**Experiment Number: 7 Date: 17-03-2025**

# **Implement User Authentication in an Express Application**

## **PRE LAB EXERCISE**

### **Objective:**

* Understand authentication and authorization in web applications.
* Learn about **bcrypt** for password hashing and **JWT (JSON Web Token)** for authentication.
* Set up a basic Express.js application with **user registration and login**.

### **QUESTIONS:**

1. **What is authentication, and why is it important?**

Authentication is the process of verifying the identity of a user, system, or application before granting access to resources. It is crucial because it prevents unauthorized access, ensuring that only legitimate users can interact with sensitive data and functionalities.

1. **How does JWT help in securing web applications?**

JSON Web Token (JWT) is a compact, self-contained token used for securely transmitting information between parties. It helps in web security by:

* Enabling stateless authentication (no need to store session data on the server).
* Ensuring data integrity with digital signatures (HMAC or RSA).
* Reducing server-side overhead, making it scalable for distributed systems.

1. **What is the role of bcrypt in password security?**

Bcrypt is a cryptographic hashing function used for securely storing passwords. It enhances security by:

* Using a slow hashing algorithm to resist brute-force attacks.
* Incorporating salt to prevent precomputed attacks (rainbow tables).
* Supporting automatic work factor adjustments to strengthen security over time.

1. **What are the key differences between session-based and token-based authentication?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Session-Based Auth** | **Token-Based Auth (JWT)** |
| **Storage** | Server-side | Client-side |
| **Scalability** | Less scalable | Highly scalable |
| **State** | Stateful | Stateless |
| **Security Risks** | Session hijacking, CSRF | Token theft if exposed |
| **Best For** | Traditional web apps | APIs, SPAs, mobile apps |

1. **How does middleware help in authentication?**

Middleware acts as an intermediary that processes requests before they reach the application logic. In authentication, middleware:

* Verifies user credentials or tokens.
* Restricts access to protected routes.
* Handles redirections for unauthorized users.
* Enhances security by centralizing authentication logic.

## **IN LAB EXERCISE**

### **Objective:**

* Implement **user authentication** in an Express.js application.
* Use **bcrypt** for password hashing.
* Implement **JWT-based authentication** for secure access.

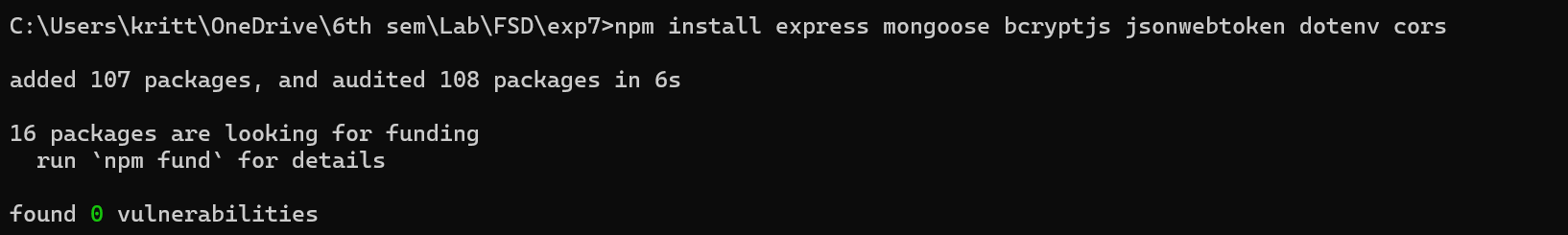
### **Resources Required:**

* **Node.js**, **Express.js**, **MongoDB**, **Postman**, **bcrypt**, **jsonwebtoken**.

### **Step 1: Install Required Packages**

npm init -y

npm install express mongoose bcryptjs jsonwebtoken dotenv cors



### **Step 2: Set Up Express and MongoDB**

**Create a .env file:**

A black screen with white text

AI-generated content may be incorrect.

### **Step 3: Create** server.js

A screen shot of a computer program

AI-generated content may be incorrect.

### **Step 4: Create Authentication Routes**

**Create routes/authRoutes.js**

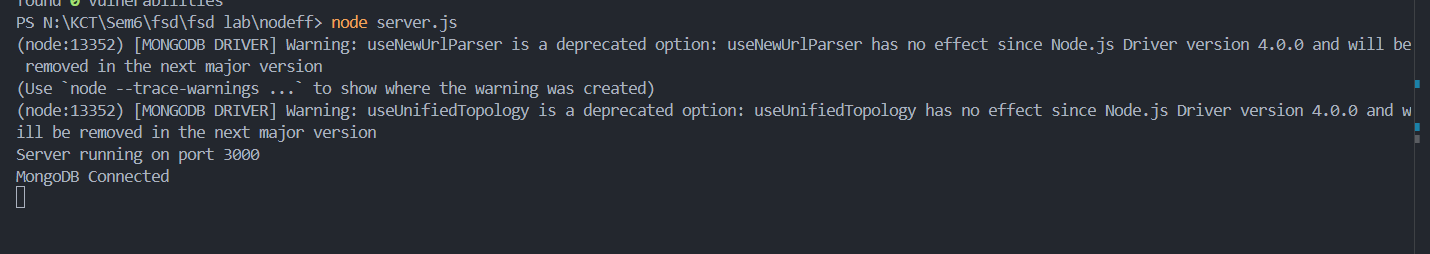
### **A screen shot of a computer program AI-generated content may be incorrect.**

### **Step 5: Create User Model**

**Create models/User.js**

A screen shot of a computer code

AI-generated content may be incorrect.

**Run command in terminal: node server.js**

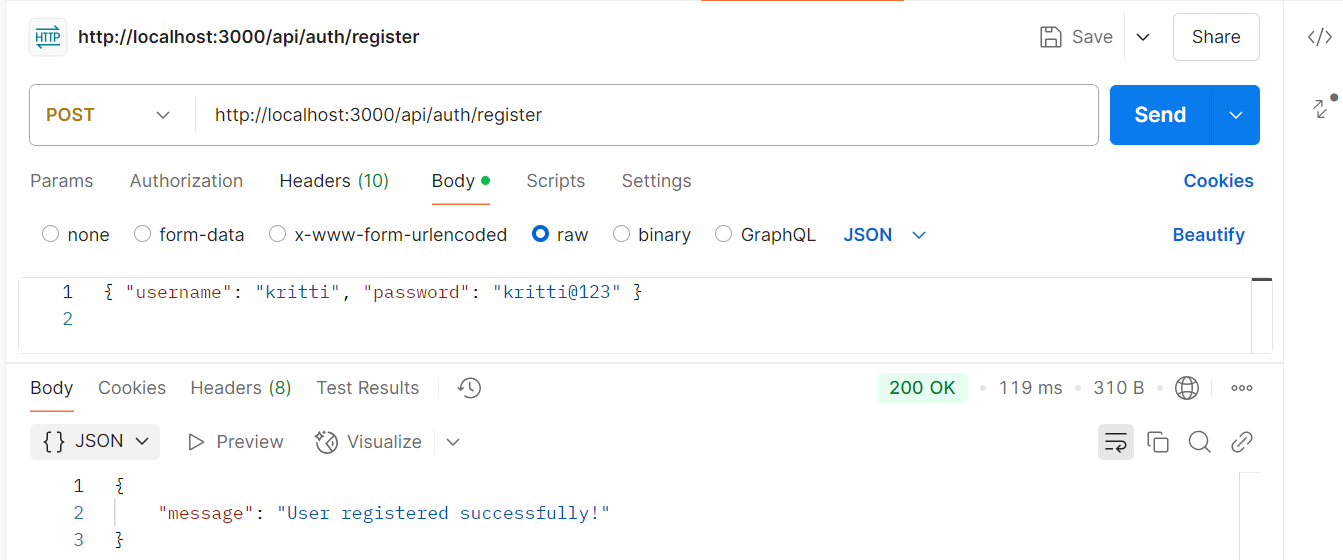
### **Step 6: Test with Postman**

1. **Register a user:**
   * URL: http://localhost:3000/api/auth/register
   * Method: POST

{ "username": "kritti", "password": "kritti@123" }

* + Expected Response:

{ "message": "User registered successfully!" }



1. **Login:**
   * URL: http://localhost:3000/api/auth/login
   * Method: POST
   * Body:

{ "username": "kritti", "password": "kritti@123" }

* + Expected Response:

{ "message": "Login successful", "token": "your\_jwt\_token" }

A screenshot of a computer

AI-generated content may be incorrect.

## **POST LAB EXERCISE**

### **QUESTIONS:**

1. **How does bcrypt ensure password security?**

bcrypt ensures password security by:

* **Hashing**: Converts passwords into a fixed-length cryptographic hash.
* **Salting**: Adds a random string (salt) to each password before hashing to prevent rainbow table attacks.
* **Key Stretching**: Uses multiple hashing rounds (default is 10) to make brute-force attacks computationally expensive.
* **Automatic Rehashing**: When system security needs improvement, bcrypt allows upgrading the hash without affecting existing passwords.

1. **What is the significance of jsonwebtoken in authentication?**

jsonwebtoken (JWT) is used for secure **stateless authentication** in web applications. Its significance includes:

* **Token-based authentication**: Users receive a token upon login, which is used for subsequent requests instead of sessions.
* **Statelessness**: No need to store session data on the server, improving scalability.
* **Integrity**: Uses digital signatures (HMAC or RSA) to prevent tampering.
* **Role Management**: Encodes user roles/permissions within the token payload.

1. **How can we implement role-based authentication in an Express app?**

**models/Muser.js**:

A screen shot of a computer code

AI-generated content may be incorrect.

**routes/mauthRoutes.js – modify the register to get role from the req**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**middleware/roleMiddleware.js – for role based access control**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**routes/protectedRoutes.js**

**A screen shot of a computer code

AI-generated content may be incorrect.**

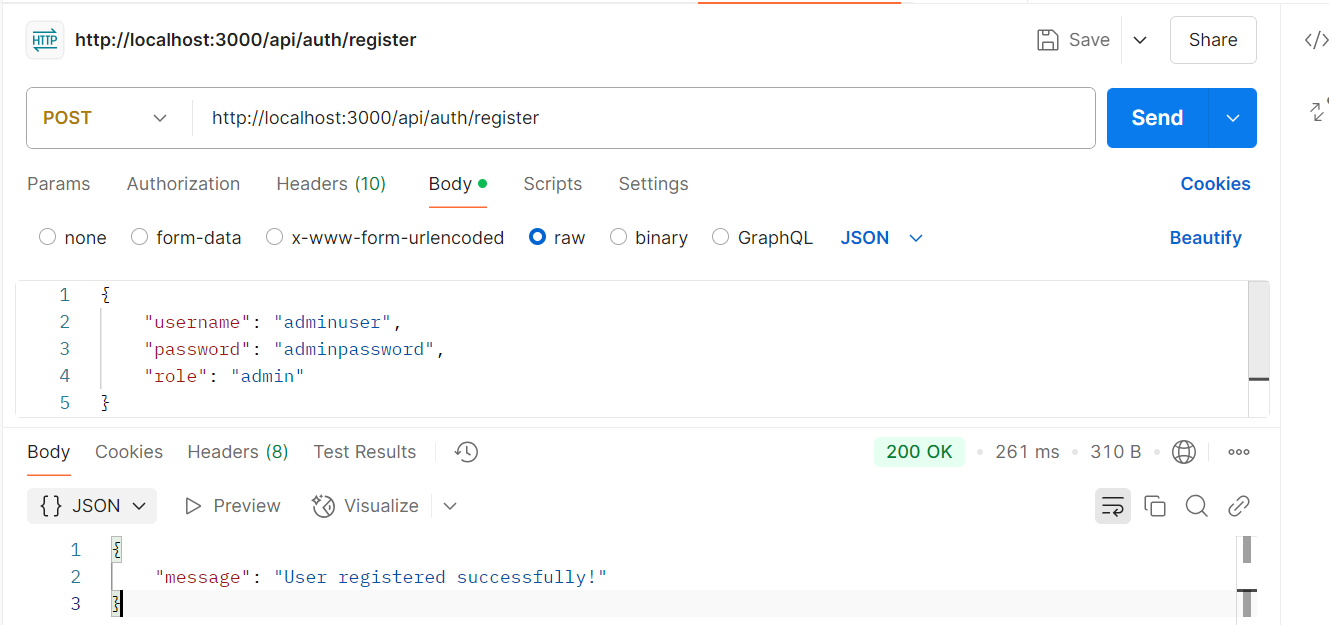
**Updated server.js**

**A screen shot of a computer code

AI-generated content may be incorrect.**

**Testing using postman :**

**Registering an admin user**



**Registering a normal user:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Login an admin user**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Login a normal user**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **How would you store JWT tokens securely on the client side**

* **HTTP-only Cookies** (Most Secure)
  + Stored in an HTTP-only cookie to prevent XSS attacks.
  + Example:

res.cookie("token", jwtToken, {

    httpOnly: true,

    secure: true, // Use only in HTTPS

    sameSite: "Strict",

});

* **Local Storage** (Less Secure, vulnerable to XSS)
  + Only use if implementing additional security (e.g., rotating tokens).
* **Session Storage** (Temporary, slightly better than Local Storage)
* **Token Rotation**: Use refresh tokens for enhanced security.

1. **What additional security measures should be taken in authentication systems?**

* Use HTTPS to encrypt data in transit.
* Enable 2FA (Two-Factor Authentication) for additional security.
* Rate Limiting & Brute-force Protection using tools like express-rate-limit.
* Secure Password Storage with bcrypt (minimum 10 rounds).
* Prevent Token Theft by implementing short expiration times and refresh tokens.
* Use CSRF Protection if storing JWTs in cookies (csurf middleware).
* Monitor for Suspicious Activity and log failed login attempts.

**ASSESSMENT PATTERN.**

|  |  |  |
| --- | --- | --- |
| **Description** | **Max Marks** | **Marks Awarded** |
| Pre Lab Exercise | **5** |  |
| In Lab Exercise | **10** |  |
| Post Lab Exercise | **5** |  |
| Viva | **10** |  |
| **Total** | **30** |  |
| **Faculty Signature** | |  |