# Setup Instructions:

* First of all created a Django project and activated the virtual env.
* Downloaded Django and required packages (DRF, Celery etc)
* Created the Django app
* Made changes to settings.py (shifted Secret key to .env file, Included Celery and Celery beat settings, JWT Auth settings).
* Setup the Urls.

# API Documentation:

## User Registration API (POST req):

"""

    Register a new user.

    Required fields:

    - username (string)

    - email (string)

    - role (string)

    - password (string)

    Returns:

    - Success message with user data if registration is successful.

    - Error message if the user already exists or invalid data.

    """

## User Login API:

"""

    Log in an existing user.

    Required fields:

    - username (string)

    - password (string)

    Returns:

    - A token for session authentication if login is successful.

    - Error message if login fails.

    """

## User Logout API:

"""

Log out the currently authenticated user.

Required fields:

- None (the request must be authenticated)

Returns:

- Success message indicating that the user has been logged out successfully.

- Error message if the user is not authenticated or if there is an issue with the logout process.

"""

## CRUD API’S

## Create Task API:

"""

    Create a new task.

    Required fields:

    - title (string): Title of the task

    - description (string): Description of the task

    - duedate (date): Due date of the task

    - priority (string): Priority level (High/Medium/Low)

    - category (string): Category the task belongs to

    - status (string): Task status (e.g., To Do, In Progress, Completed)

    - assigned\_to (user): User assigned to the task

    Returns:

    - Success message with task details.

"""

## Read ALL Task API (GET):

"""

    Get details of all tasks.

    Returns:

    - Task details if the task exists.

    """

## Read a Specific Task API (GET):

"""

    Get details of a task.

    Required fields:

    - task\_id (int): Task ID

    Returns:

    - Task details if the task exists.

    """

## Update Task API (PUT):

"""

    Update an existing task.

    Required fields (for both User and Admin):

    - status (string): New status of the task (e.g., "In Progress", "Completed")

    - tags (array of strings): Tags related to the task

    For Admins, additional fields can be updated (status, title, description, etc.).

    """

## Delete Task API (DELETE):

"""

    Delete an existing task.

    Required fields:

    - task\_id (int): Task ID

    Only the admin can delete it.

    """

## Filters and Search Api’s

## Priority filter API (GET):

"""

Filter tasks by priority.

Required fields:

- priority (string): The priority level of the tasks to filter (e.g., "High", "Medium", "Low").

Role-based behavior:

- Admins can view all tasks that match the given priority.

- Users can only view tasks assigned to them that match the given priority.

Returns:

- List of tasks matching the specified priority for the authenticated user.

- Error message if no tasks are found or if the user is not authenticated.

"""

## Category filter API (GET):

"""

Filter tasks by category.

Required fields:

- category (string): The category of the tasks to filter.

Role-based behavior:

- Admins can view all tasks that belong to the specified category.

- Users can only view tasks assigned to them that match the given category.

Returns:

- List of tasks matching the specified category for the authenticated user.

- Error message if no tasks are found or if the user is not authenticated.

"""

## Filter by Tag-name API (GET):

"""

Filter tasks by tag name.

Required fields:

- tags (string): The name of the tag to filter tasks by.

Role-based behavior:

- Admins can view all tasks that are associated with the specified tag.

- Users can only view tasks assigned to them that are associated with the specified tag.

Returns:

- List of tasks matching the specified tag for the authenticated user.

- Error message if no tasks are found for the given tag, if the tag does not exist, or if the user is not authenticated.

"""

## Filter by Search API (GET):

"""

Filter tasks by search query.

Required fields:

- search (string): The search term to filter tasks (e.g., task title, description, or tags).

Role-based behavior:

- Admins can view all tasks that match the search query.

- Users can only view tasks assigned to them that match the search query.

Returns:

- List of tasks matching the specified search query for the authenticated user.

- Error message if no tasks are found for the given search term or if the user is not authenticated.

"""

## Task Comment Api’s

## Add Comment API (POST):

"""

Create a comment on a task.

Required fields:

- task\_id (string or integer): The ID of the task to which the comment is being added.

- comment (string): The content of the comment to be added.

Role-based behavior:

- Admins can add comments to any task.

- Users can only add comments to tasks that are assigned to them.

Returns:

- Success message with the created comment details if the comment is added successfully.

- Error message if the task does not exist, if the user is not authorized to comment on the task, or if any required fields are missing.

"""

## View Comment API (GET):

"""

Retrieve comments for a specific task.

Required fields:

- task\_id (string or integer): The ID of the task for which comments are being retrieved.

Role-based behavior:

- Admins can view all comments for any task.

- Users can only view comments for tasks assigned to them.

Returns:

- List of comments associated with the specified task.

- Error message if the task does not exist, if there are no comments for the task, or if the user is not authenticated.

"""

## SubTask Api’s

## Create Subtask API (POST):

"""

Create a subtask under a main task.

Required fields:

- task\_id (string or integer): The ID of the main task under which the subtask is to be created.

- subtask\_title (string): The title of the subtask to be created.

Role-based behavior:

- Admins can create subtasks under any main task.

- Users can only create subtasks under tasks assigned to them.

Returns:

- Success message with the created subtask details if the subtask is added successfully.

- Error message if the main task does not exist, if the user is not authorized to create a subtask under the main task, or if any required fields are missing.

"""

## View Subtask API (GET):

"""

Retrieve subtasks for a specific main task.

Required fields:

- task\_id (string or integer): The ID of the main task for which subtasks are being retrieved.

Role-based behavior:

- Admins can view all subtasks for any main task.

- Users can only view subtasks for main tasks assigned to them.

Returns:

- List of subtasks associated with the specified main task.

- Error message if the main task does not exist, if there are no subtasks for the main task, or if the user is not authenticated.

"""

## Track Status Api’s

## Track status API (POST):

"""

Update the status of a task.

Required fields:

- task\_id (string or integer): The ID of the task for which the status is being updated.

- status (string): The new status of the task.

Role-based behavior:

- Admins can update the status of any task.

- Users can only update the status of tasks assigned to them.

Returns:

- Success message with the updated task details if the status is updated successfully.

- Error message if the task does not exist, if the user is not authorized to update the task status, or if the status is invalid.

"""

## Due Date Extension Api’s

## Due Date Extension API (POST):

"""

Request an extension on a task's due date.

Required fields:

- task\_id (string or integer): The ID of the task for which the extension is being requested.

- new\_due\_date (string): The requested new due date in ISO 8601 format (YYYY-MM-DD).

Role-based behavior:

- Users can request an extension on tasks assigned to them.

- Admins can view and manage all extension requests.

Returns:

- Success message with the details of the extension request if the request is submitted successfully.

- Error message if the task does not exist, if the user is not authorized to request an extension for the task, or if the new due date is invalid.

"""

## Dashboard Api’s

## Dashboard API (GET):

"""

Retrieve user dashboard data, displaying task statistics.

Endpoint:

- GET /dashboard/

Returns:

- Summary statistics of tasks.

Role-based behavior:

- Only admins can access the dashboard.

Error Handling:

- Returns an error message if the user is not authenticated or if there are issues retrieving the data.

"""

# Assumptions and notes:

It is a role based system in which there is an admin and a user. And system works accordingly for both user and admin.