INTRODUCTION

In this digital age, the seamless exchange of information and data has become an integral part of our personal and professional lives. As individuals and organizations increasingly rely on sharing various types of files, the need for a reliable, efficient, and secure file sharing and downloading platform becomes evident. To address this demand, the **"File Sharing and Downloading Web Application"** is developed as a comprehensive solution that empowers users to effortlessly share, manage, and access files with utmost ease and security.

The aim of this project is to create a user-friendly web application that caters to the diverse file sharing needs of individuals, teams, and businesses. With a robust and intuitive interface, the platform is designed to facilitate seamless navigation and interaction for users of all technical backgrounds. The application's primary objective is to provide a centralized space where users can securely upload, organize, and share various files, ensuring that information flow is efficient and streamlined.

A key aspect of the **"File Sharing and Downloading Web Application"** is the implementation of a comprehensive user account system. Through this system, users can register, create personalized profiles, and manage their account information. This approach enhances data security and privacy while enabling users to customize their sharing preferences and access controls.

In conclusion, the **"File Sharing and Downloading Web Application"** aims to revolutionize the way users share, access, and collaborate on files. By offering a secure, user-friendly, and feature-rich platform, this project seeks to enhance productivity, foster collaboration, and elevate the overall file sharing experience for users from various walks of life.

1.1 OBJECTIVE

The primary objective of the **"File Sharing and Downloading Web Application"** is to create a robust, user-friendly, and secure platform that facilitates seamless sharing and downloading of files among users. The project aims to address the need for a reliable file sharing solution, enabling individuals and businesses to efficiently exchange various types of files, regardless of their size or format. The key objectives of the project are as follows:

1. User-Friendly Interface: The web application will be designed with an intuitive and user-friendly interface to ensure easy navigation and a smooth user experience. Users will be able to perform file sharing and downloading tasks with minimal effort and technical expertise.

2. Account Creation and Management: The project will provide a user account system, allowing users to register, create their profiles, and manage their personal information. User authentication and access control mechanisms will be implemented to ensure data security and privacy.

3. File Upload and Organization: Users will have the ability to upload multiple files of various formats and organize them into folders for easy retrieval and management. The application will support file versioning to allow users to track changes and revisions.

4. File Searching and Filtering: A robust search functionality will be implemented to enable users to quickly find specific files based on keywords, file types, or other relevant criteria. Advanced filtering options will enhance the efficiency of file discovery.

5. Download Management: The project will generate unique and secure download links for shared files, ensuring that only authorized users can access the content. The application will track download activities and provide usage statistics to users.

6. Commenting and Feedback: Users will be able to interact through comments and feedback on shared files, promoting collaboration and communication among users. The platform will foster an interactive environment for discussions and knowledge sharing.

7. Data Security and Privacy: The project will implement robust security measures to safeguard user data and prevent unauthorized access. Data encryption, secure socket layer (SSL) technology, and best practices in data storage will be employed to ensure data privacy.

8. Scalability and Performance: The application will be designed to handle a large number of users and files with high performance. Scalability measures will be implemented to accommodate future growth and increased user demands.

9. API Integration: The project will leverage DjangoRestFramework to create API endpoints, allowing seamless integration with external applications and services, enhancing the application's versatility.

10. Documentation and Support: Comprehensive documentation will be provided, detailing the application's functionalities, installation, and usage instructions. Additionally, user support will be offered to address any queries or issues that may arise during usage.

\

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The existing system of online file sharing often involves navigating multiple platforms to find specific files. Users may encounter challenges when searching for desired items, leading to delays and inconvenience. Moreover, there is little to no interaction between file owners and downloaders, making it challenging to address issues like file quality or missing files. Additionally, hidden costs and shipping charges may discourage users from using existing platforms.

Disadvantages of Existing System :

i. Unprofessional management of some existing websites and computer systems for handling payments.

ii. Delays in the delivery of products, affecting customer satisfaction.

iii. Limited interaction between the actual seller and the customer, hindering personalized service.

iv. Receiving products that are not up to the mark or of poor quality.

v. Hidden costs and shipping charges of the products, leading to unexpected expenses for customers.

vi. Lack of a straightforward process for returning products.

vii. Absence of sales assistants and their assistance, resulting in reduced customer support.

<u>2.2 PROPOSED SYSTEM</u>

File Sharing and Downloading WebApplication" aims to address the limitations of the existing systems. It provides a user-friendly, one-stop solution for file sharing and downloading needs. With a multi-vendor e-Commerce approach, the platform connects users with a vast variety of files, making it easier for sellers to offer their products and buyers to find what they need. The application ensures on-time delivery, streamlining the process for users during urgent situations like pandemics.

Advantages of Proposed system

i. User-friendly interface for easy navigation.

ii. Time-saving, as users can visit multiple stores (file owners) simultaneously.

iii. Easy comparison of file options and prices.

iv. Enhanced interaction between file owners and downloaders.

v. User queries and feedback management for improved customer satisfaction.

2.3 REQUIREMENTS

Hardware Required:

• Processor: Intel® Core i5 7th generation

• RAM: 4GB

• Storage: 1 TB

Software Used:

• Front End: HTML, CSS, JS, Bootstrap, API

• Back End: Python, Django, Django Rest Framework

• Localhost: Django Server

• OS Platform: Windows 10

SOFTWARE SPECIFICATION

HTML:

HTML (Hypertext Markup Language) is used to create electronic documents displayed on the World Wide Web. It provides the structure for web pages and enables hyperlinking. The project utilizes HTML5, which offers better semantics and dynamic elements.

CSS:

CSS (Cascading Style Sheets) is used to describe reusable styles for presenting documents written in markup language. It allows developers to alter the layout and appearance of web pages. CSS helps create visually appealing and consistent designs.

JavaScript:

JavaScript is a high-level, interpreted programming language used for web development. It enhances the interaction of users with webpages, making them more dynamic and interactive.

SQLite:

SQLite is one of the most popular and easy-to-use relational database systems. It possesses many features over other relational databases.SQLite is an embedded, server-less relational database management system. It is an in-memory open-source library with zero configuration and does not require any installation.

Bootstrap:

Bootstrap is a free and open-source front-end framework used for designing responsive and visually appealing websites and web applications.

Python:

Python is a versatile and powerful programming language used for various purposes, including web development. It provides a clean and easy-to-read syntax, making it popular among developers for building robust applications.

Django:

Django is a high-level Python web framework that follows the "Don't Repeat Yourself" (DRY) principle, enabling rapid development and clean, pragmatic design. It provides a rich set of built-in features for handling common web development tasks, such as URL routing, database management, and template rendering.

Django Rest Framework (DRF):

Django Rest Framework is an extension of Django that simplifies the process of building RESTful APIs. It enables seamless integration between the front-end and back-end of the web application, allowing smooth data communication between the client and the server.

FEASIBLITY STUDY

4.1 INTRODUCTION:

The feasibility study is a crucial step in assessing the viability of the "File Sharing and Downloading Web Application" project. It involves evaluating the practicality and potential success of the project from various perspectives. This section presents a comprehensive feasibility study that delves into the operational, technical, and economic aspects of the proposed web application.

4.2 OPERATIONAL FEASIBLITY:

The operational feasibility study assesses the project's alignment with the organization's goals, objectives, and requirements. It examines whether the application's functionalities and features meet the needs of the target users effectively. In the case of the "File Sharing and Downloading Web Application," the operational feasibility is favorable for the following reasons:

User-Centric Approach: The application is designed with a user-centric approach, ensuring that it caters to the diverse file sharing requirements of individuals, teams, and businesses. Extensive research and user feedback have influenced the application's design to offer an intuitive and seamless user experience.

Scalability and Flexibility: The web application is built to be scalable and flexible, enabling it to accommodate a growing user base and adapt to changing demands. Its modular architecture allows for easy integration of new features and enhancements in the future.

Comprehensive User Account System: The application's robust user account system ensures that users have full control over their files and sharing preferences. It fosters user engagement and collaboration while maintaining data security and privacy.

Interactive Feedback Mechanism: The incorporation of an interactive commenting and feedback mechanism promotes effective communication and collaboration among users, enhancing the overall file sharing experience.

4.3 TECHNICAL FEASIBLITY:

The technical feasibility study evaluates the practicality of implementing the proposed project from a technological standpoint. It analyzes the availability of resources, technology stack, and expertise required for successful development and deployment. The "File Sharing and Downloading Web Application" demonstrates strong technical feasibility due to the following factors:

Appropriate Technology Stack: The project utilizes Django and DjangoRestFramework, which are well-established and widely-used frameworks for web application development. These technologies offer extensive features, libraries, and community support.

API Integration: The application's integration with DjangoRestFramework enables the creation of robust API endpoints, facilitating seamless communication with other applications and services.

Data Security Measures: The project prioritizes data security, implementing encryption protocols, and SSL technology to safeguard sensitive user information and shared files.

Performance Optimization: The application is designed with performance optimization in mind, ensuring fast response times and smooth user interactions even with large file volumes and concurrent users.

4.4 ECONOMIC FEASIBLITY:

The economic feasibility study focuses on evaluating the financial viability of the project. It considers the estimated project cost, potential return on investment (ROI), and cost-benefit analysis. The "File Sharing and Downloading Web Application" demonstrates strong economic feasibility due to the following factors:

Cost-Effective Development: The utilization of open-source technologies like Django and DjangoRestFramework significantly reduces development costs without compromising the application's quality.

Potential Revenue Streams: The application's scalable nature and user-centric features create opportunities for revenue generation through premium user subscriptions, advertisements, or value-added services.

Increased Productivity and Collaboration: By enhancing file sharing efficiency and promoting seamless collaboration, the application can lead to increased productivity, saving valuable time and resources for users.

Competitive Advantage: The implementation of a robust, feature-rich, and secure file sharing platform provides a competitive advantage, attracting more users and potential partnerships.

In conclusion, the feasibility study confirms that the "File Sharing and Downloading Web Application" is both operationally, technically, and economically viable. The project aligns well with the organization's goals and requirements, utilizes appropriate and scalable technologies, and offers significant potential for return on investment. With strong feasibility across all aspects, the project holds promise to revolutionize the file sharing experience and become a valuable asset for users seeking a secure and efficient file sharing and downloading solution.

PROJECT DESCRIPTION

File Sharing and Downloading WebApplication" is a multivendor E-Commerce website that provides a fast and secure platform for users to share and download files. Users can register and upload their files, and other users can access, download, and comment on these files. The application is designed to benefit both small file owners and users, providing a seamless experience for all parties involved.

The website features a robust search system, allowing users to easily find specific files. Additionally, it provides file owners with the necessary tools to manage and track their orders. The system emphasizes security, with encrypted passwords for both customers and vendors, ensuring data protection. "File Sharing and Downloading WebApplication" aims to solve common file sharing challenges, providing a convenient and efficient service to its users.

5.2 Module Description :

5.2.1 Admin Module

The Admin Module is a powerful tool that grants administrators full control over the platform. Admins have the authority to manage all users and posts, enabling them to oversee and moderate activities on the website. From user account management to post moderation, admins play a central role in ensuring the platform's smooth operation.

5.2.2 User Module

The User Module empowers individual users to manage their own accounts and personal information. Users can perform actions like updating their profiles, changing passwords, and modifying account details according to their preferences. Additionally, users have access to a dedicated section where they can exclusively manage the files they have posted on the platform.

5.2.3 File Management Module

The File Management Module allows users to upload and manage files that they wish to share with others. Uploaded files are automatically converted into links, simplifying the process of privately sharing files with specific recipients. Users have full control over the files they upload and can easily manage and organize them.

5.2.4 Comment Section

The Comment Section enables users to interact and engage with posts shared by others. Users can leave comments, feedback, or reviews on posts, fostering a vibrant and interactive community within the platform.

5.2.5 Post Module

The Post Module is responsible for showcasing all posts on the platform. Users can view posts on the home page and access detailed information about each post through the post detail page. This module ensures that all users can discover and access a wide range of shared files and information.

SYSTEM DESIGN

6.1 Introduction

The system design for the "File Sharing and Downloading Web Application" is a comprehensive and detailed plan that outlines the architecture, components, and functionality of the application. This design phase plays a pivotal role in transforming the project's requirements and objectives into a tangible and scalable system. The primary goal of the system design is to create a robust, efficient, and user-friendly platform that allows users to securely share, upload, and download files seamlessly.

The system design focuses on various aspects, including database structure, user interface, application flow, and integration of essential modules. By considering these crucial elements, the design aims to optimize performance, enhance user experience, and ensure smooth interactions throughout the application.

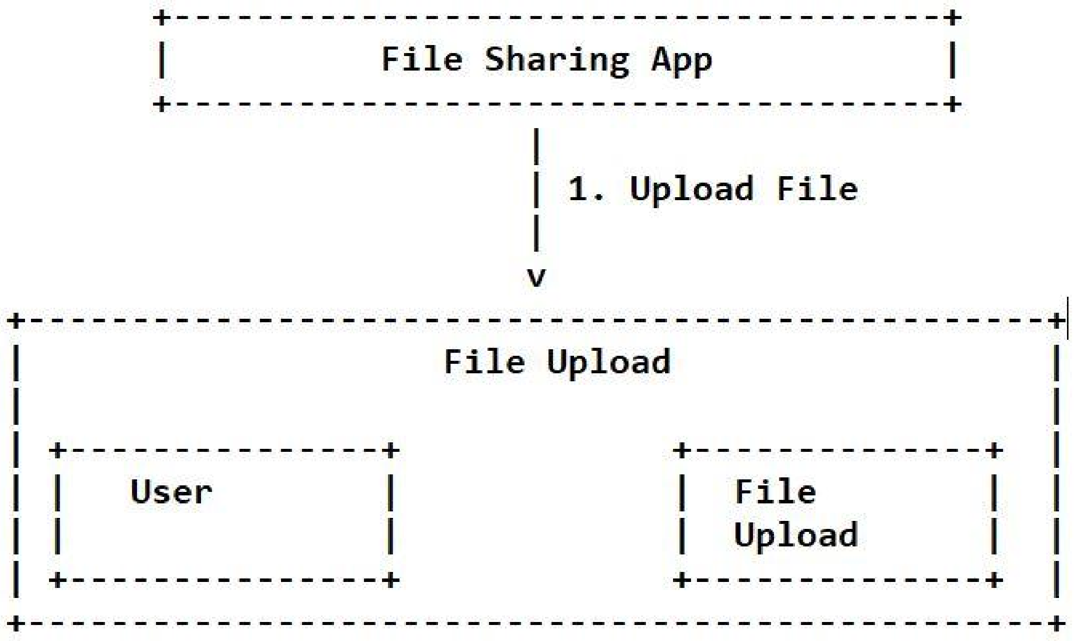
This document provides a comprehensive overview of the system's architecture, detailing the database schema, entity relationships, and data flow. Additionally, it outlines the user interface design, illustrating how users will navigate through the application and access its features. The system design also incorporates various security measures to safeguard user data and ensure privacy.

Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a graphical representation of the flow of the data through an information system. DFDs can be used for the visualization of data processing. A Data flow Diagram is a significant modeling technique for analyzing and construction information processes. DFD literally means and illustration that explains the course or movement of information in process. DFD illustrates this flow of information in process based on the inputs and outputs. A DFD can be referred to as process Model.

Unlike details flowchart, DFD’s do not supply details description of modules that graphically describes a system’s data and how the data interreact with the system.

DFD Level 0 –

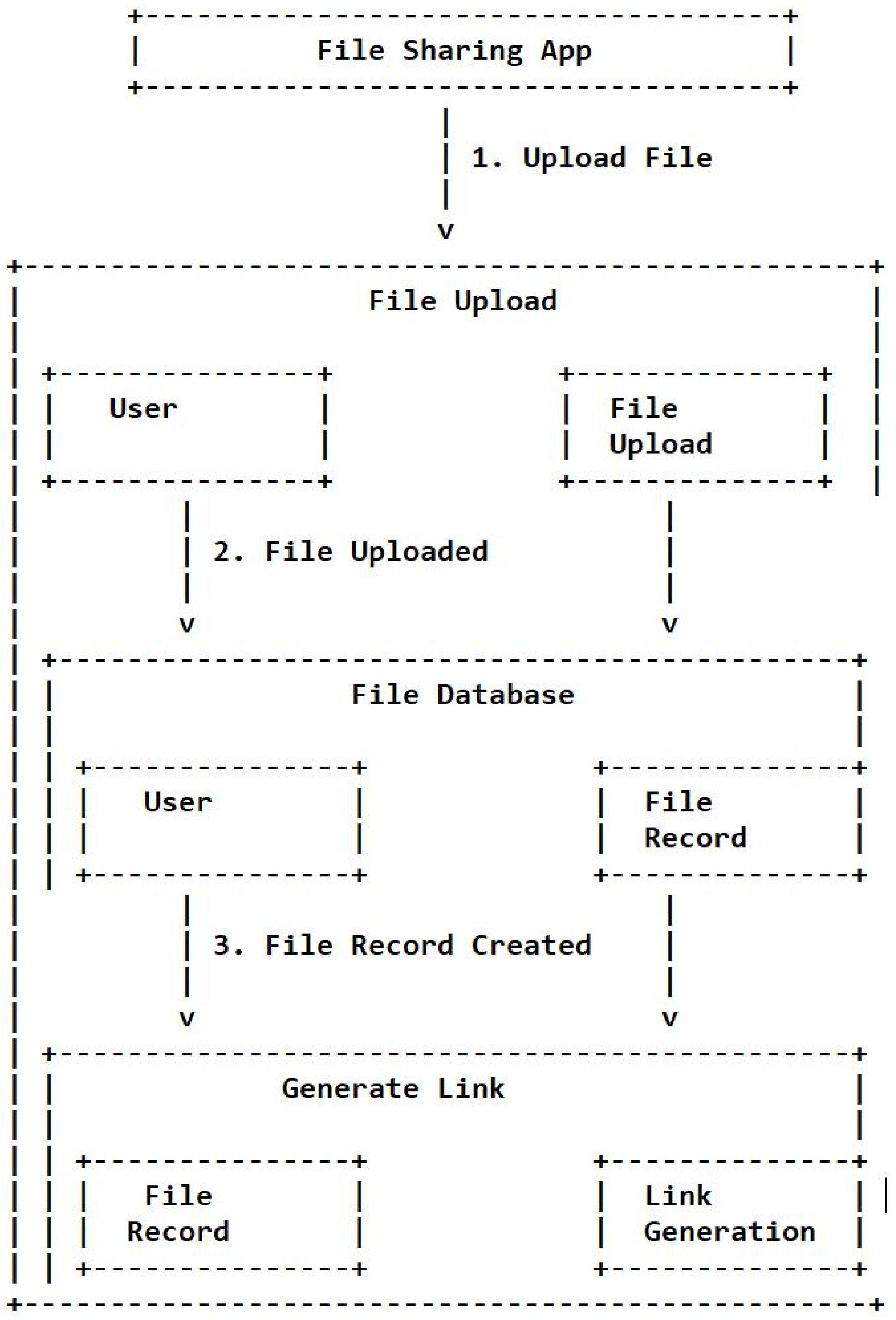


```
          +---------------------------------+
          |        File Sharing App         |
          +---------------------------------+
                         |
                         | 1. Upload File
                         |
                         v
+--------------------------------------------+
|                  File Upload               |
|                                            |
| +-------------+          +------------+    |
| |   User      |          |  File      |    |
| |             |          |  Upload    |    |
| +-------------+          +------------+    |
+--------------------------------------------+
```

DFD Level 1-

```
          +--------------------------------------+
          |           File Sharing App           |
          +--------------------------------------+
                            |
                            | 1. Upload File
                            |
                            v
   +-------------------------------------------------+
   |                    File Upload                  |
   |                                                 |
   | +---------------+              +---------------+  |
   | |   User        |              |  File         |  |
   | |               |              |  Upload       |  |
   | +---------------+              +---------------+  |
   |          |                            |
   |          | 2. File Uploaded           ||
   |          |                            |
   |          v                            v
   | +----------------------------------------------+
   | |                 File Database                |
   | |                                              |
   | | +---------------+          +---------------+
   | | |   User        |          |  File         |
   | | |               |          |  Record       |
   | | +---------------+          +---------------+
   | +----------------------------------------------+
```

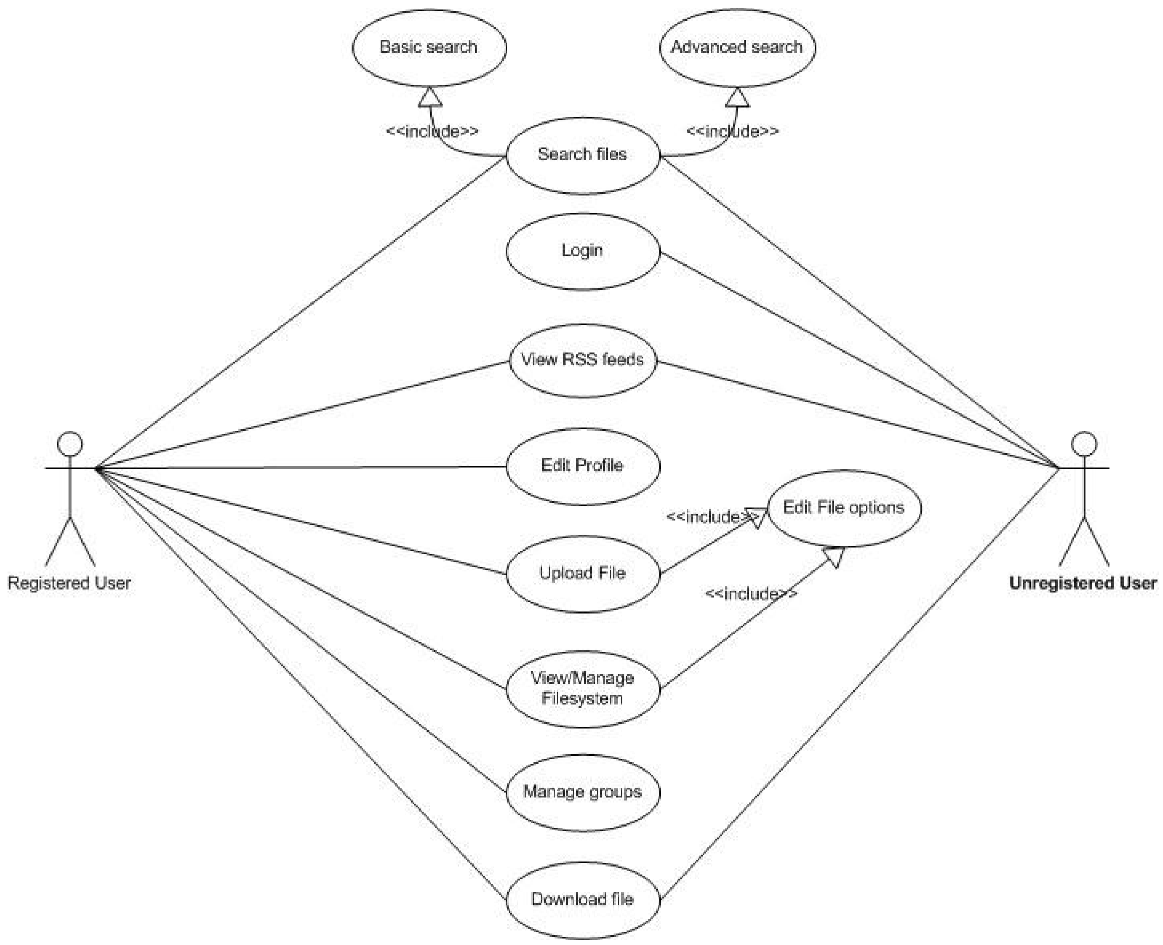
DFD Level 2 -



```
          +----------------------------------+
          |         File Sharing App         |
          +----------------------------------+
                           |
                           | 1. Upload File
                           |
                           v
   +---------------------------------------------+
   |                 File Upload                 |
   |                                             |
   | +---------------+           +-------------+ |
   | |   User        |           | File        | |
   | |               |           | Upload      | |
   | +---------------+           +-------------+ |
   |        |                          |
   |        | 2. File Uploaded         |
   |        |                          |
   |        v                          v
   | +-----------------------------------------+
   | |               File Database             |
   | |                                         |
   | | +---------------+       +-------------+ |
   | | |   User        |       | File        | |
   | | |               |       | Record      | |
   | | +---------------+       +-------------+ |
   |        |                          |
   |        | 3. File Record Created   |
   |        |                          |
   |        v                          v
   | +-----------------------------------------+
   | |           Generate Link                 |
   | |                                         |
   | | +---------------+       +-------------+ |
   | | |   File        |       | Link        | |
   | | |  Record       |       | Generation  | |
   | | +---------------+       +-------------+ |
   +---------------------------------------------+
```

USE-CASE DIAGRAM :

The Use Case Diagram for the **"File Sharing and Downloading Web Application"** provides a visual representation of the various interactions between users and the system. It illustrates the different use cases or functionalities that users can perform within the application. The main actors in the diagram are the "User" and the "System." The "User" can perform actions such as "Registering an Account," "Logging In," "Uploading Files," "Creating Posts," "Viewing Posts," and "Downloading Files." The "System" handles these actions and responds accordingly, ensuring a seamless and user-friendly experience for the users. The Use Case Diagram serves as a roadmap for understanding the application's functionalities and helps in designing and developing an efficient and effective file sharing and downloading platform.



NAVIGATION FLOW DIAGRAM:

The Navigation Flow Diagram for the "File Sharing and Downloading Web Application" outlines the user's journey and interactions within the system. It depicts the step-by-step flow of navigation and the various functionalities accessible to users. Here's the description of the Navigation Flow Diagram for your application:

1. Home Page:

- The user starts the navigation from the Home Page, where they can view recent posts and files shared by other users.

2. Register:

- If the user is not registered, they can click on the "Register" button to create a new account.

- After providing the necessary details like name, email, and password, the user can register successfully.

3. Login:

- For registered users, the "Login" option allows them to access their accounts by entering their credentials.

4. Profile Page:

- The user's profile page displays their profile photo, username, and a list of posts they have created.

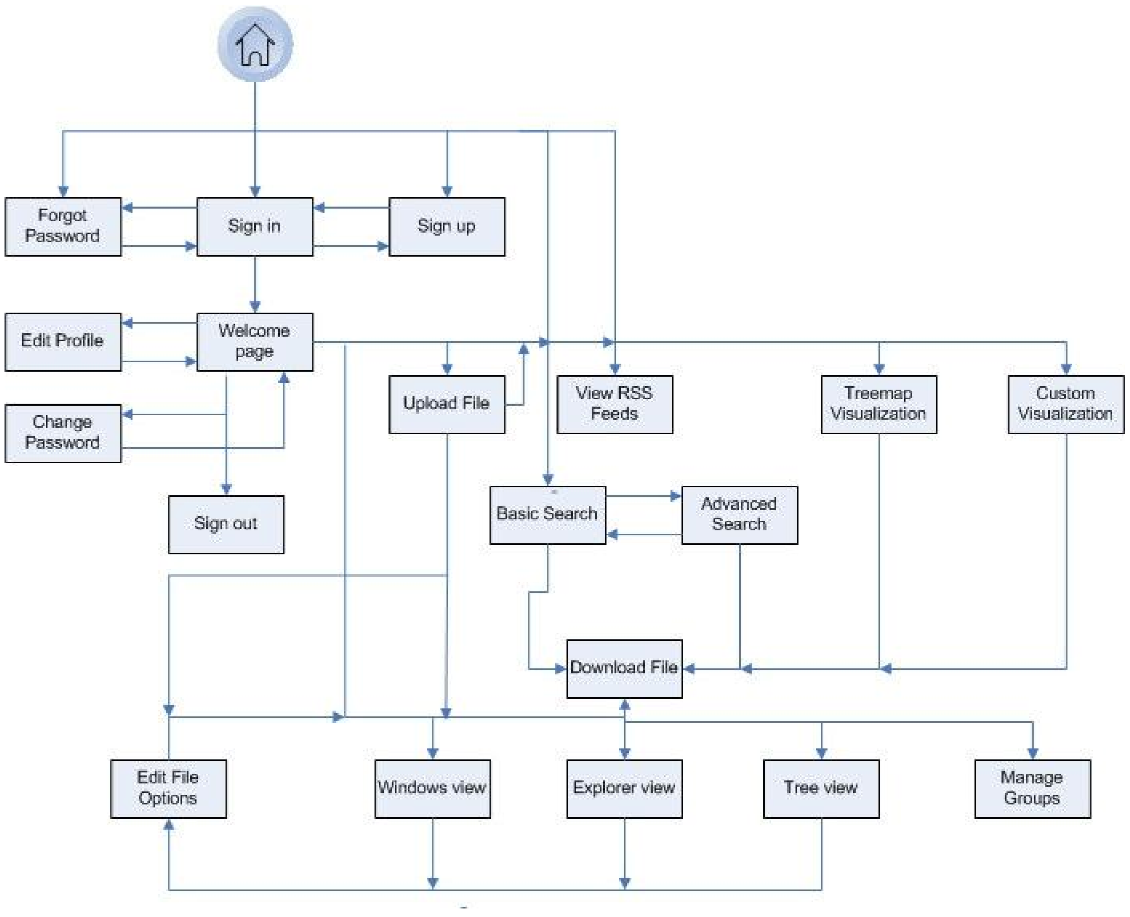
5.Create Post:

- By selecting the "Create Post" option, the user can upload a file and provide a file name and description.

6. View Post Details: - Clicking on a post in the profile page leads to the "Post Details" page, showing the file name, description, and a download button.

7. Logout:

- The "Logout" option allows the user to securely log out of their account and return to the Home Page.



The Navigation Flow Diagram illustrates the user's seamless journey through the application, facilitating easy access to file sharing, downloading, and post management functionalities. This user-friendly navigation ensures a positive user experience and enhances the efficiency of the file sharing and downloading web application.

MODEL STRUCTURE:

The model structure for the **"File Sharing and Downloading Web Application"** is designed to efficiently store and manage the relevant information required to support the application's functionalities. It consists of several fields, each representing a specific entity or model in the application. The main entities include User, Profile, Post, File, and Folder, along with their associated attributes. Here's an overview of the database structure:

1. User:

- id: Unique identifier for each user.
- name: Name of the user.
- email: Email address of the user.
- password: Encrypted password for user authentication.

2. Profile:

- id: Unique identifier for each profile.
- user_id: Foreign key linking to the User table to establish a one-to-one relationship.
- profile_photo: Stores the profile photo of the user.
- username: Username chosen by the user for their profile.

3. Post:

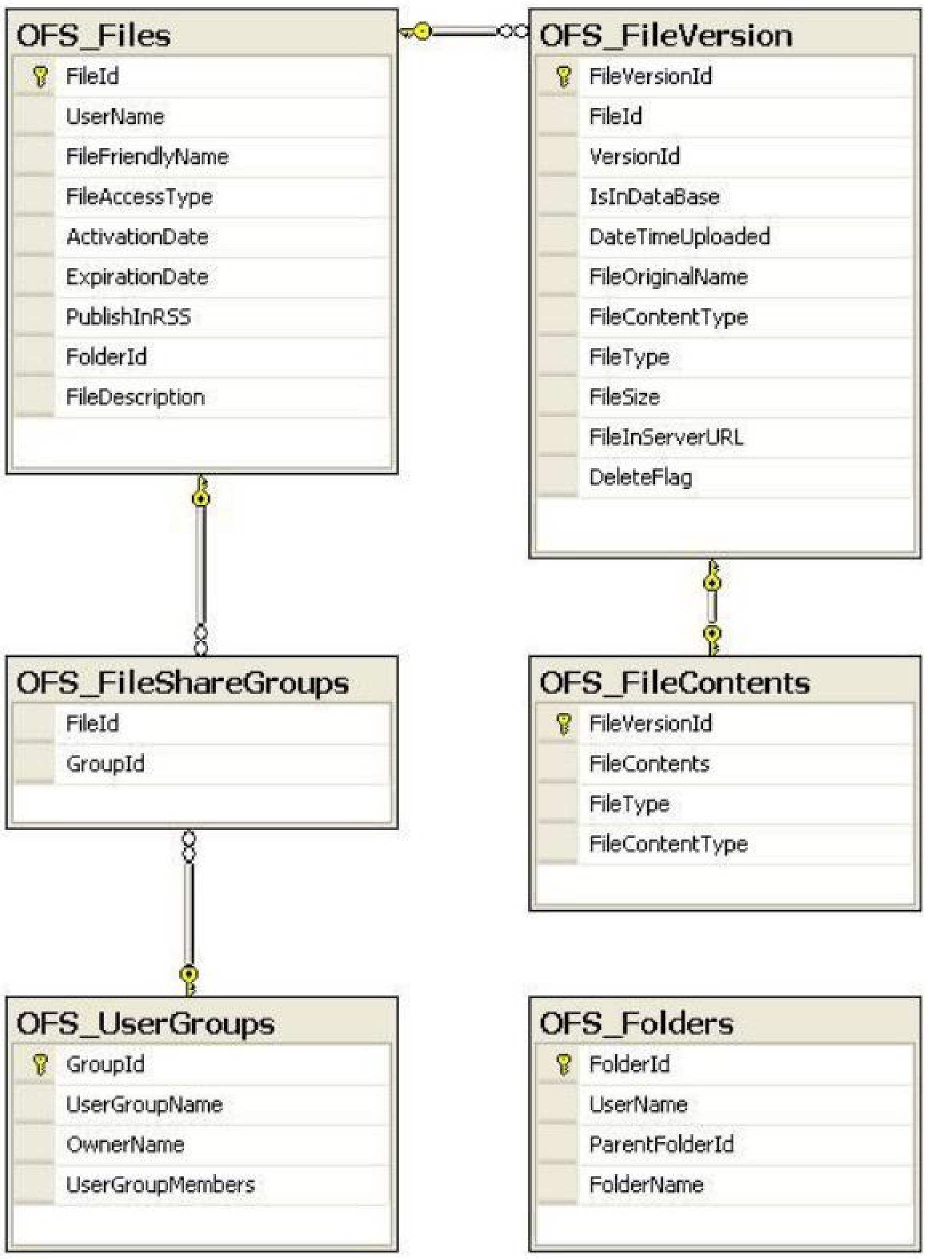
- id: Unique identifier for each post.
- user_id: Foreign key linking to the User table to establish a many-to-one relationship.
- created_at: Timestamp indicating when the post was created.
- file_name: Name of the uploaded file.
- file_description: Description of the uploaded file.

4. File:

- id: Unique identifier for each file.
- post_id: Foreign key linking to the Post table to establish a one-to-one relationship.

5. Folder:

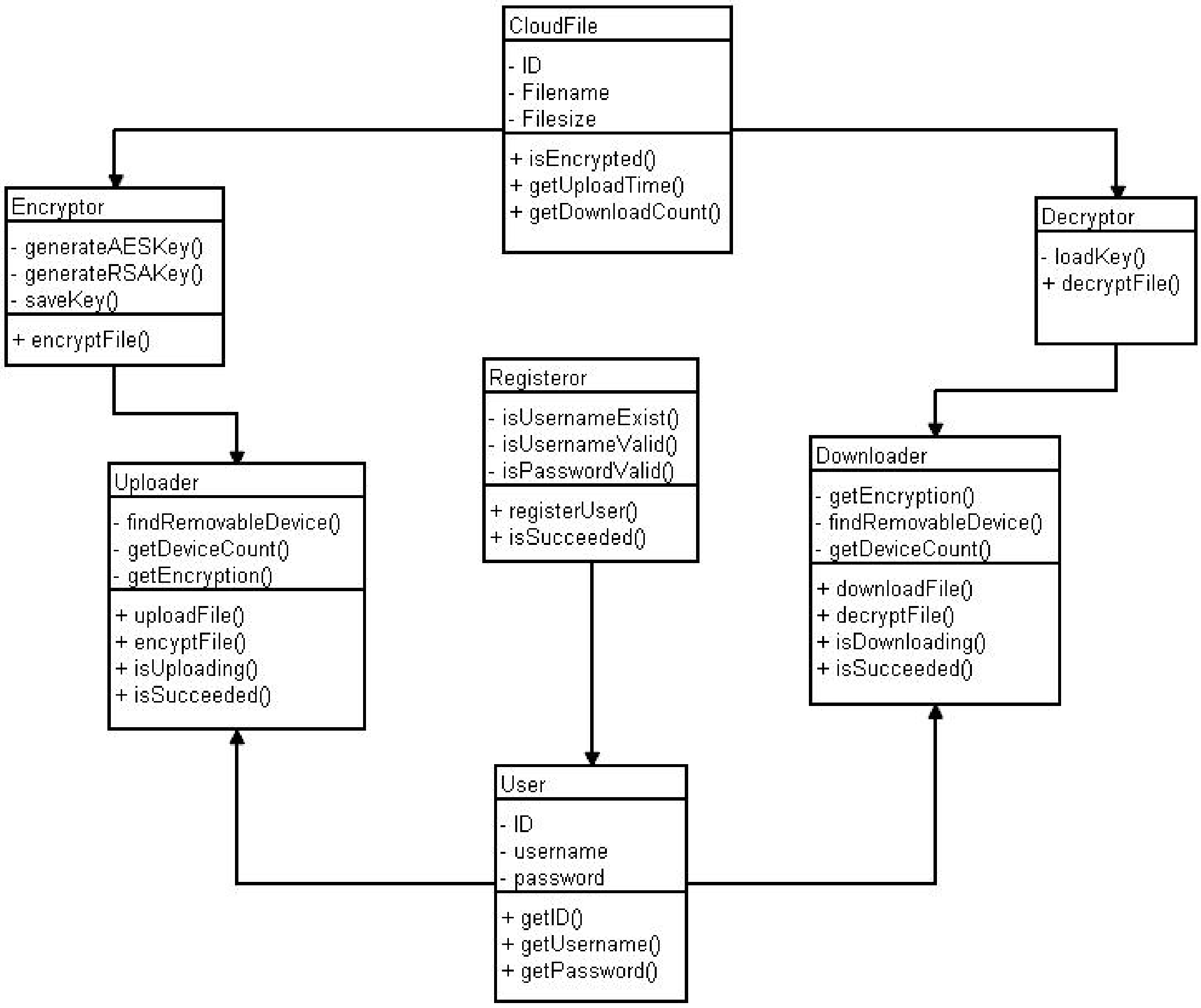
- id: Unique identifier for each folder.
- post_id: Foreign key linking to the Post table to establish a one-to-many relationship.
- folder_name: Name of the folder.



The database structure is designed to ensure data integrity, minimize redundancy, and support efficient data retrieval and manipulation.

CLASS DIAGRAM:

The class diagram for the **"File Sharing and Downloading Web Application"** represents a static view of the application's structure, illustrating the various classes, their attributes, and the relationships between them. The class diagram encompasses the core entities of the application, such as User, Profile, Post, File, and Folder, along with their associated attributes and methods. It also illustrates the relationships, such as associations and inheritance, defining the interactions and dependencies among the classes. This comprehensive representation aids developers and stakeholders in understanding the application's architecture and lays the foundation for smooth and efficient development, ensuring a robust, maintainable, and scalable file sharing and downloading solution.



SYSTEM TESTING

7.1 Introduction

Software Testing is an essential process in the "FILE SHARING AND DOWNLOADING WEBAPPLICATION," aiming to evaluate its quality and provide valuable insights to stakeholders. It ensures that the application meets its requirements and functions as expected. Various types of testing include:

Integration Testing: Verifying interactions between different modules like file posting

7.2 Test Cases

Component Testing:

Component Testing (Unit Testing) focuses on testing individual features of the "FILE SHARING AND DOWNLOADING WEBAPPLICATION" independently.

Test Criteria	Test Case	Expected Output	Pass/Fail
User Registration	Registering with existing UserName	Error Message	Fail
File Upload	Upload a file or genarating link of uploaded files	File displayed in users account or Link generated	Pass

Unit Testing:

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

Test Criteria	Test Case	Expected Output	Actual Output	Pass/Fail
Register with wrong password	Wrong password	Error Message	Error Message	Fail
New Registration	Check the Database	Registration done and viewed intable	Registration details added and viewed.	Pass
Update	Check the Status	Status Updated	Status Updated	Pass

System Testing:

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input.

System Testing evaluates the overall performance and functionality of the entire **"FILE SHARING AND DOWNLOADING WEBAPPLICATION."**

Test Criteria	Test Case	Expected Output	Actual Output	Pass/Fail
Click on Generate	Genarating zip file link	Link of Zipped File	Files inside a zip file	Pass

CODE

VIEWS

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm
def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Your account has been created! You are now
able to log in')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})
@login_required
def profile(request):
    if request.method == 'POST':
        u_form = UserUpdateForm(request.POST, instance=request.user)
        p_form = ProfileUpdateForm(request.POST,
                                   request.FILES,
                                   instance=request.user.profile)
        if u_form.is_valid() and p_form.is_valid():
            u_form.save()
            p_form.save()
            messages.success(request, f'Your account has been updated!')
            return redirect('profile')

    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

    context = {
        'u_form': u_form,
        'p_form': p_form
    }
    return render(request, 'users/profile.html', context)

from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.mixins import LoginRequiredMixin, UserPassesTestMixin
from django.contrib.auth.models import User
from django.views.generic import (
    ListView,
    DetailView,
    CreateView,
    UpdateView,
    DeleteView
)
```

```
from .models import Post

from .forms import CommentForm

from django.db.models import Q



def home(request):

    context = {

        'posts': Post.objects.all()

    }

    return render(request, 'blog/home.html', context)

def search(request):

    template = 'blog/home.html'

    query = request.GET.get('q')

    result          =          Post.objects.filter(Q(title_icontains=query)          |
Q(authorusernameicontains=query) | Q(content_icontains=query))

    paginate_by = 2

    context = {'posts': result}

    return render(request, template, context)


class PostListView(ListView):

    model = Post

    template_name = 'blog/home.html'

    context_object_name = 'posts'

    ordering = ['-date_posted']

    paginate_by = 6
```

```
class UserPostListView(ListView):
    model = Post
    template_name = 'blog/user_posts.html'
    context_object_name = 'posts'
    paginate_by = 6

    def get_queryset(self):
        user = get_object_or_404(User, username=self.kwargs.get('username'))
        return Post.objects.filter(author=user).order_by('-date_posted')

class PostCreateView(LoginRequiredMixin, CreateView):
    model = Post
    template_name = 'blog/post_form.html'
    fields = ['title', 'content', 'file']

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)
class PostUpdateView(LoginRequiredMixin, UserPassesTestMixin,
UpdateView):
    model = Post
    template_name = 'blog/post_form.html'
    fields = ['title', 'content', 'file']

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
    model = Post
    success_url = '/'
    template_name = 'blog/post_confirm_delete.html'

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False

def about(request):
    return render(request, 'blog/about.html', {'title': 'About'})

class PostDetailView(DetailView):
    model = Post
    template_name = 'blog/post_detail.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['form'] = CommentForm()
        return context

    def post(self, request, *args, **kwargs):
        post = self.get_object()
        if request.user.is_authenticated:
            form = CommentForm(request.POST)
```

RESTAPI VIEWS

```
from django.shortcuts import render
from rest_framework.views import APIView
from rest_framework.response import Response

from .serializers import *
from rest_framework.parsers import MultiPartParser

def home(request):
    return render(request ,'home.html')

def download(request , uid):
    return render(request , 'download.html' , context = {'uid' : uid})

class HandleFileUpload(APIView):
    parser_classes = [MultiPartParser]
```

```
def post(self , request):
    try:
        data = request.data

        serializer = FileListSerializer(data = data)

        if serializer.is_valid():
            serializer.save()
            return Response({
                'status' : 200,
                'message' : 'files uploaded successfully',
                'data' : serializer.data
            })

        return Response({
            'status' : 400,
            'message' : 'something went wrong',
            'data'  : serializer.errors
        })
    except Exception as e:
        print(e)
```

MODELS

```
from django.db import models

from django.utils import timezone

from django.contrib.auth.models import User

from django.urls import reverse

import os


class Post(models.Model):

        title = models.CharField(max_length=100)

        file = models.FileField(null=True,blank=True,upload_to='Files')

        content = models.TextField()

        date_posted = models.DateTimeField(default=timezone.now)

        author = models.ForeignKey(User, on_delete=models.CASCADE)


        def _str_(self):

                return self.title


        def extension(self):

                name, extension = os.path.splitext(self.file.name)

                return extension


        def get_absolute_url(self):

                return reverse('post-detail', kwargs={'pk': self.pk})
```

```
class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE,
related_name='comments')
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    content = models.TextField()
    date_posted = models.DateTimeField(default=timezone.now)

    def _str_(self):
        return f"Comment by {self.author.username} on {self.post.title}
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    image = models.ImageField(default='default.jpg', upload_to='profile_pics')

    def _str_(self):
        return f'{self.user.username} Profile'

    def save(self, *args, **kwargs):
        super(Profile, self).save(*args, **kwargs)

        img = Image.open(self.image.path)

        if img.height > 300 or img.width > 300:
            output_size = (300, 300)
            img.thumbnail(output_size)
            img.save(self.image.path)
```

TEMPLATES

Home.html

```
{% extends "blog/base.html" %}
{% block content %}
{% load static %}

<!-- Header -->
<div class="header">
  <img src="{% static 'images/fi.jpg' %}" alt="">
</div>
<br>
<!-- Rest of your content -->
<div class="container">
  <div class="row">
    {% for post in posts %}
      <div class="col-md-4">
        <article class="media content-section">
          <a href="{{ post.author.profile.image.url }} " target="_blank">
            <img src="{{ post.author.profile.image.url }}" alt="" class="article-img
rounded-circle img-thumbnail shadow">
          </a>
          <div class="media-body">
            <div class="article-metadata d-flex justify-content-between">
              <a class="mr-2" href="{% url 'user-posts' post.author.username %}">
                <h3>{{ post.author }}</h3>
              </a>
              <small class="text-muted">{{ post.date_upload|date:"F d,
Y" }}</small>
            </div>
          {% if post.file %}
           <a href="{{ post.file.url }}" download class="text-dark text-justify">
            <h5>{{ post.blog }}</h5>
           </a>
          {% endif %}
          <h2>
           <a class="article-title text-justify" href="{% url 'post-detail'
post.id %}">{{ post.title }}</a>
          </h2>
          <div class="d-flex justify-content-between">
           <p class="article-content text-justify text-truncate overflow-
hidden">{{ post.content }}</p>
            {% if post.file %}
             <div class="form-group mt-0 pt-0 m-2">
              <a class="btn btn-outline-success btn-d" href="{{ post.file.url }}"
download type="submit">
               <i style="color: chartreuse;" class="fas fa-download"></i>
              </a>
             </div>
            {% endif %}
          </div>
         </div>
        </article>
       </div>
     {% endfor %}
   </div>

  {% if is_paginated %}
    <div class="row justify-content-center mt-4">
     <ul class="pagination">
       {% if page_obj.has_previous %}
        <li class="page-item">
          <a class="page-link " style=" border: 1px solid #FFD700; background-
color: black; color: #FFD700;" href="?page=1">First</a>
```

```
        </li>
        <li class="page-item">
          <a class="page-link" style=" border: 1px solid #FFD700; background-
color: black; color: #FFD700;"
href="?page={{ page_obj.previous_page_number }}">Previous</a>
        </li>
       {% endif %}

       {% for num in page_obj.paginator.page_range %}
        {% if page_obj.number == num %}
         <li class="page-item active">
          <a class="page-link" style=" border: 1px solid #FFD700; background-
color: black; color: #FFD700;" href="?page={{ num }}">{{ num }}</a>
         </li>
        {% else %}
         <li class="page-item">
          <a class="page-link" style=" border: 1px solid #FFD700; background-
color: black; color: #FFD700;" href="?page={{ num }}">{{ num }}</a>
         </li>
        {% endif %}
       {% endfor %}
</ul>
    </div>
  {% endif %}
</div>
{% endblock content %}
{% extends "blog/base.html" %}
{% block content %}
  <div class="content-section">
     <form method="POST" style="padding: 40px;">
        {% csrf_token %}
        <fieldset class="form-group">
          <legend class="border-bottom mb-4"> <p style="margin-left:
30px;">Log In</p></legend>
```

```
            {{ form.as_p }}
          </fieldset>
          <div class="form-group">
            <button class="btn btn-outline-info" type="submit">Login</button>
          </div>
        </form>
        <div class="border-top pt-3">
          <small class="text-muted">
            <b style="color: beige; margin-left: 40px;">Need An Account?</b> <a
class="ml-2" href="{% url 'register' %}">Sign Up Now</a>
          </small>
        </div>
    </div>
{% endblock content %}
```

LINK GENARATOR HOME

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>File sharing | Home</title>
   <link     href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
       integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
   <!--                                                                    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"                                              integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous"> -->

   <style>
       /**
 * FilePond Custom Styles
 */
       .filepond--drop-label {
           color: #4c4e53;
       }

       .filepond--label-action {
           -webkit-text-decoration-color: #babdc0;
           text-decoration-color: #babdc0;
       }
```

```
    .filepond--panel-root {
      border-radius: 2em;
      background-color: #edf0f4;
      height: 1em;
    }

    .filepond--item-panel {
      background-color: #595e68;
    }

    .filepond--drip-blob {
      background-color: #7f8a9a;
    }

    /**
* Page Styles
*/
    html {
      padding: 3vh 0 0;
    }

    body {
      max-width: 28em;
      margin: 0 auto;
      background-color: #880E4F;
    }

    .background {


      border-radius: 5px;
      display: flex;
      align-items: center;
      justify-content: center;
```

```
        }

        .clipboard {
            border: 0;
            margin: 5px;
            padding: 15px;
            border-radius: 3px;
            background-image: linear-gradient(135deg, #536DFE 10%, #304FFE 100%);
            cursor: pointer;
            color: #fff;
            font-family: "Karla", sans-serif;
            font-size: 16px;
            position: relative;
            top: 0;
            transition: all 0.2s ease;
        }

        body .clipboard:hover {
            top: 2px;
        }

        body p {
            font-weight: 700;
        }
    </style>
</head>

<body>
    <link href="https://unpkg.com/filepond/dist/filepond.css" rel="stylesheet">
    <link     href="https://unpkg.com/filepond-plugin-image-preview/dist/filepond-plugin-
image-preview.css"
        rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/filepond/4.30.3/filepond.min.js"
```

```
        integrity="sha512-
AuMJiyTn/5k+gog21BWPrcoAC+CgOoobePSRqwsOyCSPo3Zj64eHyOsqDev8Yn9H4
5WUJmzbe9RaLTdFKkO0KQ=="
        crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.js"></script>
    <script src="https://unpkg.com/jquery-filepond/filepond.jquery.js"></script>
    <script    src="https://unpkg.com/filepond-plugin-image-preview/dist/filepond-plugin-
image-preview.js"></script>


    <div class="text-center">

        <img    src="/media/image/undraw.svg"    class="img-fluid    img-responsive"
style="height: 300px;">
    </div>

    <input type="file" class="my-pond mt-4" multiple name="filepond" />

    <div class="text-center">
        <button class="btn btn-success" onclick="upload_file()">Upload File</button>
    </div>

    <div id="panel" class="carad mt-3" id="btn" style="display: none;">
        <div class="card-bodqy">


            <div class="background">
                <center>
                    <button onclick="copyClip()" class="clipboard">Click me to copy current
Url</button>
                </center>
            </div>

        </div>
```

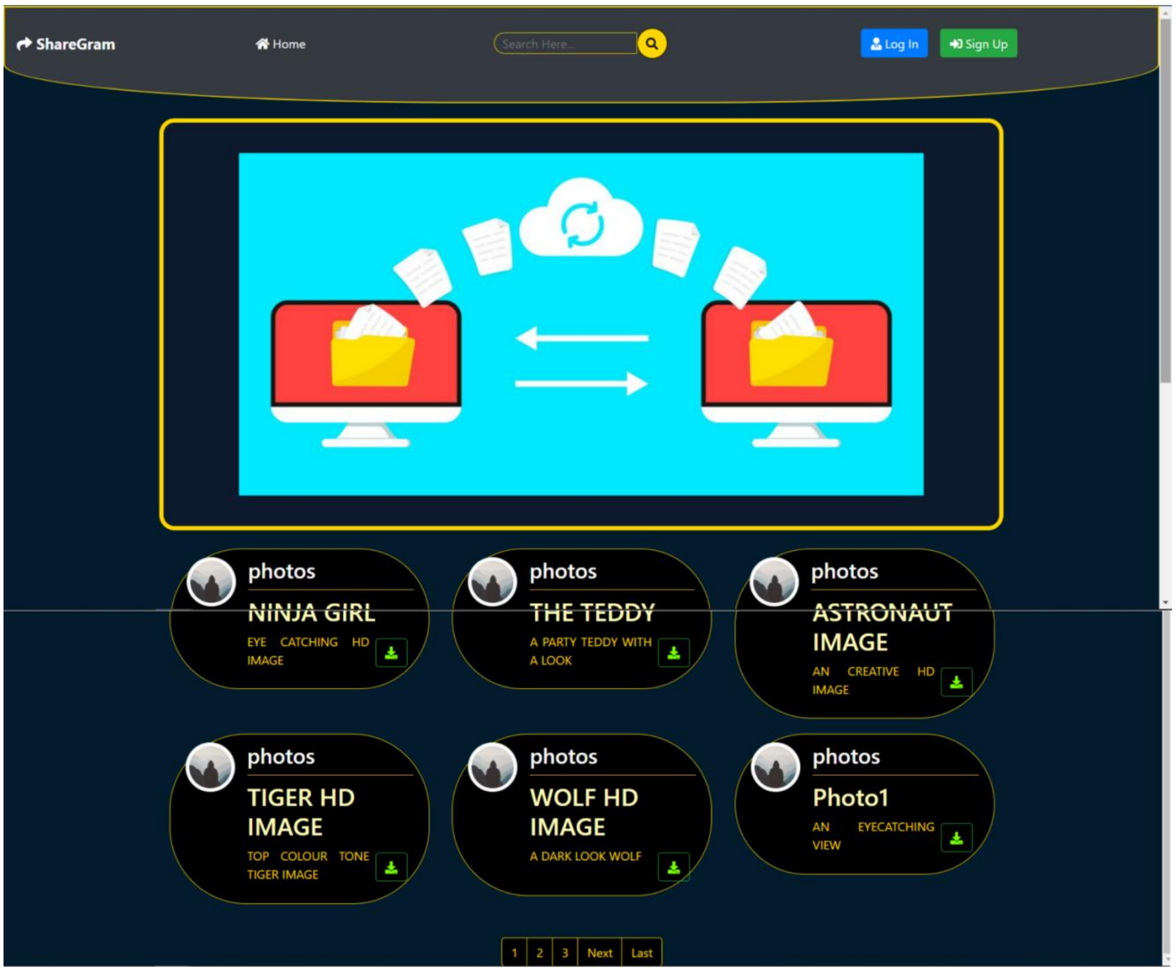
```
</div>
</div>



<script src="https://unpkg.com/filepond/dist/filepond.min.js"></script>

<script>
  FilePond.registerPlugin(
    FilePondPluginImagePreview,

  );

  const pond = FilePond.create(
    document.querySelector('.my-pond')
  );

  var url = null

  function upload_file() {
    document.getElementById("panel").style.display = "block";

    var files = pond.getFiles()

    var formdata = new FormData()

    for (var i = 0; i < files.length; i++) {
      formdata.append('files', files[i].file)
    }

    fetch('/handle/', {
        method: 'POST',
        headers: {
          'X-CSRFToken': "{{csrf_token}}"
        },
                body: formdata
            }).then(res => res.json())
            .then(result => {
                console.log(result)

                url = `http://127.0.0.1:5000/download/${result.data.folder}`

                document.getElementById('btn').style.display = 'df -h'



            })

        }

        function copyClip() {
            console.log('dd')



            /* Copy the text inside the text field */
            navigator.clipboard.writeText(url);

            /* Alert the copied text */
            alert("Copied the text: " + url);
        }
    </script>

</body>

</html>
```

SERIALIZERS

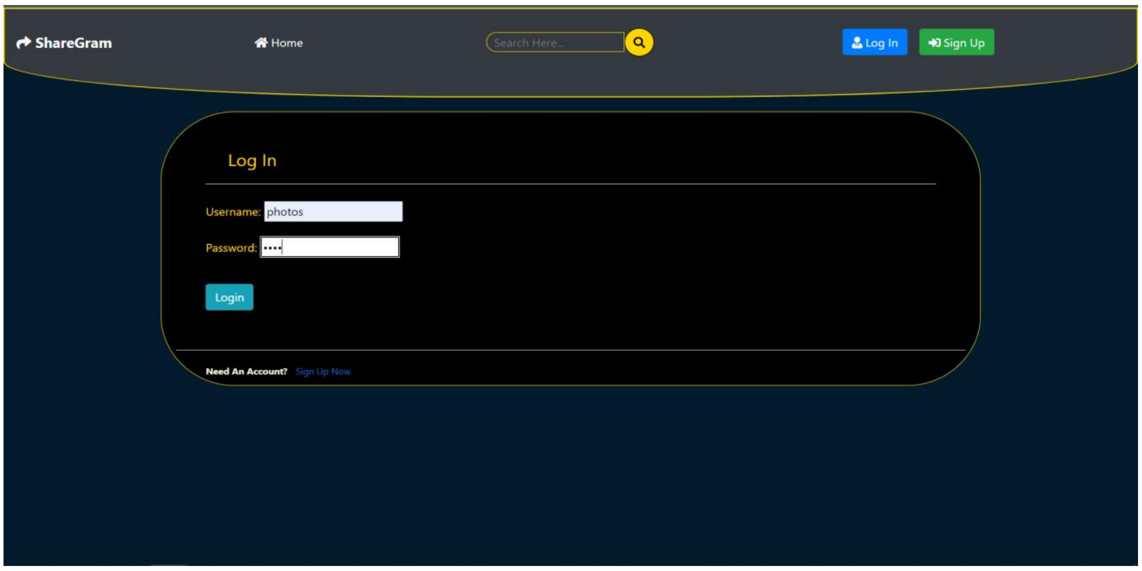
```
class FileSerializer(serializers.ModelSerializer):
    class Meta:
        model = Files
        fields = '__all__'

class FileListSerializer(serializers.Serializer):
    files = serializers.ListField(
            child = serializers.FileField(max_length = 100000 , allow_empty_file =
False , use_url = False)
    )
    folder = serializers.CharField(required = False)

    def zip_files(self,folder):
                                        shutil.make_archive(f'public/static/zip/{folder}'    ,
'zip' ,f'public/static/{folder}' )
    def create(self , validated_data):
        folder = Folder.objects.create()
        files = validated_data.pop('files')
        files_objs = []
        for file in files:
            files_obj = Files.objects.create(folder = folder , file = file)
            files_objs.append(files_obj)
            self.zip_files(folder.uid)

        return {'files' : {} , 'folder' : str(folder.uid)}
```

SCREENSHOTS

HOME PAGE:



LOGIN PAGE:



Log In

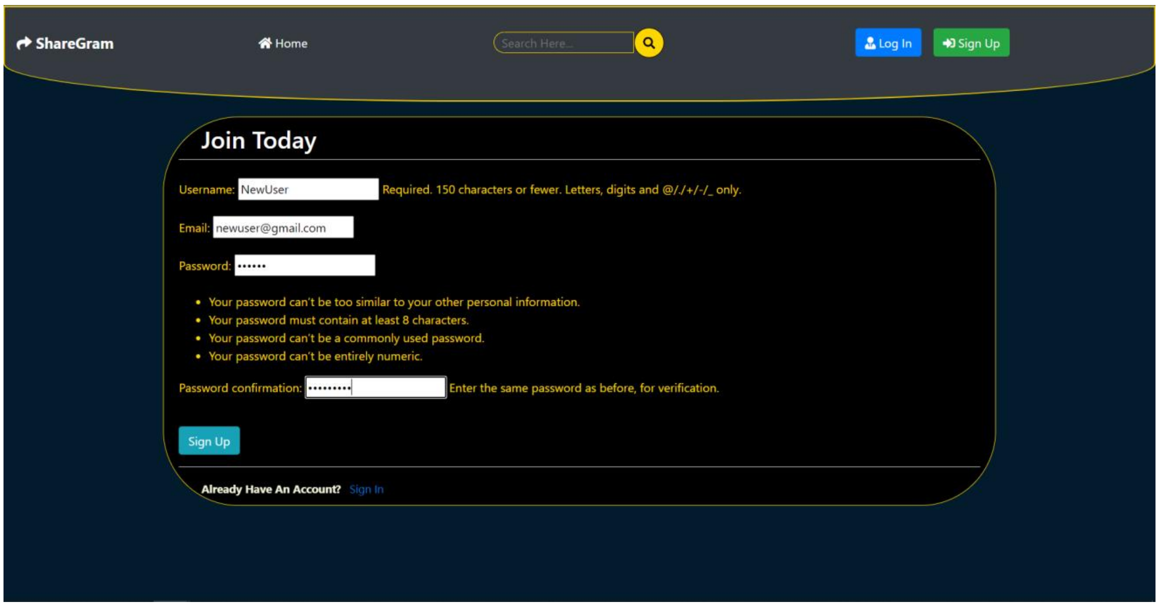
Username: photos

Password: ••••

Login

Need An Account? Sign Up Now

REGISTER PAGE:



Join Today

Username: NewUser Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email: newuser@gmail.com

Password: ••••••

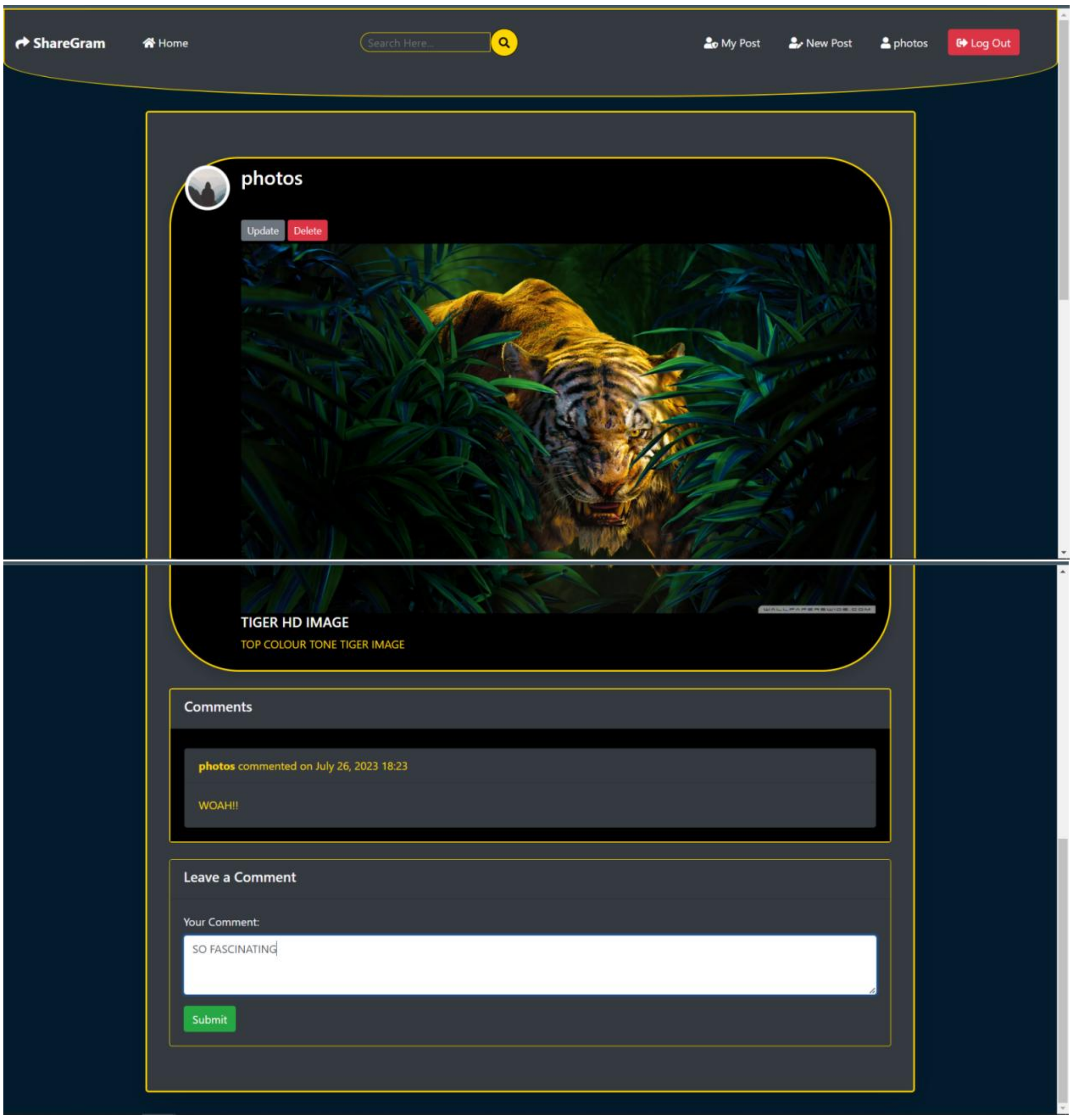
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: ••••••••• Enter the same password as before, for verification.

Sign Up

Already Have An Account? Sign In

POST DETAIL PAGE:



TIGER HD IMAGE

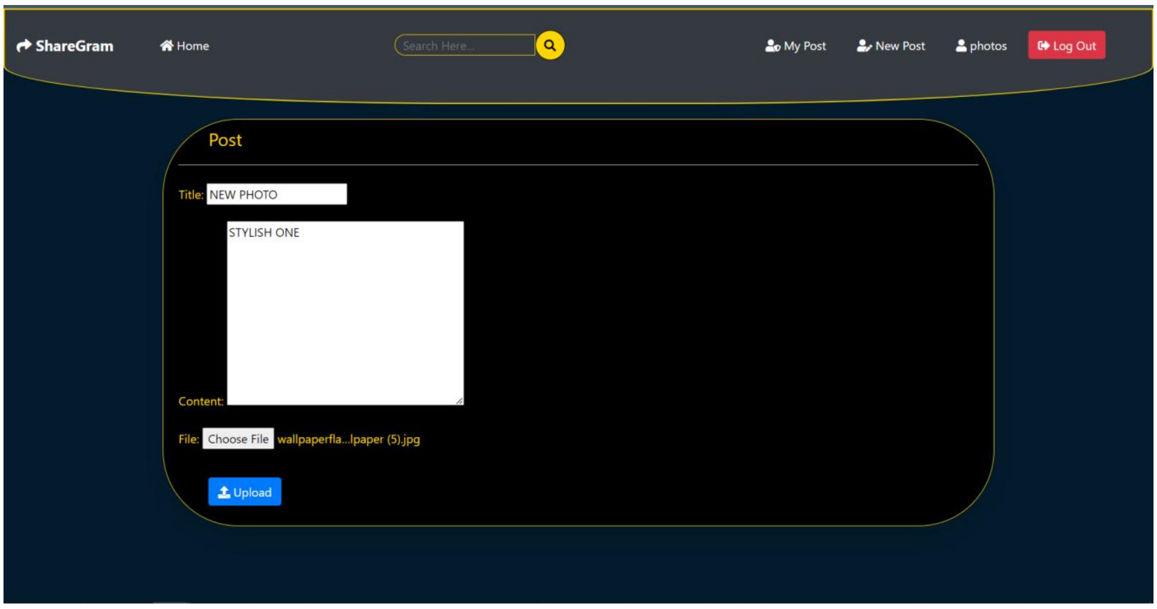
TOP COLOUR TONE TIGER IMAGE

photos commented on July 26, 2023 18:23

WOAH!!

SO FASCINATING

POST FORM PAGE:

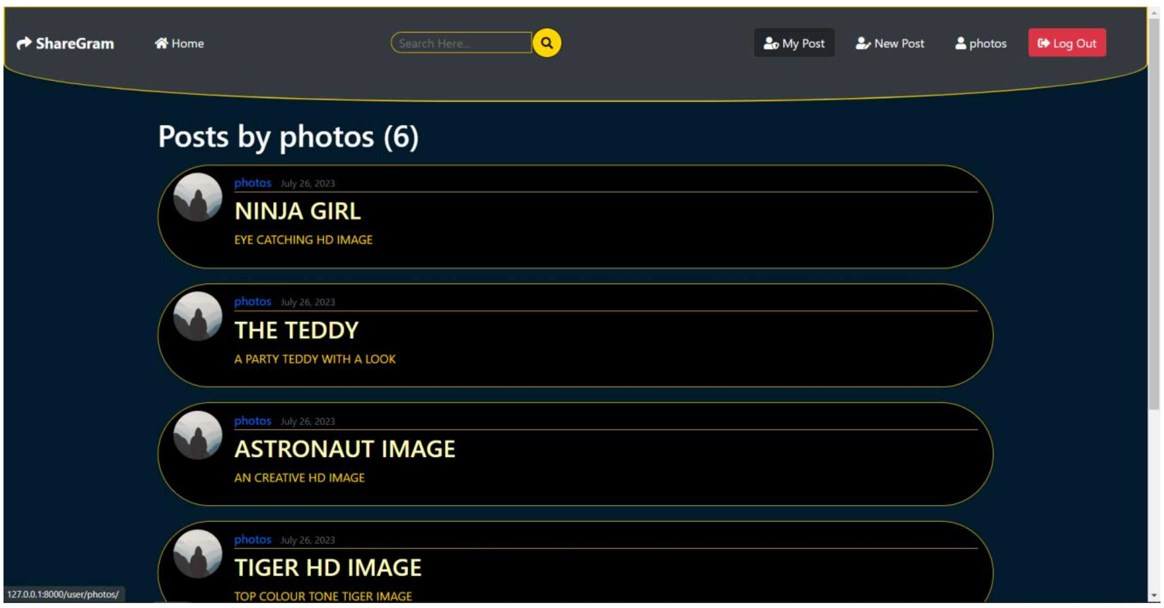


Title: NEW PHOTO

Content: STYLISH ONE

File: wallpaperfla...lpaper (5).jpg

MYPOSTS PAGE:



Posts by photos (6)

photos July 26, 2023
NINJA GIRL
EYE CATCHING HD IMAGE

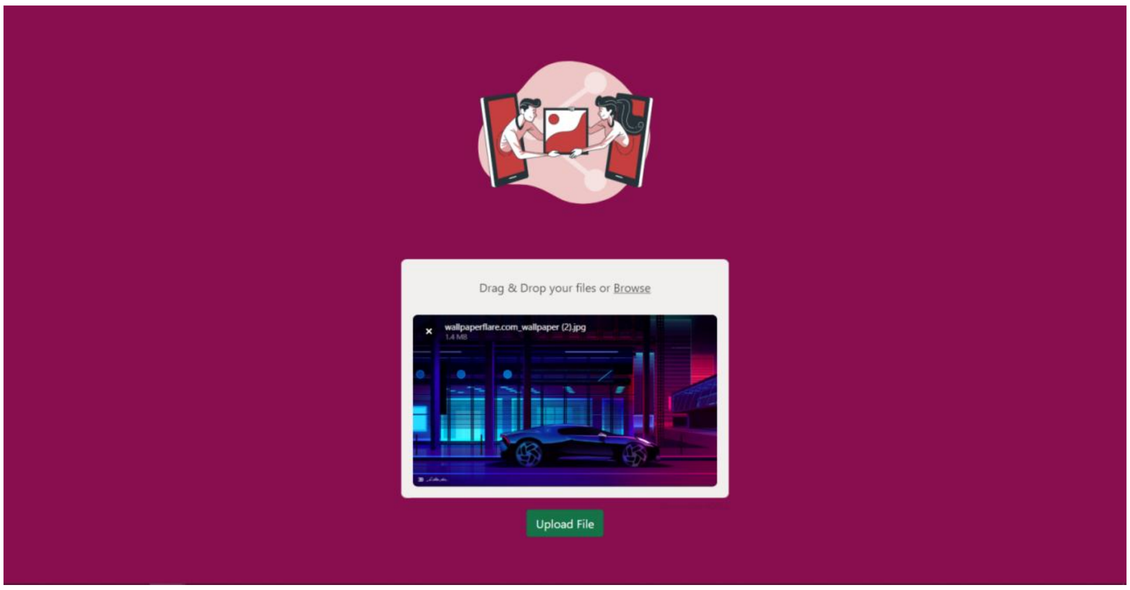
photos July 26, 2023
THE TEDDY
A PARTY TEDDY WITH A LOOK

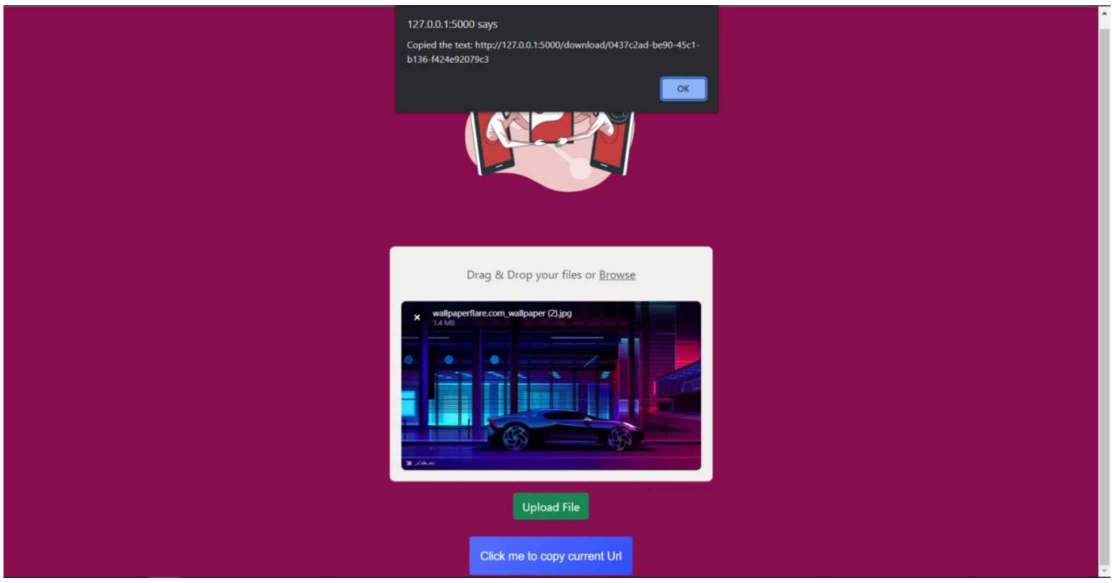
photos July 26, 2023
ASTRONAUT IMAGE
AN CREATIVE HD IMAGE

photos July 26, 2023
TIGER HD IMAGE
TOP COLOUR TONE TIGER IMAGE

127.0.0.1:8000/user/photos/

<u>LINK GENRATOR PAGE:</u>

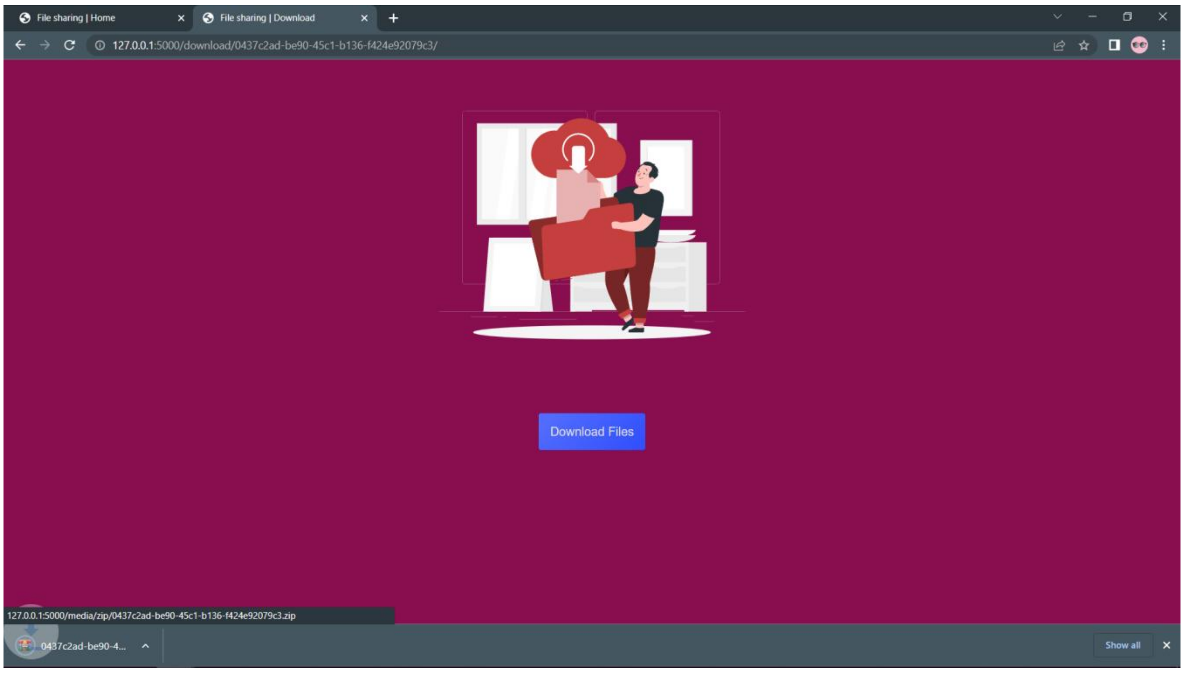

Drag & Drop your files or Browse
wallpaperflare.com_wallpaper (2).jpg
1.4 MB
Upload File


127.0.0.1:5000 says

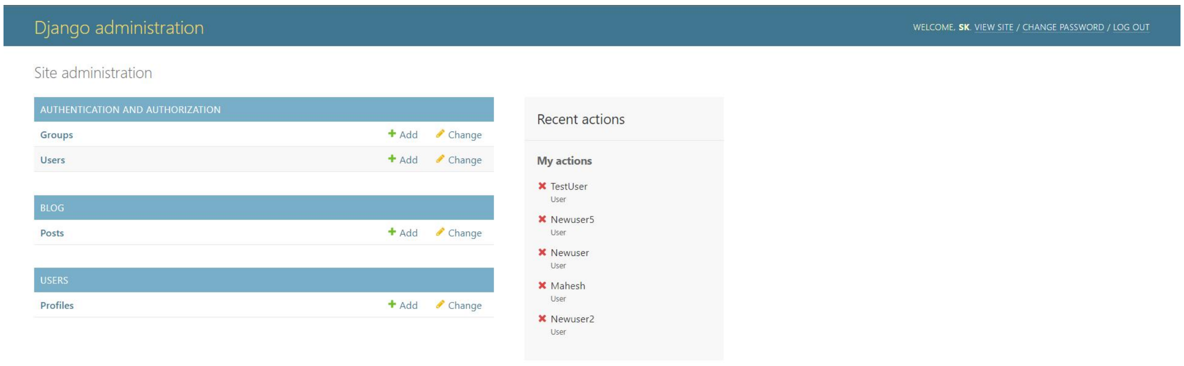
Copied the text: http://127.0.0.1:5000/download/0437c2ad-be90-45c1-b136-f424e92079c3

OK
Drag & Drop your files or Browse
wallpaperflare.com_wallpaper (2).jpg
1.4 MB
Upload File
Click me to copy current Url

DOWNLOAD PAGE:


File sharing | Home
File sharing | Download
127.0.0.1:5000/download/0437c2ad-be90-45c1-b136-f424e92079c3/
Download Files
127.0.0.1:5000/media/zip/0437c2ad-be90-45c1-b136-f424e92079c3.zip
0437c2ad-be90-4...
Show all

ADMIN PAGE:



Django administration

WELCOME, SK. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

BLOG		
Posts	+ Add	Change

USERS		
Profiles	+ Add	Change

Recent actions

My actions

- TestUser User
- Newuser5 User
- Newuser User
- Mahesh User
- Newuser2 User

CONCLUSION

In conclusion, the **'FILE SHARING AND DOWNLOADING WEBAPPLICATION'** is a dynamic project developed using a combination of HTML, CSS, JavaScript, Django, and Django Rest Framework. The platform fosters a collaborative and user-friendly environment for secure and efficient file sharing. Users can easily upload, manage files, and interact through comments and reviews. Vendors are also encouraged to join, creating a vibrant multivendor marketplace that delivers quality products.

The project is designed to meet user requirements while addressing existing file-sharing limitations. Its robust foundation allows for future enhancements. By connecting nearby marts and empowering users to share various files, the platform aims to create a vibrant community of file-sharing enthusiasts.

With a strong focus on delivering quality products, **'FILE SHARING AND DOWNLOADING WEBAPPLICATION'** is dedicated to providing tangible results to users. As the development continues, he platform will evolve to meet users' ever-changing needs and demands.

In summary, the project **'FILE SHARING AND DOWNLOADING WEBAPPLICATION'** seeks to revolutionize the file-sharing experience, offering a seamless and efficient solution for users to share, manage, and access files securely.

FUTURE ENHANCEMENTS

The **"FILE SHARING AND DOWNLOADING WEBAPPLICATION"** will continually improve to meet user demands and technological advancements. Key future enhancements include:

1. Faster Delivery: Implementing faster file transfer protocols and leveraging cloud servers for efficient delivery.

2. Expanded Content: Supporting various file types to cater to diverse user needs.

3. Enhanced User Experience: Improving the interface and navigation for seamless file-sharing.

4. Advanced Search: Empowering users with better search and filtering options.

5. Real-Time Collaboration: Introducing live editing and commenting for collaborative work.

6. Stronger Security: Enhancing data protection with advanced security measures.

7. Mobile App: Developing a dedicated app for on-the-go file management.

8. Social Media Integration: Enabling easy sharing of files and posts on social platforms.

Cloud Deployment:

- Utilizing AWS, Azure, or Google Cloud for scalability and reliability.

BIBLIOGRAPHY

https://www.w3schools.com/

https://www.newtonschool.com/

https://www.codewithharry.com/

https://www.geeksforgeeks.com/