

**AUTHENTICATION IN NODE.JS**

**By**

**Anshul Ojha Sir**

**Web Authentication BCrypt Technical Report**

**On**

**Node.js Authentication API**

Submitted in partial fulfillment of the requirements for the Boot camp Certification

**Bachelor of Technology**

**In**

**Computer Science and Engineering – Data Science**

**Of**

**Garden City University**

By

Gangothri Devi M

**Under the guidance of Anshul Ojha Sir(Bootcamp mentor).**

**1. Project Title**

Node.js Authentication API using bcrypt and JWT

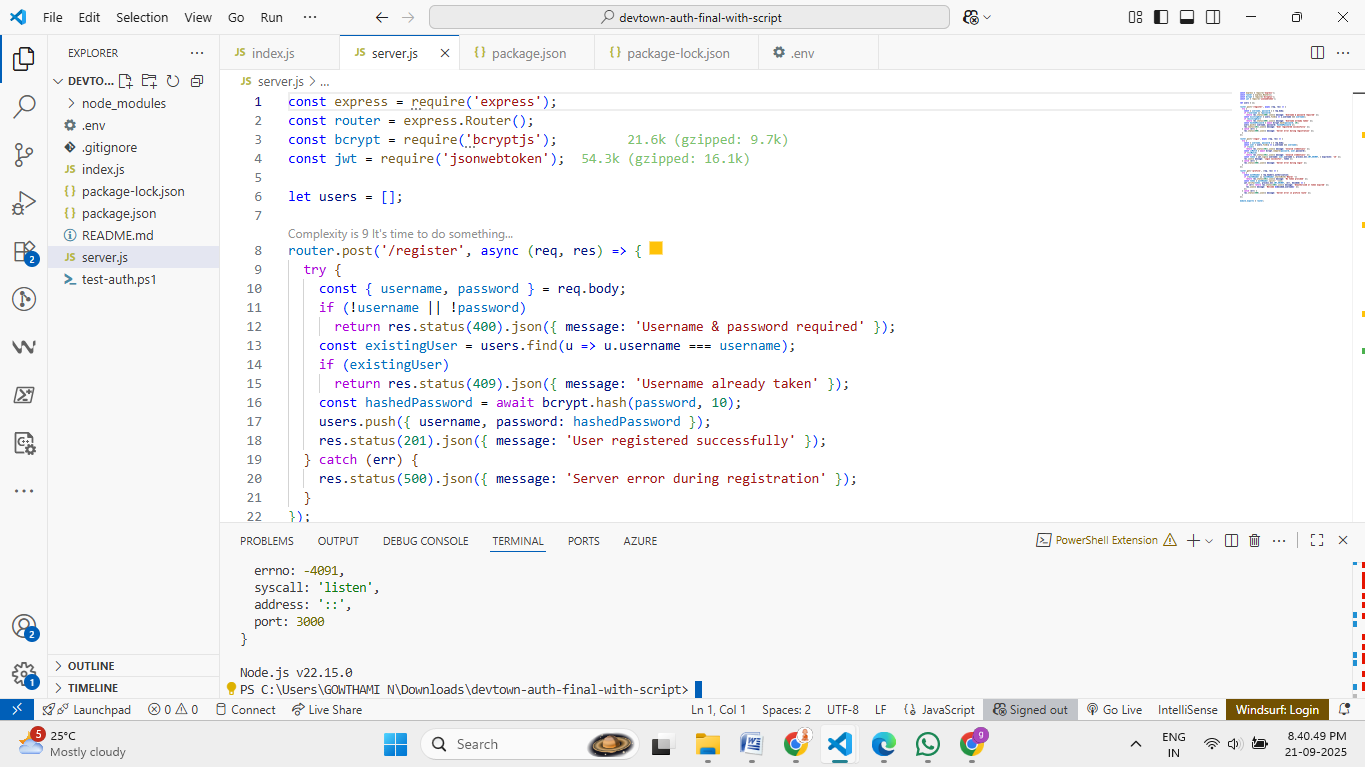
**2. Objective**

The goal of this project is to design and implement a **secure user authentication system** in Node.js. This project demonstrates the ability to:

* Implement **password hashing** with bcrypt to store passwords securely.
* Implement **user login** with password verification.
* Generate and validate **JWT tokens** for secure access to protected routes.
* Structure a Node.js project for scalability and modularity.
* Handle errors gracefully and ensure clean JSON responses.

**3. Project Structure**

The project is organized as follows:



your-project/

│

├── index.js # Entry point (starts server)

├── server.js # Auth routes (register, login, profile)

├── package.json # Dependencies and start script

├── .env # Contains PORT & JWT\_SECRET

├── .gitignore # Ignore node\_modules & .env

└── README.md # Documentation & usage instructions

**Explanation of each file:**

* **index.js:** Starts the Node.js server on the specified port. Loads environment variables and middleware.
* **server.js:** Contains all **authentication routes**:
  + /register → Register new users with hashed passwords.
  + /login → Login users and generate JWT token.
  + /profile → Protected route accessible only with valid JWT token.
* **package.json:** Manages dependencies (express, bcrypt, jsonwebtoken, body-parser) and start script.
* **.env:** Stores environment variables like PORT and JWT\_SECRET securely.
* **.gitignore:** Ensures sensitive files like node\_modules and .env are ignored by Git.
* **README.md:** Provides project overview, installation instructions, and usage.

**4. Technologies Used**

| **Category** | **Technologies / Tools** |
| --- | --- |
| Backend Framework | Node.js |
| Authentication & Security | bcrypt.js, JWT, dotenv |
| Middleware | body-parser |
| Development Tools | Git, Nodemon |
| Deployment | Optional (Render / Railway) |

**5. Core Features Implemented**

**5.1 User Registration**

* Accepts username and password.
* Password is **hashed using bcrypt** before storing.
* User data stored in **in-memory array** for simplicity.

**5.2 User Login**

* Accepts username and password.
* Password verified using **bcrypt.compare()**.
* Returns a **JWT token** if credentials are valid.

**5.3 Protected Route**

* /profile route accessible **only with valid JWT token**.
* Returns "Welcome <username>" on success or "Unauthorized" if token is missing/invalid.

**5.4 Error Handling**

* Invalid credentials return "Invalid credentials".
* Unauthorized access blocked.
* Input validation handled before processing requests.

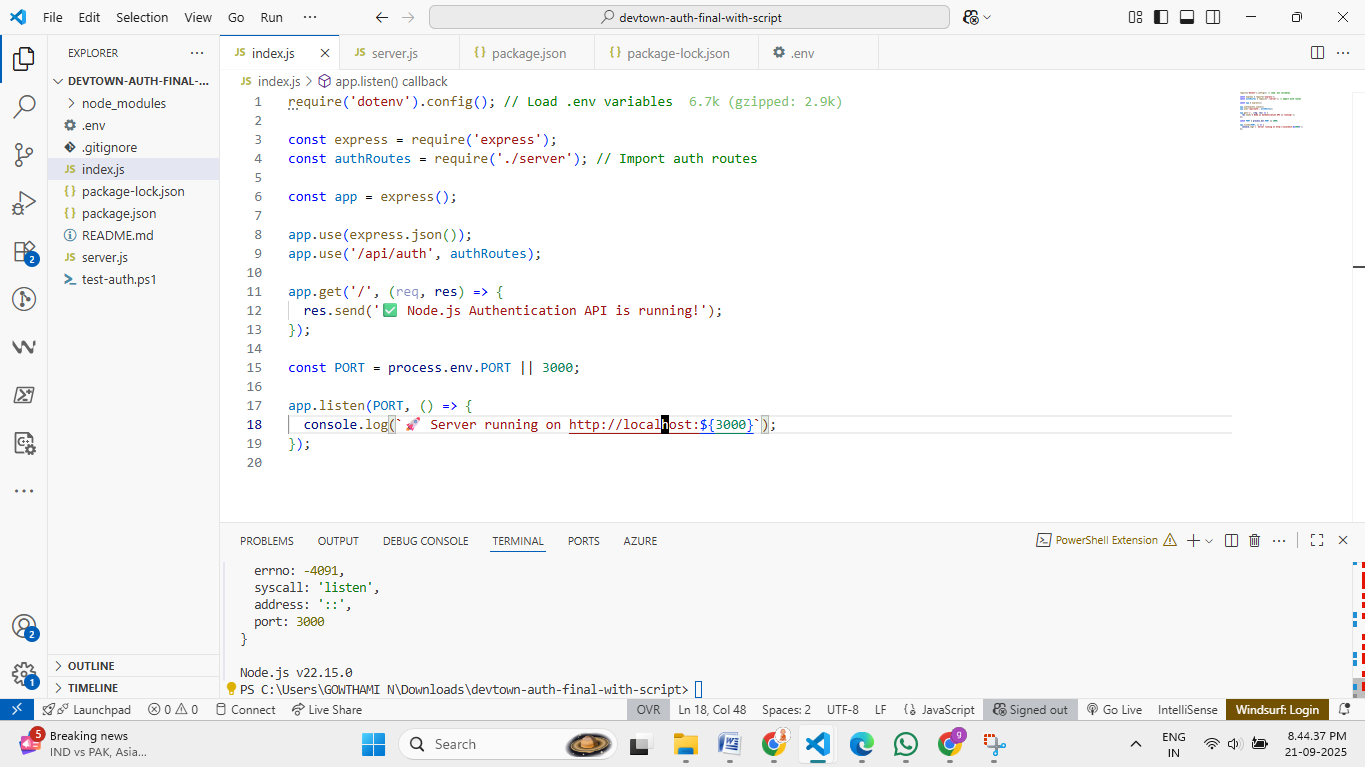
**6. Architecture & Workflow**

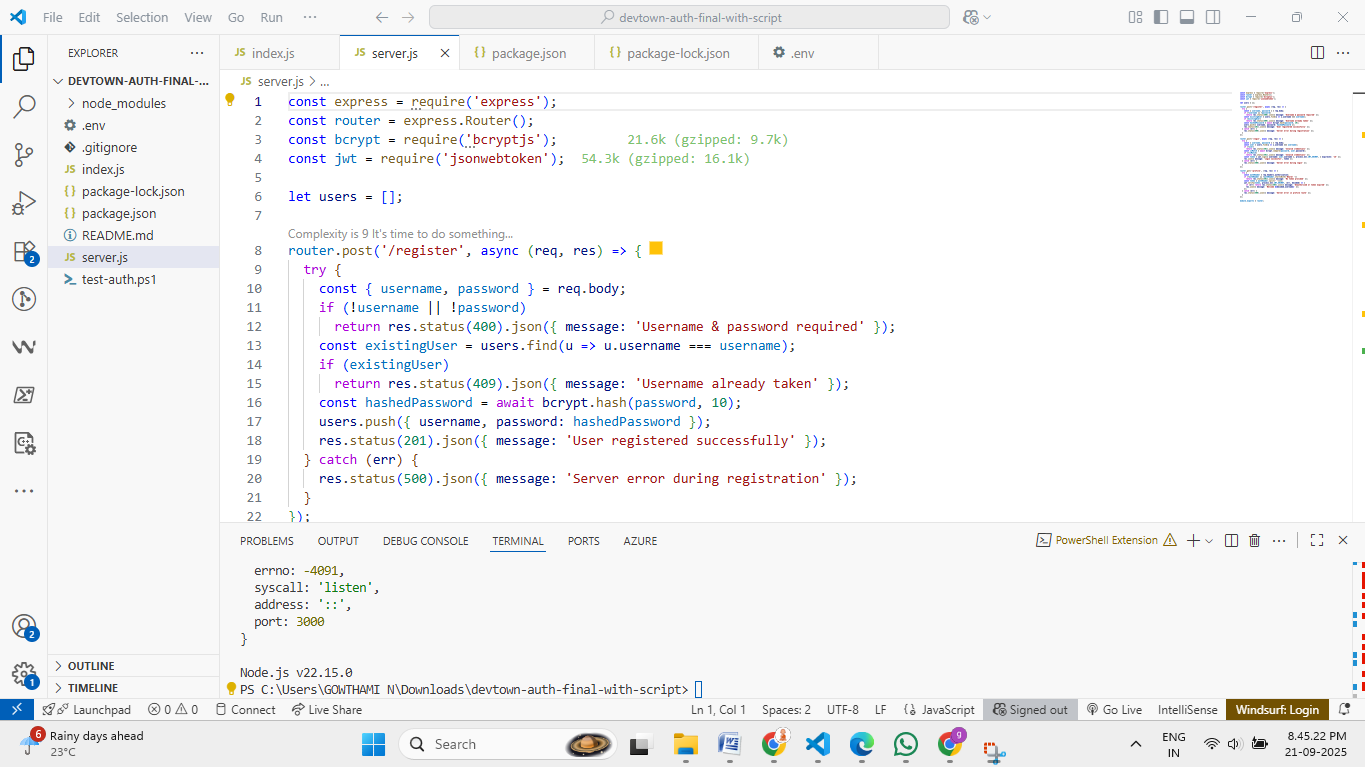
1. Client sends HTTP requests to the server.
2. index.js initializes the server and loads middleware.
3. server.js defines all authentication routes and handles logic.
4. Middleware parses JSON input and manages JWT authentication.
5. In-memory array stores users with hashed passwords.
6. JWT tokens secure protected routes.
7. Responses are sent back in JSON format.

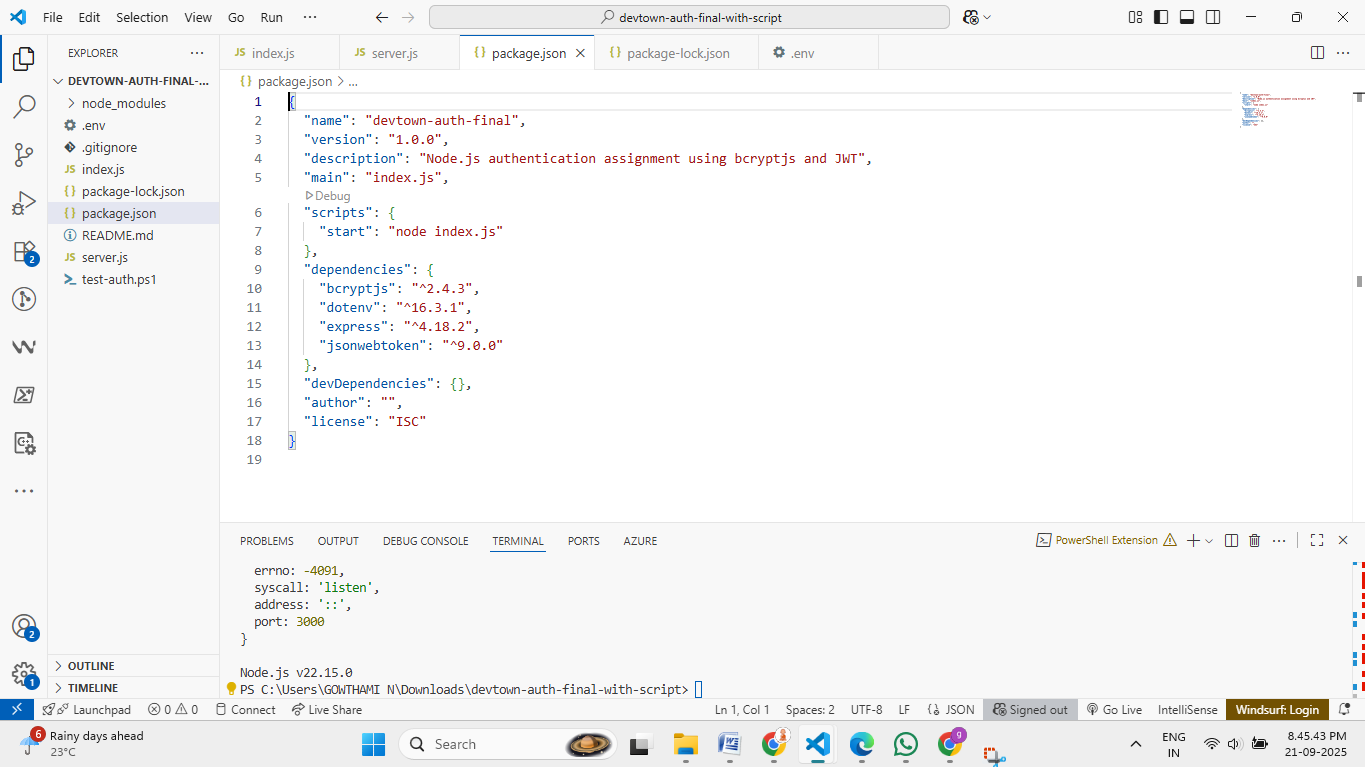
**7. API Endpoints**

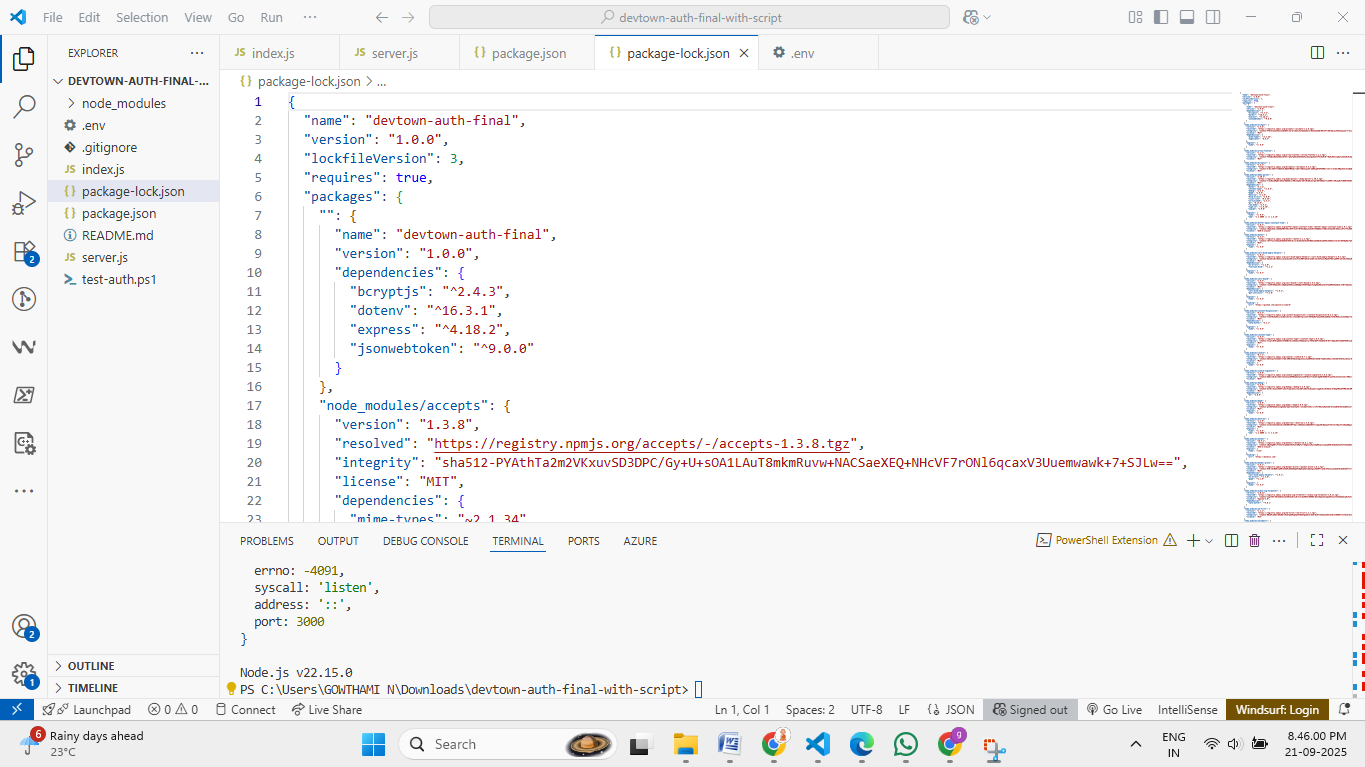
| **Route** | **Method** | **Description** |
| --- | --- | --- |
| register | POST | Register a new user with hashed password. |
| login | POST | Login user and generate JWT token. |
| profile | GET | Protected route; requires valid JWT token. |

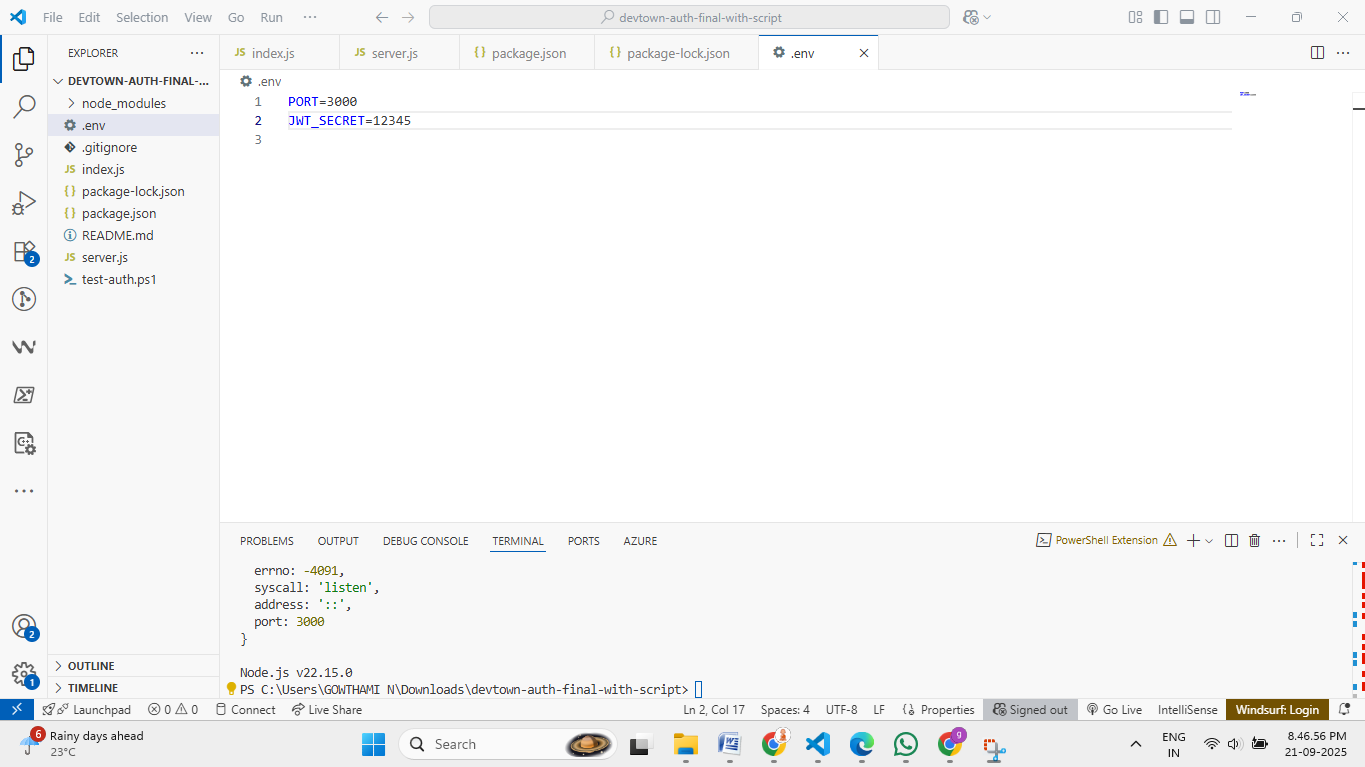
**8. Sample Code Snippets**



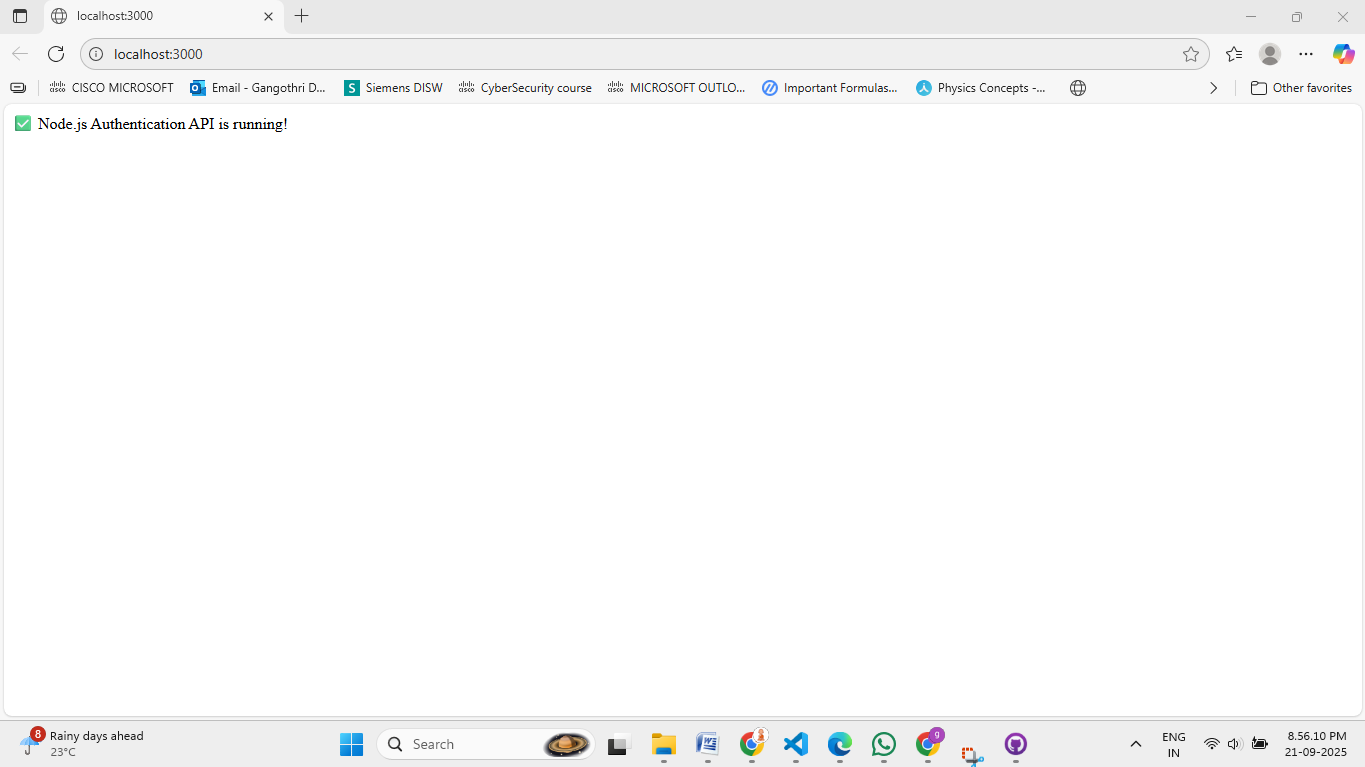








**OUTPUT:**



**9. Deployment Steps (Optional)**

1. Push project to **GitHub repository**.
2. Configure **Render / Railway deployment**.
3. Add environment variables:
   * PORT → Server port number.
   * JWT\_SECRET → Secret key for JWT token signing.
4. Set start command: npm start.
5. Test endpoints using curl or any frontend client.

**10. Testing & Validation**

* Tested **user registration, login, and JWT authentication** using curl or custom clients.
* Verified:
  + Correct registration and password hashing.
  + Successful JWT generation.
  + Protected route accessible only with valid token.
  + Unauthorized access blocked properly.

**11. Challenges & Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Password Security | Used bcrypt hashing with salt. |
| Token Validation | Implemented JWT verification for protected routes. |
| Error Handling | Returned clean JSON error messages for invalid input. |
| Project Structure | Separated index.js and server.js for clarity. |

**12. Conclusion**

This project demonstrates a **secure and modular Node.js authentication system**. It uses **bcrypt for password hashing** and **JWT for secure route protection**, with a clean, maintainable structure ready for scaling or deployment.

Github Link: