# Summer Fellowship Report

On

**Migrating python FOSSEE public website from Django to Wagtail CMS**

Submitted by

**Sohum Damani**

Under the guidance of

**Mr. Ankit R. Javalkar**
FOSSEE

August 29, 2022

## Acknowledgment

I want to express my gratitude to everyone who has assisted me in finishing this project. I am grateful to **Mr. Ankit R. Javalkar** and the entire FOSSEE team at IIT Bombay for giving me the chance to work on this project. I also want to express my gratitude to him for always supporting me and guiding me in the correct route. His advice has been crucial in assisting me in finishing the job considerably ahead of schedule. Additionally, I want to thank him for his advice and helpful recommendations. Last but not least, my internship experience both memorable and a great learning experience.

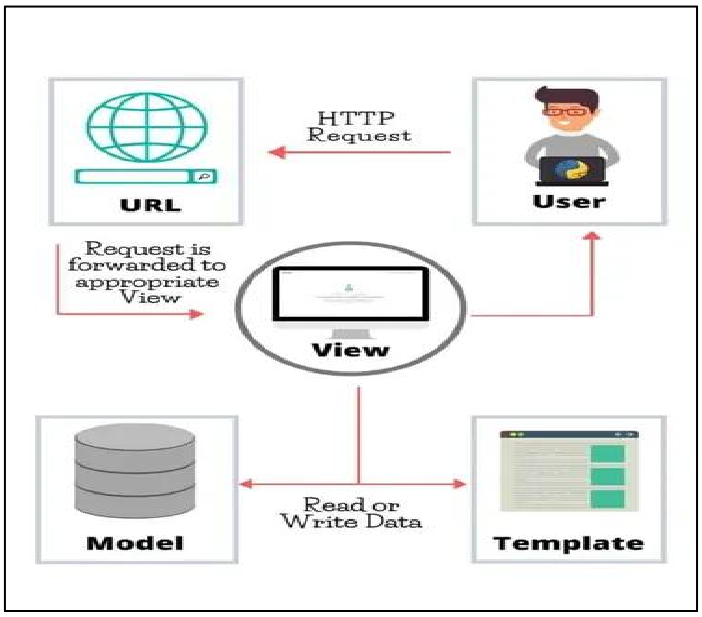
# Contents

# Chapter 1

# Introduction

## 1.1 Django

Currently python FOSSEE website is build completing using the Python Web Framework named Django. It works on the MVT(Model View Template) structure. In this the user sends an HTTP request which is mapped it to available url routes. Each url route is linked to view function which acts as a mediator to get required data from the database model and send the data to the template. This then creates a response page which is visible to the user.



## 1.2 Wagtail

The Wagtail is a Python based CMS made for Django, released in 2015 by a digital agency named Torchbox.Ai.CMS stands for Content Management System which allows us to create,modify and share digital content.Similar to Django it has an admin panel where one can create new pages or edit the existing pages.It helps to organize your content in a tree structure which can have any structure.

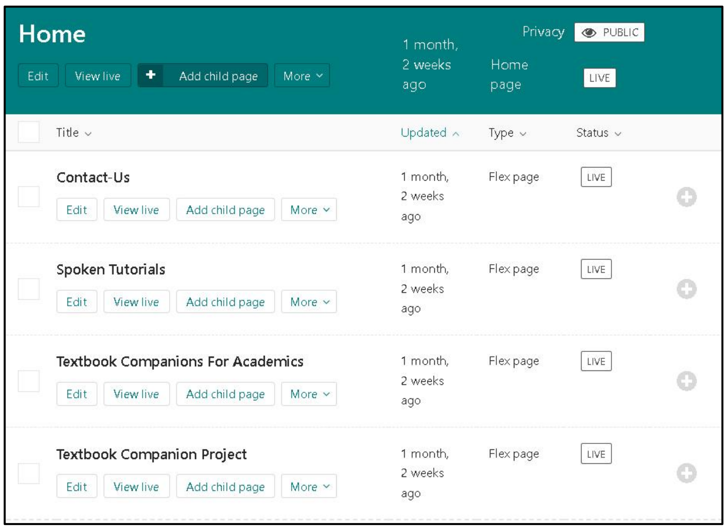
# Chapter 2

# Benefits of Wagatil

## ➢ Managing Pages:

One of the features in wagtail is to manage the tree structure of a page. Wagtail has an root page generally the home page of an website. This root page can have multiple child pages .

As we can see we have an option to edit, add pages, unpublish a page and make a duplicate copy of the page.
One more useful feature is to schedule when a page should be published and till when should it be visible.



Home — 1 month, 2 weeks ago — Home page — LIVE

| Title | Updated | Type | Status |
|---|---|---|---|
| Contact-Us | 1 month, 2 weeks ago | Flex page | LIVE |
| Spoken Tutorials | 1 month, 2 weeks ago | Flex page | LIVE |
| Textbook Companions For Academics | 1 month, 2 weeks ago | Flex page | LIVE |
| Textbook Companion Project | 1 month, 2 weeks ago | Flex page | LIVE |

## ➢ User Management:

From the back office one can manage and give access to your team members to edit content.
The roles you can assign to a user are :

- Administrator: you grant him all the access to the CMS to manage the pages, the content, the brand or the settings.
- Moderator: he can validate the work of the editor and then publish.
- Editor: he can write and modify content and a page, but he can't publish it.

# Chapter 2

- ## Embed:

  Wagtail supports generating embed code from URLs to content on external providers such as Youtube or Twitter. By default, Wagtail will fetch the embed code directly from the relevant provider’s site using the Embed protocol.This feature allows you to embed an external piece of content in your page like :

  - Imgaes
  - Documents
  - Web pages

- ## Workflow:

  Workflows allow you to configure how moderation works on your site. Workflows are sequences of tasks, all of which must be approved before the workflow completes.When editing a task, you may find that some fields - including the name - are uneditable. This is to ensure workflow history remains consistent - if you find yourself needing to change the name, it is recommended that you disable the old task, and create a new one with the name you need. Disabling a task will cause any pages currently in moderation on that task to skip to the next task. Some common task can be proofreading,drafting , publishing etc.

# Chapter 3

# Environment Setup

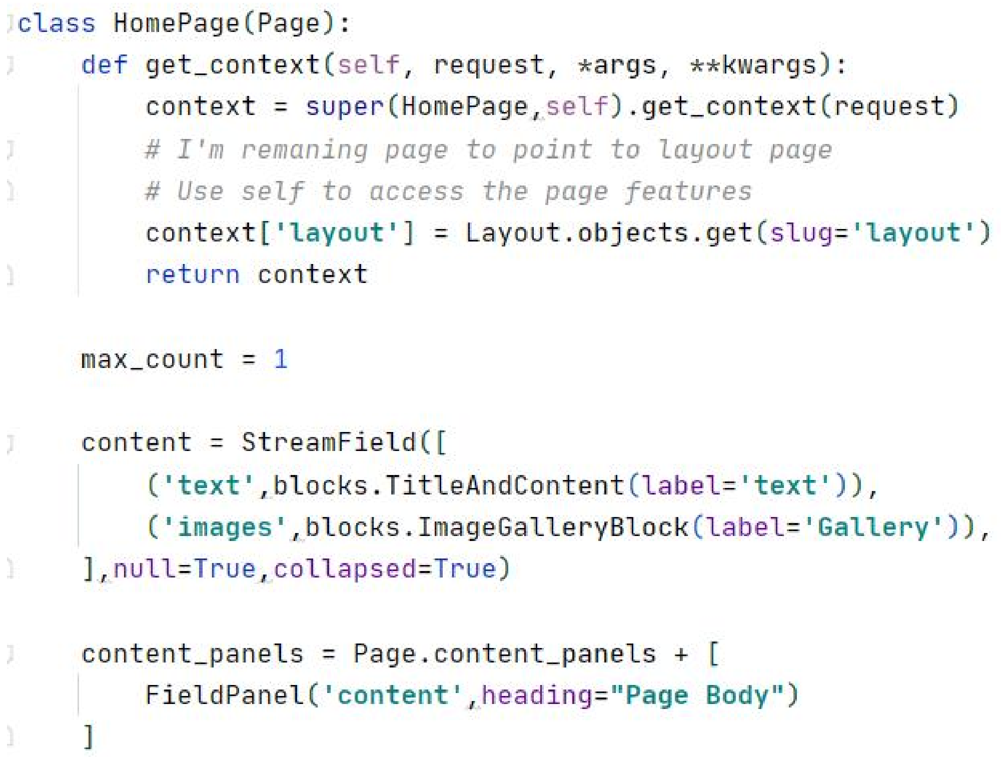
- Create and activate a virtual environment
- Install Wagtail
- Generate your website and setup database
- Create a page Model
- Create superuser
- Run server

# Chapter 4

7

# Page Model

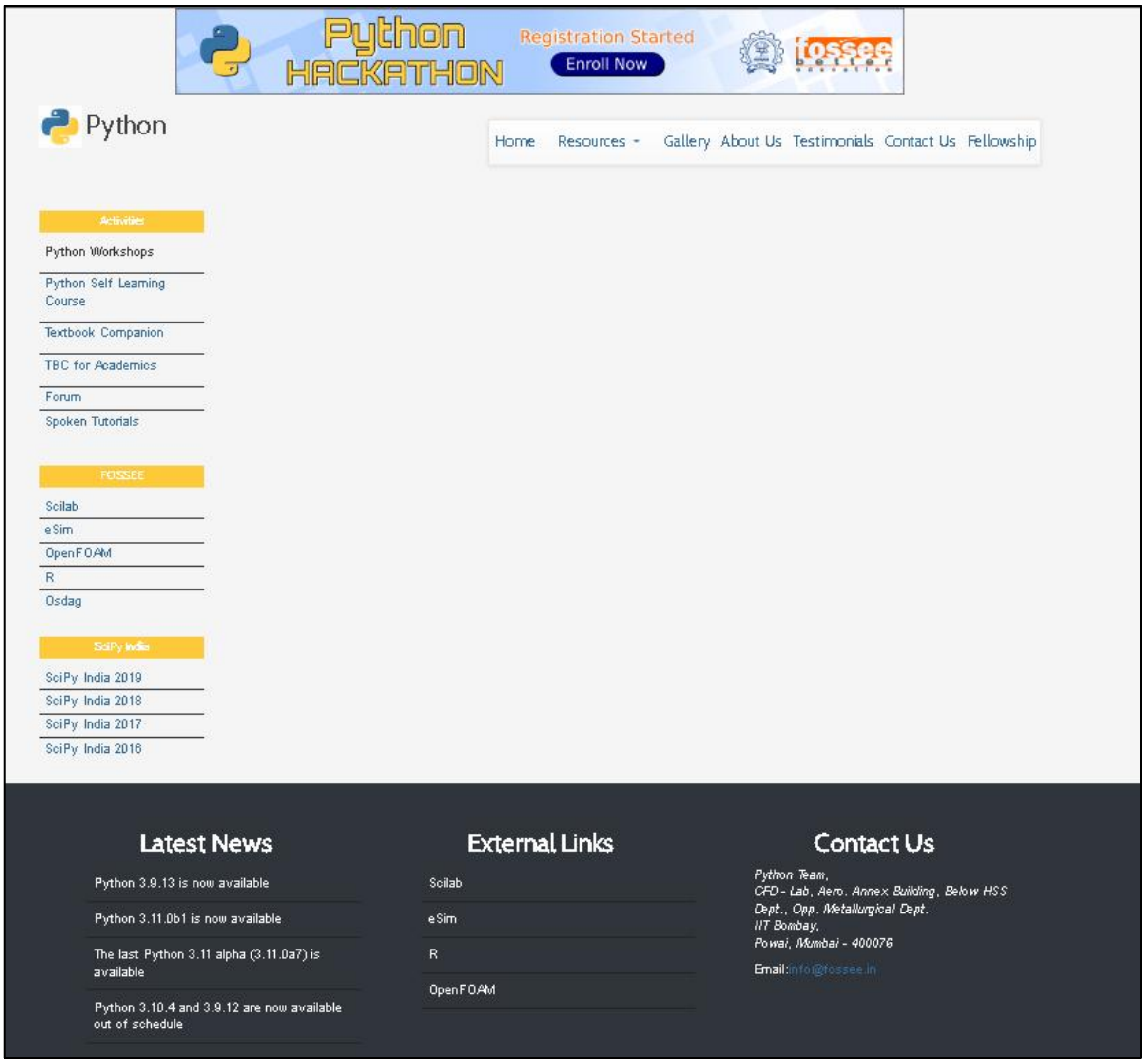
Just like Django Model wagtail has it’s own Page Model . In this we can add the features available in Django and some extra features like RichText and Stream Fields provided by Wagatil.
There are 3 page editor interface:
- ✓ content_panels - For content, such as main body text
- ✓ promote_panels - For metadata, such as tags, thumbnail image and SEO title
- ✓ settings_panels - For settings, such as publish date



```
class HomePage(Page):
    def get_context(self, request, *args, **kwargs):
        context = super(HomePage,self).get_context(request)
        # I'm remaning page to point to layout page
        # Use self to access the page features
        context['layout'] = Layout.objects.get(slug='layout')
        return context

    max_count = 1

    content = StreamField([
        ('text',blocks.TitleAndContent(label='text')),
        ('images',blocks.ImageGalleryBlock(label='Gallery')),
    ],null=True,collapsed=True)

    content_panels = Page.content_panels + [
        FieldPanel('content',heading="Page Body")
    ]
```

FieldPanel is used to register a feature in a page and be visible in the admin panel.

This is the layout page which is inherited by every other page. It consist of a banner image, navbar,sidebar and footer. Each of them is a snippet.



Activities

Python Workshops
Python Self Learning Course
Textbook Companion
TBC for Academics
Forum
Spoken Tutorials

FOSSEE

Scilab
eSim
OpenFOAM
R
Osdag

SciPy India

SciPy India 2019
SciPy India 2018
SciPy India 2017
SciPy India 2016

Latest News

Python 3.9.13 is now available
Python 3.11.0b1 is now available
The last Python 3.11 alpha (3.11.0a7) is available
Python 3.10.4 and 3.9.12 are now available out of schedule

External Links

Scilab
eSim
R
OpenFOAM

Contact Us

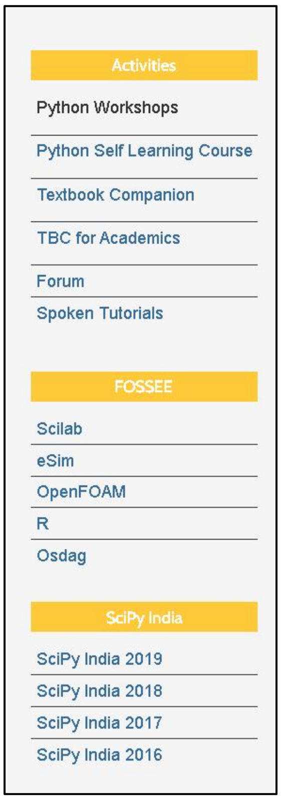
*Python Team,*
*CFD - Lab, Aero. Annex Building, Below HSS Dept., Opp. Metallurgical Dept.*
*IIT Bombay,*
*Powai, Mumbai - 400076*

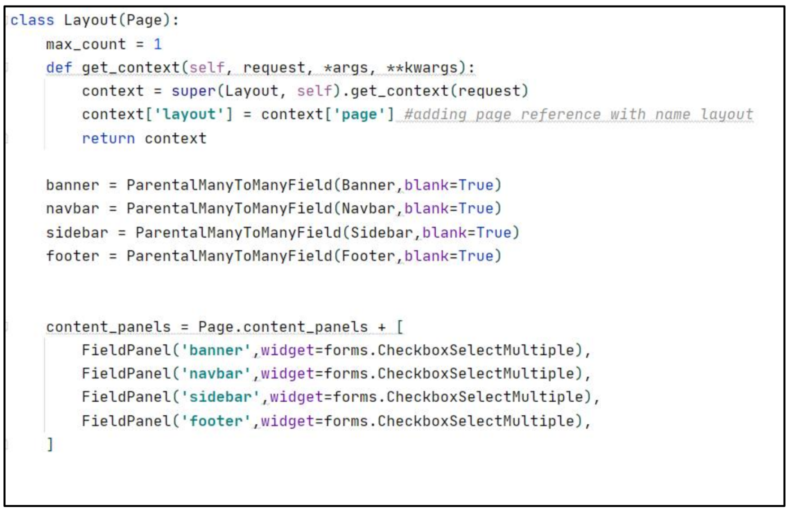
Email:info@fossee.in

# Chapter 5

## Snippets

Snippets are pieces of content which do not necessitate a full web page to render. They could be used for making secondary content, such as headers, footers, and sidebars. I have created banner,navbar,sidebar and footer as a snippets. These snippets are than called in the layout page which is an base page for every other page.

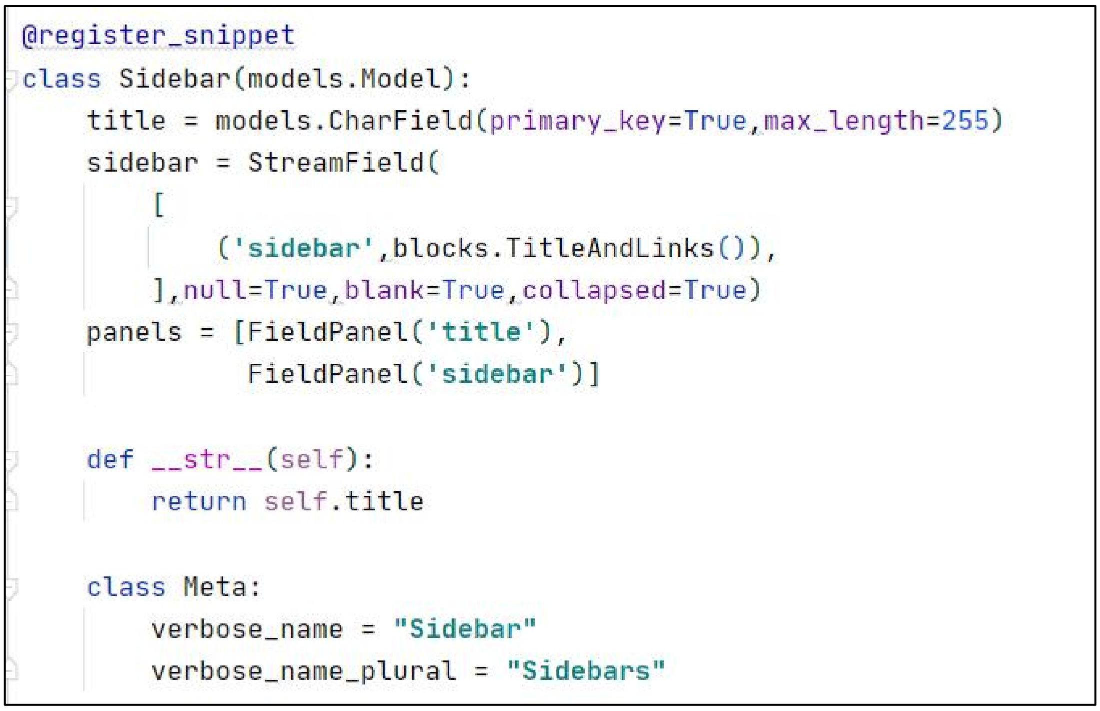




```
class Layout(Page):
    max_count = 1
    def get_context(self, request, *args, **kwargs):
        context = super(Layout, self).get_context(request)
        context['layout'] = context['page'] #adding page reference with name layout
        return context

    banner = ParentalManyToManyField(Banner,blank=True)
    navbar = ParentalManyToManyField(Navbar,blank=True)
    sidebar = ParentalManyToManyField(Sidebar,blank=True)
    footer = ParentalManyToManyField(Footer,blank=True)


    content_panels = Page.content_panels + [
        FieldPanel('banner',widget=forms.CheckboxSelectMultiple),
        FieldPanel('navbar',widget=forms.CheckboxSelectMultiple),
        FieldPanel('sidebar',widget=forms.CheckboxSelectMultiple),
        FieldPanel('footer',widget=forms.CheckboxSelectMultiple),
    ]
```

# Chapter 5

By using @register_snippet one can assign a model as a snippet. Here sidebar has two basic feature title and sidebar.
This sidebar is a stream field whose main feature is to enable user to add as many internal and external links under one title.
These link can than be used to navigate to different website.



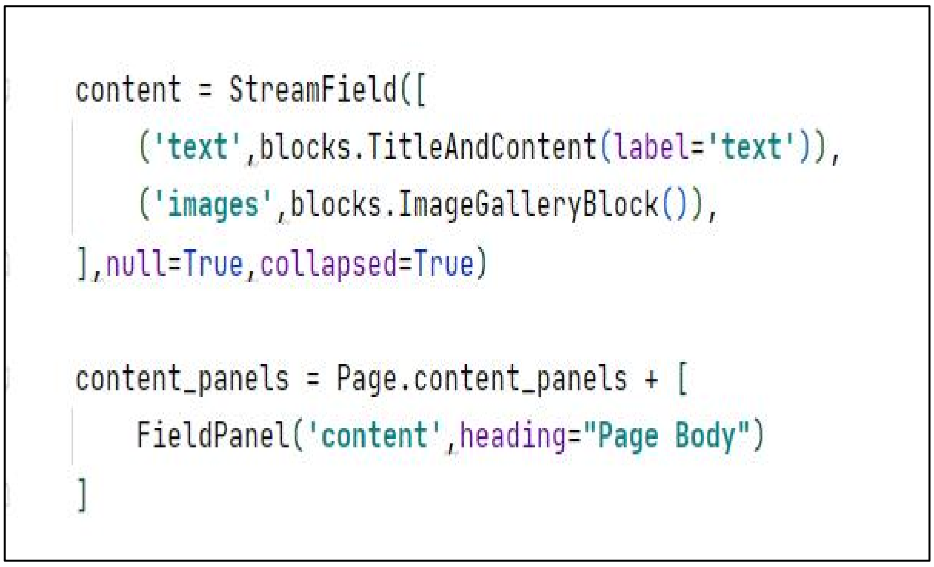
```
@register_snippet
class Sidebar(models.Model):
    title = models.CharField(primary_key=True,max_length=255)
    sidebar = StreamField(
        [
            ('sidebar',blocks.TitleAndLinks()),
        ],null=True,blank=True,collapsed=True)
    panels = [FieldPanel('title'),
              FieldPanel('sidebar')]

    def __str__(self):
        return self.title

    class Meta:
        verbose_name = "Sidebar"
        verbose_name_plural = "Sidebars"
```

# Chapter 6

# Stream Fields

Stream Field provides a content editing model suitable for pages that do not follow a fixed structure. Within the database, the Stream Field content is stored as JSON, ensuring that the full informational content of the field is preserved, rather than just an HTML representation of it..Most of the model field are available as Stream Field block.

Here content is a stream field which has two types of block text and images . In text block custom text with different size and styles like italics,bold and bullets.



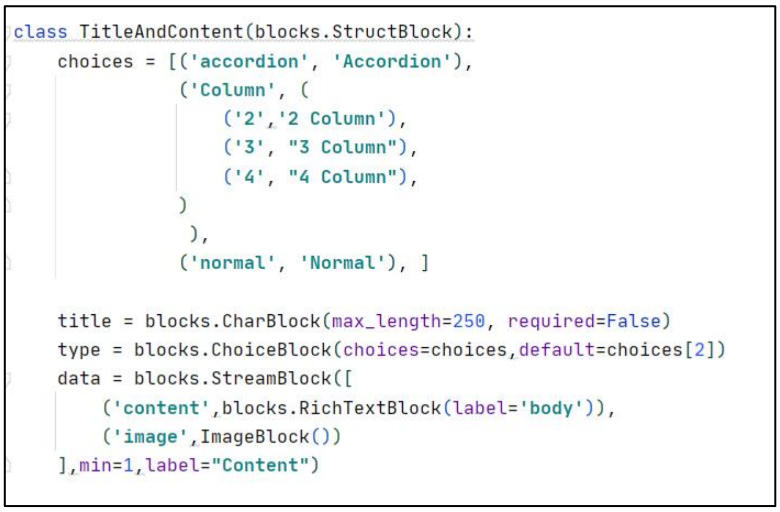
```
content = StreamField([
    ('text',blocks.TitleAndContent(label='text')),
    ('images',blocks.ImageGalleryBlock()),
],null=True,collapsed=True)

content_panels = Page.content_panels + [
    FieldPanel('content',heading="Page Body")
]
```

In addition to using the built-in block types directly within StreamField, it's possible to construct new block types by combining sub-blocks in various ways. Once a new block type has been built up in this way, you can use it anywhere where a built-in block type would be used - including using it as a component for yet another block type.

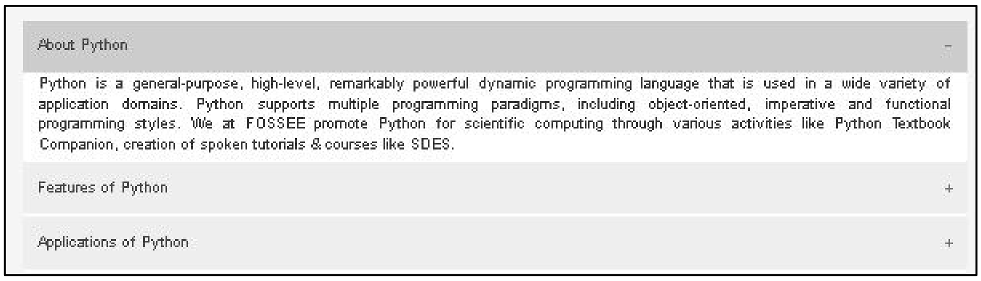
# Chapter 6

## ➢ Struct Block

Struct Block allows you to group several ‘child’ blocks together to be presented as a single block.
Here the type block have options in which the text content will be visible .



```
class TitleAndContent(blocks.StructBlock):
    choices = [('accordion', 'Accordion'),
               ('Column', (
                   ('2','2 Column'),
                   ('3', "3 Column"),
                   ('4', "4 Column"),
               )
                ),
               ('normal', 'Normal'), ]

    title = blocks.CharBlock(max_length=250, required=False)
    type = blocks.ChoiceBlock(choices=choices,default=choices[2])
    data = blocks.StreamBlock([
        ('content',blocks.RichTextBlock(label='body')),
        ('image',ImageBlock())
    ],min=1,label="Content")
```

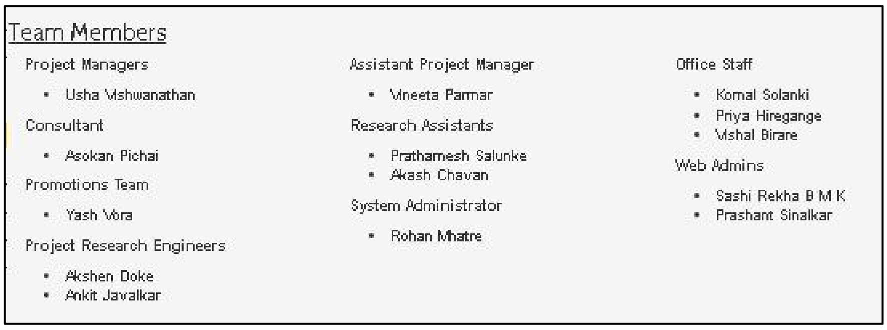


About Python –

Python is a general-purpose, high-level, remarkably powerful dynamic programming language that is used in a wide variety of application domains. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming styles. We at FOSSEE promote Python for scientific computing through various activities like Python Textbook Companion, creation of spoken tutorials & courses like SDES.

Features of Python +

Applications of Python +



Team Members

Project Managers
- Usha Vishwanathan

Consultant
- Asokan Pichai

Promotions Team
- Yash Vora

Project Research Engineers
- Akshen Doke
- Ankit Javalkar

Assistant Project Manager
- Vineeta Parmar

Research Assistants
- Prathamesh Salunke
- Akash Chavan

System Administrator
- Rohan Mhatre

Office Staff
- Komal Solanki
- Priya Hiregange
- Vishal Birare

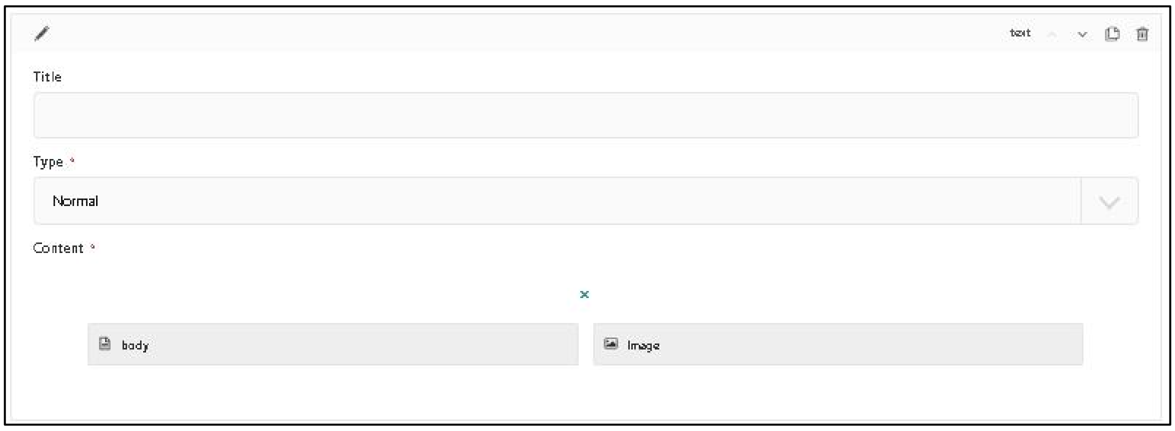
Web Admins
- Sashi Rekha B M K
- Prashant Sinalkar

Here is an example where I have implemented the accordion and column feature

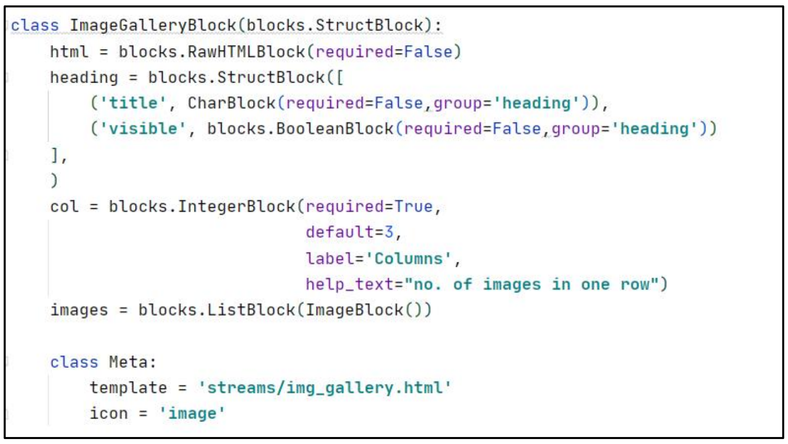
# Chapter 6

## ➢ Stream Block

Stream Block defines a set of child block types that can be mixed and repeated in any sequence, via the same mechanism as StreamField itself. This feature I have used multiple times to create multi line accordion like Textbook Companion page. Rich Text Block used inside a stream block is gives the user freedom to write the body as in writing in a word file and Django takes care of converting it to HTML.



## ➢ List Block



```
class ImageGalleryBlock(blocks.StructBlock):
    html = blocks.RawHTMLBlock(required=False)
    heading = blocks.StructBlock([
        ('title', CharBlock(required=False,group='heading')),
        ('visible', blocks.BooleanBlock(required=False,group='heading'))
    ],
    )
    col = blocks.IntegerBlock(required=True,
                              default=3,
                              label='Columns',
                              help_text="no. of images in one row")
    images = blocks.ListBlock(ImageBlock())

    class Meta:
        template = 'streams/img_gallery.html'
        icon = 'image'
```

List Block defines a repeating block, allowing content authors to insert as many instances of a particular block type as they like. In the gallery page I have used this to arrange images in a single row. Here I have also added an HTML block to customize the view of the images in the template.

# Chapter 7

## Conclusion

- Anyone having access to admin page and permission to update the website content without taking care of the HTML code behind it .
- Using the workflow team can be divided into small groups and assign specific task like editor,proofreading,moderator etc.
- Adding basic styles like accordion or row/col to text can be done with ease.
- Adding new updates regarding the upcoming events can be done by scheduling the changes in the page beforehand.
- Updating the website basic layout can done easily without deleting the existing layout.
- Creating a new page is just one click away and can be saved without publishing as a draft.

# Reference

[1] *Wagtail Documentation* - URL: https://docs.wagtail.org/en/stable/

[2] *Git documentation*. URL: https://git-scm.com/docs/git.

[3] *Github Documentation*. URL: https://docs.github.com/en/get-started/quickstart/hello-world.

[4] Github Repository : URL: https://github.com/SohumDamani/Fossee-Website