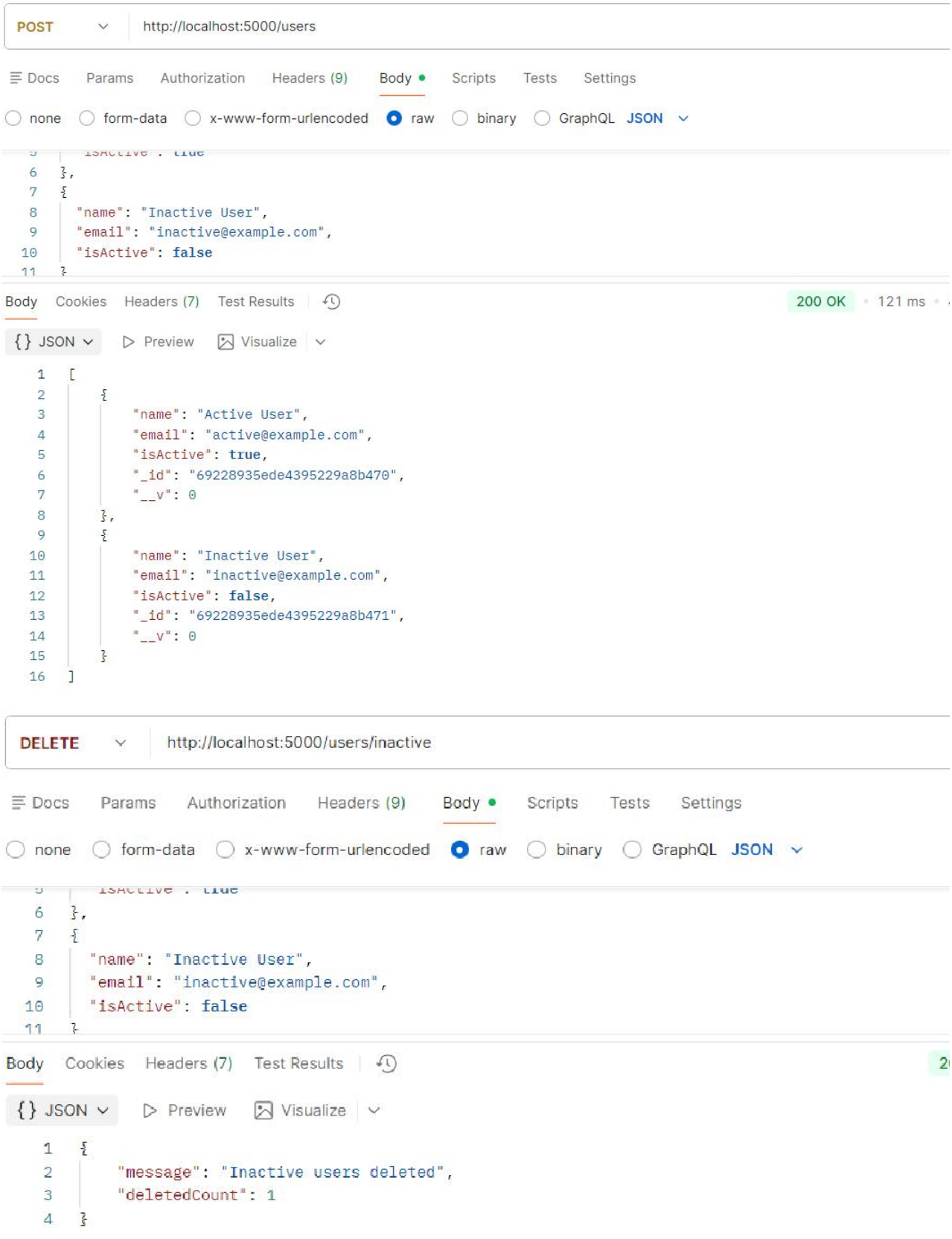


4	**Youarebuildinganadminpanelwhereadminscandeletemultipleinactiveusers from the database in one go. Write the Mongoose query to remove all documents with isActive: false and send a success response with the number of deleted documents.**

Code	// models/User.js
	const mongoose = require("mongoose");
	const userSchema = new mongoose.Schema({
	name: String,
	email: String,
	isActive: { type: Boolean, default: true },
	});
	module.exports = mongoose.model("User", userSchema);
	// controllers/userController.js
	const User = require("../models/User");
	exports.createUser = async (req, res) => {
	try {
	const u = await User.create(req.body);
	res.json(u);
	} catch (err) {
	res.status(400).json({ error: err.message });
	}
	};
	exports.deleteInactiveUsers = async (req, res) => {
	try {
	const result = await User.deleteMany({ isActive: false });
	res.json({ message: "Inactive users deleted", deletedCount: result.deletedCount });
	} catch (err) {
	res.status(500).json({ error: err.message });

```
}
};
// routes/userRoutes.js
const express = require("express");
const { createUser, deleteInactiveUsers } = require("../controllers/userController");
const router = express.Router();
router.post("/users", createUser);
router.delete("/users/inactive", deleteInactiveUsers);
module.exports = router;
// server.js
const express = require("express");
const mongoose = require("mongoose");
const userRoutes = require("./routes/userRoutes");
const app = express();
app.use(express.json());
mongoose.connect("mongodb://127.0.0.1:27017/practical5")
.then(()=>console.log("✓Q4 MongoDB connected"))
.catch(err=>console.log("✗, err.message));
app.use("/", userRoutes);
app.listen(5004, ()=>console.log("Q4 Server running on http://localhost:5004"));
```

Output



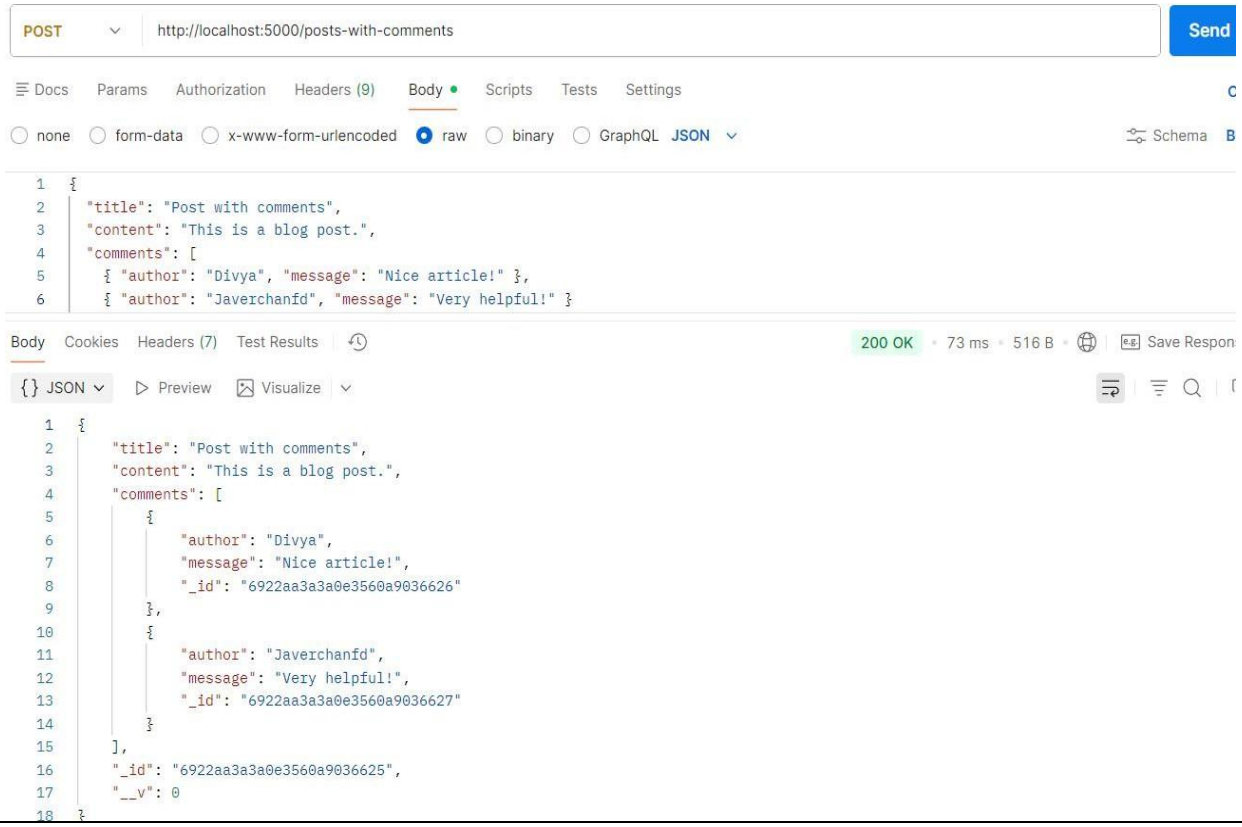
5	**Whiledesigning ablogging platform,eachblogpostshouldstoreitscomments.**

	CreateaMongooseschemathat embedscommentsinsidethepostdocument.Each commentshouldhaveauthorandmessage

Code	// models/Post.js
	const mongoose = require("mongoose");
	const commentSchema = new mongoose.Schema({
	author: String,
	message: String,
	createdAt: { type: Date, default: Date.now },
	});
	const postSchema = new mongoose.Schema({
	title: String,
	content: String,
	comments: [commentSchema],
	createdAt: { type: Date, default: Date.now },
	});
	module.exports = mongoose.model("Post", postSchema);
	// controllers/postController.js
	const Post = require("../models/Post");
	exports.createPost = async (req, res) => {
	try {
	const p = await Post.create(req.body);
	res.json(p);
	} catch (err) {
	res.status(400).json({ error: err.message });
	}
	};
	exports.addComment = async (req, res) => {
	try {
	const post = await Post.findById(req.params.id);
	post.comments.push(req.body);
	await post.save();
	res.json(post);
	} catch (err) {
	res.status(400).json({ error: err.message });
	}
	};
	exports.getPosts = async (req, res) => {
	const posts = await Post.find();
	res.json(posts);
	};
	// routes/postRoutes.js
	const express = require("express");
	const { createPost, addComment, getPosts } = require("../controllers/postController");
	const router = express.Router();
	router.post("/posts", createPost);
	router.get("/posts", getPosts);
	router.post("/posts/:id/comments", addComment);
	module.exports = router;

```
// server.js
const express = require("express");
const mongoose = require("mongoose");
const postRoutes = require("./routes/postRoutes");
const app = express();
app.use(express.json());
mongoose.connect("mongodb://127.0.0.1:27017/practical5")
.then(()=>console.log("✓Q5 MongoDB connected"))
.catch(err=>console.log("✕, err.message));
app.use("/", postRoutes);
app.listen(5005, ()=>console.log("Q5 Server running on http://localhost:5005"));
```

Output



6 **Inastudentmanagementapplication,eachstudentcanenrollinmultiplecourses.**

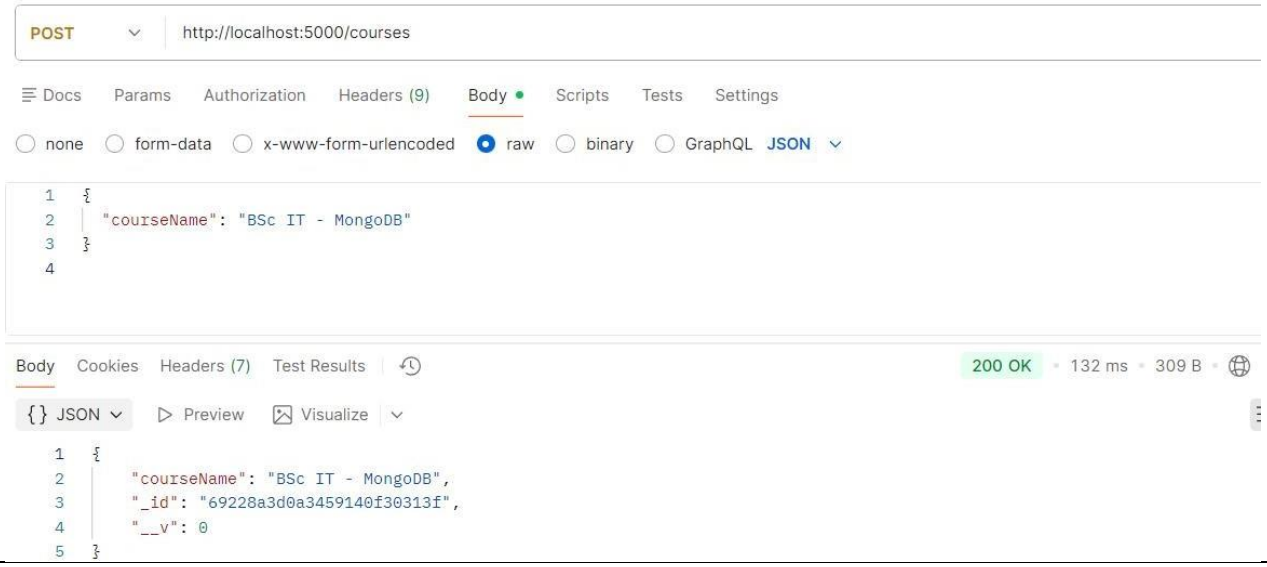
Designanormalizedschemawherestudentdocumentsreferencecoursedocuments.

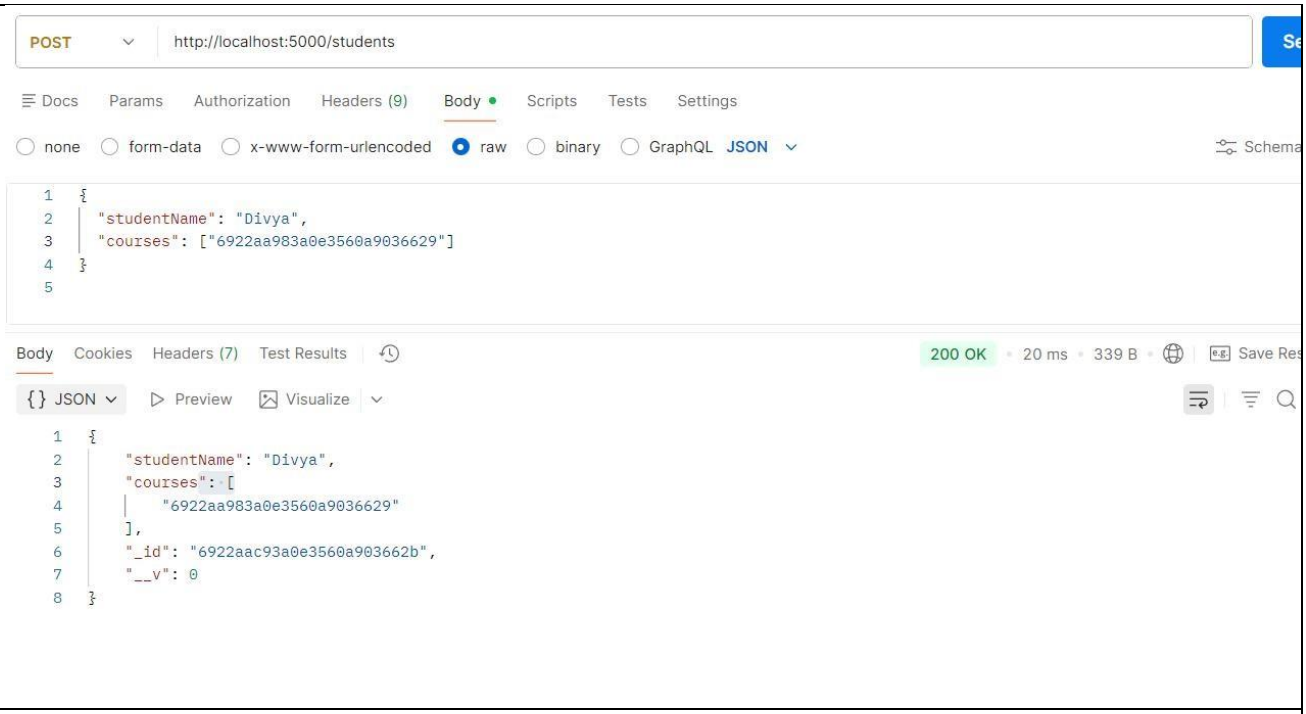
Code

```
// models/Course.js
const mongoose = require("mongoose");
const courseSchema = new mongoose.Schema({ courseName: String });
module.exports = mongoose.model("Course", courseSchema);
// models/Student.js
const mongoose = require("mongoose");
const studentSchema = new mongoose.Schema({
studentName: String,
courses: [{ type: mongoose.Schema.Types.ObjectId, ref: "Course" }],
});
module.exports = mongoose.model("Student", studentSchema);
// controllers/courseController.js
const Course = require("../models/Course");
exports.createCourse = async (req, res) => {
try { const c = await Course.create(req.body); res.json(c); }
catch(err){ res.status(400).json({ error: err.message }); }
};
exports.getCourses = async (req,res)=>{ res.json(await Course.find()); };
// controllers/studentController.js
const Student = require("../models/Student");
exports.createStudent = async (req, res) => {
try { const s = await Student.create(req.body); res.json(s); }
catch(err){ res.status(400).json({ error: err.message }); }
};
exports.getStudents = async (req,res)=> {
const students = await Student.find().populate("courses");
res.json(students);
};
// routes/courseRoutes.js
const express = require("express");
const { createCourse, getCourses } = require("../controllers/courseController");
const router = express.Router();
router.post("/courses", createCourse);
router.get("/courses", getCourses);
module.exports = router;
// routes/studentRoutes.js
const express = require("express");
const { createStudent, getStudents } = require("../controllers/studentController");
const router = express.Router();
router.post("/students", createStudent);
router.get("/students", getStudents);
module.exports = router;
// server.js
const express = require("express");
const mongoose = require("mongoose");
const courseRoutes = require("./routes/courseRoutes");
const studentRoutes = require("./routes/studentRoutes");
const app = express();
app.use(express.json());
mongoose.connect("mongodb://127.0.0.1:27017/practical5")
```

```
.then(()=>console.log("✓Q6 MongoDB connected"))
.catch(err=>console.log("✗", err.message));
app.use("/", courseRoutes);
app.use("/", studentRoutes);
app.listen(5006, ()=>console.log("Q6 Server running on http://localhost:5006"));
```

Output



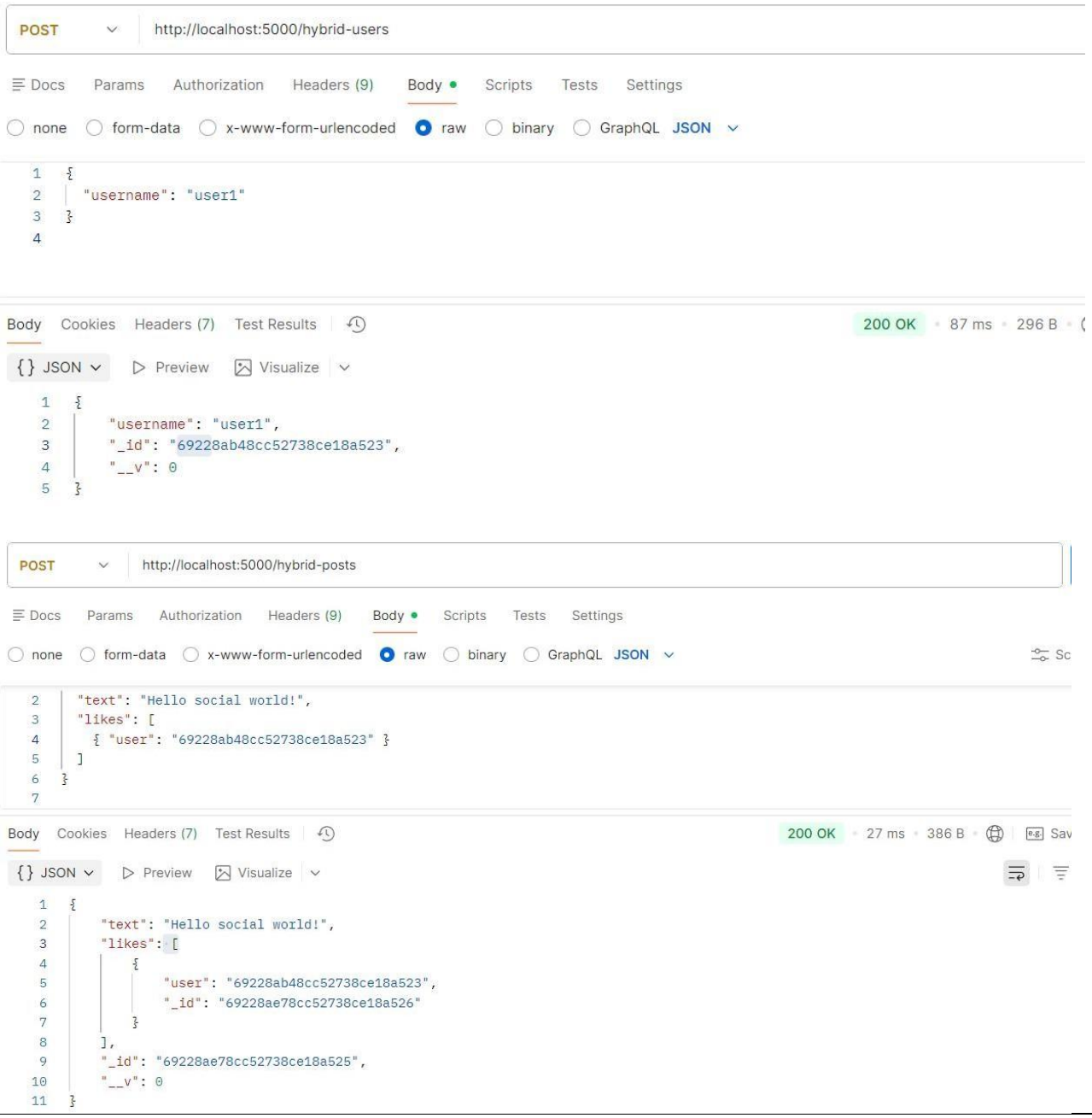


7	**You'rebuildingasocialmediaappwhereuserscanlikepostsandinteractwitheach other's profiles.** **Your task is to design the database schema using Mongoose (a MongoDB ODM for Node.js)** **with a hybrid data model, where:** **1. Likesarestoreddirectlyinthepostdocument(embeddeddata).** **2. Userprofilesarestoredasreferencesinthedatabase,linkinguserstothe posts they** **like (referenced data).**

code	// models/HybridUser.js
	const mongoose = require("mongoose");
	const userSchema = new mongoose.Schema({ username: String });
	module.exports = mongoose.model("HybridUser", userSchema);
	// models/HybridPost.js
	const mongoose = require("mongoose");
	const likeSchema = new mongoose.Schema({
	user: { type: mongoose.Schema.Types.ObjectId, ref: "HybridUser" },
	likedAt: { type: Date, default: Date.now }
	});
	const postSchema = new mongoose.Schema({
	text: String,
	likes: [likeSchema],
	createdAt: { type: Date, default: Date.now }
	});
	module.exports = mongoose.model("HybridPost", postSchema);
	// controllers/hybridController.js
	const HybridUser = require("../models/HybridUser");
	const HybridPost = require("../models/HybridPost");
	exports.createUser = async (req,res)=>{
	const u = await HybridUser.create(req.body);
	res.json(u);

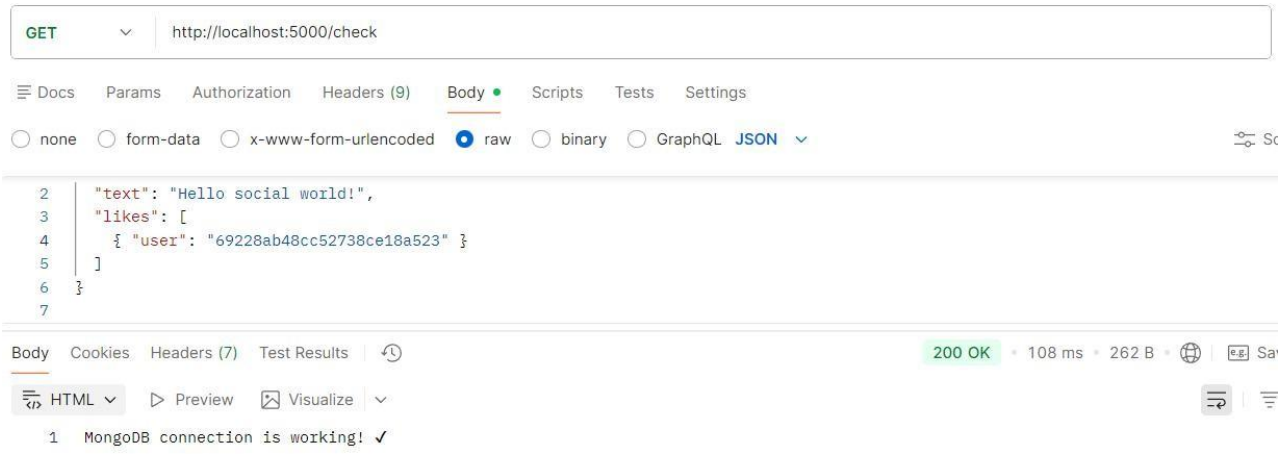
```
};
exports.createPost = async (req,res)=>{
const p = await HybridPost.create(req.body);
res.json(p);
};
exports.likePost = async (req,res)=>{
const post = await HybridPost.findById(req.params.id);
post.likes.push({ user: req.body.user });
await post.save();
res.json(post);
};
exports.getPosts = async (req,res)=>{
const posts = await HybridPost.find().populate("likes.user", "username");
res.json(posts);
};
// routes/hybridRoutes.js
const express = require("express");
const { createUser, createPost, likePost, getPosts } = require("../controllers/hybridController");
const router = express.Router();
router.post("/hybrid-users", createUser);
router.post("/hybrid-posts", createPost);
router.post("/hybrid-posts/:id/like", likePost);
router.get("/hybrid-posts", getPosts);
module.exports = router;
// server.js
const express = require("express");
const mongoose = require("mongoose");
const hybridRoutes = require("./routes/hybridRoutes");
const app = express();
app.use(express.json());
mongoose.connect("mongodb://127.0.0.1:27017/practical5")
.then(()=>console.log("✓Q7 MongoDB connected"))
.catch(err=>console.log("✕, err.message));
app.use("/", hybridRoutes);
app.listen(5007, ()=>console.log("Q7 Server running on http://localhost:5007"));
```

Output


POST
http://localhost:5000/hybrid-users
Docs Params Authorization Headers (9) Body Scripts Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
{
"username": "user1"
}
Body Cookies Headers (7) Test Results
200 OK 87 ms 296 B
JSON Preview Visualize
{
"username": "user1",
"_id": "69228ab48cc52738ce18a523",
"__v": 0
}
POST
http://localhost:5000/hybrid-posts
Docs Params Authorization Headers (9) Body Scripts Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
"text": "Hello social world!",
"likes": [
{ "user": "69228ab48cc52738ce18a523" }
]
}
Body Cookies Headers (7) Test Results
200 OK 27 ms 386 B
JSON Preview Visualize
{
"text": "Hello social world!",
"likes": [
{
"user": "69228ab48cc52738ce18a523",
"_id": "69228ae78cc52738ce18a526"
}
],
"_id": "69228ae78cc52738ce18a525",
"__v": 0
}

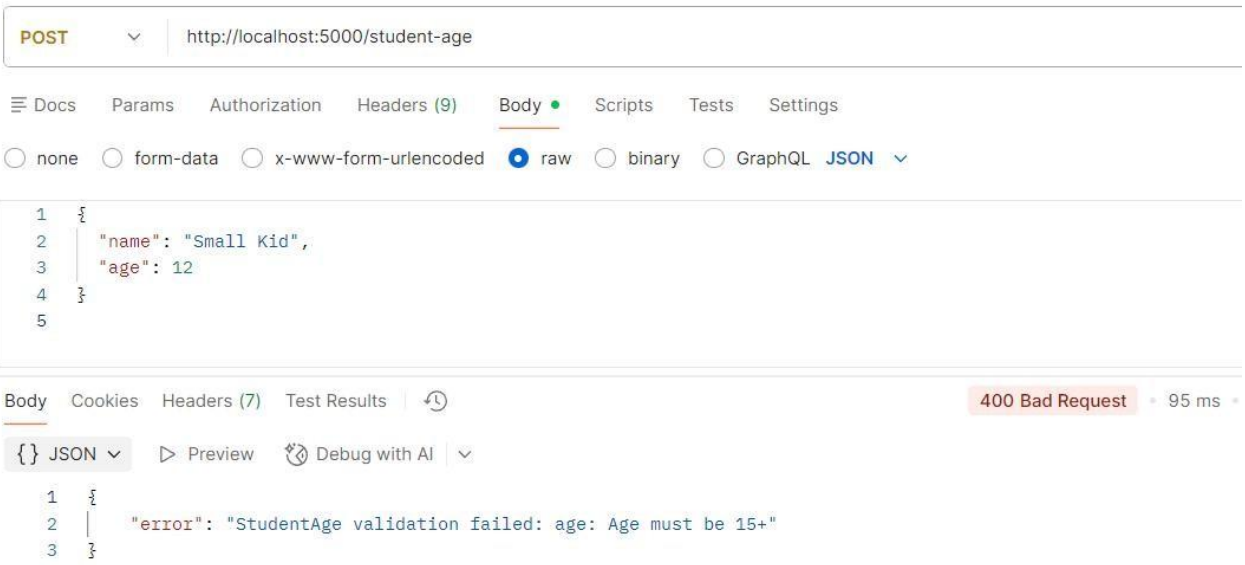
8	**WhiledevelopingaNode.jsapplication,connectittoalocalMongoDBinstance using Mongoose.Writethecodetoestablishtheconnectionandlog"Connectedto MongoDB" on success, or an error message otherwise.**

Code	// server.js
	const express = require("express");
	const mongoose = require("mongoose");
	const app = express();
	mongoose.connect("mongodb://127.0.0.1:27017/practical5")
	.then(()=>console.log("✓Q8 MongoDB connected"))
	.catch(err=>console.log("✕, err.message));
	app.get("/check", (req,res)=>res.send("MongoDB connection is working! ✓"));
	app.listen(5008, ()=>console.log("Q8 Server running on http://localhost:5008"));

Output	
9	**Whiledevelopingaschoolapp,ensure thatstudentsyoungerthan15cannot be savedtothedatabase.Implementthisusing Mongooseschemavalidation.**

Code

```
// models/StudentAge.js
const mongoose = require("mongoose");
const studentAgeSchema = new mongoose.Schema({
name: String,
age: { type: Number, min: [15, "Age must be 15+"] }
});
module.exports = mongoose.model("StudentAge", studentAgeSchema);
// controllers/studentAgeController.js
const StudentAge = require("../models/StudentAge");
exports.create = async (req,res) => {
try {
const s = await StudentAge.create(req.body);
res.json(s);
} catch (err) {
res.status(400).json({ error: err.message });
}
};
exports.getAll = async (req,res) => res.json(await StudentAge.find());
// routes/studentAgeRoutes.js
const express = require("express");
const { create, getAll } = require("../controllers/studentAgeController");
const router = express.Router();
router.post("/student-age", create);
router.get("/student-age", getAll);
module.exports = router;
// server.js
const express = require("express");
const mongoose = require("mongoose");
const routes = require("./routes/studentAgeRoutes");
const app = express();
app.use(express.json());
mongoose.connect("mongodb://127.0.0.1:27017/practical5")
.then(()=>console.log("✓Q9 MongoDB connected"))
.catch(err=>console.log("✕, err.message));
app.use("/", routes);
app.listen(5009, ()=>console.log("Q9 Server running on http://localhost:5009"));
```

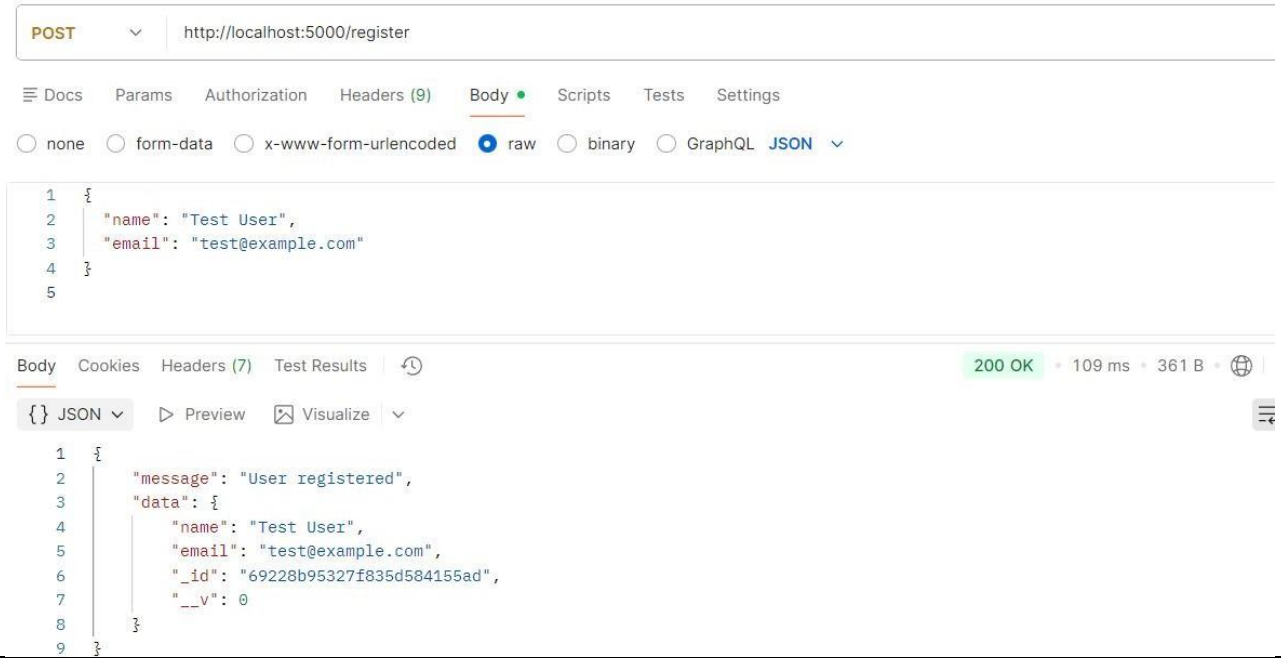
Output	
10	**YouarebuildingauserregistrationAPIforawebapplication.Whenanewusertries to register, you need to ensure that the email address they provide is unique. Implementtheregistrationlogicsothatifauserattemptstoregisterwithan email thatalreadyexistsin thedatabase,theAPIshouldrespondwithaJSONerror**

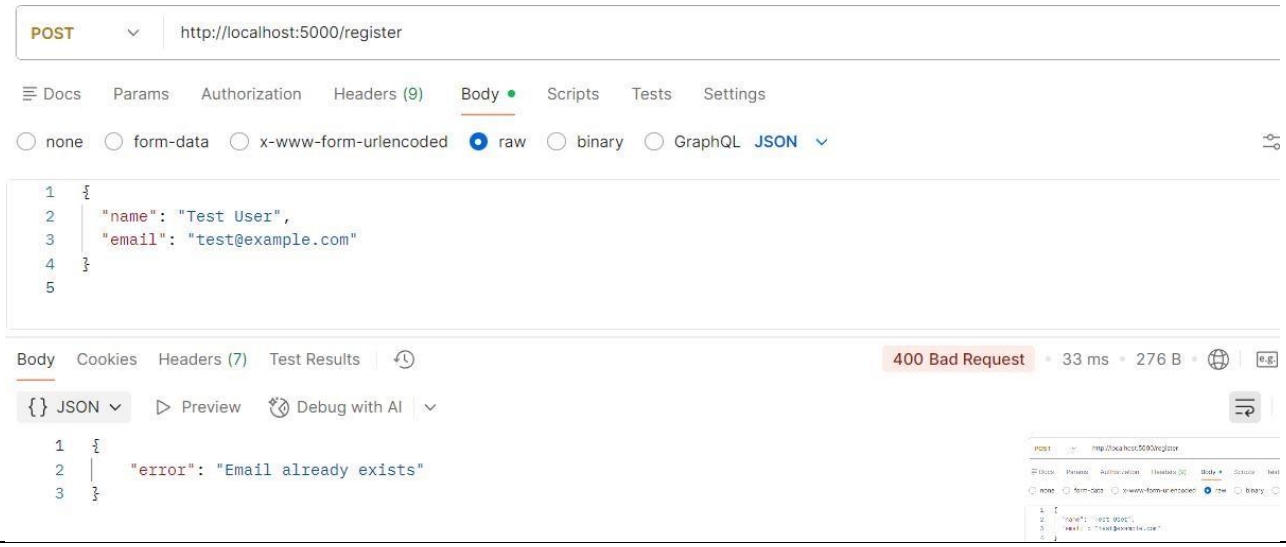
Code	// models/RegUser.js
	const mongoose = require("mongoose");
	const regUserSchema = new mongoose.Schema({
	name: String,
	email: { type: String, unique: true, required: true }
	});
	module.exports = mongoose.model("RegUser", regUserSchema);
	// controllers/regController.js
	const RegUser = require("../models/RegUser");
	exports.register = async (req,res) => {
	try {
	const u = await RegUser.create(req.body);
	res.json({ message: "User registered", data: u });
	} catch (err) {
	if (err.code === 11000) return res.status(400).json({ error: "Email already exists" });
	res.status(500).json({ error: err.message });
	}
	};
	exports.getUsers = async (req,res) => res.json(await RegUser.find());
	// routes/regRoutes.js
	const express = require("express");
	const { register, getUsers } = require("../controllers/regController");
	const router = express.Router();
	router.post("/register", register);

```
router.get("/register", getUsers);
module.exports = router;
// server.js
const express = require("express");
const mongoose = require("mongoose");
const regRoutes = require("./routes/regRoutes");
const app = express();
app.use(express.json());
mongoose.connect("mongodb://127.0.0.1:27017/practical5")
.then(()=>console.log("✓Q10 MongoDB connected"))
.catch(err=>console.log("✗", err.message));
app.use("/", regRoutes);

  app.listen(5010, ()=>console.log("Q10 Server running on http://localhost:5010"));
```

O/p





11 **YouarebuildingasimpleusermanagementsystemusingExpress.jsandMongoDB. Perform the following tasks:**

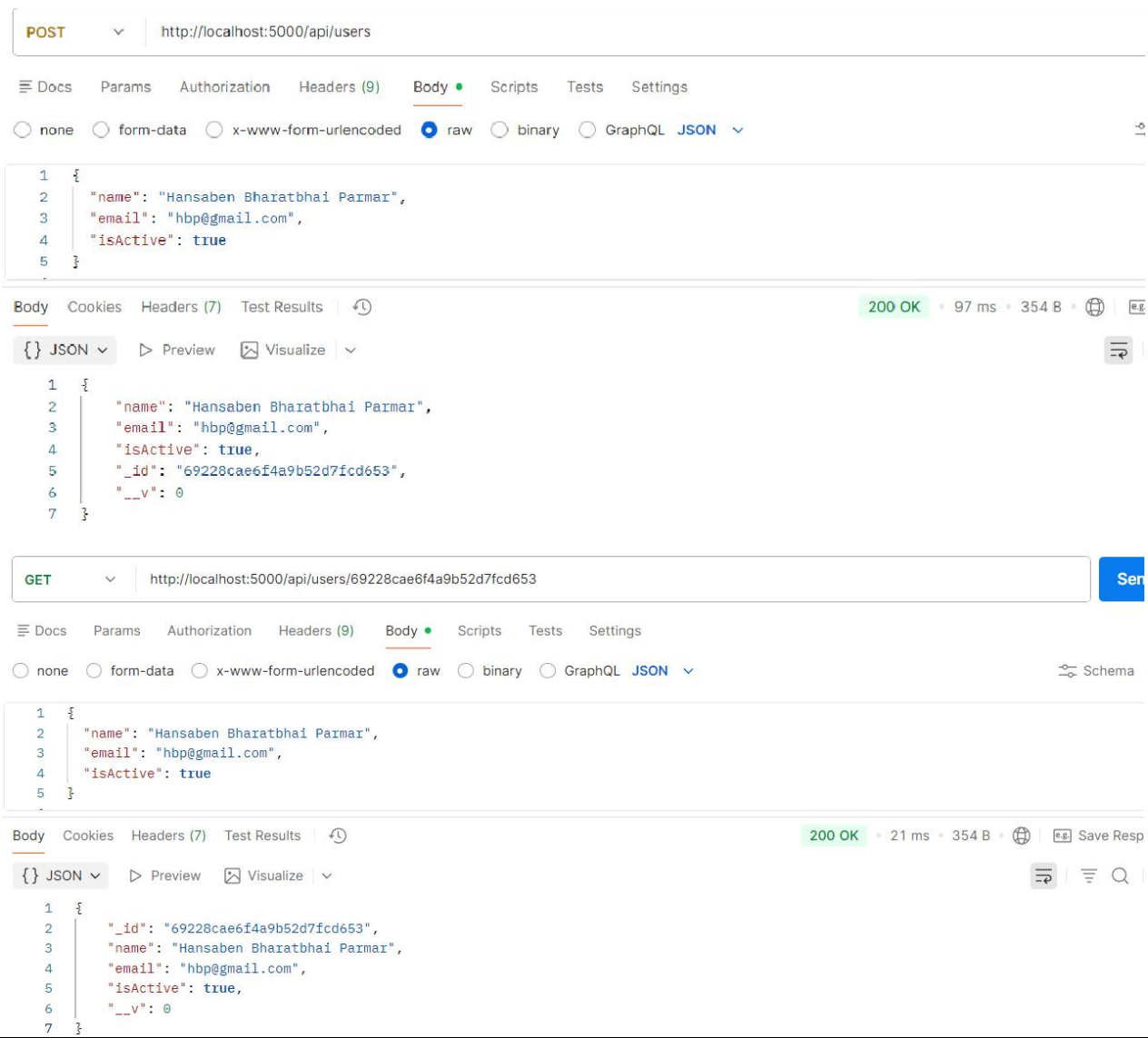
- **ConnecttoMongoDBusing MongooseinyourExpress.js application.**
- **CreateaUsermodelwithfields:name,email,andpassword.**

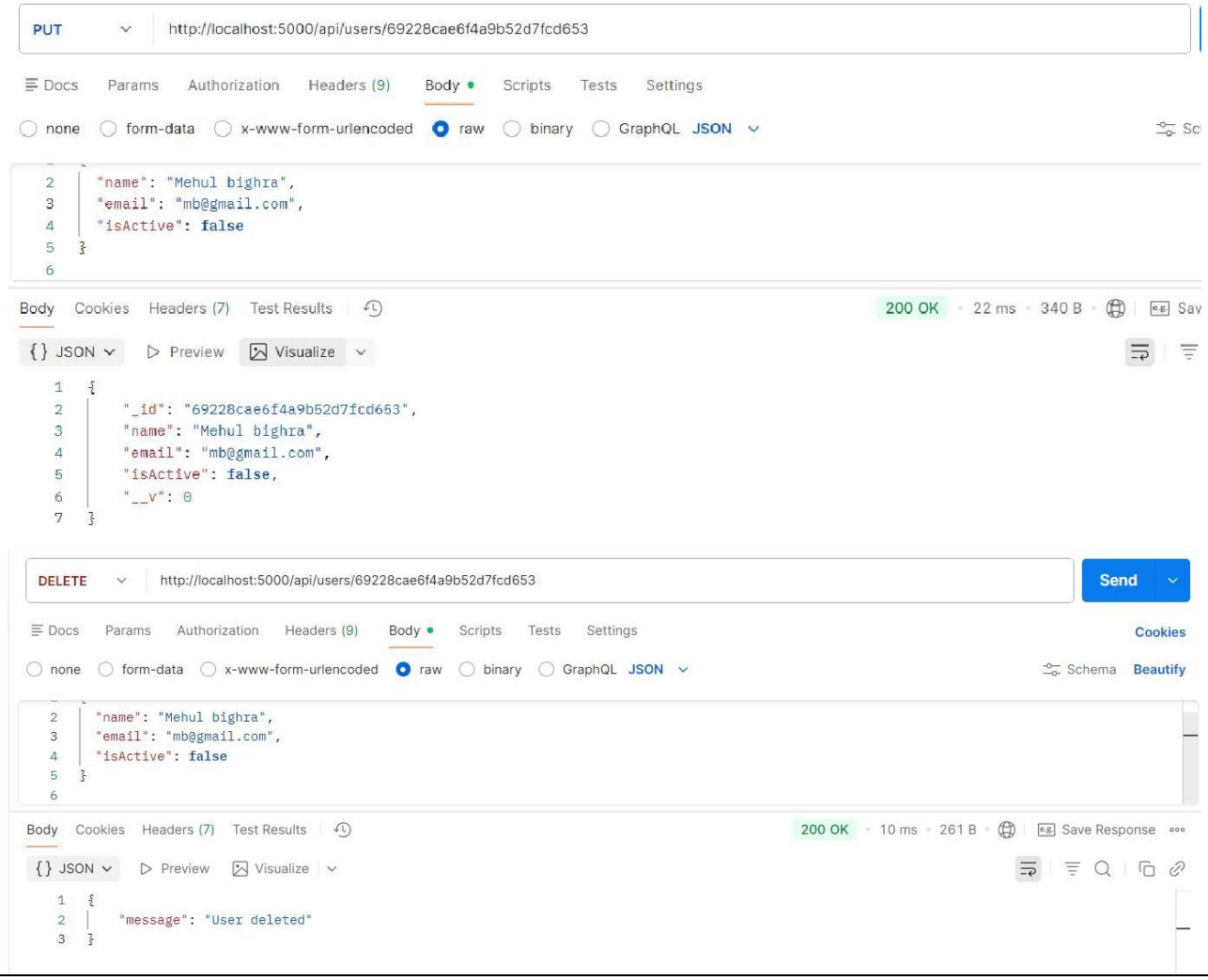
Code

```
// models/User.js
const mongoose = require("mongoose");
const userSchema = new mongoose.Schema({
name: String,
email: String,
isActive: { type: Boolean, default: true }
});
module.exports = mongoose.model("User", userSchema);
// controllers/userController.js
const User = require("../models/User");
exports.create = async (req,res)=> res.json(await User.create(req.body));
exports.getAll = async (req,res)=> res.json(await User.find());
exports.getById = async (req,res)=> res.json(await User.findById(req.params.id));
exports.update = async (req,res)=> {
const u = await User.findByIdAndUpdate(req.params.id, req.body, { new: true });
res.json(u);
};
exports.remove = async (req,res)=> {
await User.findByIdAndDelete(req.params.id);
res.json({ message: "User deleted" });
};
// routes/userRoutes.js
const express = require("express");
const { create, getAll, getById, update, remove } = require("../controllers/userController");
const router = express.Router();
router.post("/api/users", create);
router.get("/api/users", getAll);
```

```
router.get("/api/users/:id", getById);
router.put("/api/users/:id", update);
router.delete("/api/users/:id", remove);
module.exports = router;
// server.js
const express = require("express");
const mongoose = require("mongoose");
const userRoutes = require("./routes/userRoutes");
const app = express();
app.use(express.json());
mongoose.connect("mongodb://127.0.0.1:27017/practical5")
.then(()=>console.log("✓Q11 MongoDB connected"))
.catch(err=>console.log("✗, err.message));
app.use("/", userRoutes);
app.listen(5011, ()=>console.log("Q11 Server running on http://localhost:5011"));
```

o/p




PUT
http://localhost:5000/api/users/69228cae6f4a9b52d7fcd653
Docs Params Authorization Headers (9) Body Scripts Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
{
"name": "Mehul bighra",
"email": "mb@gmail.com",
"isActive": false
}
Body Cookies Headers (7) Test Results
200 OK 22 ms 340 B
JSON Preview Visualize
{
"_id": "69228cae6f4a9b52d7fcd653",
"name": "Mehul bighra",
"email": "mb@gmail.com",
"isActive": false,
"__v": 0
}
DELETE
http://localhost:5000/api/users/69228cae6f4a9b52d7fcd653
Send
Docs Params Authorization Headers (9) Body Scripts Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Schema Beautify
{
"name": "Mehul bighra",
"email": "mb@gmail.com",
"isActive": false
}
Body Cookies Headers (7) Test Results
200 OK 10 ms 261 B
Save Response
JSON Preview Visualize
{
"message": "User deleted"
}