## Assignment Title: Task Management API

Your task is to build a simple Task Management API using Django. The API should allow users to create, update, delete, and retrieve tasks. Each task can have a title, description, status, priority level, and due date.

## Requirements:

1. **API Endpoints**:
    - **Create a task**:
      Endpoint: `POST /tasks/`
      Request Body:

```
{
  "title": "Task Title",
  "description": "Task description",
  "status": "pending",  // ["pending", "in-progress", "completed"]
  "priority": "high",    // ["low", "medium", "high"]
  "due_date": "2024-09-30"
}
```

    - **Retrieve all tasks**:
      Endpoint: `GET /tasks/`
    - **Retrieve a single task**:
      Endpoint: `GET /tasks/{task_id}/`
    - **Update a task**:
      Endpoint: `PUT /tasks/{task_id}/`
      Request Body (optional):

```
{
  "title": "Updated Title",
  "description": "Updated description",
  "status": "in-progress",
  "priority": "medium",
  "due_date": "2024-10-15"
}
```

- **Delete a task**:
  Endpoint: `DELETE /tasks/{task_id}/`

2. **Task Attributes**:

   - `title` (string, required)
   - `description` (text, optional)
   - `status` (choices: “pending”, “in-progress”, “completed”)
   - `priority` (choices: “low”, “medium”, “high”)
   - `due_date` (date, optional)

3. **Database Design**:

   - Design the models appropriately for the tasks, including fields for each attribute.

4. **Validation**:

   - Ensure that the due date cannot be in the past when creating or updating a task.
   - Status transitions should follow a logical flow (e.g., from “pending” to “in-progress” to “completed”).

5. **Authentication**:

   - Implement JWT-based authentication for accessing the API. Users should only be able to manage their own tasks.
   - Provide an endpoint for user registration and login:
     - **Register**: `POST /auth/register/`
     - **Login**: `POST /auth/login/`

6. **Sorting & Filtering**:

   - Add filtering options for tasks by status and priority.
   - Allow sorting tasks by due date, priority, or creation date.

## Bonus (Optional):

1. Implement pagination for the task retrieval endpoint.

2. Add support for assigning tasks to different users (many-to-one relationship).
3. Implement search functionality for tasks based on the title or description.

## Deliverables:

- A GitHub repository with the project code and a `README.md` file explaining how to set up and run the project.
- Clear instructions for API usage (you can provide a Postman collection or Swagger/OpenAPI documentation).
- Any extra assumptions or improvements you made to the original requirements.

## Evaluation Criteria:

1. Code quality, structure, and adherence to best practices.
2. Proper API design and usage of HTTP methods.
3. Correctness and efficiency of database design.
4. Handling of edge cases, validations, and error responses.
5. Bonus points for implementing the optional tasks.

## Tech Stack:

- Use **SQLite** or **PostgreSQL** as the database (preferably PostgreSQL).
- Use **JWT for authentication** (Django Rest Framework's JWT integration).

## Time to Complete:

You have **4 days** to complete this task.
Good luck!