**Task Management API**

This Django-based Task Management API enables users to create tasks, assign tasks to users, and retrieve tasks assigned to specific users. Built using Django and Django Rest Framework (DRF), this API is simple, efficient, and extendable.

**Features**

1. **Task Creation**: Create tasks with a name and description.
2. **Assign Task to Users**: Assign tasks to one or multiple users.
3. **Retrieve Tasks for a Specific User**: Fetch all tasks assigned to a specific user.

**Project Structure**

project-directory/

│

├── tasks/ # App containing models, views, serializers, etc.

│ ├── migrations/ # Django migrations for database setup

│ ├── models.py # Task and User models

│ ├── serializers.py # Serializers for Task and User models

│ ├── views.py # API views for task management

│ └── urls.py # App-specific URL patterns

│

├── project\_name/ # Django project configuration

│ ├── settings.py # Project settings

│ ├── urls.py # Main URL configuration

│

├── manage.py # Django management commands

├── requirements.txt # Python dependencies

└── README.md # Project documentation (this file)

**Setup Instructions**

**1. Clone the Repository**

bash

git clone <repository-link>

cd <repository-directory>

**2. Set Up a Virtual Environment**

bash

python -m venv env

source env/bin/activate # On Windows: env\Scripts\activate

**3. Install Dependencies**

Install all the required Python libraries using:

bash

pip install -r requirements.txt

**4. Database Migration**

Run the following commands to set up the database:

bash

python manage.py makemigrations

python manage.py migrate

**5. Create a Superuser**

Create an admin account for the Django admin interface:

bash

python manage.py createsuperuser

Follow the prompts to create a username, email, and password.

**6. Run the Development Server**

Start the Django development server:

bash

python manage.py runserver

Your project will be available at http://127.0.0.1:8000.

**API Endpoints**

**1. Create a Task**

* **URL**: POST /api/tasks/create/
* **Request Example**:

json

{

"name": "Test Task",

"description": "This is a test task."

}

* **Response Example**:

json

{

"id": 1,

"name": "Test Task",

"description": "This is a test task.",

"created\_at": "2025-03-25T20:05:18.481885Z",

"task\_type": "",

"completed\_at": null,

"status": "Pending",

"assigned\_users": []

}

**2. Assign a Task to Users**

* **URL**: POST /api/tasks/assign/<task\_id>/
* **Request Example**:

json

{

"user\_ids": [1, 2]

}

* **Response Example**:

json

{

"message": "Task assigned successfully"

}

**3. Retrieve Tasks for a Specific User**

* **URL**: GET /api/tasks/user/<user\_id>/
* **Response Example**:

json

[

{

"id": 1,

"name": "Test Task",

"description": "This is a test task.",

"created\_at": "2025-03-25T20:05:18.481885Z",

"task\_type": "",

"completed\_at": null,

"status": "Pending",

"assigned\_users": [1, 2]

}

]

**Testing the API**

**Using curl**

You can use curl commands to test the API endpoints:

1. **Create a Task**:

bash

curl -X POST "http://127.0.0.1:8000/api/tasks/create/" \

-H "Content-Type: application/json" \

-d "{\"name\": \"Test Task\", \"description\": \"This is a test task.\"}"

1. **Assign a Task**:

bash

curl -X POST "http://127.0.0.1:8000/api/tasks/assign/1/" \

-H "Content-Type: application/json" \

-d "{\"user\_ids\": [1, 2]}"

1. **Retrieve Tasks for a User**:

bash

curl -X GET "http://127.0.0.1:8000/api/tasks/user/1/"

**Test Credentials**

Use the Django admin interface to add test users:

1. Access the admin panel at http://127.0.0.1:8000/admin/.
2. Log in with your superuser credentials.
3. Add users with the following details:

**File Overview**

**tasks/models.py**

Defines the User and Task models, including relationships and fields.

**tasks/serializers.py**

Defines serializers for User and Task to handle JSON serialization and deserialization.

**tasks/views.py**

Implements the API endpoints for task creation, assignment, and retrieval.

**tasks/urls.py**

Maps API endpoints to the respective views.