

TO-DO-API BACKEND

DOCUMENTATION USING

POSTMAN

1) Check a server is properly running using “GET” Method.

=

1. Open **Postman**
2. Create a **New Request**
3. Select HTTP Method → **GET**

□ Enter the Server URL

In the URL field, enter:

`http://localhost:5000/`

□ Important:

- Use **http**, not https
- Use the correct port (**5000**)

□ Send the Request

Click the **Send** button.

✓Check the Response

Expected Response:

Smart ToDo API running

Status Code:

200 OK

The screenshot shows the Postman application interface. On the left, the sidebar lists various API endpoints under the collection 'Smart TO-DO API'. The selected endpoint is 'server run check' (GET). In the main workspace, the 'Params' tab is active, showing a single parameter 'Key' with a value 'Value'. Below this, the 'Body' tab shows the response: '200 OK' with a status message 'Smart ToDo API running'. The bottom navigation bar includes options like 'HTML', 'Preview', 'Visualize', and 'Logs'.

2) Register a User Using “Post” Method.

1. Launch **Postman**
2. Click **New → HTTP Request**
3. Select **HTTP Method → POST**

□ Enter the Registration API URL

In the URL field, enter:

```
http://localhost:5000/api/auth/register
```

□ Set Request Headers

Go to the **Headers** tab and add:

Key	Value
Content-Type	application/json

□ Add Request Body

1. Click the **Body** tab
2. Select **raw**
3. Choose **JSON** from the dropdown
4. Enter the following data:

```
{  
  "name": "Ranjan Das",  
  "email": "Ranjan@example.com",  
  "password": "123457"  
}
```

□ Send the Request

Click the **Send** button.

❖ Check the Response

Successful Response:

```
{  
  "message": "User created",  
  "user": {
```

```

        "id": "694f85046b6d20e0c06c1784",
        "name": "Ranjan Das",
        "email": "Ranjan@example.com"
    }
}

```

Status Code:

201 Created

The screenshot shows the Postman interface with the following details:

- Collection:** Smart TO-DO API
- Request Type:** POST
- URL:** http://localhost:5000/api/register
- Headers:** (8 items)
- Body (JSON):**

```

1 {
2     "name": "Ranjan Das",
3     "email": "Ranjan@example.com",
4     "password": "123457"
5 }
```
- Response:**
 - Status: 201 Created
 - Time: 287 ms
 - Size: 357 B
 - Content:

```

1     "message": "User created",
2     "user": {
3         "id": "694f85046b6d20e0c06c1784",
4         "name": "Ranjan Das",
5         "email": "Ranjan@example.com"
6     }
7 }
```

3) Login a User Using “Post” Method.

1. Open Postman
2. Click New → HTTP Request
3. Select HTTP Method → POST

Enter Login API URL

http://localhost:5000/api/auth/login

Set Request Headers

Go to **Headers** tab and add:

Key	Value
Content-Type	application/json

Add Request Body

1. Go to **Body**
2. Select **raw**
3. Choose **JSON**
4. Enter login credentials:

```
{  
  "email": "john@example.com",  
  "password": "123456"  
}
```

Send the Request

Click **Send**.

Check the Response

Successful Response:

```
{  
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
  "user": {  
    "id": "694eac8a34fb2cab2a32ef3",  
    "name": "John Doe",  
    "email": "john@example.com"  
  }  
}
```

Status Code:

200 OK

Important: Save the JWT Token

Copy the token value.

You will use it as **Bearer Token** for:

- Create Task
- Update Task
- Delete Task

The screenshot shows the Postman interface. On the left, a sidebar lists various API endpoints under the category 'Smart TO-DO API'. The 'user login' endpoint is selected. The main area shows a POST request to 'http://localhost:5000/api/login'. The 'Body' tab is active, showing a JSON payload:

```

1  {
2    "email": "john@example.com",
3    "password": "123456"
4  }

```

Below the request, the response is displayed with a status of '200 OK'. The response body is also in JSON format:

```

1  {
2    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY5NGVhYzhmZRMnQyY2F1MmEzMmVmMyIsImhlhdCI6MTc2NjgyMDk0MiwiZXhwIjoxNzY2OTAwMzQyfQ.apx_00-jZn10M_n7POcA2xPMpt9St107TYboCb-r0cE",
3    "user": {
4      "id": "694eac8a34fb2cab2a32ef3",
5      "name": "John Doe",
6      "email": "john@example.com"
7    }

```

4) Task creation using JWT , “Post” Method.

Ensure User Is Logged In

- Login using the **POST /api/auth/login** API
- Copy the **JWT token** from the response

Example:

```
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
```

Open Postman

1. Open **Postman**
2. Create a **New HTTP Request**
3. Select HTTP Method → **POST**

Enter Task Creation API URL

`http://localhost:5000/api/tasks`

Set Authorization (JWT)

1. Go to **Authorization** tab
2. Select **Type** → Bearer Token
3. Paste the **JWT token** (without quotes)

✓ Postman will automatically set the header:

Authorization: Bearer <JWT_TOKEN>

□ Set Request Headers

Key	Value
Content-Type	application/json

□ Add Request Body

1. Go to **Body**
2. Select **raw**
3. Choose **JSON**
4. Enter task details:

```
{  
  "title": "Finish Smart ToDo API project",  
  "completed": false  
}
```

□ Send the Request

Click **Send**.

✓ Check the Response

Successful Response:

```
{  
  "_id": "694eb99da964b59aa33ab7cb",  
  "title": "Finish Smart ToDo API project",  
  "completed": false,  
  "userId": "694eac8a34fbd2cab2a32ef3",  
  "__v": 0  
}
```

Status Code:

201 Created

The screenshot shows the Postman interface with the following details:

- Collection:** Smart TO-DO API
- Request Type:** POST
- URL:** http://localhost:5000/tasks
- Headers:** Authorization: Bearer eyJhbGciOiJIUz...
- Body:** (Raw JSON) {
 "title": "Finish Smart ToDo API project",
 "completed": false,
 "userId": "694eac8a34fbd2cab2a32ef3",
 "_id": "694f8cedb19ec68817ee8bea",
 "__v": 0
}
- Response Status:** 201 Created
- Response Time:** 107 ms
- Response Size:** 377 B
- Response Body:** (Raw JSON) {
 "title": "Finish Smart ToDo API project",
 "completed": false,
 "userId": "694eac8a34fbd2cab2a32ef3",
 "_id": "694f8cedb19ec68817ee8bea",
 "__v": 0
}

5) update task using “PUT” Method.

= Login and Get JWT Token

1. Login using:

```
POST /api/auth/login
```

2. Copy the **JWT token** from the response.

□ Get Task ID

1. Send a **GET request**:

```
GET http://localhost:5000/api/tasks
```

2. Copy the `_id` of the task you want to update.

Example:

```
"_id": "694eb99da964b59aa33ab7cb"
```

□ Open Postman

1. Create a **New Request**
2. Select HTTP Method → **PUT**

□ Enter Update Task API URL

Replace `<taskId>` with the actual task ID:

```
http://localhost:5000/api/tasks/694eb99da964b59aa33ab7cb
```

□ Set Authorization (JWT)

1. Go to **Authorization** tab
2. Select **Type** → `Bearer Token`
3. Paste the JWT token

□ Set Request Headers

Key	Value
Content-Type	application/json

□ Add Request Body

Go to **Body** → **raw** → **JSON** and enter updated fields:

```
{
  "title": "Finish Smart ToDo API project (Updated)",
  "completed": true
}
```

You can update:

- title
- completed

□ Send the Request

Click **Send**.

✓ Check the Response

Successful Response:

```
{
  "_id": "694eb99da964b59aa33ab7cb",
  "title": "Finish Smart ToDo API project (Updated)",
  "completed": true,
  "userId": "694eac8a34fbcd2cab2a32ef3",
  "__v": 0
}
```

Status Code:

200 ok

The screenshot shows the Postman interface with the following details:

- Request URL:** http://localhost:5000/tasks/694f8cedb19ec68...
- Method:** PUT
- Headers:** (9 items)
- Body (raw JSON):**

```
{
  "completed": true,
  "title": "Finish Smart ToDo API ASAP123"
}
```
- Response Status:** 200 OK
- Response Time:** 99 ms
- Response Size:** 371 B
- Response Content:**

```
{
  "_id": "694f8cedb19ec68817ee8bea",
  "title": "Finish Smart ToDo API ASAP123",
  "completed": true,
  "userId": "694eac8a34fbcd2cab2a32ef3",
  "__v": 0
}
```

6) Delete Task using “DELETE” method.

= Login and Obtain JWT Token

1. Send a login request:

```
POST http://localhost:5000/api/auth/login
```

2. Copy the **JWT token** from the response.

□ Get the Task ID

1. Fetch all tasks:

```
GET http://localhost:5000/api/tasks
```

2. Copy the `_id` of the task you want to delete.

Example:

```
"_id": "694eb99da964b59aa33ab7cb"
```

□ Open Postman

1. Create a **New HTTP Request**
2. Select HTTP Method → **DELETE**

□ Enter Delete Task API URL

Replace `<taskId>` with the actual task ID:

```
http://localhost:5000/api/tasks/694eb99da964b59aa33ab7cb
```

□ Set Authorization (JWT)

1. Go to **Authorization** tab
2. Select **Type** → `Bearer Token`
3. Paste the JWT token

□ Send the Request

Click **Send**.

✓ Check the Response

Successful Response:

```
{  
  "message": "Task deleted successfully"  
}
```

Status Code:

200 OK

The screenshot shows the Postman interface for the Smart TO-DO API's Delete Task endpoint. The request method is set to **DELETE** and the URL is `http://localhost:5000/tasks/694f8cedb19ec68...`. In the Headers tab, there is a single header `Authorization` with the value `Bearer eyJhbGciOiJIUz...`. The Body tab is set to `JSON` and contains the message `"message": "Task Deleted"`. The response tab shows a **200 OK** status with a response time of 82 ms and a body size of 261 B. The response body is `{ "message": "Task deleted" }`.

This screenshot is identical to the one above, showing the Postman interface for the Smart TO-DO API's Delete Task endpoint. The request method is **DELETE**, URL is `http://localhost:5000/tasks/694f8cedb19ec68...`, and the Headers tab shows the `Authorization` header with the value `Bearer eyJhbGciOiJIUz...`. The Body tab is `JSON` with the message `"message": "Task deleted"`. The response tab shows a **200 OK** status with a response time of 82 ms and a body size of 261 B, containing the message `{ "message": "Task deleted" }`.