

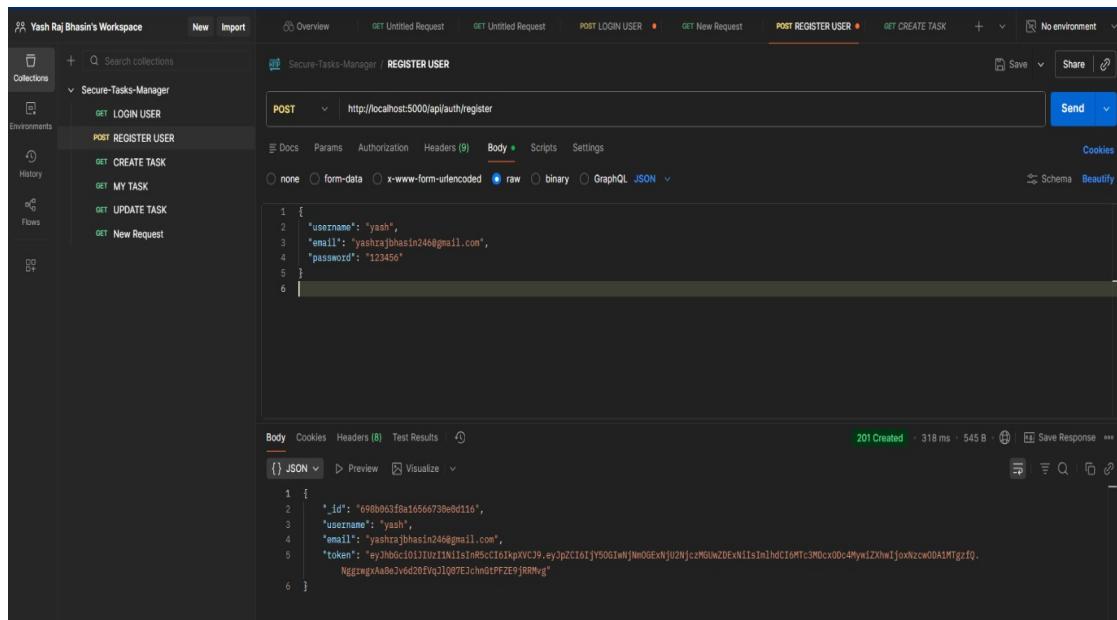
SECURE TASKS MANAGER

POSTMAN API DOCUMENTATION

Prepared By: Yash Raj Bhasin

This document presents a detailed demonstration of the Secure Tasks Manager REST API using Postman. Each section includes request-response outputs along with clear explanations to showcase authentication, authorization, and complete task management functionality implemented using Node.js, Express, and MongoDB.

REGISTER USER API



The screenshot shows the Postman interface with the 'REGISTER USER' API endpoint selected. The request method is POST, the URL is `http://localhost:5000/api/auth/register`, and the body is set to raw JSON with the following content:

```
1 {
2   "username": "yash",
3   "email": "yashrajbhasin24@gmail.com",
4   "password": "123456"
5 }
```

The response tab shows a successful `201 Created` status with a response time of 318 ms and a response size of 545 B. The response body is a JSON object containing user details and a JWT token:

```
1 {
2   "_id": "698b063f8a16566738e0d116",
3   "username": "yash",
4   "email": "yashrajbhasin24@gmail.com",
5   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC16IjY5OGlwIiJmOGEwNjU2NjczMGUwZDExNjIsImhdCI6MTc3MDcxOjc4MywiZXhwIjoxNzcwODA1MTgzeQ.NggzwgxAu8eJvd20FVqJlQ07EJchmGtFZE9jRRMvg"
```

This API endpoint is responsible for registering a new user into the system. The user provides a username, email, and password in the request body. Upon successful registration, the system stores the user securely in the database and returns the user details along with a JWT authentication token. This token is later used to authorize protected routes.

LOGIN USER API

The screenshot shows the API testing interface for the `POST LOGIN USER` endpoint. The URL is `http://localhost:5000/api/auth/login`. The request body is set to `raw` JSON format, containing:

```
1 {
2   "email": "yashrajbhasin24@gmail.com",
3   "password": "123456"
4 }
```

The response status is `200 OK`, with a response time of `180 ms` and a size of `540 B`. The response body is a JSON object:

```
1 {
2   "id": "698b063fb1a16566730e0d116",
3   "username": "yash",
4   "email": "yashrajbhasin24@gmail.com",
5   "token": "eyJhbGciOiJIUzI1NiJzIm5CcCi6IkpxVC39.eyJpZC16IjY50GIwNjN0GEExNjU2NjczMGUwZDExN1isImhCi6Mt3M0cx00k10cWlZXhW1joXNzw0DA1MzU4fQ.onZo030cpH10ymTggz-FvqLKHByzDHBivg_1Qm-x0"
```

This endpoint authenticates an existing user using their registered email and password. If the credentials are valid, the server generates and returns a JWT token. This token proves the user's identity and is required to access task-related APIs.

CREATE TASK API

The screenshot shows the Postman interface with the following details:

- Collection:** Secure-Task-Manager
- Request Type:** POST
- URL:** http://localhost:5000/api/tasks
- Body:** raw JSON
- JSON Data:**

```
1 {
2   "title": "Complete Assignment",
3   "description": "Finish Secure Task Manager project"
4 }
```
- Response Status:** 201 Created
- Response Headers:** 65 ms · 508 B
- Response Body:**

```
1 {
2   "_id": "698b803f3a1566073e0d11a",
3   "user": "698b803f3a1566073e0d11a",
4   "title": "Complete Assignment",
5   "description": "Finish Secure Task Manager project",
6   "createdAt": "2026-02-10T10:38:21.246Z",
7   "updatedAt": "2026-02-10T10:38:21.246Z",
8   "__v": 0
9 }
```

This API allows an authenticated user to create a new task. The request body contains the task title and description. The task is linked to the logged-in user and stored in the database with timestamps for creation and updates.

GET ALL TASKS API

The screenshot shows a dark-themed interface for a workspace named "Yash Raj Bhasin's Workspace". On the left, a sidebar lists collections: "Secure-Tasks-Manager" (selected), "POST LOGIN USER", "POST REGISTER USER", "POST CREATE TASK", "GET Get All Tasks" (highlighted in blue), "GET UPDATE TASK", and "GET New Request". The main area displays the "Secure-Tasks-Manager / Get All Tasks" endpoint. The request method is set to "GET" and the URL is "http://localhost:5000/api/tasks". Below the URL, there are tabs for "Docs", "Params", "Authorization", "Headers (8)", "Body", "Scripts", and "Settings". The "Body" tab is selected, showing the response body. The response status is "200 OK", with a timestamp of "83 ms" and a size of "505 B". The response body is a JSON array containing one task object:

```
[{"id": "698b88bd8a16566738e8d1a", "title": "Complete Assignment", "description": "Finish Secure Task Manager project", "user": "698b88bd8a16566738e8d115", "createdAt": "2026-02-10T18:30:21.246Z", "updatedAt": "2026-02-10T18:30:21.246Z", "_v": 0}]
```

This endpoint retrieves all tasks created by the authenticated user. It ensures data privacy by returning only tasks associated with the user's account, making the system secure and user-specific.

UPDATE TASK API

The screenshot shows the API documentation for the `PUT /api/tasks/{task_id}` endpoint. The URL is `http://localhost:5000/api/tasks/698b08bd5a16566730e0d11a`. The request body is a JSON object:

```
1 {
2   "title": "My Task",
3   "description": "Testing"
4 }
```

The response status is `200 OK`, with a response time of `130 ms` and a size of `464 B`. The response body is:

```
1 {
2   "_id": "698b08bd5a16566730e0d11a",
3   "title": "My Task",
4   "description": "Testing",
5   "user": "698b08bd5a16566730e0d116",
6   "createdAt": "2022-02-10T09:38:21.246Z",
7   "updatedAt": "2022-02-10T09:44:07.956Z",
8   "__v": 0
9 }
```

This API updates an existing task using its unique task ID. The user can modify the task title or description. Updated timestamps help track changes, ensuring accurate task management.

DELETE TASK API

The screenshot shows the Postman interface for the 'Delete Task' endpoint. The left sidebar lists collections like 'Secure-Tasks-Manager' with methods such as POST LOGIN USER, POST REGISTER USER, POST CREATE TASK, GET Get All Tasks, PUT UPDATE TASK, and the selected GET Delete Task. The main panel shows a DELETE request to `http://localhost:5000/api/tasks/698b08bd8a16566730e0d1a`. The 'Body' tab is selected, showing the message "This request does not have a body". Below the request, the response section shows a 200 OK status with a JSON body containing the message "Task removed".

```
{} JSON
1 {
2   "message": "Task removed"
3 }
```

200 OK · 67 ms · 293 B · Save Response

This endpoint permanently deletes a task based on its unique ID. Once deleted, the task is removed from the database, confirming successful cleanup and proper task lifecycle management.