|  |  |
| --- | --- |
| **COURSE CODE: DJS22ITL604** | **DATE:** |
| **COURSE NAME: Full Stack Web Development Laboratory** | **CLASS: TYBTech** |
| **NAME: Anish Sharma** |  |

**Roll no: I011**

# EXPERIMENT NO. 09

**CO/LO:** CO1-Develop a full stack web application.

**AIM / OBJECTIVE:** Enhance APIs with middleware, validation, and security mechanisms such as rate limiting and CORS.

**THEORY**:

**Introduction to API Security & Middleware**

APIs (Application Programming Interfaces) play a crucial role in modern web applications, enabling communication between frontend and backend services. Ensuring API security is essential to protect data integrity, prevent unauthorized access, and maintain application performance. **Key Enhancements for Secure APIs**

1. **Middleware in Express.js**: Middleware functions process requests before they reach the route handlers, commonly used for logging, authentication, validation, and security.
2. **Validation**: Ensuring that incoming data follows predefined formats and constraints, reducing vulnerabilities like SQL injection and XSS attacks.
3. **Rate Limiting**: Restricting the number of API requests from a user within a timeframe to prevent abuse.
4. **CORS (Cross-Origin Resource Sharing)**: Controls access to resources from different origins, preventing unauthorized requests.
5. **Authentication & Authorization**: Securing APIs with JWT (JSON Web Tokens) and OAuth to restrict access.

**TECHNOLOGIES/PLATFORMS USED:**

* **Backend Framework:** Express.js (Node.js)
* **Security Packages:** Helmet, Express-rate-limit, CORS, JSON Web Token (JWT), Joi (for validation)
* **Database:** MongoDB (with Mongoose for schema validation)
* **Version Control:** Git & GitHub

**COURSE CODE: DJS22ITL604 DATE:**

**COURSE NAME: Full Stack Web Development Laboratory CLASS: TYBTech**

**STEP-BY-STEP IMPLEMENTATION:**

**Step 1: Setting Up Middleware in Express.js**

const express = require('express'); const helmet = require('helmet'); const cors = require('cors');

const rateLimit = require('express-rate-limit'); const app = express();

// Middleware for security headers app.use(helmet());

// Middleware for CORS app.use(cors());

// Rate Limiting const limiter = rateLimit({ windowMs: 15 \* 60 \* 1000, // 15 minutes max: 100, // Limit each IP to 100 requests per window message: "Too many requests from this IP, please try again later."

});

app.use(limiter);

// Middleware for JSON parsing app.use(express.json());

|  |  |
| --- | --- |
| **COURSE CODE: DJS22ITL604** | **DATE:** |
| **COURSE NAME: Full Stack Web Development Laboratory** | **CLASS: TYBTech** |

**Step 2: Data Validation using Joi**

const Joi = require('joi');

const userSchema = Joi.object({ name: Joi.string().min(3).required(), email: Joi.string().email().required(), password: Joi.string().min(6).required()

});

app.post('/register', (req, res) => { const { error } = userSchema.validate(req.body);

if (error) return res.status(400).json({ message: error.details[0].message }); res.status(201).json({ message: "User registered successfully" });

});

**Step 3: Implementing JWT Authentication** const jwt = require('jsonwebtoken');

const SECRET\_KEY = "your\_secret\_key";

app.post('/login', (req, res) => { const { email, password } = req.body;

if (email !== "test@example.com" || password !== "password") { return res.status(401).json({ message: "Invalid credentials" });

}

const token = jwt.sign({ email }, SECRET\_KEY, { expiresIn: '1h' });

|  |  |
| --- | --- |
| **COURSE CODE: DJS22ITL604** | **DATE:** |
| **COURSE NAME: Full Stack Web Development Laboratory** | **CLASS: TYBTech** |

res.json({ token });

});

const authenticateToken = (req, res, next) => { const token = req.header('Authorization');

if (!token) return res.status(401).json({ message: "Access Denied" });

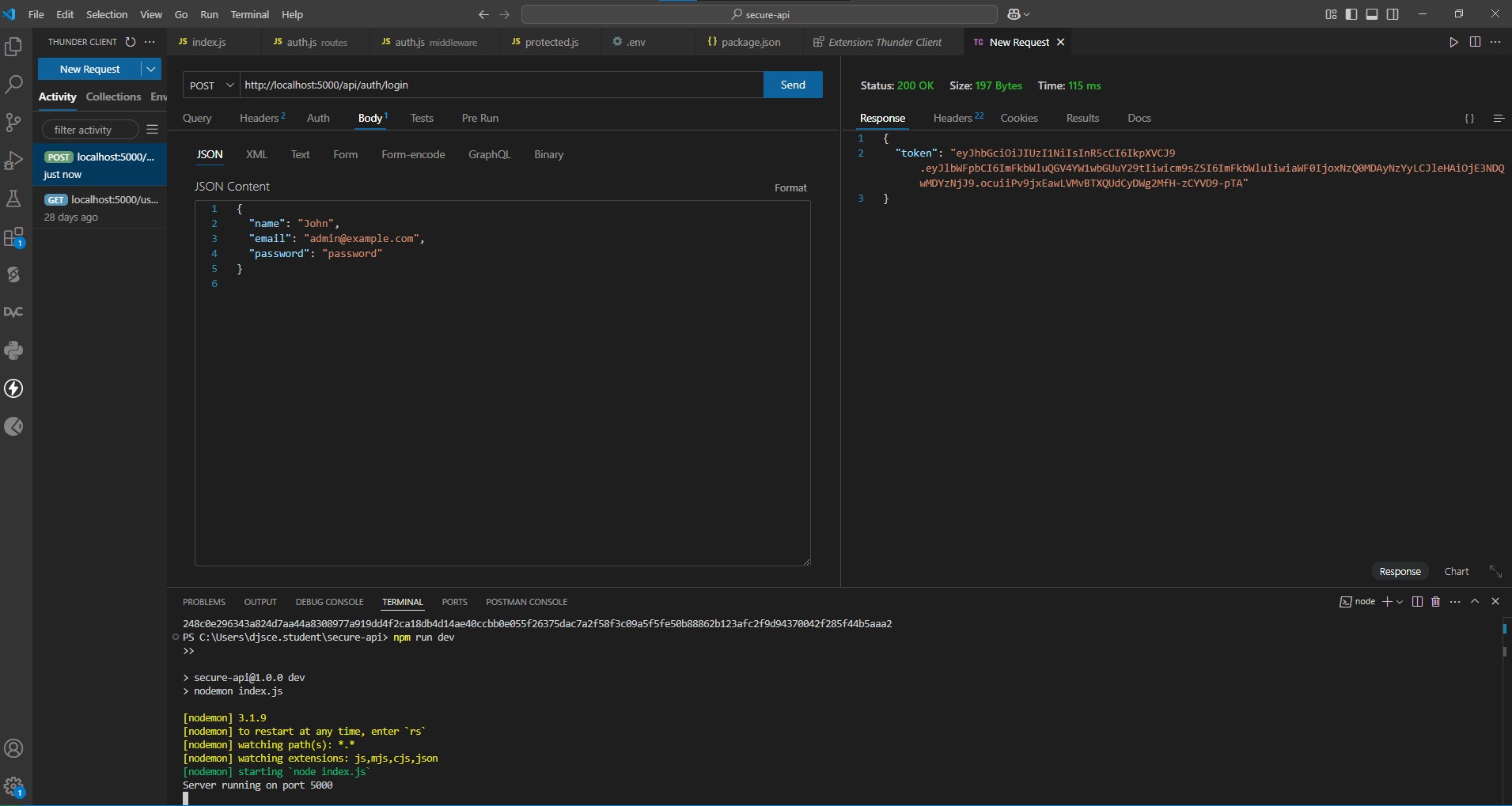
jwt.verify(token.split(" ")[1], SECRET\_KEY, (err, user) => { if (err) return res.status(403).json({ message: "Invalid Token" }); req.user = user; next();

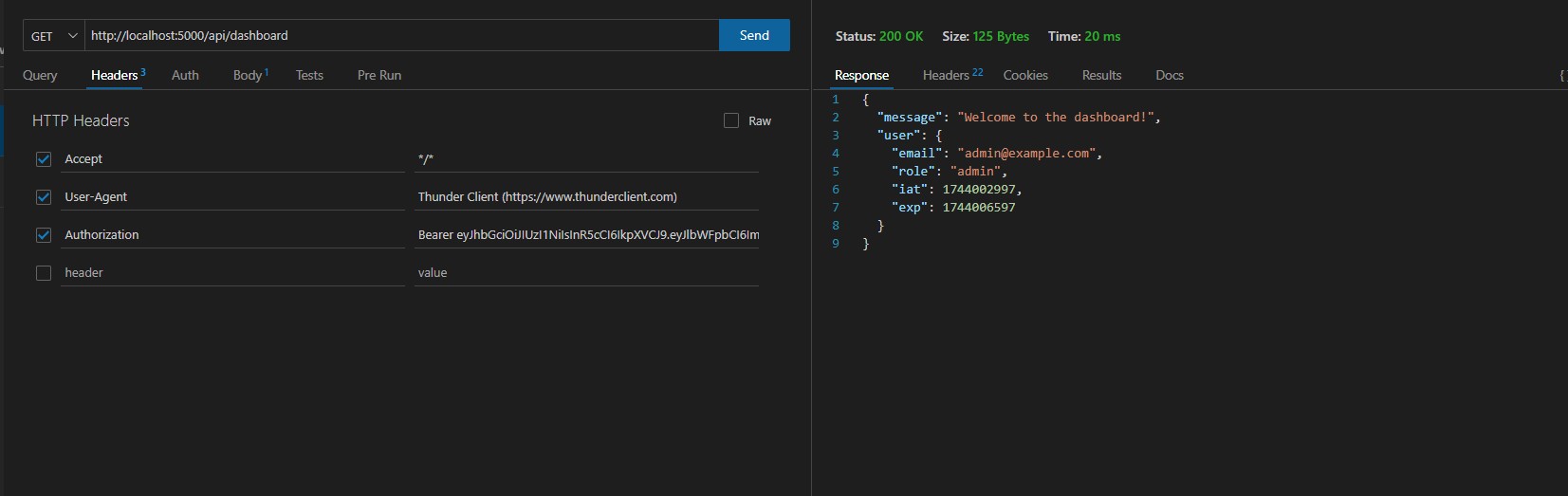
});

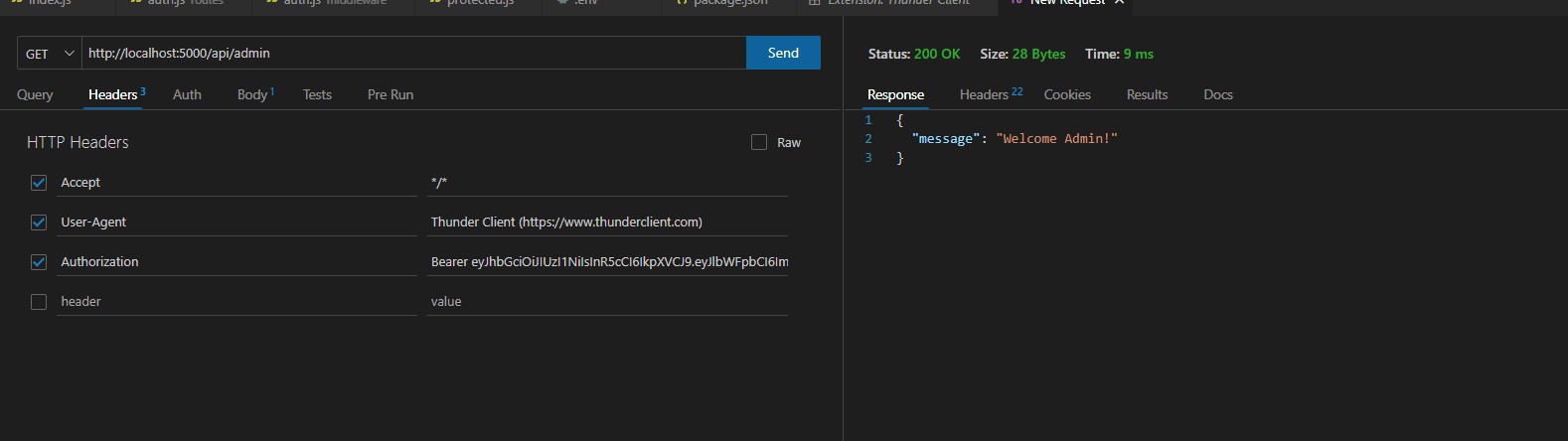
};

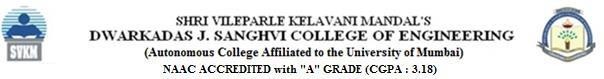
**Step 4: Protecting Routes with JWT Authentication** app.get('/dashboard', authenticateToken, (req, res) => { res.json({ message: "Welcome to the secure dashboard!", user: req.user });

});









**DEPARTMENT OF INFORMATION TECHNOLOGY**

**COURSE CODE: DJS22ITL604 DATE:**

**COURSE NAME: Full Stack Web Development Laboratory CLASS: TYBTech**

**BOOKS AND WEB RESOURCES:**

1. Express.js Documentation
2. Helmet Security Middleware
3. Express-rate-limit
4. CORS Documentation
5. JSON Web Tokens (JWT)
6. Joi Validation Library

**TASK:**

1. Implement Role-Based Access Control (RBAC): Restrict access to certain routes based on user roles (admin, user).
2. Protect API from excessive requests and enforce CORS policy.
   1. Implement rate limiting using express-rate-limit.
   2. Restrict API access to only allowed origins using cors.

1. Implement HTTP Security Headers
   1. Prevent security vulnerabilities using HTTP headers.
   2. Install helmet to set security headers:
   3. Apply Content-Security-Policy (CSP), X-XSS-Protection, and X-Frame-Options.

**WRITE-UP QUESTIONS:**

1. Explain the importance of rate limiting and how it prevents abuse.
2. What is CORS, and how does it impact API accessibility?
3. Why is validation important in API development, and how does Joi h

Academic Year 2024-25 SAP ID: