API Documentation

This document provides a comprehensive guide to the Todo Application's REST API. The application is composed of two primary services: a **User Service** for user authentication and management, and a **Todo Service** for handling todo items.

All API interactions are performed over HTTP using **JSON** for both request bodies and responses.

1. Authentication

The API uses **JWT (JSON Web Token) Bearer Token** authentication. A valid JWT must be included in the `Authorization` header of every request to the Todo Service.

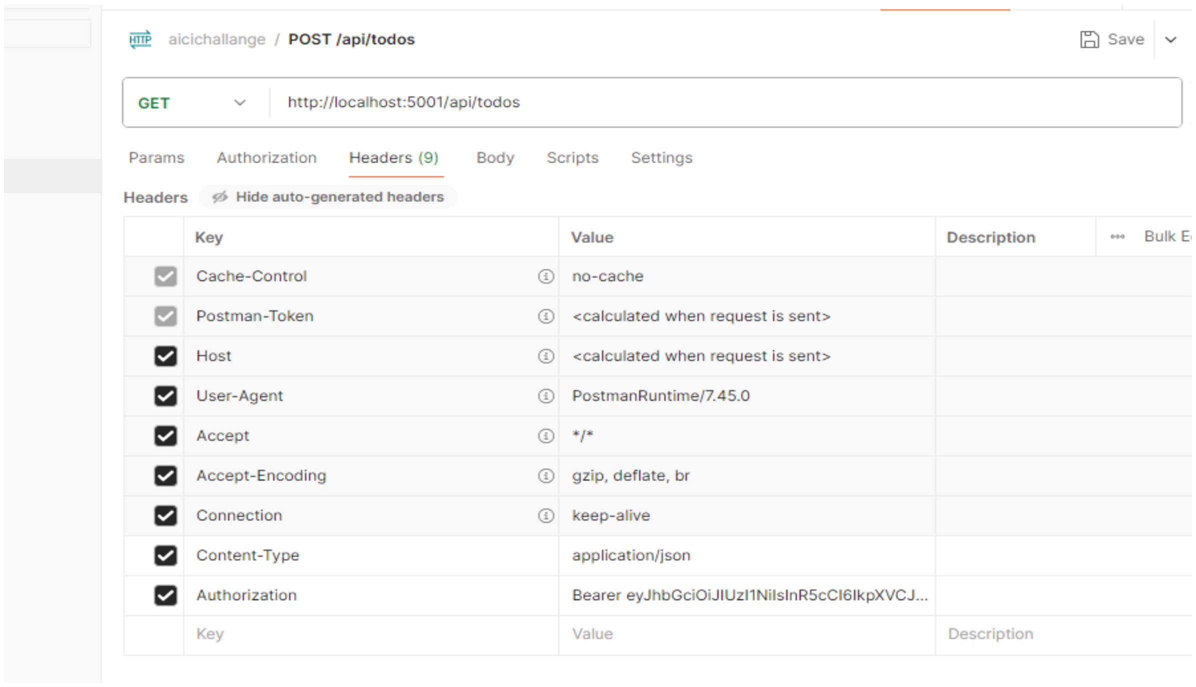
How to Obtain a Token

1. **Register a new user** using the `POST /api/users/register` endpoint.
2. **Log in an existing user** using the `POST /api/users/login` endpoint.

How to Use the Token

Once you have a token, include it in your request headers as follows:

```
Authorization: Bearer <YOUR_JWT_TOKEN>
```


aicichallange / POST /api/todos

Save

GET http://localhost:5001/api/todos

Params Authorization Headers (9) Body Scripts Settings

Headers Hide auto-generated headers

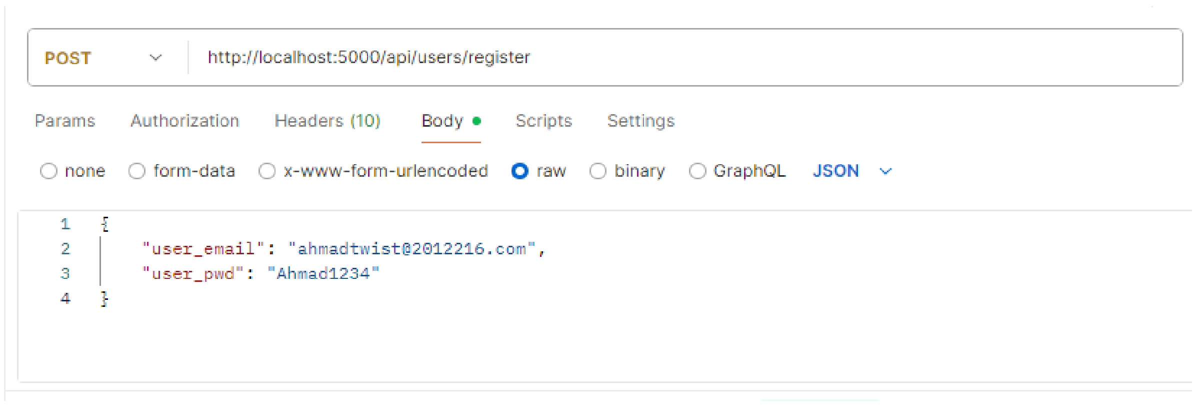
	Key	Value	Description
✓	Cache-Control	no-cache	
✓	Postman-Token	<calculated when request is sent>	
✓	Host	<calculated when request is sent>	
✓	User-Agent	PostmanRuntime/7.45.0	
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	Content-Type	application/json	
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	
	Key	Value	Description

2. User Service

Base URL: http://localhost:5000/api/users

Register a New User

- **Method**: POST
- **Path**: /register
- **Description**: Creates a new user account.
- **Request Body**:

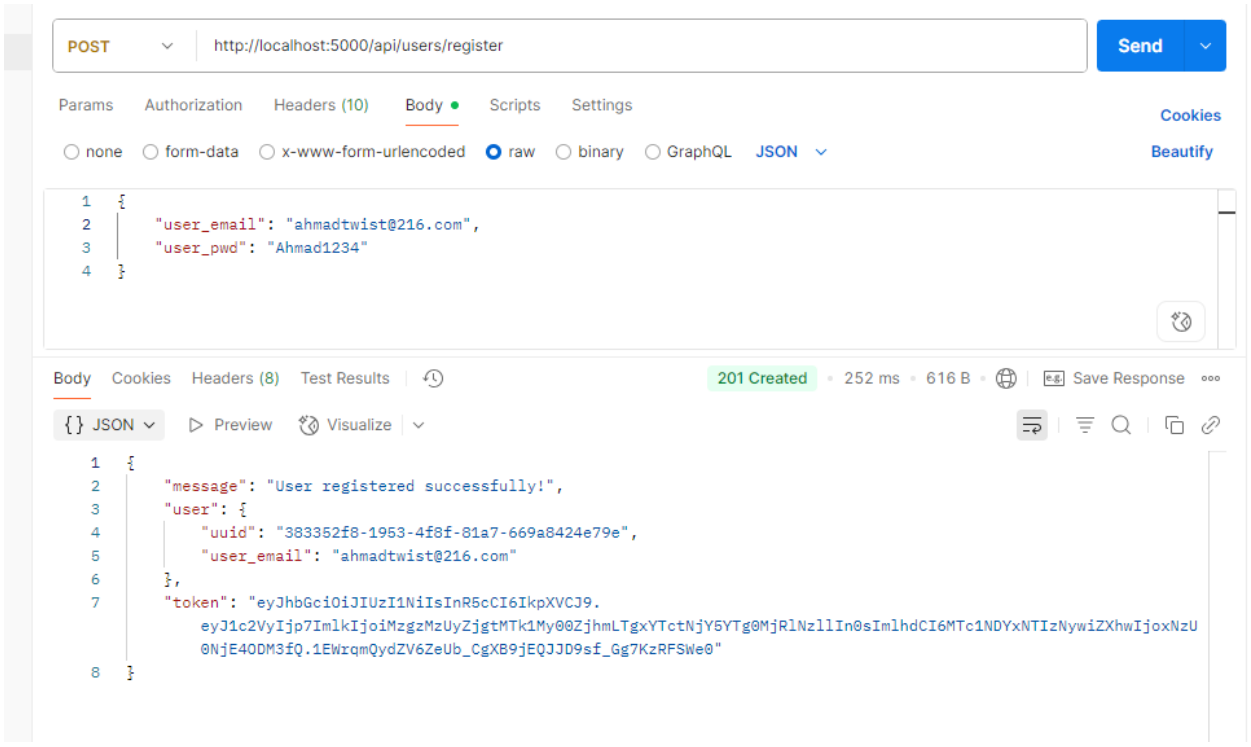

POST http://localhost:5000/api/users/register

Params Authorization Headers (10) Body ● Scripts Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "user_email": "ahmadtwist@2012216.com",
    "user_pwd": "Ahmad1234"
}
```

Success Response: 201 Created



POST http://localhost:5000/api/users/register

```
{
    "user_email": "ahmadtwist@216.com",
    "user_pwd": "Ahmad1234"
}
```

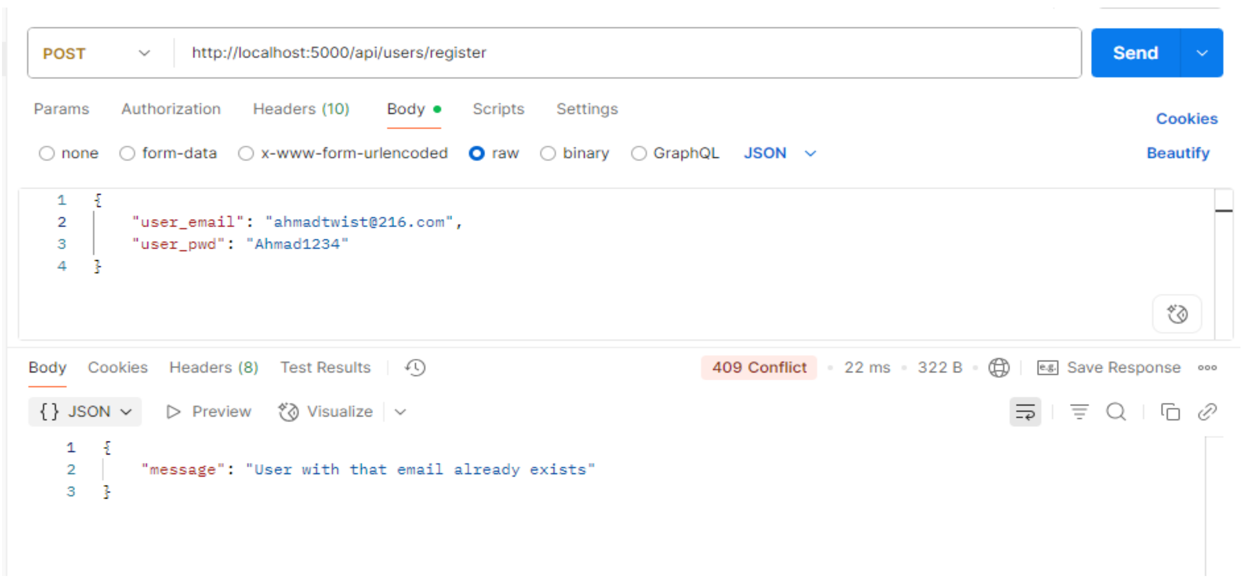
201 Created · 252 ms · 616 B

```
{
    "message": "User registered successfully!",
    "user": {
        "uuid": "383352f8-1953-4f8f-81a7-669a8424e79e",
        "user_email": "ahmadtwist@216.com"
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VyIjp7ImlkIjoiMzgzMzUyZjgtMTk1My00ZjhmLTgxYTctNjY5YTg0MjRlNzllIn0sImlhdCI6MTc1NDYxNTIzNywiZXhwIjoxNzU
        0NjE4ODM3fQ.1EWrqmQydZV6ZeUb_CgXB9jEQJJD9sf_Gg7KzRFSWe0"
}
```

Error Responses:

- `400 Bad Request`: Missing email or password.
- `409 Conflict`: User with the provided email already exists.

Error Responses Screen-Shots:

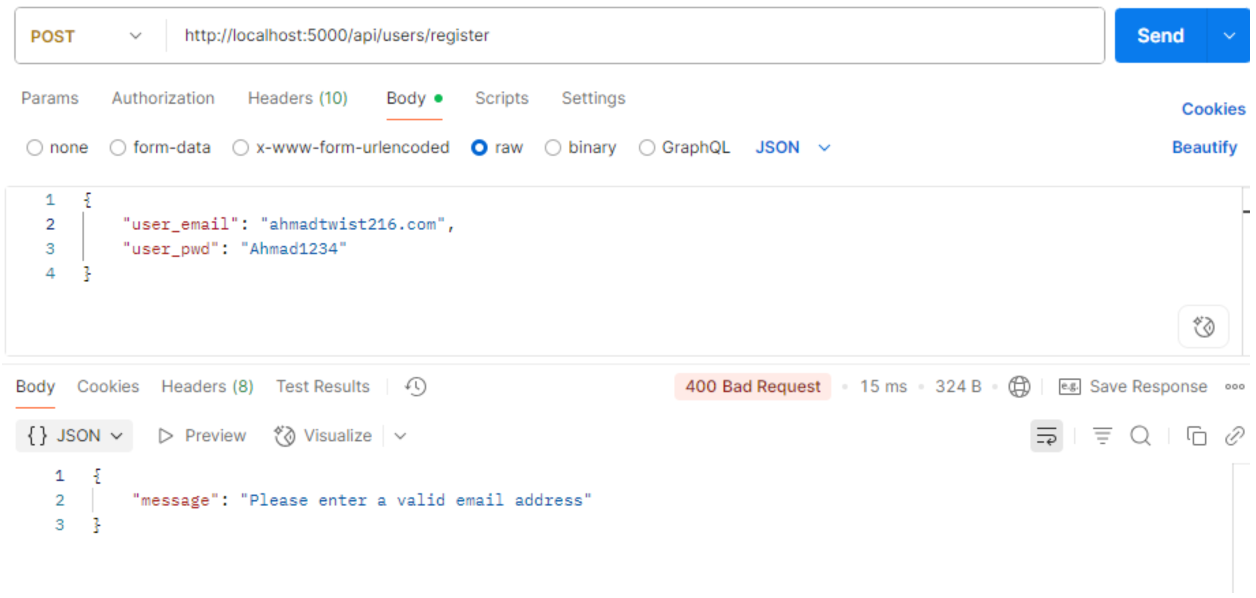


http://localhost:5000/api/users/register

```
{
    "user_email": "ahmadtwist@216.com",
    "user_pwd": "Ahmad1234"
}
```

409 Conflict • 22 ms • 322 B

```
{
    "message": "User with that email already exists"
}
```



http://localhost:5000/api/users/register

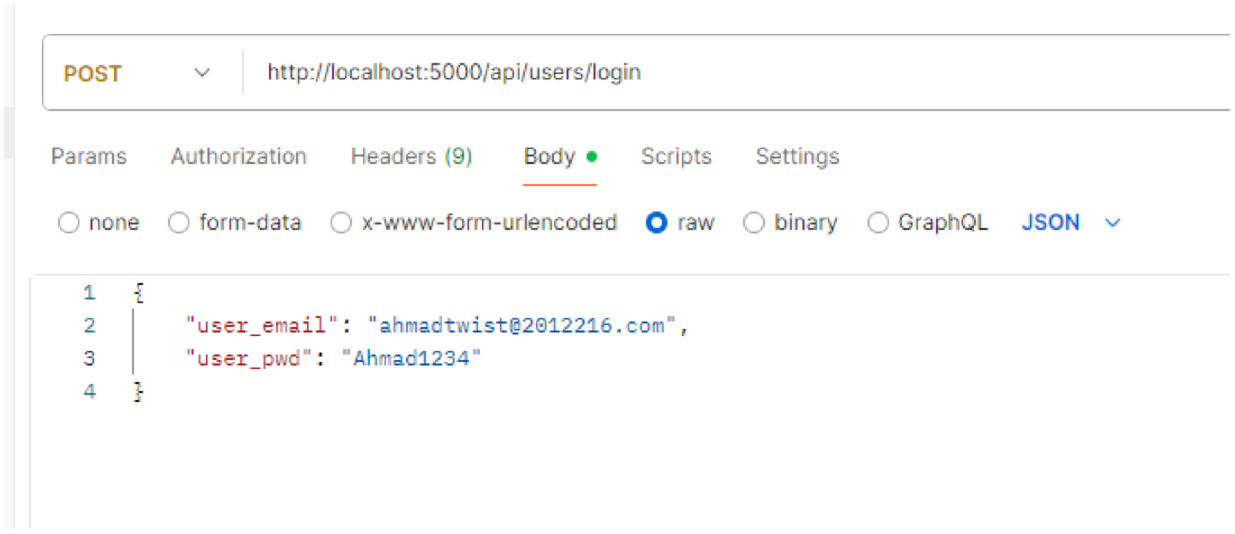
```
{
    "user_email": "ahmadtwist216.com",
    "user_pwd": "Ahmad1234"
}
```

400 Bad Request • 15 ms • 324 B

```
{
    "message": "Please enter a valid email address"
}
```

User Login

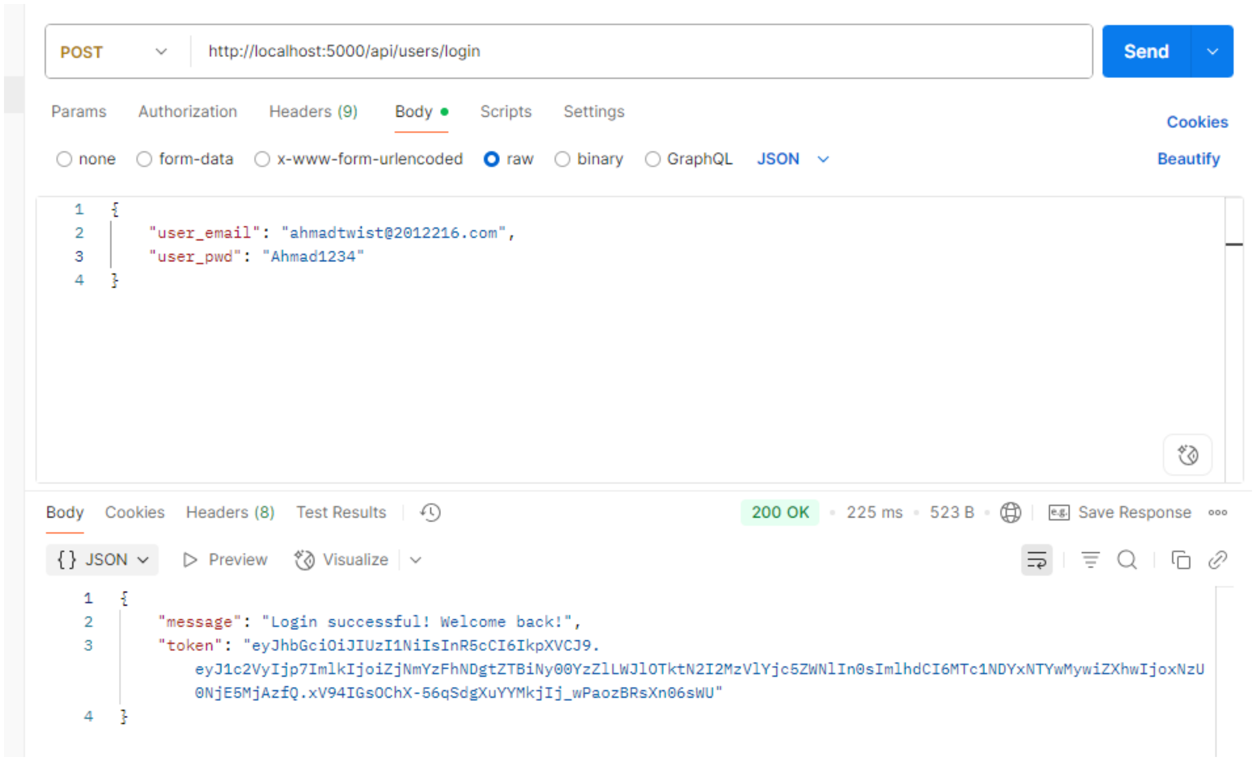
- **Method**: POST
- **Path**: `/login`
- **Description:** Authenticates a user and returns a JWT for subsequent requests.
- **Request Body**:


POST http://localhost:5000/api/users/login

Body: raw, JSON

```json
{
    "user_email": "ahmadtwist@2012216.com",
    "user_pwd": "Ahmad1234"
}
```

Success Response: `200 OK`


POST http://localhost:5000/api/users/login

Body: raw, JSON

```json
{
    "user_email": "ahmadtwist@2012216.com",
    "user_pwd": "Ahmad1234"
}
```

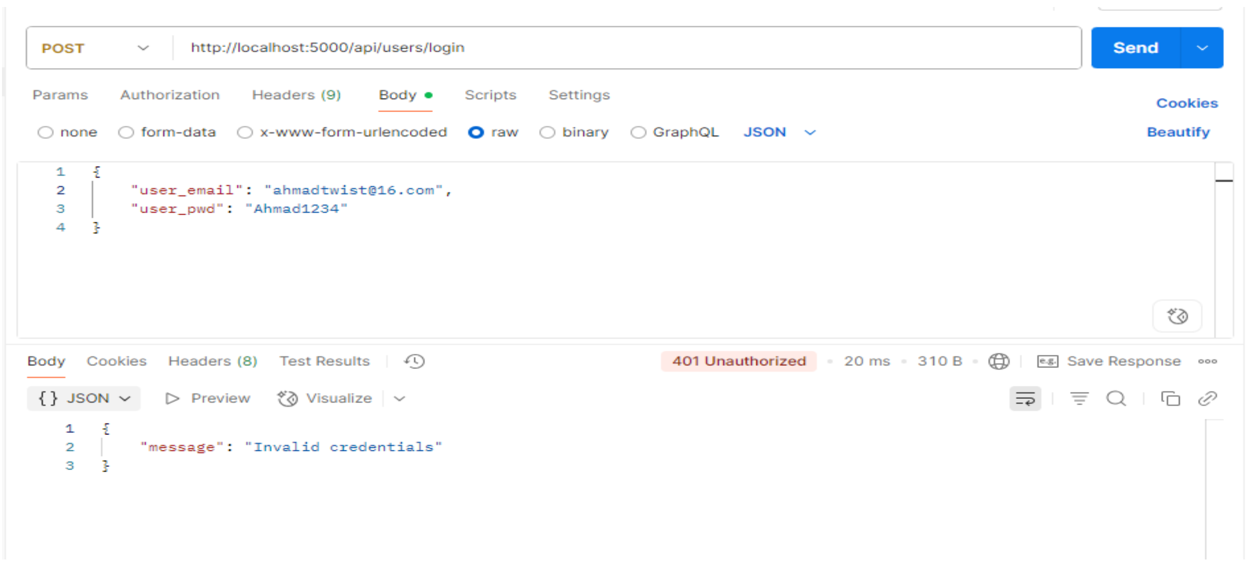
200 OK · 225 ms · 523 B

```json
{
    "message": "Login successful! Welcome back!",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VyIjp7ImlkIjoiZjNmYzFhNDgtZTBiNy00YzZlLWJlOTktN2I2MzVlYjc5ZWNlIn0sImlhdCI6MTc1NDYxNTYwMywiZXhwIjoxNzU
        0NjE5MjAzfQ.xV94IGsOChX-56qSdgXuYYMkjIj_wPaozBRsXn06sWU"
}
```

Error Responses:

- `400 Bad Request`: Missing email or password.
- `401 Unauthorized`: Incorrect email or password.

Error Responses Screen-Shots:

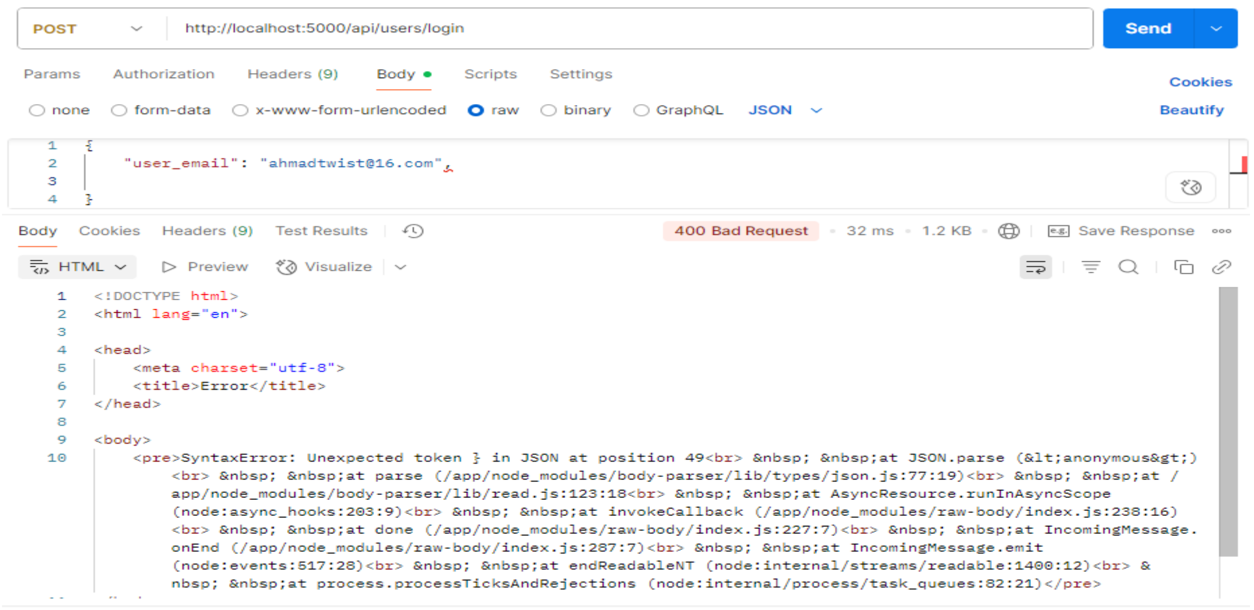


http://localhost:5000/api/users/login

```json
{
    "user_email": "ahmadtwist@16.com",
    "user_pwd": "Ahmad1234"
}
```

401 Unauthorized · 20 ms · 310 B

```json
{
    "message": "Invalid credentials"
}
```



http://localhost:5000/api/users/login

```json
{
    "user_email": "ahmadtwist@16.com",

}
```

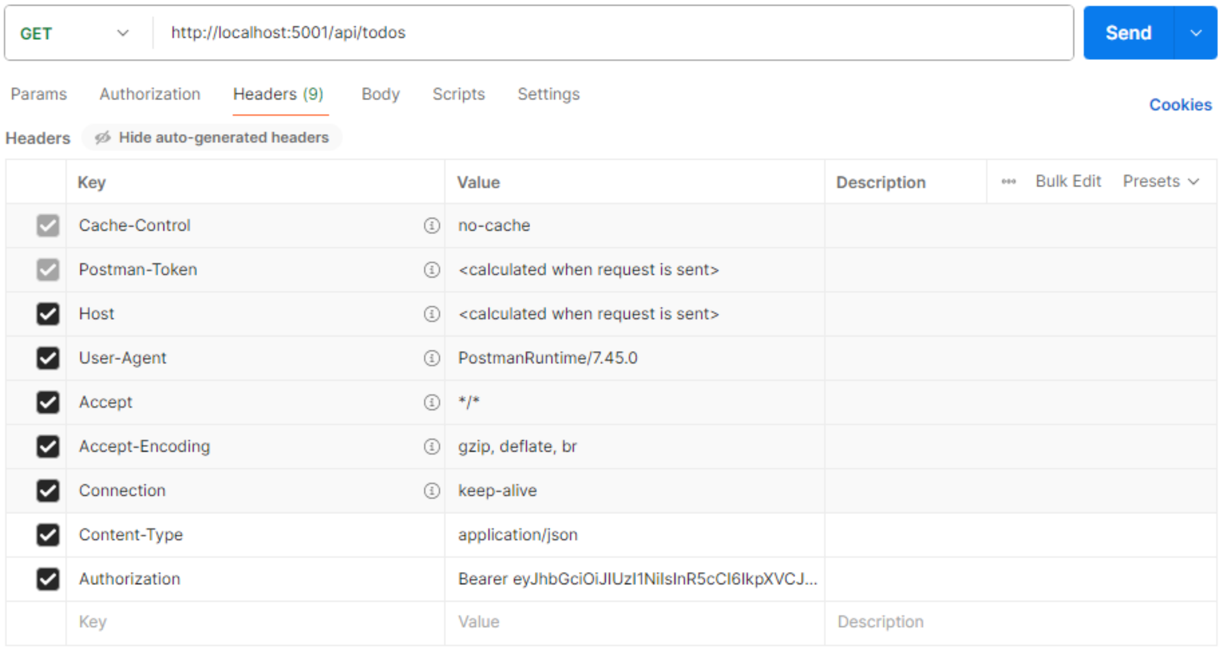
400 Bad Request · 32 ms · 1.2 KB

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Error</title>
</head>

<body>
    <pre>SyntaxError: Unexpected token } in JSON at position 49<br> &nbsp; &nbsp;at JSON.parse (&lt;anonymous&gt;)
        <br> &nbsp; &nbsp;at parse (/app/node_modules/body-parser/lib/types/json.js:77:19)<br> &nbsp; &nbsp;at /
        app/node_modules/body-parser/lib/read.js:123:18<br> &nbsp; &nbsp;at AsyncResource.runInAsyncScope
        (node:async_hooks:203:9)<br> &nbsp; &nbsp;at invokeCallback (/app/node_modules/raw-body/index.js:238:16)
        <br> &nbsp; &nbsp;at done (/app/node_modules/raw-body/index.js:227:7)<br> &nbsp; &nbsp;at IncomingMessage.
        onEnd (/app/node_modules/raw-body/index.js:287:7)<br> &nbsp; &nbsp;at IncomingMessage.emit
        (node:events:517:28)<br> &nbsp; &nbsp;at endReadableNT (node:internal/streams/readable:1400:12)<br> &
        nbsp; &nbsp;at process.processTicksAndRejections (node:internal/process/task_queues:82:21)</pre>
```

3. Todo Service

Base URL: `http://localhost:5001/api/todos`

- **Create a New Todo**
- **Method**: POST
- **Path**: /
- **Description**: Creates a new todo item for the authenticated user.
- **Headers**:
  - `Content-Type`: `application/json`
  - `Authorization`: `Bearer JWT_TOKEN`

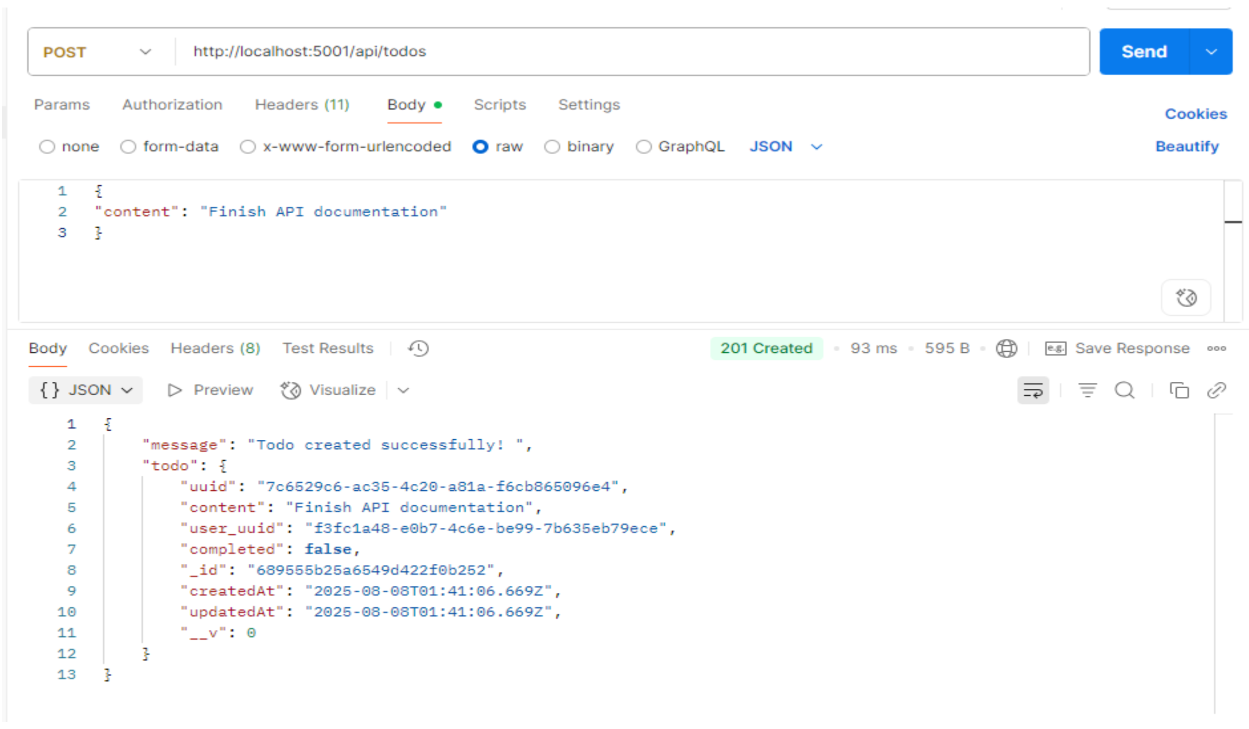


http://localhost:5001/api/todos

Headers

	Key	Value	Description
✓	Cache-Control	no-cache	
✓	Postman-Token	<calculated when request is sent>	
✓	Host	<calculated when request is sent>	
✓	User-Agent	PostmanRuntime/7.45.0	
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	Content-Type	application/json	
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	
	Key	Value	Description

Success Response: `201 Created`



http://localhost:5001/api/todos

```
{
"content": "Finish API documentation"
}
```

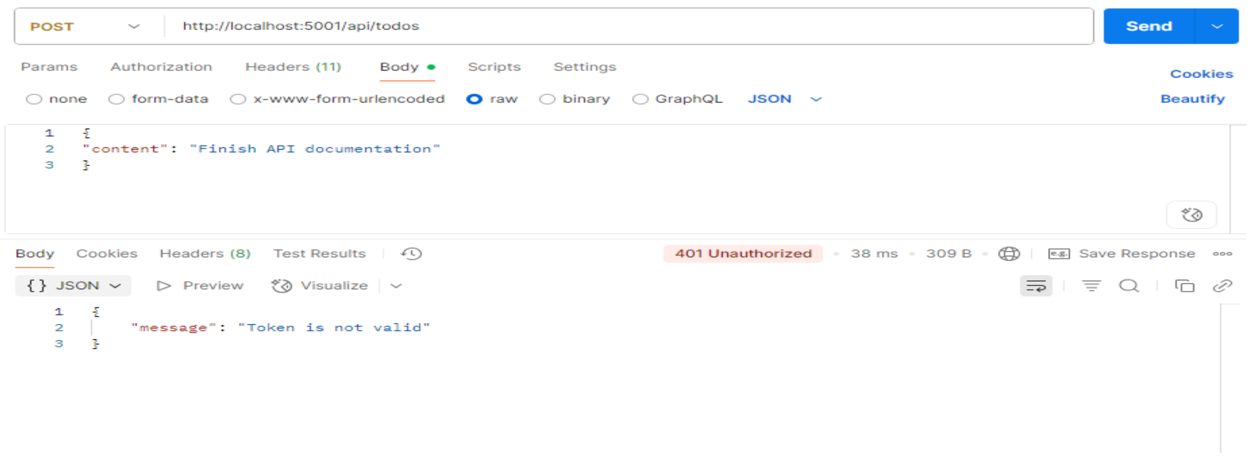
201 Created • 93 ms • 595 B

```
{
    "message": "Todo created successfully! ",
    "todo": {
        "uuid": "7c6529c6-ac35-4c20-a81a-f6cb865096e4",
        "content": "Finish API documentation",
        "user_uuid": "f3fc1a48-e0b7-4c6e-be99-7b635eb79ece",
        "completed": false,
        "_id": "689555b25a6549d422f0b252",
        "createdAt": "2025-08-08T01:41:06.669Z",
        "updatedAt": "2025-08-08T01:41:06.669Z",
        "__v": 0
    }
}
```

Error Responses:

- `400 Bad Request`: content field is missing.
- `401 Unauthorized`: Missing or invalid JWT

Error Responses Screen-Shots:

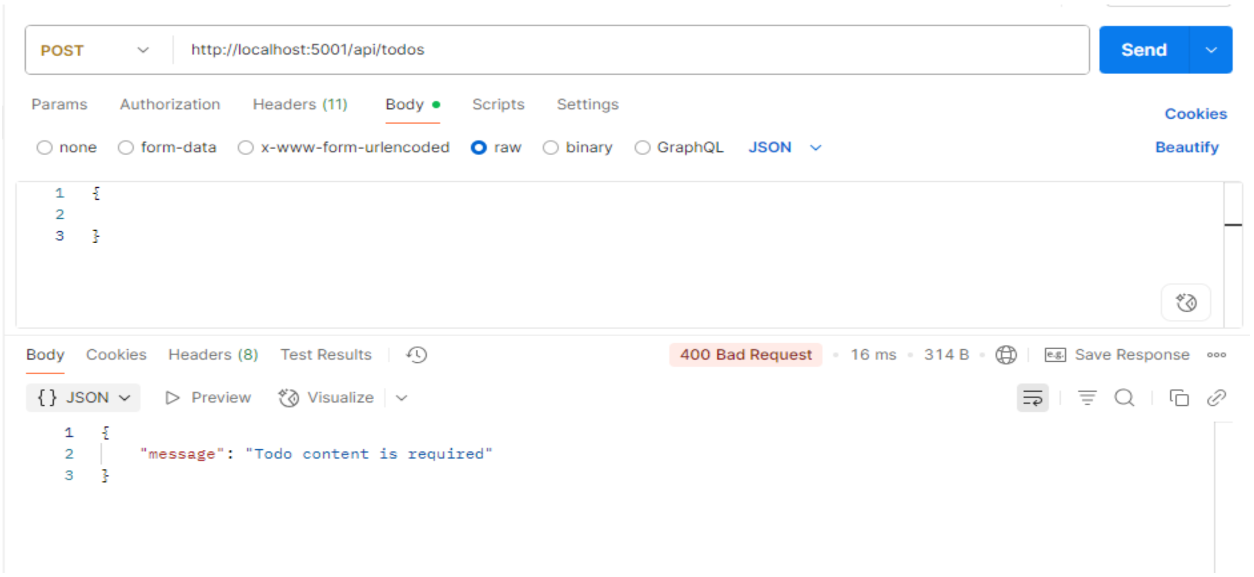


http://localhost:5001/api/todos

```
{
"content": "Finish API documentation"
}
```

401 Unauthorized • 38 ms • 309 B

```
{
    "message": "Token is not valid"
}
```



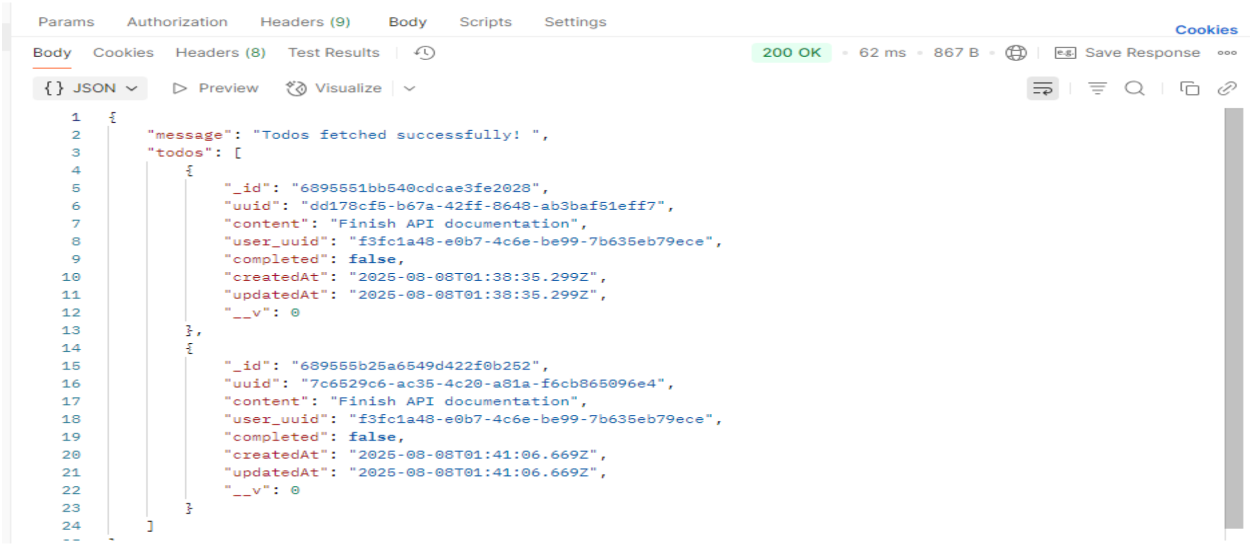
```
{
}
```

```
{
    "message": "Todo content is required"
}
```

Get All Todos

- **Method**: GET
- **Path**: /
- **Description:** Retrieves all todo items owned by the authenticated user.
- **Headers**: `Authorization: Bearer JWT_TOKEN`
- **Success Response**: `200 OK`

Success Response: `201 Created`

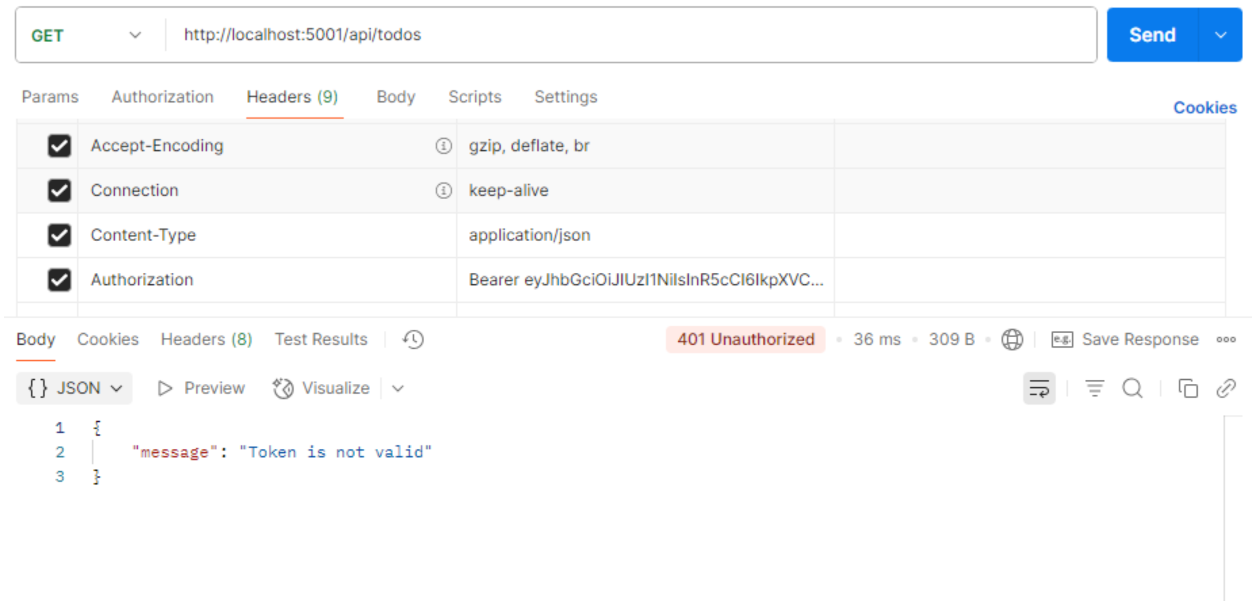


```
{
    "message": "Todos fetched successfully! ",
    "todos": [
        {
            "_id": "6895551bb540cdcae3fe2028",
            "uuid": "dd178cf5-b67a-42ff-8648-ab3baf51eff7",
            "content": "Finish API documentation",
            "user_uuid": "f3fc1a48-e0b7-4c6e-be99-7b635eb79ece",
            "completed": false,
            "createdAt": "2025-08-08T01:38:35.299Z",
            "updatedAt": "2025-08-08T01:38:35.299Z",
            "__v": 0
        },
        {
            "_id": "689555b25a6549d422f0b252",
            "uuid": "7c6529c6-ac35-4c20-a81a-f6cb865096e4",
            "content": "Finish API documentation",
            "user_uuid": "f3fc1a48-e0b7-4c6e-be99-7b635eb79ece",
            "completed": false,
            "createdAt": "2025-08-08T01:41:06.669Z",
            "updatedAt": "2025-08-08T01:41:06.669Z",
            "__v": 0
        }
    ]
```

Error Responses:

- `401 Unauthorized`: Missing or invalid JWT.

Error Responses Screen-Shots:



http://localhost:5001/api/todos

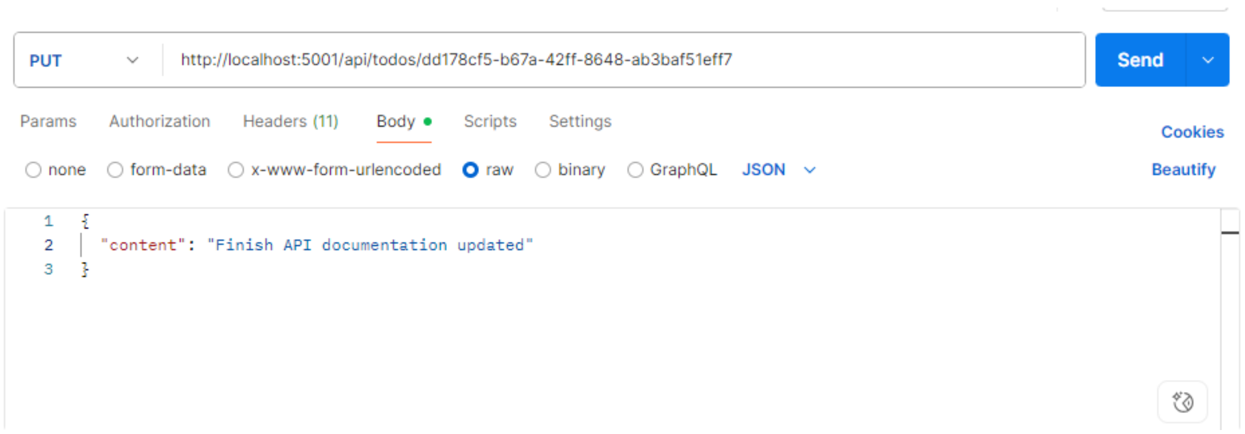
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Content-Type	application/json
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC...

```
{
    "message": "Token is not valid"
}
```

Update a Todo Item

- **Method**: PUT
- **Path**: `/:uuid`
- **Description**: Updates the content or completion status of a todo item.
- **Headers**:
  - `Content-Type`: `application/json`
  - `Authorization`: `Bearer JWT_TOKEN`

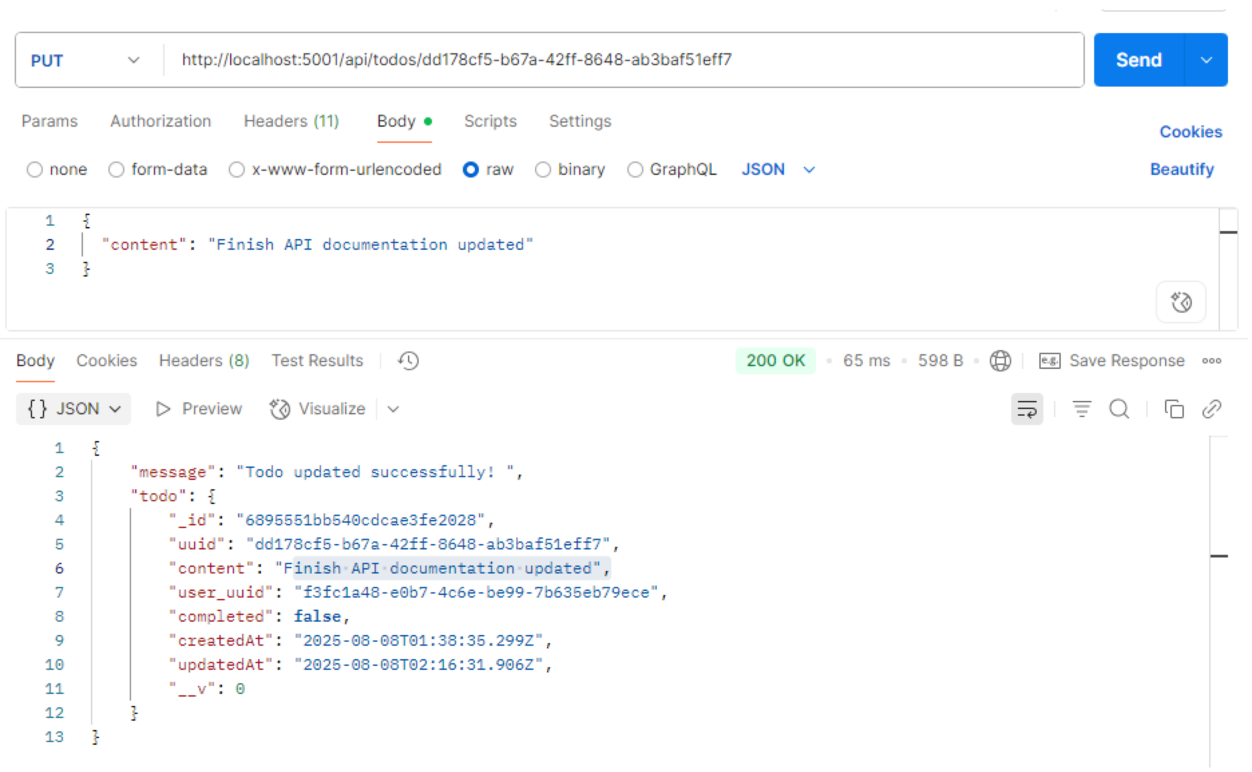
- **Request Body**:



```
http://localhost:5001/api/todos/dd178cf5-b67a-42ff-8648-ab3baf51eff7
```

```
{
  "content": "Finish API documentation updated"
}
```

Success Response: `201 Created`



```
http://localhost:5001/api/todos/dd178cf5-b67a-42ff-8648-ab3baf51eff7
```

```
{
  "content": "Finish API documentation updated"
}
```

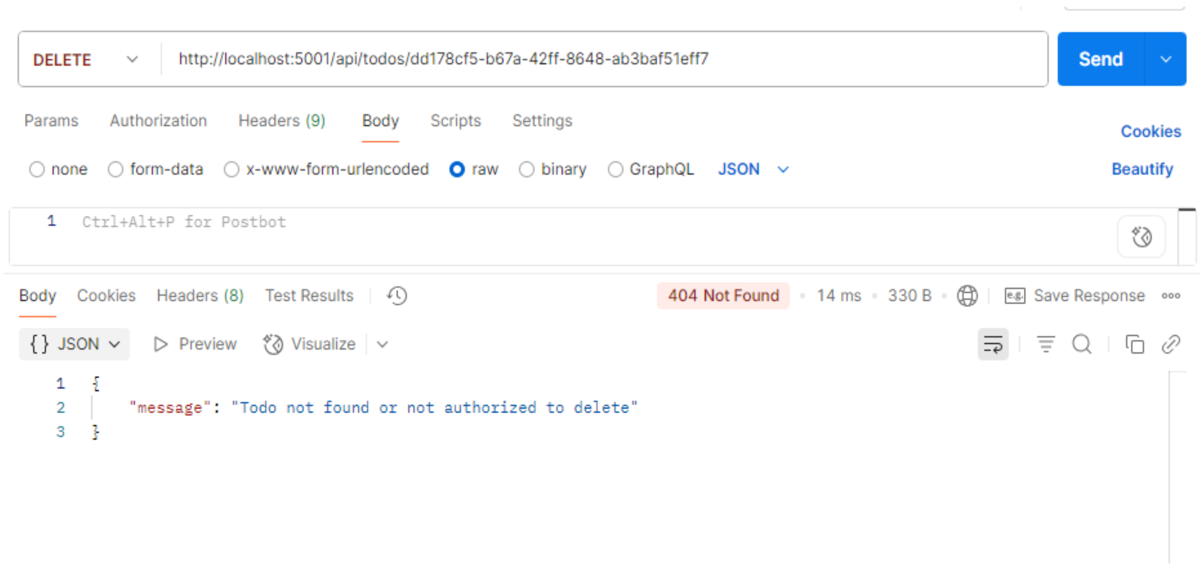
```
{
    "message": "Todo updated successfully! ",
    "todo": {
        "_id": "6895551bb540cdcae3fe2028",
        "uuid": "dd178cf5-b67a-42ff-8648-ab3baf51eff7",
        "content": "Finish API documentation updated",
        "user_uuid": "f3fc1a48-e0b7-4c6e-be99-7b635eb79ece",
        "completed": false,
        "createdAt": "2025-08-08T01:38:35.299Z",
        "updatedAt": "2025-08-08T02:16:31.906Z",
        "__v": 0
    }
}
```

Error Responses:

- `400 Bad Request`: No update data provided.
- `401 Unauthorized`: Missing or invalid JWT.
- `403 Forbidden`: The todo does not belong to the authenticated user.
- `404 Not Found`: The specified `uuid` does not exist.

Delete a Todo Item

- **Method**: `DELETE`
- **Path**: `/:uuid`
- **Description**: Deletes a todo item by its UUID.
- **Headers**: `Authorization: Bearer JWT_TOKEN`
- **Success Response**: `204 No Content`


DELETE http://localhost:5001/api/todos/dd178cf5-b67a-42ff-8648-ab3baf51eff7

```
Ctrl+Alt+P for Postbot
```

404 Not Found · 14 ms · 330 B

```
{
    "message": "Todo not found or not authorized to delete"
}
```

Error Responses

- `401 Unauthorized`: Missing or invalid JWT.
- `403 Forbidden`: The todo does not belong to the authenticated user.
- `404 Not Found`: The specified `uuid` does not exist.

4. Running the Application with Docker

For a streamlined setup that avoids environment conflicts, you can use **Docker** and **Docker Compose**. This method runs each service in its own isolated container.

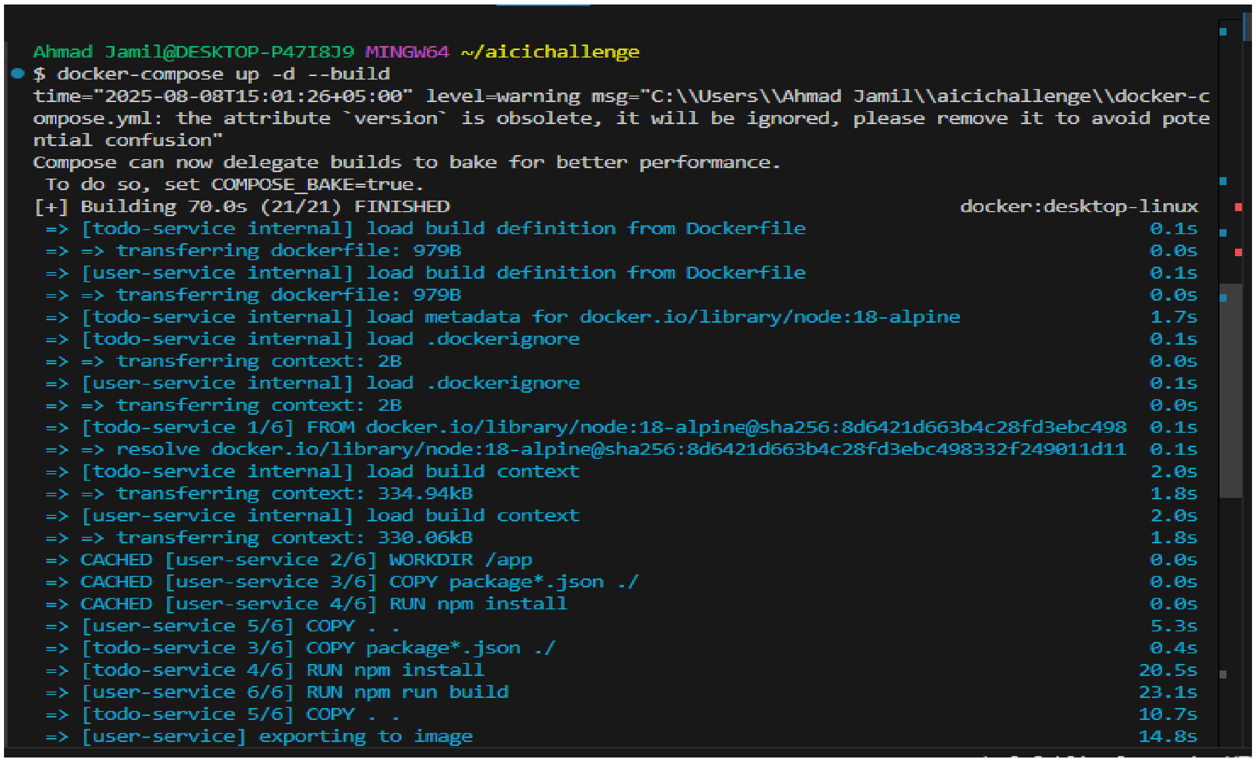
Prerequisites

- **Docker & Docker Compose**: Ensure you have both installed on your system.

Instructions

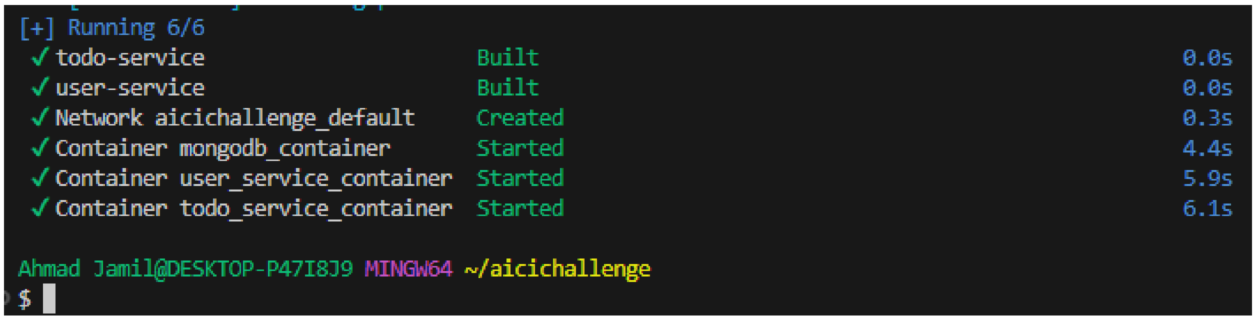
1. **Start all services**: From the root directory of your project (the folder containing the `docker-compose.yml` file), run the following command. This will build the images for each service and start them.

```
docker-compose up -d -build
```

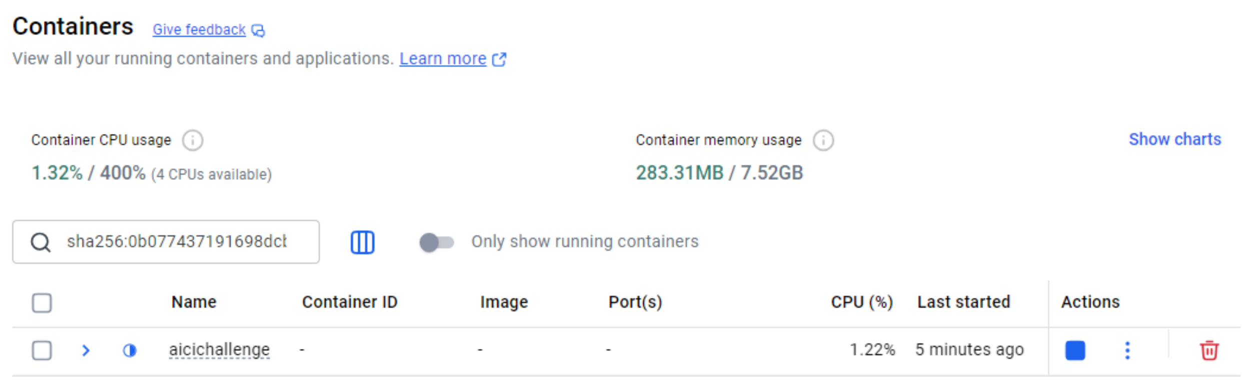


```
Ahmad Jamil@DESKTOP-P47I8J9 MINGW64 ~/aicichallenge
$ docker-compose up -d --build
time="2025-08-08T15:01:26+05:00" level=warning msg="C:\\Users\\Ahmad Jamil\\aicichallenge\\docker-c
ompose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid pote
ntial confusion"
Compose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 70.0s (21/21) FINISHED                                       docker:desktop-linux
 => [todo-service internal] load build definition from Dockerfile                         0.1s
 => => transferring dockerfile: 979B                                                      0.0s
 => [user-service internal] load build definition from Dockerfile                         0.1s
 => => transferring dockerfile: 979B                                                      0.0s
 => [todo-service internal] load metadata for docker.io/library/node:18-alpine            1.7s
 => [todo-service internal] load .dockerignore                                            0.1s
 => => transferring context: 2B                                                           0.0s
 => [user-service internal] load .dockerignore                                            0.1s
 => => transferring context: 2B                                                           0.0s
 => [todo-service 1/6] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498  0.1s
 => => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d11  0.1s
 => [todo-service internal] load build context                                            2.0s
 => => transferring context: 334.94kB                                                     1.8s
 => [user-service internal] load build context                                            2.0s
 => => transferring context: 330.06kB                                                     1.8s
 => CACHED [user-service 2/6] WORKDIR /app                                                0.0s
 => CACHED [user-service 3/6] COPY package*.json ./                                       0.0s
 => CACHED [user-service 4/6] RUN npm install                                             0.0s
 => [user-service 5/6] COPY . .                                                           5.3s
 => [todo-service 3/6] COPY package*.json ./                                              0.4s
 => [todo-service 4/6] RUN npm install                                                   20.5s
 => [user-service 6/6] RUN npm run build                                                 23.1s
 => [todo-service 5/6] COPY . .                                                          10.7s
 => [user-service] exporting to image                                                    14.8s
```

a. up: Starts the containers.
b. `-d`: Runs the containers in **detached mode** (in the background).
c. `--build`: Forces a rebuild of the images, which is useful if you've changed the code.



```
[+] Running 6/6
 ✓ todo-service                       Built                                    0.0s
 ✓ user-service                       Built                                    0.0s
 ✓ Network aicichallenge_default      Created                                  0.3s
 ✓ Container mongodb_container        Started                                  4.4s
 ✓ Container user_service_container   Started                                  5.9s
 ✓ Container todo_service_container   Started                                  6.1s

Ahmad Jamil@DESKTOP-P47I8J9 MINGW64 ~/aicichallenge
$
```



Containers
Give feedback
View all your running containers and applications. Learn more
Container CPU usage
1.32% / 400% (4 CPUs available)
Container memory usage
283.31MB / 7.52GB
Show charts
sha256:0b077437191698dct
Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
aicichallenge	-	-	-	1.22%	5 minutes ago	

2. **Access the application:** Once the containers are running, you can access the frontend in your browser .