Experiment 7

Source code:

```
const jwt = require('jsonwebtoken');
const JWT_SECRET = "yourSecretKey";

function auth(requiredRole) {
    return (req, res, next) => {
        const authHeader = req.headers.authorization;
        if (!authHeader || !authHeader.startsWith('Bearer ')) {
            return res.status(401).json({ success: false, message: "No token provided" });
        }

        const token = authHeader.split(' ')[1];
        try {
            const decoded = jwt.verify(token, JWT_SECRET);
            req.user = decoded;

            if (requiredRole && req.user.role !== requiredRole) {
                return res.status(403).json({ success: false, message: "Access denied" });
            }

            next();
        } catch (err) {
            res.status(401).json({ success: false, message: "Invalid token" });
        }
    };
}

module.exports = auth;
```

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
    name: { type: String, required: true, trim: true },
    email: {
        type: String,
        required: true,
        unique: true,
        lowercase: true,
        match: [/^\S+@\S+\.\S+$/, 'Please enter a valid email']
    },
    password: { type: String, required: true, minlength: 6 },
    role: { type: String, enum: ['user', 'admin'], default: 'user' }
}, { timestamps: true });

module.exports = mongoose.model('User', userSchema);
```

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/user');
const JWT_SECRET = "yourSecretKey";

// REGISTER
router.post('/register', async (req, res) => {
    try {
        const { name, email, password, role } = req.body;
        const hashedPassword = await bcrypt.hash(password, 10);
        const newUser = new User({ name, email, password: hashedPassword, role });
        await newUser.save();
        res.status(201).json({ success: true, message: 'User registered successfully' });
    } catch (err) {
        res.status(400).json({ success: false, message: err.message });
    }
});
```

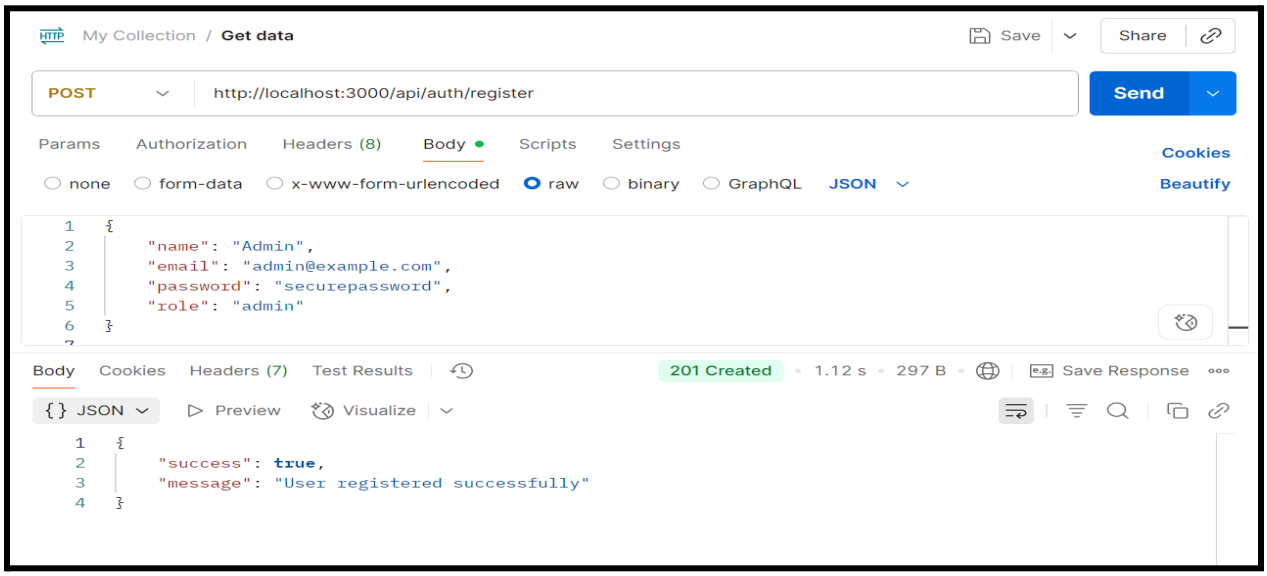
```
router.post('/login', async (req, res) => {
    try {
        const { email, password } = req.body;
        const user = await User.findOne({ email });
        if (!user) return res.status(400).json({ success: false, message: "Invalid credentials" });

        const isMatch = await bcrypt.compare(password, user.password);
        if (!isMatch) return res.status(400).json({ success: false, message: "Invalid credentials" });

        const token = jwt.sign({ id: user._id, role: user.role }, JWT_SECRET, { expiresIn: '1h' });
        res.json({ success: true, token });
    } catch (err) {
        res.status(500).json({ success: false, message: err.message });
    }
});

module.exports = router;
```

```
const express = require('express');
const router = express.Router();
const User = require('../models/user');
const auth = require('../middleware/auth');

// GET all users (admin only)
router.get('/', auth('admin'), async (req, res) => {
    try {
        const users = await User.find();
        res.json({ success: true, data: users });
    } catch (err) {
        res.status(500).json({ success: false, message: err.message });
    }
});

// DELETE user (admin only)
router.delete('/:id', auth('admin'), async (req, res) => {
    try {
        await User.findByIdAndDelete(req.params.id);
        res.json({ success: true, message: 'User deleted' });
    } catch (err) {
        res.status(500).json({ success: false, message: err.message });
    }
});
```

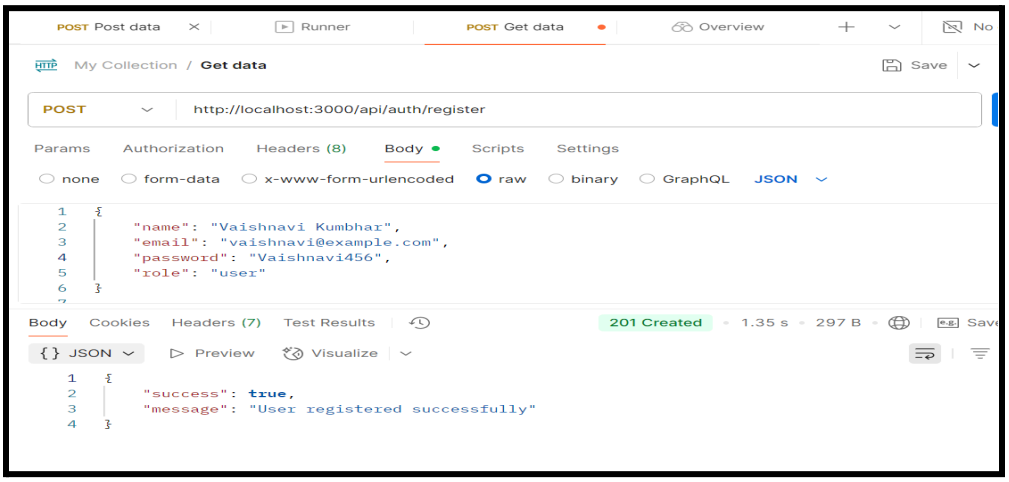
```
// UPDATE user (admin can update anyone, users only their own)
router.put('/:id', auth(), async (req, res) => {
    try {
        if (req.user.role !== 'admin' && req.user.id !== req.params.id) {
            return res.status(403).json({ success: false, message: "You can only update your own profile" });
        }
        const updatedUser = await User.findByIdAndUpdate(req.params.id, req.body, { new: true });
        res.json({ success: true, data: updatedUser });
    } catch (err) {
        res.status(500).json({ success: false, message: err.message });
    }
});
```

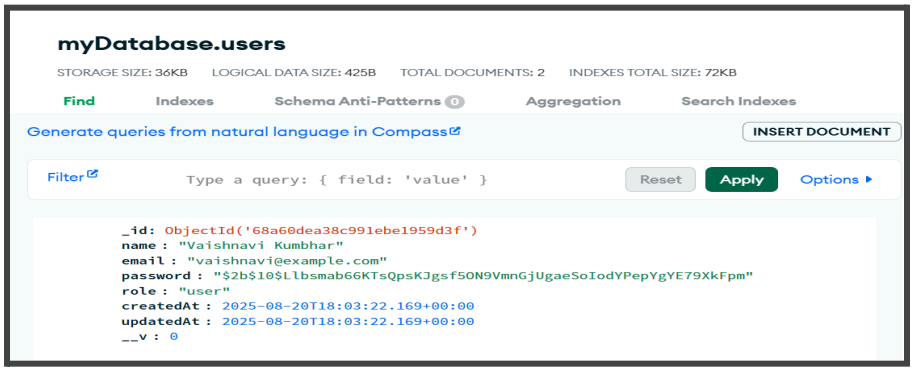
```
const express = require('express');
const mongoose = require('mongoose');
const app = express();
app.use(express.json());

// Connect to MongoDB
mongoose.connect("mongodb+srv://<username>:<password>@cluster0.mongodb.net/myDatabase", {
    useNewUrlParser: true,
    useUnifiedTopology: true
})
.then(() => console.log('✅ MongoDB Connected'))
.catch(err => console.error(err));

// Import Routes
const userRoutes = require('./routes/userRoutes');
const authRoutes = require('./routes/auth');

app.use('/api/users', userRoutes);
app.use('/api/auth', authRoutes);

// Start Server
app.listen(3000, () => {
    console.log('🚀 Server running on port 3000');
});
```

Output:



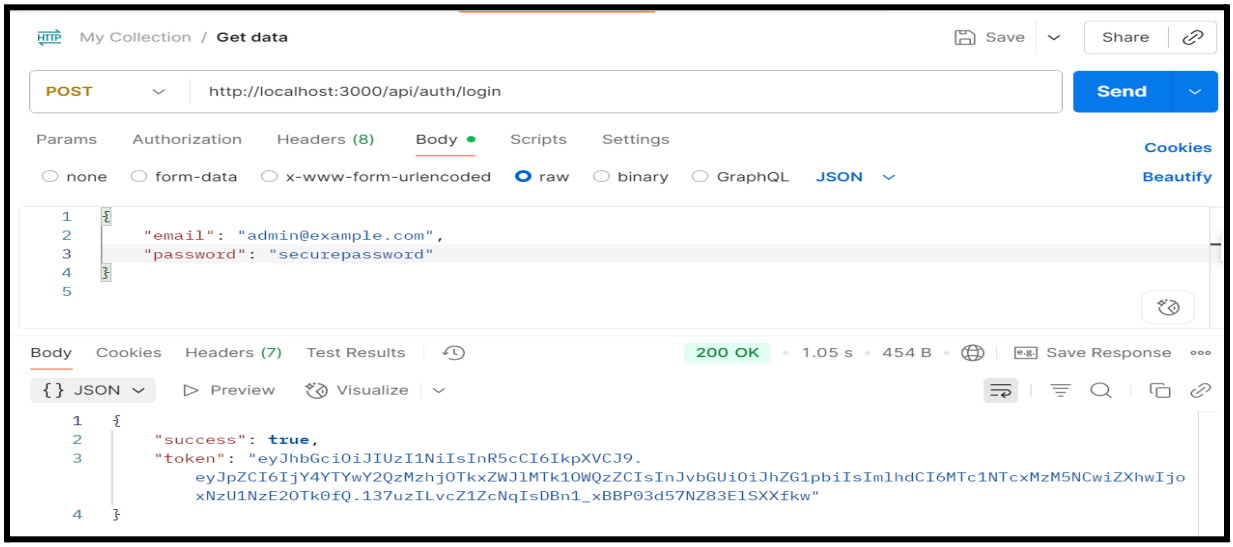
1.2. Register a Regular User



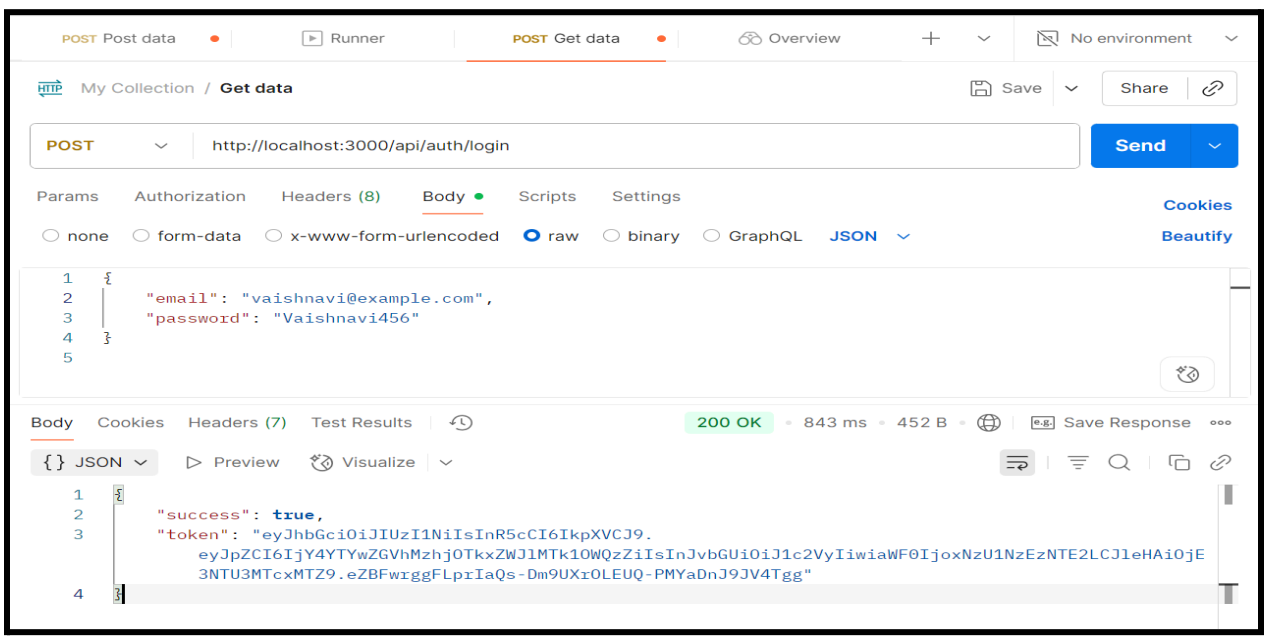


2. Authenticate and Read Operations

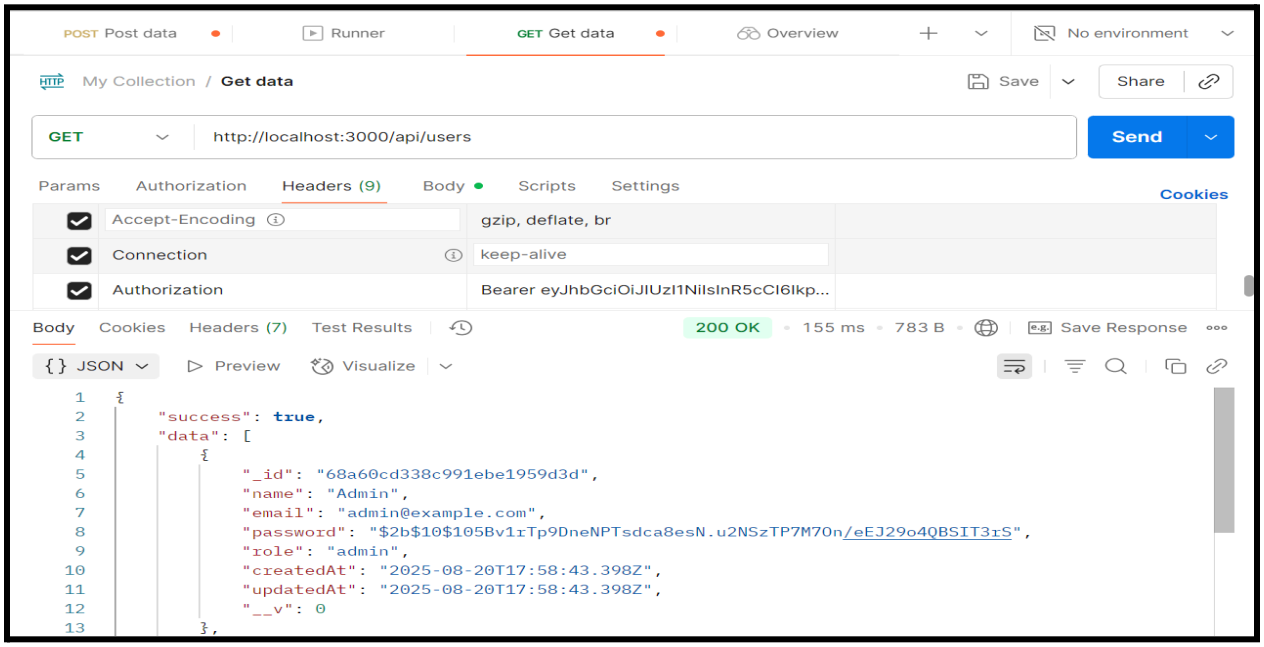
2.1. Log In to Obtain an Admin Token



2.2. Log In as a Regular User



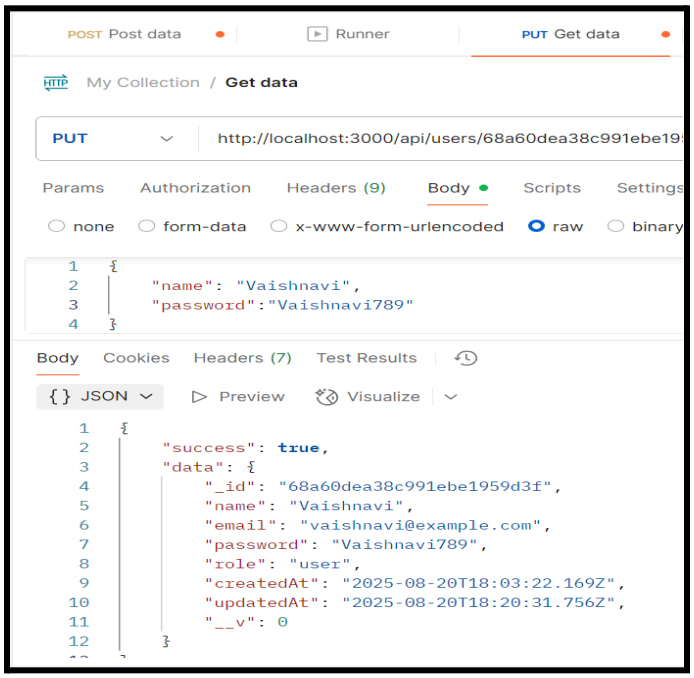
2.3. Get All Users



3. Update Operation

This part of the demonstration shows the ability to modify an existing user's data using the PUT method.

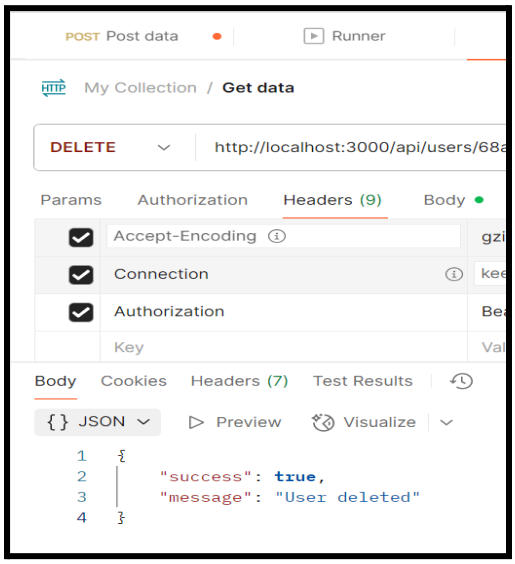
3.1. Update the Regular User's Name



4. Delete Operation

This final part demonstrates the deletion of a user from the database and verifies that the operation was successful.

4.1. Delete the Regular User



4.2. Verify Deletion