📌 Todo App API Documentation

Base URL:

`http://localhost:3000/api`

🔐 Authentication

Register User

`POST /user/register`
Request Body:

```
{
  "name": "John Doe",
  "email": "john@test.com",
  "password": "123456"
}
```

Response:

```
{
    "message": "User registered successfully"
}
```

Login User

`POST /user/login`
Request Body:

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

Response:

```
{
  "token": "JWT_TOKEN_HERE"
}
```

Logout User

`POST /user/logout`

Response:

```
{

    "message": "User logout successfully"

}
```

✅ Todos (Protected Routes)

⚠️ All todo routes require authentication.
Send JWT token in header:

```
Authorization: Bearer <token>
```

Create Todo

```
POST /todo
```
Request Body:

```
{
  "task": "Learn Node.js"
}
```

Response:

```
{
    "message": "Sucessfully Added task"
}
```

Get All Todos (of logged-in user)

```
GET /todo
```
Response:

```
[
  "data": [
        {
            "_id": "68a7007bea3bb2b5f61a5269",
            "task": "new task by admin",
            "userId": "68a5a269863d40e16d9ae630",
            "createdAt": "2025-08-21T11:18:19.313Z",
            "updatedAt": "2025-08-21T11:18:19.313Z",
            "__v": 0
        },
        {
```

```
            "_id": "68a70674ad7f646665f59f03",
            "task": "new task by admin",
            "userId": "68a5a269863d40e16d9ae630",
            "createdAt": "2025-08-21T11:43:48.859Z",
            "updatedAt": "2025-08-21T11:43:48.859Z",
            "__v": 0
        }
    ]]
```

Update Todo

`PUT /todo/:id`
Request Body:

```
{
  "task": "Learn TypeScript"
}
```

Response:

```
{
    "message": "Updated record at id = 68a6fd1bea3bb2b5f61a5251"
}
```

Delete Todo

`DELETE /todo/:id`
Response:

```
{
    "message": "Deleted record at id = 68a6fd1bea3bb2b5f61a5251"
}
```

⚠ Error Responses

The backend return errors in this format:

```
{ "error": "Invalid or Expire Token"}
{ "error": "Invalid ID"}
{ "error": "Email already in use" }
{ "error": "Invalid credentials" }
```

🛠 Setup for Frontend Developer

- Base API URL → `http://localhost:3000/api`

- Add `Content-Type: application/json` in headers.
- Use `Authorization: Bearer <token>` for protected routes.
- `userId` is auto-handled by backend → frontend should **not send it**.
