# Task Management System

## About Me :

**Seeking a challenging Software Engineer position at a leading technology company where I can utilize my strong foundation in Python and Django to develop innovative and impactful software solutions. With a proven track record in backend development and a passion for continuous learning, I am eager to contribute to cutting-edge projects and collaborate with talented teams to drive technological advancements.**

RESUME LINK GitHub Project Link

## Tech Stack used :

- Python
- Django
- Django REST framework
- Swagger

## Database used :

- SQLite

**Rahul Bandi**
**Email : rahulsmrd@gmail.com**
**Phone : +91 93981 49614**

**19 November 2024**

# Task Management System API Documentation

**Base URL** : http://127.0.0.1:8000/api/v1

## Endpoints

### Key Parameters

**id** : Database Indexing
**user** : ID of the Authenticated User
**task_id** : Unique ID for particular task
**title** : Task Title
**description** : Task Description
**status** : Status of the Task
**created_at** : Task creation Timestamp
**updated_at** : Last Modified Timestamp
**email** : email of the user should contain '@'
**password** : strong password
**name** : name of the user

### 1. GET /task

**Success Status Code : 200**

- Returns a JSON of all the Tasks associated to User.
- If user is a Superuser (Admin User) then returns all the Tasks in Database.
- Authentication through token is mandatory.

**Request (JSON)**

```
{}
```

**Response (JSON)**


```
{
  "id": 0,
  "user": 0,
  "task_id": 0,
  "title": "string",
  "description": "string",
  "due_date": "2024-11-19",
  "status": "pending",
  "created_at": "2024-11-19T11:18:11.490Z",
  "updated_at": "2024-11-19T11:18:11.490Z"
}
```

## 2. POST /task

**Success Status Code : 201**

- Creates a Task for a Authenticated User.
- Authentication through token is mandatory.

**Request (JSON)**

```
{
  "title": "string",
  "description": "string",
  "due_date": "2024-11-19",
  "status": "pending"
}
```

**Response (JSON)**

```
{
  "id": 0,
  "user": 0,
  "task_id": 0,
  "title": "string",
  "description": "string",
  "due_date": "2024-11-19",
  "status": "pending",
  "created_at": "2024-11-19T10:25:11.259Z",
  "updated_at": "2024-11-19T10:25:11.259Z"
}
```

## 3. GET /task/{id}

**Success Status Code : 200**

- Returns a JSON of the Tasks with id = id associated to User.
- Returns only if User of the task and Authenticated User are same
- Authentication through token is mandatory.

**Request (JSON)**

```
{}
```

**Response (JSON)**

```
{
  "id": 0,
  "user": 0,
  "task_id": 0,
  "title": "string",
  "description": "string",
  "due_date": "2024-11-19",
  "status": "pending",
  "created_at": "2024-11-19T10:25:11.259Z",
  "updated_at": "2024-11-19T10:25:11.259Z"
}
```

## 4. PUT /task/{id}

**Success Status Code : 200**

- Updates entire task instance (with id) with new updated values provided.
- Updates only if User of the task and Authenticated User are same
- Authentication through token is mandatory.

**Request (JSON)**

```
{
  "title": "string",
  "description": "string",
  "due_date": "2024-11-19",
  "status": "pending"
}
```

**Response (JSON)**

```
{
  "id": 0,
  "user": 0,
  "task_id": 0,
  "title": "string",
  "description": "string",
  "due_date": "2024-11-19",
  "status": "pending",
  "created_at": "2024-11-19T10:25:11.259Z",
  "updated_at": "2024-11-19T10:25:11.259Z"
}
```

## 5. PATCH /task/{id}/completed

**Success Status Code : 200**

- Partial update of task instance (with id) with new updated values provided.
- Updates only if User of the task and Authenticated User are same
- Authentication through token is mandatory.

**Request (JSON)**

**Response (JSON)**

```
{
  "id": 0,
  "user": 0,
  "task_id": 0,
  "title": "string",
  "description": "string",
  "due_date": "2024-11-19",
  "status": "completed",
  "created_at": "2024-11-19T10:25:11.259Z",
  "updated_at": "2024-11-19T10:25:11.259Z"
}
```

## 6. DELETE /task/{id}

**Success Status Code : 204**

- Deletes the task instance with id.
- Deletes only if User of the task and Authenticated User are same
- Authentication through token is mandatory.

**Request (JSON)**

```
{
  "title": "string",
  "description": "string",
  "due_date": "2024-11-19",
  "status": "pending"
}
```

**Response (JSON)**

```
{
  "id": 0,
  "user": 0,
  "task_id": 0,
  "title": "string",
  "description": "string",
  "due_date": "2024-11-19",
  "status": "pending",
  "created_at": "2024-11-19T10:25:11.259Z",
  "updated_at": "2024-11-19T10:25:11.259Z"
}
```

## 7. POST /register

**Success Status Code : 201**

- User can Register providing the Email, Password and Name.
- To get Auth Token user should register with credentials.

**Request (JSON)**

```
{
  "email": "user@example.com",
  "password": "string",
  "name": "string"
}
```

**Response (JSON)**

```
{
  "email": "user@gmail.com",
  "name": "string"
}
```

## 8. POST /token

**Success Status Code : 201**

- Returns a Auth Token for a user which is used for Authentication.
- Credentials of the user are given as a Request.

**Request (JSON)**

```
{
  "email": "user@example.com",
  "password": "string"
}
```

**Response (JSON)**

```
{
  "token":
"3321e61f856073ad6ed907ada3d3745740ebf7b6"
}
```

## 9. GET /user

**Success Status Code : 200**

- Returns the details of present Authenticated User.
- Authentication through token is mandatory.

**Request (JSON)**

```
{
  "email":
"user@example.com",
  "password": "string",
  "name": "string"}
```

**Response (JSON)**

```
{
  "email": "user@gmail.com",
  "name": "string"
}
```

## 10. PUT /user

**Success Status Code : 200**

- Updates the entire profile data of the Authenticated user.
- Authentication through token is mandatory.
- Can Change password from here.

**Request (JSON)**

```
{
 "email": "user@example.com",
 "password": "string",
 "name": "string"
}
```

**Response (JSON)**

```
{
  "email": "user@gmail.com",
  "name": "string"
}
```

## 11. PATCH /user

**Success Status Code : 200**

- Partial update of profile data of Authenticated User with new updated values provided.
- Authentication through token is mandatory.
- Can Change Password from here.

**Request (JSON)**

```
{
 "email":
"user@example.com",
 "password": "string",
 "name": "string"
}
```

**Response (JSON)**

```
{
 "email": "user@example.com",
 "name": "string"
}
```

# Status Codes:

| Status Code | Type | Reason |
|---|---|---|
| 400 | Bad Request | The request was malformed or missing required parameters |
| 401 | Unauthorized | The API key provided was invalid or missing |
| 403 | Forbidden | You are not allowed to perform the task as user information mismatches |
| 500 | Internal Server Error | An unexpected error occurred on the server. |
| 201 | Created | The Data given has been stored into the database |
| 200 | OK | All the processes has gone well and output is given as desired |
| 204 | No Content | The instance that you are looking for has been deleted |

# Swagger Documentation

**All the 11 APIs can be tested with the integrated Swagger API in the website itself.**

## URL: /docs

### Points to be Noted:

1. **To Authenticate**
   a. Register with Email, Password, Name at /register.
   b. Get the Auth token by providing Email, Password at /token
   c. Click on the button **Authorize** located on the top of the APIs, scroll to **tokenAuth (apiKey)** and paste the token copied in there with a added Keyword "Token "
      i. **Example : token =** "abcdefghijklmnopqrstuvwxyz"
      ii. **Then paste** "Token abcdefghijklmnopqrstuvwxyz"
   d. Click **Authorize.**
2. **To Execute a API**
   a. Click on the API
   b. Click Try
   c. Enter the data (if applicable)
   d. click on Execute.
   e. Status code with response is seen below.
3. **To see the Schema**
   a. At the bottom of the page you can find all the schema of different requests.

# Steps to clone and Run the application

**Open Command Prompt and execute in the same order**

- **git clone {link}**

**With the folder path opened in command prompt**

**Create and Activate Virtual Environment**

- **python -m venv venv**
- **venv\Script\activate**

**Install the Requirements**

- **pip install -r requirements.txt**

**Migrate the Database**

- **python manage.py makemigrations**
- **python manage.py migrate**

**Run the Tests**

- **python manage.py test**

**Run the server**

- **python manage.py runserver**

Navigate to **http://127.0.0.1:8000/api/v1/docs/**