### **Node.js Task Management API Documentation**

#### **Base URL**

***For starting the API application:***

1. Navigate to the source directory:  
   bash

cd /assignment/src

npm start

***For starting the test application:***

1. Navigate to the tests directory:  
   bash

cd /assignment/tests

npm test

**Postman documentation URL**: [Postman Documentation](https://documenter.getpostman.com/view/21723795/2sA3QqhYhe)

**Default Application URL**:

https://localhost:5000

To specify a different port, change the PORT value in the .env file:

PORT=5000

### **Authentication**

All endpoints require authentication using a JWT token. The token must be included in the Authorization header as a Bearer token.

### **Endpoints**

#### **Authentication**

**Register**

* **URL**: /auth/register
* **Method**: POST
* **Description**: Registers a new user.
* **Request Body**:

{

"email": "user@example.com",

"password": "yourpassword"

}

**Responses**:

* 201 Created: Returns the JWT token.  
  {

"token": "your.jwt.token"

}

* 400 Bad Request: Validation error or email already used.

{

"message": "Validation error message or Email address is already used."

}

**Login**

* **URL**: /auth/login
* **Method**: POST
* **Description**: Logs in an existing user.
* **Request Body**:  
  {

"email": "user@example.com",

"password": "yourpassword"

}

**Responses**:

* 200 OK: Returns the JWT token.

{

"token": "your.jwt.token"

}

* 400 Bad Request: Invalid credentials.

{

"message": "Invalid credentials"

}

#### **Tasks**

**Create Task**

* **URL**: /task
* **Method**: POST
* **Description**: Creates a new task.
* **Headers**: Authorization: Bearer your.jwt.token
* **Request Body**:  
  {

"title": "Task Title",

"description": "Task Description"

}

**Responses**:

* 201 Created: Returns the created task.

{

"id": "task\_id",

"title": "Task Title",

"description": "Task Description",

"userId": "user\_id",

"createdDate": "2023-01-01T00:00:00Z"

}

* 400 Bad Request: Validation error.

{

"message": "Validation error message"

}

**Get All Tasks**

* **URL**: /task
* **Method**: GET
* **Description**: Retrieves all tasks for the authenticated user.
* **Headers**: Authorization: Bearer your.jwt.token
* **Responses**:
  + 200 OK: Returns an array of tasks.  
    [

{

"id": "task\_id",

"title": "Task Title",

"description": "Task Description",

"userId": "user\_id",

"createdDate": "2023-01-01T00:00:00Z"

},

...

]

**Get Task by ID**

* **URL**: /task/:id
* **Method**: GET
* **Description**: Retrieves a task by its ID.
* **Headers**: Authorization: Bearer your.jwt.token
* **Responses**:
  + 200 OK: Returns the task.  
    {

"id": "task\_id",

"title": "Task Title",

"description": "Task Description",

"userId": "user\_id",

"createdDate": "2023-01-01T00:00:00Z"

}

* 404 Not Found: Task not found.  
  {

"message": "Task not found"

}

**Update Task**

* **URL**: /task/:id
* **Method**: PUT
* **Description**: Updates a task by its ID.
* **Headers**: Authorization: Bearer your.jwt.token
* **Request Body**:  
  {

"title": "Updated Task Title",

"description": "Updated Task Description"

}

**Responses**:

* 200 OK: Returns the updated task.-

{

"id": "task\_id",

"title": "Updated Task Title",

"description": "Updated Task Description",

"userId": "user\_id",

"createdDate": "2023-01-01T00:00:00Z"

}

* 400 Bad Request: Validation error.

{

"message": "Validation error message"

}

**Delete Task**

* **URL**: /task/:id
* **Method**: DELETE
* **Description**: Deletes a task by its ID.
* **Headers**: Authorization: Bearer your.jwt.token
* **Responses**:
  + 204 No Content: Task deleted successfully.
  + 404 Not Found: Task not found.  
    {

"message": "Task not found"

}

### **Validation**

* **Email**: Must be a valid email address.
* **Password**: Must be between 10 and 50 characters.
* **Title**: Required field, non-empty string.
* **Description**: Optional field, non-empty string.

### **Error Handling**

All errors are returned in the following format:

{

"message": "Error message"

}

### **Security Considerations**

* **SQL Injection**: Use ORM/ODM like Mongoose to interact with the database to prevent SQL injection.
* **Cross-Site Scripting (XSS)**: Ensure data is properly sanitized and validated.
* **JWT Security**: Use a strong secret for JWT and set appropriate expiration times.
* **Rate Limiting**: Implement rate limiting to prevent brute force attacks.

### **Example Usage**

**Register a User**

bash

curl -X POST http://localhost:5000/api/auth/register \

-H "Content-Type: application/json" \

-d '{"email":"user@example.com", "password":"yourpassword"}'

**Login a User**

bash

curl -X POST http://localhost:5000/api/auth/login \

-H "Content-Type: application/json" \

-d '{"email":"user@example.com", "password":"yourpassword"}'

**Create a Task**

bash

curl -X POST http://localhost:5000/api/task \

-H "Authorization: Bearer your.jwt.token" \

-H "Content-Type: application/json" \

-d '{"title":"Task Title", "description":"Task Description"}'

This documentation provides an overview of the endpoints, request and response formats, validation rules, and security considerations for your Node.js Task Management API.