# **Daily Internship Report – April 23 - Wednesday**

**Date:** April 23 wednesday  
**Intern Name:** Pankaj Basnet  
**Department/Team:** Python Django  
**Mentor:** Rohan Raj Shrestha

### **Task Overview:**

1. Git

Objective: Learn modules, exception handling, and Git basics.

Topics: List/dict comprehensions, os, datetime, Git workflow.

Tasks:

Write a mini file organizer that moves files by extension.

Push all code to GitHub.

Daily git practice: branch creation, commits, pull/push.

1. Django Introduction

Objective: Set up and understand a Django project.

Topics: MTV architecture, project vs app, admin panel.

Tasks:

Set up a blog project.

Add 2 models: Post and Comment.

Enable CRUD via admin panel.

**Assignment:**

**Write a mini file organizer that moves files by extension.:**

### **2. Tasks Completed & In Progress**

#### 

#### **[Task 1: mini file organizer that moves files by extension.]**

Step 1: Setup

* Define paths for the source and destination directories.
* Convert these paths to Path objects for easier manipulation.

#### Step 2: List Directory Contents

#### Use Path.iterdir() to list contents of the source and destination directories.

#### Convert the iterators to lists to avoid exhaustion.

#### Step 3: Define move\_file\_to\_destination Function

#### Purpose: Moves a file to a subdirectory based on its extension.

#### Logic:

#### Construct the path for the subdirectory using the file's extension.

#### Check if the subdirectory exists:

#### If it does, move the file there.

#### If it doesn't, create the subdirectory and then move the file.

#### Step 4: Iterate Over Source Directory

#### Loop through each item in the source directory.

#### For each file:

#### Extract its extension.

#### If the extension is in the predefined list (["txt", "py", "md"]), call move\_file\_to\_destination.

**Repo Link: https://github.com/pankaj-basnet/internship-projects--April-2025--Django-/blob/main/250423--week-2--/task01/task\_1\_e\_\_SEND\_FOR\_REPORT\_.py**

**Task 2: Set up and understand a Django project**

Set up a blog project..

**DJANGO PROJECT SETUP**

* + Install Django using pip install django.
  + Check installation with django-admin –version
  + Create a Django project called “blogapp”

django-admin startproject blogapp

* + Navigate into the project folder

cd blogapp

* + Start a new Django app named blog

python manage.py startapp blog

* + .Understand that the project has the configuration like settings.py

Add 'blog' to INSTALLED\_APPS in blogapp /settings.py.

* + Run initial migrations

python manage.py migrate

* + Run the development server

python manage.py runserver

* + Visit http://127.0.0.1:8000/ in your browser to check if it's working.
* MODELS (MTV - Model)
  + Open blog/models.py and define the Post model

class Post(models.Model):

title = models.CharField(max\_length=100)

content = models.TextField()

created\_at = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.title

* + Define the Comment model below Post named, variable and default arguments

class Comment(models.Model):

post = models.ForeignKey(Post, on\_delete=models.CASCADE, related\_name='comments')

author = models.CharField(max\_length=50)

content = models.TextField()

created\_at = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return f'Comment by {self.author}'

* + Understand ForeignKey relationship from Comment to Post (Many-to-One)
  + python manage.py makemigrations blog to generate migration files
  + Apply migrations with python manage.py migrate.
* DJANGO ADMIN PANEL (Enabling CRUD)

1. python manage.py createsuperuser

enter username, email, and password.git add: Stages changes to be committed.

1. Open blog/admin.py.
2. Register the Post model:

admin.site.register(Post)