

DataFortress – A Python centric file sharing approach

A Minor Project

Submitted in partial fulfilment of the requirement for the award of the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

By

Kartik Gupta

(Registration No.: 11020210076)



SRM
UNIVERSITY
DELHI-NCR, SONEPAT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM UNIVERSITY DELHI-NCR

Plot no. 39, Rajiv Gandhi Educational City, Sonapat, Haryana – 131029

December, 2023

Approval Sheet

This Project work entitled **DataFortress- A python centric file sharing approach** by **Kartik Gupta** is approved for the degree of **Bachelor of Technology (B.Tech CSE)**.

Examiners

Supervisor(s)

Dr. Puneet Goswami

Ms. Lakshita Aggarwal

Project Coordinator

Head of the Department

Dr. Puneet Goswami

Date: _____

Place: _____

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project entitled **“DATAFORTRESS- A PYTHON CENTRIC FILE SHARING APPROACH”** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering and submitted in the department of Computer Science and Engineering of SRM University, Delhi-NCR, Sonapat, Haryana, India is an authentic record of my own work carried out under the supervision of **Dr. Puneet Goswami & Ms. Lakshita Aggarwal**, as a minor project in 7th semester during the academic year 2023-24. The matter presented in this project has not been submitted for the award of any other degree of this or any other Institute/University.

Kartik Gupta
(11020210076)

CERTIFICATE

This is to certify that the project entitled **“DATAFORTRESS- A PYTHON CENTRIC FILE SHARING APPROACH”** submitted by **Kartik Gupta, 11020210076** to the department of Computer Science and engineering of SRM University, Delhi-NCR, Sonipat, Haryana, (India) in partial fulfillment of the requirements for the award of the degree in Bachelor in Technology in Computer Science and Engineering under the Faculty of Engineering is an authentic record of the work carried out by him under my supervision. In my opinion, this work full films the requirement for which it has been submitted. This project has not been submitted to any other University or Institution for any other degree and is submitted as minor project in 7th semester during the academic year 2023-2024.

Dr. Puneet Goswami
(Supervisor)

H.O.D Dept. of Computer Science and Engineering

Ms. Lakshita Aggarwal
(Co-Supervisor)

Asst. Professor Dept. of Computer Science and Engineering

ACKNOWLEDGEMENT

I would like to thank to **Dr. Puneet Goswami & Ms. Lakshita Aggarwal**, our internal project guides who guided us a lot in completing this project. I would like to thank our institution and faculty members without whom this project would not have been possible.

I would like to acknowledge with much appreciation, the crucial role of the staff of Computer Science and Engineering department, who gave me the permission to use all the required equipment and the necessary resources to complete the project. I would also like to thank my parents for guiding and encouraging me throughout the duration of project.

I am highly indebted and graceful to my teachers' strict supervision, constant encouragement, inspiration and guidance which ensured the worthiness of my work. Working under them was an enriching experience.

ABSTRACT

In this age of developing and sharing loads of content and media on the internet, multiple challenges follow along in the race to develop the trendiest content. The process of content creation requires the effort of sometimes single but multiple times the involvement of a team of multiple people which can include the creators themselves, the editors, the script writers, etc.

The workspace for a content developer is filled up from unstructured data which is difficult to manage and there is a constant need to increase the content pushed on social media platforms more and more. For content creators to post quality content with consistency, they need to hire editors and have to manually verify every file themselves before uploading it to a social media platform. One can provide access to their workspace to an editor but that creates uncertainty and paranoia.

To tackle this, we will be creating a layer which is a python-secure workspace-sharing web application that allows the editor to share the files to a centralized server with permission to post it and the content creator can approve or deny the request to avoid mischief and enable confidentiality, integrity, availability and security of the content to be posted. A data-sharing platform with an Intrusion detection system that works as data storage and uses sessions to create easy communication and efficient means of content sharing.

TABLE OF CONTENTS

TITLE PAGE	I
DECLARATION	II
CERTIFICATE	III
ACKNOWLEDGEMENT	IV
ABSTRACT	V
TABLE OF CONTENTS	VI
LIST OF FIGURES	VII
LIST OF TABLES	VIII
LIST OF ABBREVIATIONS	IX
Chapter 1- Introduction	1
1.1 General Introduction	
1.2 Problem Statement	
1.3 Objective	
1.4 Technologies used	
1.5 Scope	
Chapter 2- Literature Survey	9
2.1 Introduction	
2.2 State of art	
2.3 Tabular Comparison	

Chapter 3- Proposed Methodology	12
3.1 Introduction	
3.2 Block Diagram	
3.3 Algorithms	
3.4 Working details of the system	
3.5 Project Summary	
Chapter 4- Results and Discussions	20
4.1 Implementation details	
4.2 Impact on Society	
4.3 Result	
Chapter 5- Conclusion and Future Work	34
5.1 Conclusion	
5.2 Future Work	
REFERENCES	35

LIST OF FIGURES

Fig. 3.1 Working of the system.....	13
Fig. 3.2 User Login algorithm.....	15
Fig. 3.3 signup algorithm.....	16
Fig. 3.4 Video uploading algorithm.....	17
Fig. 4.1 Landing Page for Datafortress.....	29
Fig. 4.2 Login and signup page for the application.....	30
Fig. 4.3 Video Upload page.....	31
Fig. 4.4 Video list page.....	32
Fig. 4.5 Notifications page.....	33

LIST OF TABLES

Table 2.1 Research paper Highlights	11
---	----

Chapter 1

INTRODUCTION

1.1 General Introduction

In the fast-paced world of content generation, problems with data organization and security regularly arise during the exchange of ideas and creations between editors and content providers. As a result, DataFortress presents itself as a strong online application developed using Python Django [6] with the goal of enhancing data sharing efficiency and improving communication. At its core, DataFortress is a centralized platform that makes it easier for content producers and editors to collaborate easily. With the programme, editors can publish content and media files with titles and descriptions more easily because they have access to the content creator's workspace.

DataFortress's key feature is its content approval system. Editors can upload videos and send relevant notifications, seeking permission from the content creator to make the content public. The content creator, has control through a notification message, allowing them to either approve or deny further processing and publication.

DataFortress fully takes care of the issue of security in this digital world. The online application protects against cyber threats [7] by utilizing an Intrusion Detection System strengthened with capabilities, providing a secure environment for cooperative work. Moreover, DataFortress skillfully handles user sessions, using Django's inbuilt user management system to provide secure connection that maintains integrity and secrecy.

We discuss about the uses of DataFortress, exploring its functionalities, security measures, and the pivotal role it plays to provide a seamless collaboration between editors and content creators. We aim to showcase how DataFortress not only addresses the current challenges in content development but also sets a new standard for secure and efficient workspace sharing.

DataFortress is a Python-based workspace-sharing web application that will allow communication and efficient data sharing between two parties: an editor and a content creator. Nowadays, too much network data and physical/cloud storage are being wasted while fulfilling the needs of a content development page. DataFortress provides a centralized system where both the content creator and the editor can sign up. With DataFortress having access to the content creator's workspace, the editor can upload the content with the required prerequisite details such as title/description for a media file, and further ask the content creator for permission to upload the content publicly. The owner of the workspace can then approve/deny further processing and publishing the content by accessing a notification dialog on their desired platform. The web application will be secured using an Intrusion Detection System[8] which identifies cyberattacks. It will also manage sessions and store keys which will be secured by cryptography to enable secure communication and maintain integrity and security.

1.2 Problem Statement

We encounter a lot of difficulties in sharing and storing media on the internet in the current content creation and sharing scenario. Unstructured data is a major obstacle to effective management in the workspace of content developers, no matter what the source is. The constant requirement for more content on social media platforms makes organization even more necessary. In an effort to maintain consistency and quality, content providers frequently have to deal with the difficulties of employing editors and manually checking each file before uploading it. Editor access puts the content's security and confidentiality at risk by creating a sense of uncertainty and fear.

- In the era of online content, managing files for creators, editors, and writers becomes chaotic due to unstructured data in workspaces.
- Content creators face manual verification headaches, hiring editors, and ensuring file quality before sharing on social media.
- Granting editors access to workspaces raises concerns about confidentiality, creating uncertainty and paranoia.
- The constant need to push more content on social media adds pressure on creators and editors.

1.3 Objective

DataFortress is a secure web application based on Python that aims to transform the way editors and creators cooperate, offering an answer to the complex problems encountered in content production and collaboration. Our main goal is to implement a centralized system that will enable smooth file sharing [9] while giving content creators a degree of control over their work. By granting editors permission to share files and giving content producers the power to accept or reject requests, this promotes collaboration while reducing the possibility of unauthorized changes.

DataFortress goes beyond conventional data-sharing platforms by integrating a Intrusion Detection System. This advanced security measure transforms DataFortress into a secure data storage hub, actively identifying potential cyber threats. By prioritizing the CIA triad, our application ensures that content remains safeguarded against unauthorized access and tampering.

- To build a workspace-sharing web application using the Python Django framework.
- To implement cybersecurity by using cryptography.
- To use an Intrusion detection system for the web-based application.
- To enable Python scripting/APIs for notifications and updates.

1.4 Technologies Used

1. Visual Studio Code (VS Code):

Visual Studio Code [10], developed by Microsoft, stands as a preeminent source code editor, which is lightweight and offers great versatility. With its rich feature set and support for a wide range of programming languages, Visual Studio Code has established itself as a top tool for developers worldwide. It can integrate with various extensions that makes it a powerful tool for coding tasks. With cross-platform compatibility, Visual Studio Code has been a pioneer in the realm of code editing, providing a user-friendly environment for developers to write, debug, and collaborate on their projects.

2. Python:

Python, a high-level and interpreted programming language, received much praise for its readability, simplicity, and versatility. Python is widely used by developers for web development, data science, machine learning, automation, and scripting. It is a powerful tool with many uses. Python's vast library and framework ecosystem, along with its simple syntax, make it a useful language for programmers of all skill levels. Python, which emphasizes code readability and offers an expressive syntax, is still an essential tool in the programming world, powering everything from complex algorithms to expansive web frameworks.

3. Django:

Django is a high-level Python web framework that encourages rapid development and clean, logical design. It handles a lot of the hassle of web development and was built by experienced developers, allowing you to concentrate on developing your app instead of having to start from scratch. It is open source and free. Django was created to assist developers in completing applications as soon as possible, from conception to completion. Django takes security seriously and helps developers prevent many usual security mistakes. Django reduces the amount of duplicate code by including an admin interface, a templating engine, and an Object-Relational Mapping (ORM) [11] framework. As a result, Django has become the framework of choice for developers seeking a powerful and elegant solution for web application development.

4. DBMS (Database Management System):

Database Management Systems (DBMS) is a fundamental technology used in applications for data management, retrieval, and storage. These systems, which include SQLite, PostgreSQL, and MySQL, are essential for preserving data security, allowing efficient data retrieval, and preserving data integrity. Whether employed in web applications, business systems, or analytical tools, DBMS provides a structured and organized approach to handling vast amounts of information. It is crucial for establishing a smooth interface between apps and data so that programmers may work with information in a structured and safe way.

5. HTML/CSS:

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) create the bedrock of web development, providing the means to structure and style web page content. HTML defines the skeletal structure of a webpage, outlining elements such as headings, paragraphs, and images. CSS, on the other hand, dictates the visual presentation and layout, allowing developers to create aesthetically pleasing and responsive user interfaces. Together, HTML and CSS form an integral part of the web development stack, enabling developers to craft engaging and visually appealing websites and applications.

6. JavaScript:

JavaScript, a dynamic scripting language, plays a pivotal role in enabling interactivity and dynamic content on the client side of web applications. JavaScript is widely used to improve user experience via asynchronous communication, real-time updates, and dynamic content manipulation. JavaScript has become more widely used in server-side programming with the introduction of potent frameworks and libraries like React, Angular, and Vue.js. These tools provide a full toolkit for creating feature-rich, responsive, and modern web apps.

7. Machine Learning:

Machine Learning (ML) represents a new era in computing, enabling systems to learn from data and make informed decisions without explicit programming. Machine learning algorithms are revolutionizing technology through their application in various fields such as image identification, recommendation systems, natural language processing, data analysis, and more. Strong frameworks for creating and deploying machine learning models are provided by libraries like TensorFlow and scikit-learn, enabling developers to use predictive analytics and pattern recognition in their applications.

8. Cryptography:

Cryptography, the science of securing communication and data, employs algorithms and keys to transform information into a secure format. Cryptography is used in secure communication, password hashing, digital signatures, and many other areas of information security. It is crucial for maintaining secrecy and integrity. Developers may build a foundation of confidence in digital transactions and communications, safeguard against unauthorized access, and guarantee the privacy of sensitive information by utilizing cryptographic techniques [12].

1.5 Scope

The goal of DataFortress is to transform data sharing and communication between editors and content creators by developing and deploying a powerful web application for workspace sharing based on Python. The project intends to solve the lack of physical and cloud storage and unnecessary network data that are present in content production. A streamlined registration process is offered by DataFortress, a centralized platform for content authors and editors.

- **Centralized Collaboration:**

Establishing a centralized system to facilitate efficient collaboration between content creators and editors.

- **User Registration and Workspace Access:**

Enabling content creators and editors to register on the platform, granting DataFortress access to the content creator's workspace.

- **Content Upload and Approval Workflow:**

Allowing editors to upload content with necessary details (title/description) and seek permission from content creators for public upload.

Implementing a workflow for content creators to approve or deny requests through a notification dialog on their preferred platform.

- **Security Measures:**

Integrating an Intrusion Detection System to identify and thwart potential cyberattacks.

- **Key Management and Cryptography:**

Ensuring secure communication by managing sessions and storing keys, utilizing cryptography to maintain data integrity and security.

Chapter-2

LITERATURE SURVEY

2.1 Introduction

This chapter will go over some papers in order to gain a better grasp of the strategies that have been offered. All of these strategies have the same goal in mind. These papers will help us easily understand the project and the functionalities to be used.

2.2 State of Art

Leho tedersoo et al [1] examines data sharing practices across scientific disciplines, revealing disparities despite overall improvement. Leho tedersoo et al explains moving away from vague data availability statements and advocates for motivating researchers with tangible benefits like recognition. The study, focusing on articles in Nature and Science, found increased data availability from 54.2% to 71.8% over two 10-year intervals. However, variations exist across fields, with energy, catalysis, psychology, optics, and photonics showing lower availability. Nature had significantly less data accessible. The paper underscores the importance of refining data sharing principles and policies for a more collaborative scientific community.

Adamya Shyam et al [2] examines the creation of an online platform for academic resource sharing, focusing on class notes and question papers. The platform uses software engineering principles and follows an iterative model, ensuring its viability. The feasibility study identifies key objectives, including user registration, educational categories, and subcategories. The project is described as efficient, user-friendly, and simple, with an ER diagram and flow chart for guidance. The project successfully implemented using the iterative model and black box testing.

Himanshu Gore et al [3] explains the Django web framework, presenting its advantages and providing step-by-step instructions for beginners. It covers app creation, and form implementation within views, showcasing Django's efficiency in interacting with databases for dynamic web development. The introduction of class-based views as an alternative for handling HTTP requests adds depth to the exploration. Emphasizing code organization, the paper stresses the placement of business logic in view.py and URL definitions in urls.py for effective project structure. The inclusion of form URLs in the

project's `urls.py` file is highlighted, allowing readers to run the server and access the form. Himanshu Gore et al underscores the role of Django templates in crafting dynamic web pages, demystifying the parameters involved. Additionally, it sheds light on the use of templates, constructed with HTML, CSS, and JavaScript, offering a powerful means to structure websites. A practical example illustrates Django's form functionality, offering a code snippet for form template creation and explaining data handling in views.

Muthi Reddy P. et al [4] examines cloud computing and data security, delving into key topics such as secure access control, index generation techniques, anti-collusion data sharing, and user revocation in cloud storage. It covers privacy-preserving information distribution, resource allocation in cloud gaming, and data confidentiality in distributed computing. It underscores the significance of addressing challenges in user revocation and access control within cloud environments. It advocates for efficient and secure techniques like role-based access control and attribute-based encryption. Future research directions are suggested, including improving leakage resilience and exploring self-destructing encryption methods. Overall, the paper contributes valuable insights to recent research in cloud computing and data security, emphasizing the importance of protecting sensitive data in the evolving landscape of cloud technology.

Yiming Sun et al [5] explains the Marshall McLuhan's seminal concepts of "media as the message" and "media as human extension," providing a foundational understanding of the role of media technology in shaping human culture. Focusing on digital media, the paper conducts a thorough analysis of IoT technology's application in content creation. It explores WSN and NB-IoT technologies separately, combining them to investigate the networking structure and related fusion technologies. The paper highlights the implementation of the random forest (RF) algorithm for pattern classification and regression prediction. It intricately examines the roles and functions of each layer in IoT technology, emphasizing the perception layer and system parameter self-tuning curve. It reflects on the past, present, and future of IoT, adopting a user-centered and smart environment perspective. This paper contributes to the field by not only grounding its analysis in McLuhan's influential theories but also by providing a comprehensive examination of IoT technologies, including their application in digital media content creation and their potential impact on the evolving landscape of smart environments.

2.3 Tabular Comparison

Table 2.1 Research paper Highlights

Ref.	Year	Highlights of the paper
[1]	2021	<ul style="list-style-type: none"> Examines data sharing practices across disciplines, revealing disparities Focuses on Nature and Science articles, noting increased data availability Underscores the importance of refining data sharing principles for collaboration
[2]	2020	<ul style="list-style-type: none"> Explores creation of an online platform for academic resource sharing Uses software engineering principles and follows iterative model for viability Identifies key objectives: user registration, educational categories, and subcategories Described as efficient, user-friendly, and simple with successful implementation and black box testing
[3]	2021	<ul style="list-style-type: none"> Explains Django web framework, focusing on advantages and step-by-step instructions for beginners Covers app creation, form implementation, and database interaction for dynamic web development Introduces class-based views for handling HTTP requests Emphasizes code organization, Django templates, and practical example of form functionality
[4]	2017	<ul style="list-style-type: none"> Examines cloud computing and data security, covering secure access control, index generation, and anti-collusion Addresses challenges in user revocation and access control in cloud environments Advocates for efficient techniques like role-based access control and attribute-based encryption Suggests future research directions including improving leakage resilience and exploring self-destructing encryption
[5]	2022	<ul style="list-style-type: none"> Highlights the random forest algorithm, layer roles in IoT, and system parameter self-tuning curve Reflects on past, present, and future of IoT from user-centered and smart environment perspective Contributes insights grounded in McLuhan's theories and comprehensive examination of IoT technologies

Table 2.1 illustrates a comparison between different research papers. The papers highlight the usage of python Django web framework, it's advantages, how Django is used in data sharing, cloud computing and data security in cloud sharing environments and the use of Django's inbuilt features for creating a web application.

Chapter-3

PROPOSED METHODOLOGY

3.1 Methodology

To create the web application, We'll use Python Django framework and it's features to create the Backend functionality [13] for the application and Django templates and static files to build the Frontend web pages.

We use the following features in our web application:

1. Platform Setup:

- Employ Python Django for web application development.
- Establish a centralized collaboration platform for content creators and editors.

2. User Access and Registration:

- Enable secure user registration for content creators and editors.
- Facilitate DataFortress access to the content creator's workspace.

3. Content Approval System:

- Develop a streamlined content approval process for editors.
- Implement a notification system for content creators to approve or deny content processing.

4. Security Measures:

- Utilize Intrusion Detection System.
- Secure sessions and keys with cryptography for data integrity.

5. File Upload Process:

- Enable editors to upload content with prerequisite details.
- Implement permission-seeking mechanism for public content upload.

6. Testing and Continuous Improvement:

- Conduct thorough testing, including black-box testing, for robustness.
- Implement iterative development for continuous improvement based on feedback [14].

3.2. Block Diagram

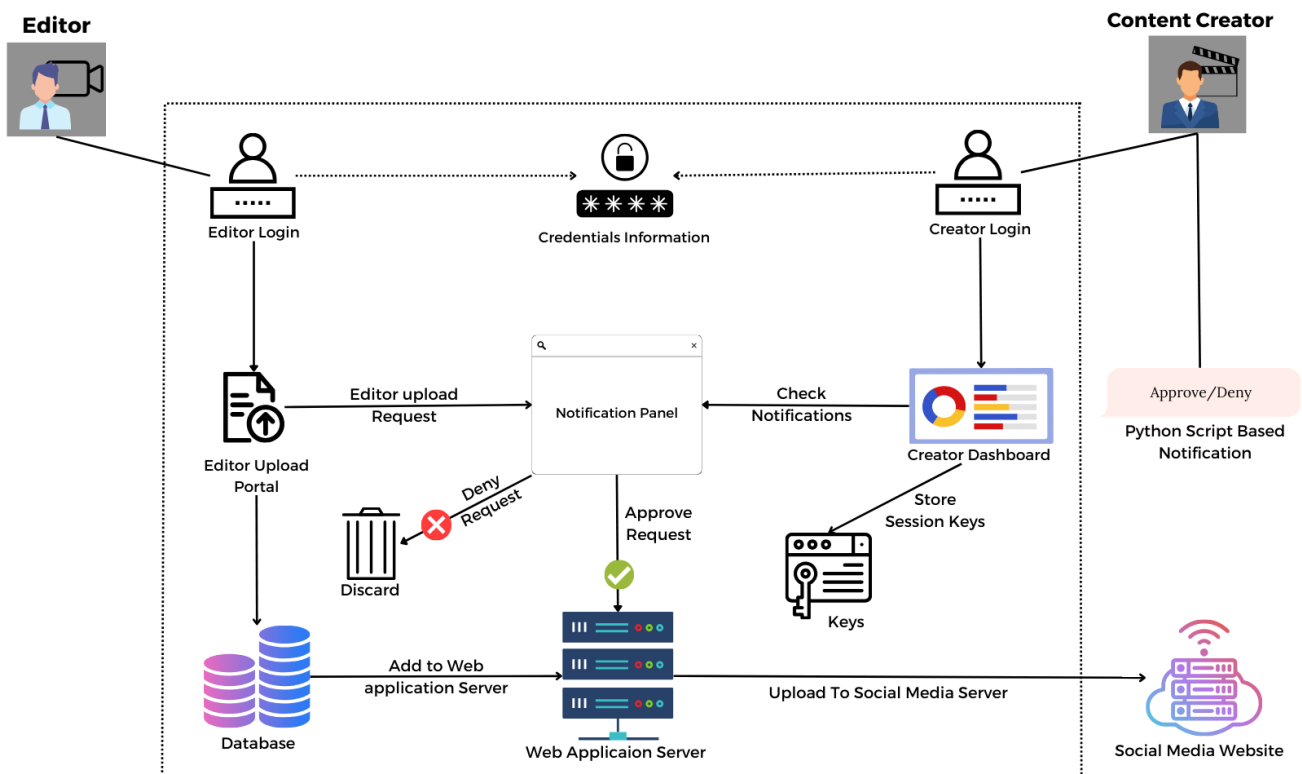


Fig. 3.1 Working of the system

Fig. 3.1 depicts the flow of the working of the system. The blocks represent entities which includes editor and creator as master users, pages including the landing page, editor login, creator login, upload portal, creator dashboard, notification panel and database file viewer. This architecture displays how the system workflow is designed and how it will be used to perform designated tasks.

To summarize our study, we intend to create a python based data sharing system that enables authentication and authorization for secure access to certain features. We use the application to share video files among users registered on the application. The app employs a video upload, a video list and a notification list system all of which have separate functionalities.

1. **Editor Login:** Upon launching the website, users are presented with a login screen where they can enter their credentials (username and password) to access the editor dashboard.
2. **Editor Upload:** After successfully logging in, users can upload their content to the social media website using the upload feature. The application supports multiple file formats for content upload.
3. **Creator Login:** The creator login feature allows content creators to access their personal dashboard where they can manage all aspects of the content creation process.
4. **Creator Dashboard:** The creator dashboard is the primary workspace for content creators, enabling them to upload content, manage content upload requests, and approve/deny user-generated content.
5. **Notification Panel:** The notification panel is a built-in feature of the application that provides users with real-time updates on the status of their content upload requests. It includes features such as:
 - Displaying and Checking notifications
 - Deleting Notifications
6. **Python Script Based Notification:** The application features a Python script-based notification system that allows users to receive real-time updates on the status of their content upload requests.
7. **Database:** The database is responsible for storing and retrieving information related to the content creation process. It stores information such as:
 - User credentials
 - Content upload requests
 - Session keys
 - Social media website notifications

3.3 Algorithms

3.3.1 login_user

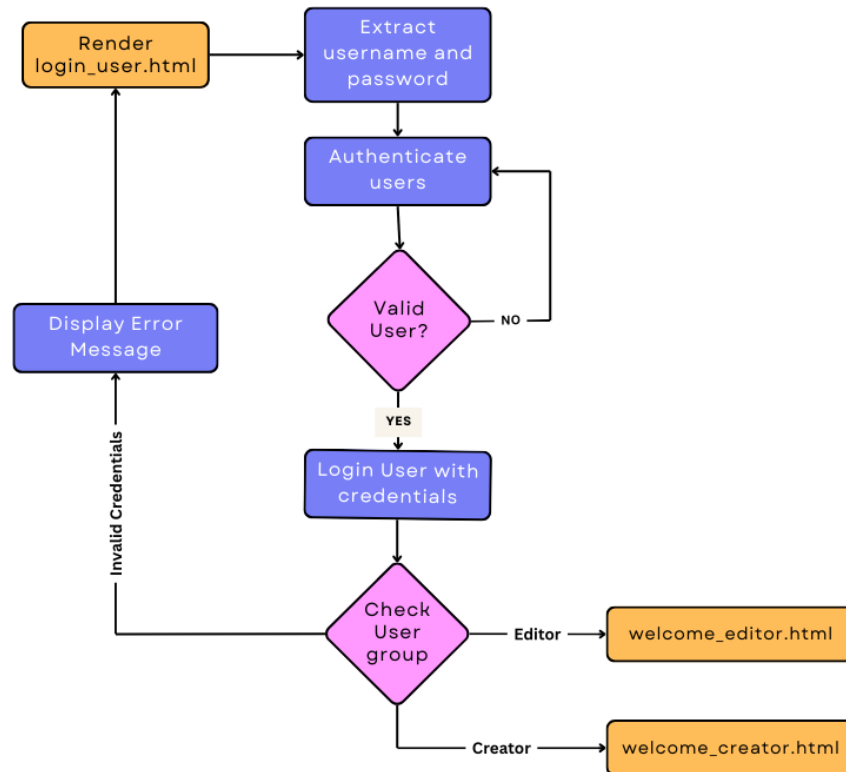


Fig. 3.2 User Login algorithm

Fig. 3.2 depicts the steps the code takes to render the `login_user.html` page. First, the user submits their login credentials, which are then extracted and used to authenticate the user. If the authentication is successful, the user is logged in and redirected to a corresponding welcome page based on their group. If the authentication fails, the user is presented with an error message prompting them to enter their correct login credentials.

3.3.2 signup

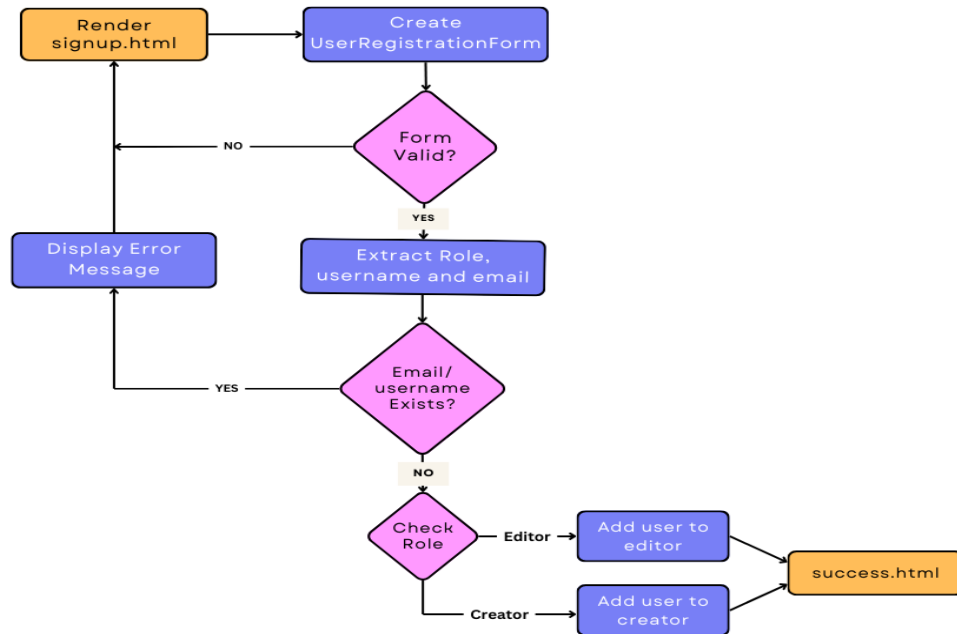


Fig. 3.3 signup algorithm

Fig. 3.3 depicts the signup form, where they are required to input their desired role, username, and email. Once the form is submitted, the application validates the user's input to ensure that it meets the specified criteria. If the user's input passes the validation, the system proceeds to check if the provided email or username already exists within the system. If either of these is already in use, the application generates an error message to notify the user of the duplication. However, if both the email and username are unique, the application evaluates the user's selected group. Based on the role, the system then adds the user to the appropriate group.

3.3.3 upload_video

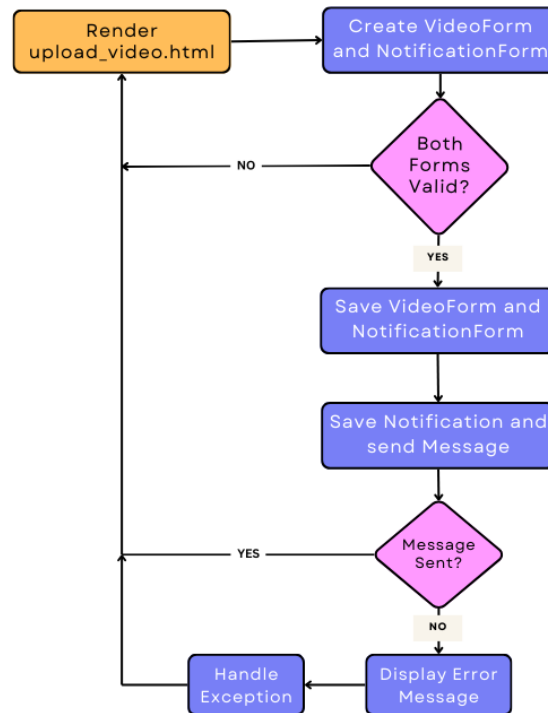


Fig. 3.4 Video uploading algorithm

Fig 3.4 illustrates the process of uploading a video to a database. It starts by rendering the upload_video.html page, which includes two forms: a VideoForm and a NotificationForm. If both forms are valid, the video and notification are saved to the database, and a message is sent. If either form is invalid, an error message is displayed.

3.4 Working details of the system

This section discusses about the system entities, how they combine and work together to enable the purpose of the application. The Django templates [16] and the static files work in order to create the front end for the web application. The back end consists of python Django files that include URLs, views, models, forms, settings, databases etc. to render the static and template files. The system contains authentication and authorization [15] algorithms with basic validators and assists in creating accounts. The database sharing is done through resource sharing and user grouping. The notifications and media files are managed by the Django's database handling features. The security is maintained using Django admin and external intrusion detection systems.

3.4.1 Security:

The following features are offered in order to guarantee the security of user credentials and content:

- **Credential Information:** Users must authenticate themselves with a special username and password in order to access this protected credential information system.
- **Login:** To access the editor dashboard, users must provide their login information (password and username).
- **Secure File Upload and Storage:** Using Django's built-in file handling, this system allows files to be uploaded and stored safely.

3.4.2 Usability:

The following features are offered in order to improve user experience and make the platform more user-friendly:

- **User-Friendly Interface:** A user-friendly interface that makes use of responsive design principles to guarantee functionality and accessibility on a range of screens and devices.

3.4.3 Scalability:

The following features are suggested in order to effectively manage content upload requests and accommodate a high number of users:

- **Role-Based Access Control:** Using Django's built-in support for permissions and groups, this role-based access control system uses roles to allocate users based on their duties and roles inside the DataFortress platform.
- **Notification Panel:** An integrated function that gives users up-to-date information on the progress of their requests for content uploads.
- **Python Script Based Notification:** Users can get real-time updates on the progress of their requests to submit content using this PyWhatKit notification system[17].

3.4.4 Efficiency:

The following features are offered in order to enhance the platform's overall effectiveness and performance:

- **Web Application Server:** This scalable web application server handles requests from social networking websites and user interfaces.
- **Database:** A database used to store and retrieve data about the process of creating content.

3.4.5 Accessibility:

To ensure the platform is accessible [18] to all users, including those with disabilities, the following features are proposed:

- **Keyboard Navigation:** The platform is fully accessible via keyboard navigation.

3.5 Project Summary

DataFortress, a Python-based workspace-sharing web application, works as a solution to modern day content creation and data sharing problems. DataFortress establishes a platform to completely remove security and data storage issues. It facilitates a more effective and streamlined process for uploading and sharing content [19]. The content approval feature keeps the process systematic and free of errors.

DataFortress gives strong security measures a priority in response to the data security industry's growing importance. It uses an intrusion detection system to successfully defend against cyberattacks. This guarantees the integrity and confidentiality of the transferred content in addition to a secure connection. DataFortress establishes a new standard for secure and effective collaborative content development with these security safeguards in place.

In essence, DataFortress changes the ways of data sharing by providing a user-friendly and secure platform for both editors and content creators. Through its centralized approach, streamlined approval process, and security features, DataFortress aims to enhance the experience, ultimately contributing to a more efficient and secure environment for content sharing and creation.

Chapter-4

RESULTS AND DISCUSSION

4.1 Implementation Details

For this Project, we will be using the Python's Django framework that is used for web development in python. It contains certain necessary files including:

1. urls.py

1.a in DataFortress application

```
datafortress > urls.py > ...
1  from django.urls import path
2  from . import views
3
4
5  urlpatterns = [
6      path('', views.index , name='index'),
7      path('login_user' , views.login_user , name='login_user'),
8      path('logout' , views.logout , name='logout'),
9      path('signup' , views.signup , name='signup'),
10     path('success', views.success , name='success'),
11     path('welcome_editor', views.welcome_editor, name='welcome_editor'),
12     path('welcome_creator', views.welcome_creator, name='welcome_creator'),
13     path('upload_video', views.upload_video, name='upload_video'),
14     path('video_list', views.video_list, name='video_list'),
15     path('delete_video/<int:video_id>/', views.delete_video, name='delete_video'),
16     path('approve_video/<int:video_id>/', views.approve_video, name='approve_video'),
17     path('notification_list/', views.notification_list, name='notification_list'),
18     path('notifications/delete/<int:notification_id>/', views.delete_notification, name='delete_notification'),
19     path('video_list', views.video_list, name='video_list'),
20     path('homepage', views.homepage, name='homepage'),
21
22 ]
```

1.b in base folder

```
major > urls.py > ...
1  from django.contrib import admin
2  from django.urls import path , include
3  from django.conf import settings
4  from django.conf.urls.static import static
5
6
7  urlpatterns = [
8      path('admin/', admin.site.urls),
9      path('', include('datafortress.urls')),
10 ] + static(settings.MEDIA_URL, document_root= settings.MEDIA_ROOT)
```

2. forms.py

2.a Importing Libraries

```

1  from django import forms
2  from django.contrib.auth.forms import UserCreationForm
3  from django.contrib.auth.models import User
4  from django.core.exceptions import ValidationError
5  from django.contrib.auth.validators import ASCIIUsernameValidator
6  from django.contrib.auth.password_validation import validate_password
7  from django.utils.translation import gettext_lazy as _
8  from .validators import *
9  from .models import Video, Notification

```

2.b UserRegistrationForm

```

11 class UserRegistrationForm(UserCreationForm):
12     email = forms.EmailField(max_length=254, help_text='Required. Enter a valid email address.')
13     role = forms.ChoiceField(choices=[('editor', 'Editor'), ('creator', 'Creator')])
14     username = forms.CharField(
15         label=_("Username"),
16         validators=[ASCIIUsernameValidator()],
17         max_length=150,
18         widget=forms.TextInput(attrs={'class': 'form-control'}),
19     )
20     password1 = forms.CharField(
21         label="Password",
22         strip=False,
23         widget=forms.PasswordInput(attrs={'autocomplete': 'new-password'}),
24         validators=[validate_password_has_uppercase, validate_password_special_characters, validate_password_mixed_case],
25     )
26
27     class Meta:
28         model = User
29         fields = ['username', 'email', 'password1', 'password2', 'role']
30
31     def clean_password1(self):
32         password1 = self.cleaned_data.get('password1')
33
34         try:
35             validate_password(password1, self.instance)
36         except ValidationError as e:
37             self.add_error('password1', e.messages[0])
38
39     return password1

```

```
40
41     def clean_password2(self):
42         cleaned_data = super().clean()
43         password1 = cleaned_data.get('password1')
44         password2 = cleaned_data.get('password2')
45
46         if password1 and password2 and password1 != password2:
47             raise ValidationError("Passwords do not match.")
48
49         return password2
50
51     def clean_username(self):
52         username = self.cleaned_data.get('username')
53         if username:
54             try:
55                 ASCIIUsernameValidator()(username)
56             except ValidationError as e:
57                 raise forms.ValidationError(str(e))
58         return username
59
```

2.c VideoForm and NotificationForm

```
61
62     class VideoForm(forms.ModelForm):
63
64         class Meta:
65             model = Video
66             fields = ['title', 'description', 'video_file', 'is_public']
67
68
69     class NotificationForm(forms.ModelForm):
70         class Meta:
71             model = Notification
72             fields = ['recipient', 'message', 'phone']
73
74         def __init__(self, editor, *args, **kwargs):
75             super(NotificationForm, self).__init__(*args, **kwargs)
76             # Limit the recipient choices to users in the "creator" group
77             self.fields['recipient'].queryset = User.objects.filter(groups__name='creator')
```


3. models.py

```
1  from django.db import models
2  from django.contrib.auth.models import User
3
4
5  class Video(models.Model):
6      title = models.CharField(max_length=255)
7      description = models.CharField(max_length=500)
8      video_file = models.FileField(upload_to='media/')
9      is_public = models.CharField(max_length=20)
10
11     def __str__(self):
12         return self.name + ": " + str(self.videofile)
13
14
15     class Notification(models.Model):
16         sender = models.ForeignKey(User, on_delete=models.CASCADE, related_name='notifications_sent')
17         recipient = models.ForeignKey(User, related_name='received_notifications', on_delete=models.CASCADE)
18         message = models.CharField(max_length=200)
19         timestamp = models.DateTimeField(auto_now_add=True)
20         phone = models.CharField(max_length=15, default='')
21         is_read = models.BooleanField(default=False)
22
23         def mark_as_read(self):
24             self.is_read = True
25             self.save()
```

4. views.py

4.a Importing Libraries

```
1  import os
2  from django.shortcuts import get_object_or_404, render, redirect
3  from django.contrib.auth.models import User, auth, Group
4  from django.contrib import messages
5  from .forms import UserRegistrationForm, VideoForm, NotificationForm
6  from django.contrib.auth import login, authenticate
7  from .models import Video, Notification
8  from django.contrib.auth.decorators import user_passes_test, login_required
9  from django.conf import settings
10 from .whatsapp_utils import send_whatsapp_message
```

4.b Functions for urls

```
12 def index(request):
13     return render(request , 'index.html')
14
15 def user_in_editor(user):
16     return user.groups.filter(name='editor').exists()
17
18 def user_in_creator(user):
19     return user.groups.filter(name='creator').exists()
20
21 def homepage(request):
22     user1 = request.user
23     user1_group = user1.groups.first()
24     if user1_group.name == 'creator':
25
29 def login_user(request):
30     if request.method == 'POST':
31         username = request.POST['username']
32         password = request.POST['password']
33         user = authenticate(request, username=username, password=password)
34         if user is not None:
35             login(request, user)
36             if user.groups.filter(name='editor').exists():
37                 return redirect('welcome_editor')
38             elif user.groups.filter(name='creator').exists():
39                 return redirect('welcome_creator')
40         else:
41             messages.info(request, "Invalid Credentials")
42             return redirect('login_user')
43     return render(request, 'login_user.html')
44
45 @user_passes_test(user_in_editor)
46 def welcome_editor(request):
47     return render(request, 'welcome_editor.html')
48
49 @user_passes_test(user_in_creator)
50 def welcome_creator(request):
51     return render(request, 'welcome_creator.html')
52
53
54 def logout(request):
55     auth.logout(request)
56     return redirect('/')
```

```
59 def signup(request):
60     if request.method == 'POST':
61         form = UserRegistrationForm(request.POST)
62         if form.is_valid():
63             x = form.cleaned_data['role']
64             u = form.cleaned_data['username']
65             e = form.cleaned_data['email']
66             if User.objects.filter(email = e).exists():
67                 messages.info(request , "Email Already in use.")
68                 return redirect('signup')
69             elif User.objects.filter(username = u).exists():
70                 messages.info(request , "Username Already in use.")
71                 return redirect('signup')
72             else:
73                 form.save()
74                 if x == 'editor' or x == 'Editor':
75                     user = User.objects.get(username = u)
76                     group = Group.objects.get(name = "editor")
77                     user.groups.add(group)
78                 elif x == 'creator' or x == 'Creator':
79                     user = User.objects.get(username = u)
80                     group = Group.objects.get(name = "creator")
81                     user.groups.add(group)
82                 return redirect('success')
83         else:
84             form = UserRegistrationForm()
85         return render(request, 'signup.html', {'form': form})
86
87
88 def success(request):
89     return render(request , 'success.html')
```

```

91 @user_passes_test(user_in_editor)
92 @login_required
93 def upload_video(request):
94     if request.method == 'POST':
95         form = VideoForm(request.POST, request.FILES)
96         notif_form = NotificationForm(request.user, request.POST)
97         if form.is_valid() and notif_form.is_valid():
98             try:
99                 form.save()
100                 title = form.cleaned_data['title']
101                 notification = notif_form.save(commit=False)
102                 notification.sender = request.user
103                 to_number = notif_form.cleaned_data['phone']
104                 new_message = notif_form.cleaned_data['message']
105                 message = f"Title: {title}, Message: {new_message}"
106                 notification.save()
107                 success, error_message = send_whatsapp_message(to_number, message)
108                 messages.info(request, "Video Uploaded Successfully.")
109                 if success:
110                     messages.info(request, "Notification Sent Successfully.")
111                     return redirect('upload_video')
112                 else:
113                     messages.info(request, "Error sending WhatsApp notification.")
114                     return redirect('upload_video')
115             except Exception as e:
116                 error_message = f"An error occurred during file upload: {str(e)}"
117                 return render(request, 'upload_error.html', {'error_message': error_message})
118         else:
119             error_message = "Form is not valid. Please check your inputs."
120             return render(request, 'upload_error.html', {'error_message': error_message})

121     else:
122         form = VideoForm()
123         notif_form = NotificationForm(request.user)
124         return render(request, 'upload_video.html', {'form': form, 'notif_form': notif_form})
125
126
127 @login_required
128 def video_list(request):
129     videos = Video.objects.all()
130     return render(request, 'video_list.html', {'videos': videos})
131
132 @login_required
133 def delete_video(request, video_id):
134     video = get_object_or_404(Video, pk=video_id)
135     video_path = os.path.join(settings.MEDIA_ROOT, str(video.video_file))
136     if request.method == 'POST':
137         video.delete()
138     if os.path.exists(video_path):
139         os.remove(video_path)
140     return redirect('video_list')
141
142 @login_required
143 def approve_video(request, video_id):
144     return redirect('video_list')

```

```

148
149 @login_required
150 def notification_list(request):
151     # Retrieve notifications for the current user (creator)
152     notifications = Notification.objects.filter(recipient=request.user).order_by('-timestamp')
153     return render(request, 'notification_list.html', {'notifications': notifications})
154
155
156 @login_required
157 def delete_notification(request, notification_id):
158     notification = get_object_or_404(Notification, pk=notification_id)
159     if notification.recipient == request.user:
160         notification.delete()
161     return redirect('notification_list')

```

5. validators.py

```

1  from django.core.exceptions import ValidationError
2  import re
3
4  def validate_password_special_characters(value):
5      if not re.search(r'[@#$%^&*(),.?":{}|<>_]', value):
6          raise ValidationError("The password must contain at least one special character.")
7
8  def validate_password_has_uppercase(value):
9      if not any(char.isupper() for char in value):
10         raise ValidationError("The password must contain at least one uppercase letter.")
11
12  def validate_password_mixed_case(value):
13      if not any(char.islower() for char in value) or not any(char.isupper() for char in value):
14         raise ValidationError("The password must contain both uppercase and lowercase characters.")

```

6. whatsapp_utils.py

```

1  import pywhatkit as kit
2  import datetime
3
4  def send_whatsapp_message(to_number, message):
5      try:
6          # Format the phone number as '+<country_code>phone_number'
7          to_number = f'+{to_number}'
8
9          # Send the WhatsApp message
10         kit.sendwhatmsg(to_number, message, datetime.datetime.now().hour, datetime.datetime.now().minute+1)
11         return True, None # Success
12     except Exception as e:
13         error_message = str(e)
14     return False, error_message

```

Static and Template Files

```

  ✓ static
    > assets
    # stylehome.css
    # stylelogin.css
    # styles.css
    # uploadstyle.css
  ✓ templates
    <> index.html
    <> login_user.html
    <> notification_list.html
    <> signup.html
    <> success.html
    <> upload_error.html
    <> upload_video.html
    <> video_list.html
    <> welcome_creator.html
    <> welcome_editor.html
```

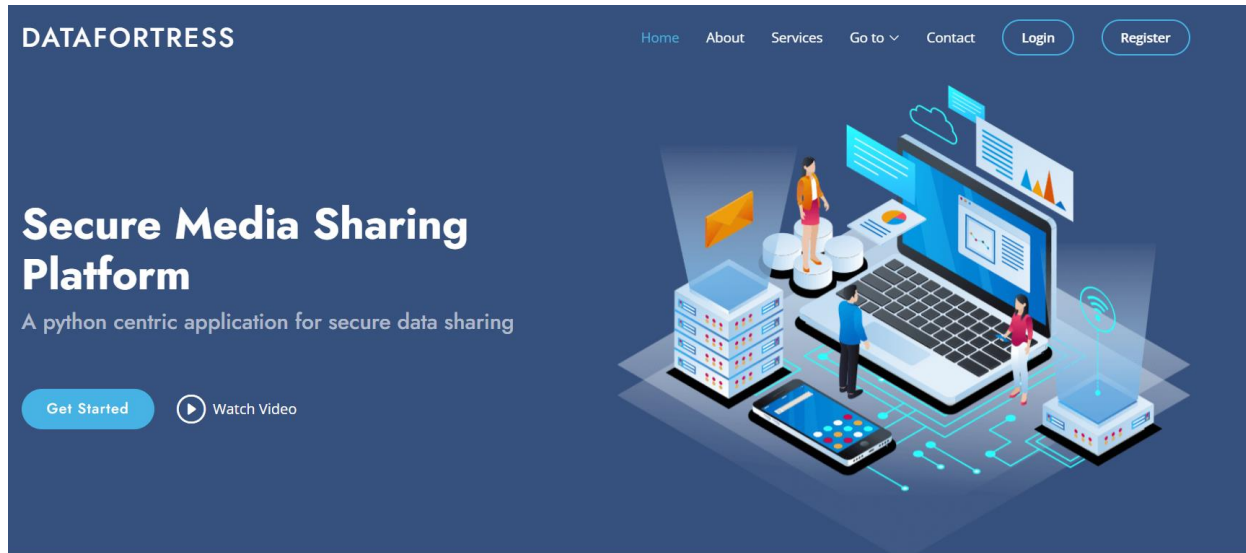
4.2 Impact on Society

We have the potential to positively impact society, As we implement more security and ease of data sharing available to more and more users. The issues related to data storage and privacy will be solved by creating secure and efficient collaboration in content development, safeguarding intellectual property, and promoting transparent communication.

4.3 Result

As a result, we have created a Python-based workspace-sharing web application, designed to ease data sharing between editors and content creators. The project aims to address the current inefficiencies in content development, where excessive network data and physical/cloud storage are underutilized. DataFortress provides a centralized platform, offering a seamless signup process for both content creators and editors.

4.3.1 Landing Page

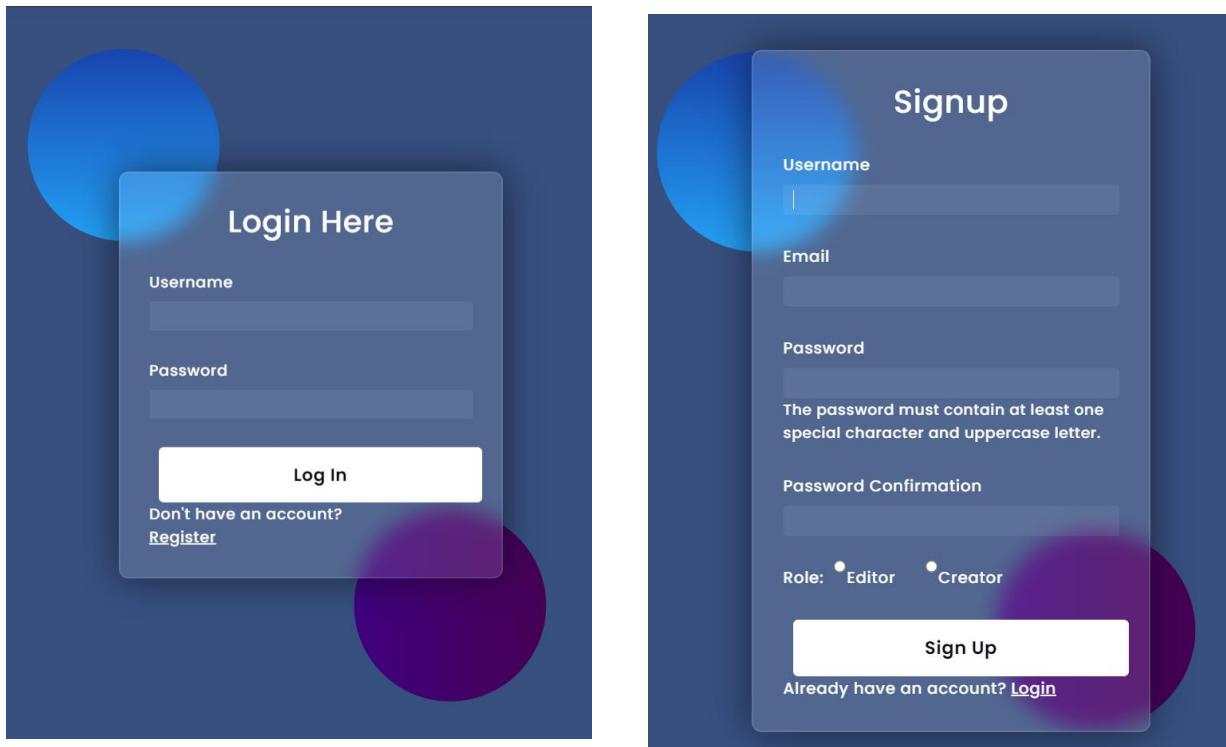


ABOUT THE INITIATIVE

Fig 4.1 Landing Page for Datafortress

Fig. 4.1 shows the landing page for the web application. It provides access to further usage of the the application by providing redirect links for logging in/ signing up or further exploring the website.

4.3.2 Login/Signup pages



Login Here

Username

Password

Log In

Don't have an account? [Register](#)

Signup

Username

Email

Password

The password must contain at least one special character and uppercase letter.

Password Confirmation

Role: ☐ Editor ☐ Creator

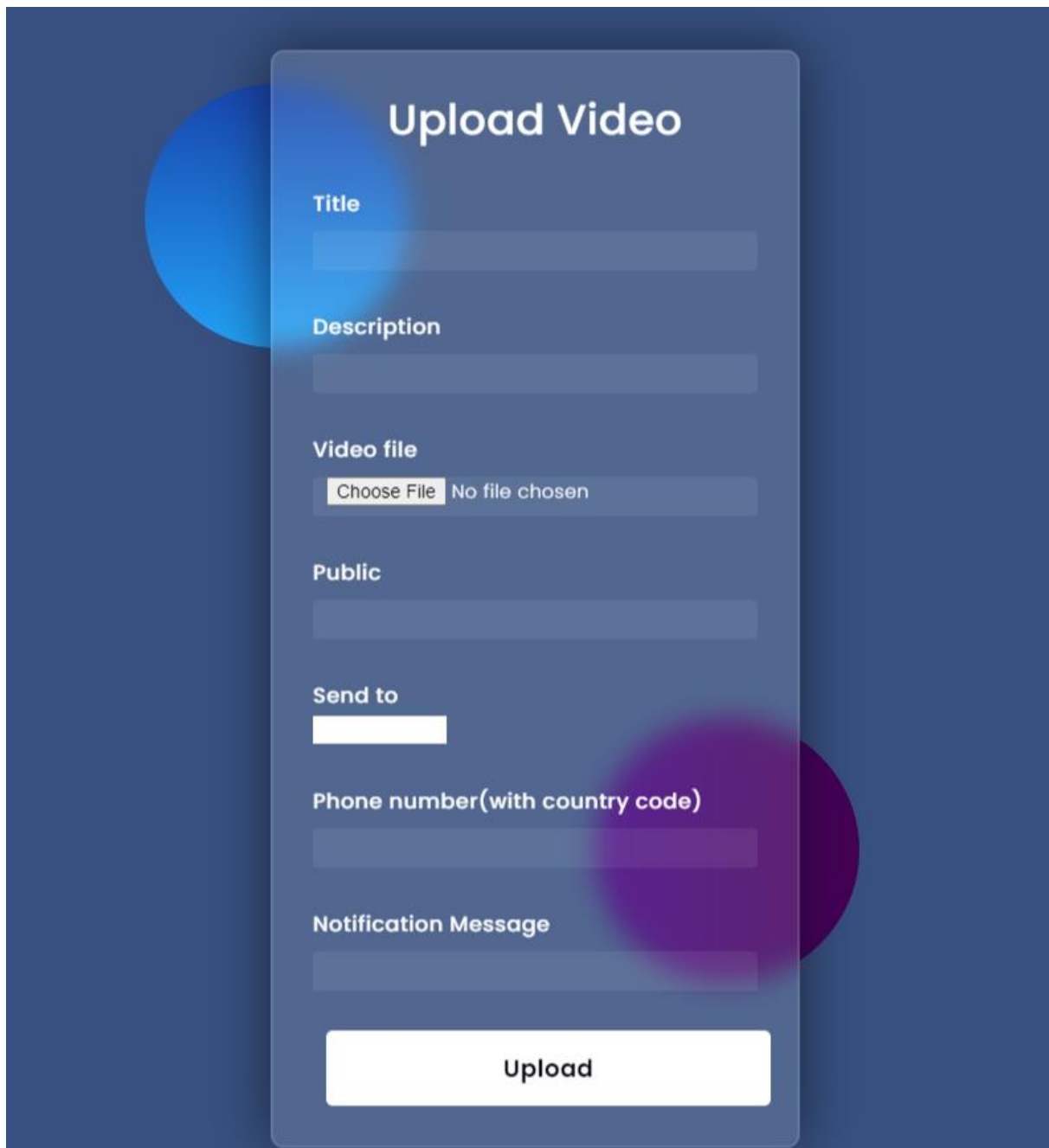
Sign Up

Already have an account? [Login](#)

4.2 Login and signup page for the application

Fig. 4.2 shows the login and signup page for the user to perform the designated tasks as per the requirement. The pages allow the user to further interact with the web application.

4.3.3 Video Upload Page



The screenshot shows a 'Video Upload' form on a dark blue background. The form is a light blue rounded rectangle with the title 'Upload Video' at the top. It contains several input fields: 'Title', 'Description', 'Video file' (with a 'Choose File' button and 'No file chosen' text), 'Public' (a dropdown menu), 'Send to' (a text input), 'Phone number(with country code)' (a text input), and 'Notification Message' (a text input). At the bottom of the form is a large white 'Upload' button.

Fig 4.3 Video Upload page

Fig. 4.3 shows the upload video page that allows the user with editor permissions to upload the video and send a notification to the creator by filling out the video form and the notification form.

4.3.4 Video List

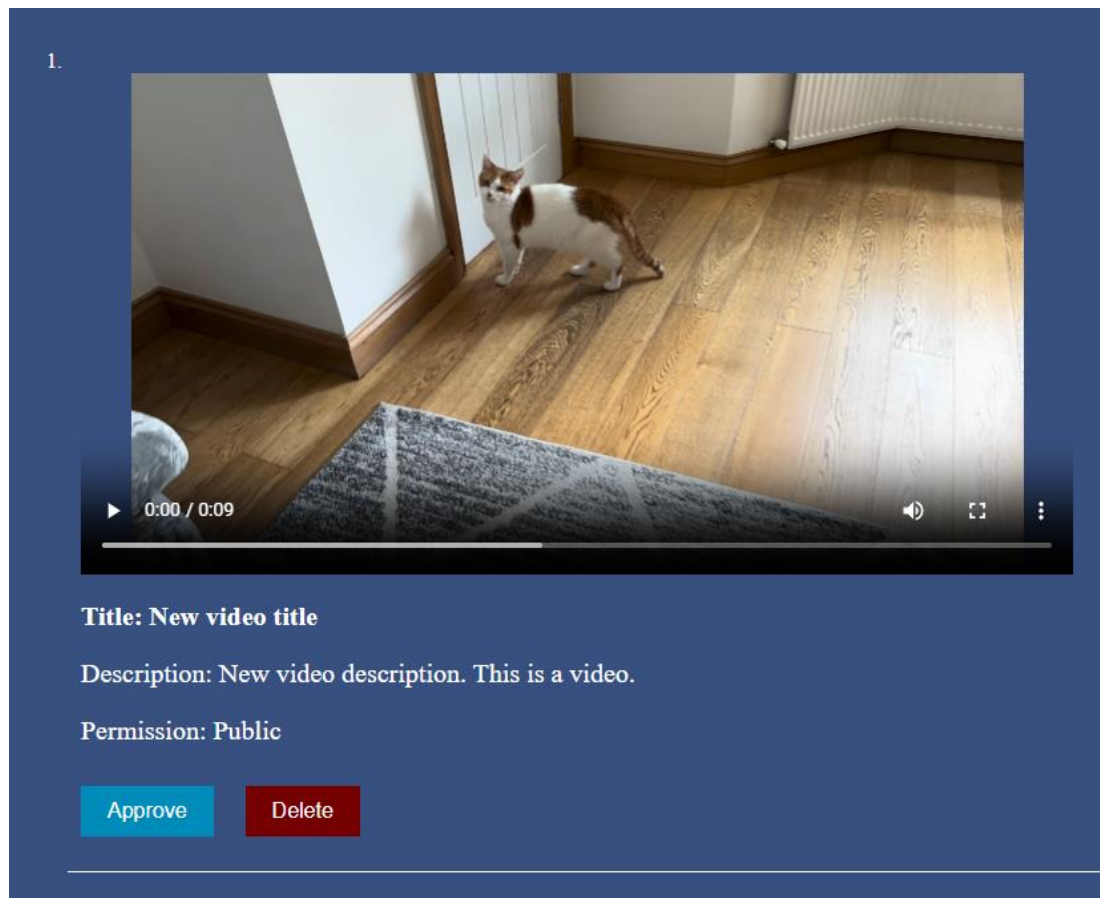


Fig. 4.4 Video list page

Fig. 4.4 shows the video list page that includes the videos uploaded by the editor for the specific user. The creator can then approve/deny the video to be further uploaded or used and the page contains the details about the video.

4.3.5 Notification List

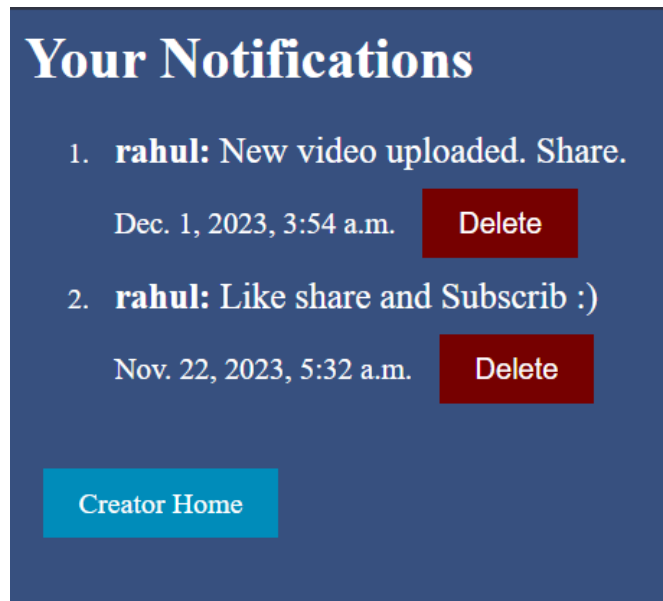


Fig. 4.5 Notifications page

Fig. 4.5 shows a notification list for the creator that displays the notifications received, the time and date of the notification, the user it was sent by and the permission to delete the notification.

Chapter-5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

In conclusion, DataFortress marks an important jump forward in changing how we approach collaborative content development. This Python-based workspace-sharing web application has solved the ongoing problems that editors and content creators face, resulting in a centralized platform that improves data security and streamlines communication. With its permission-seeking procedures and notification system, the methodical content approval process empowers editors while guaranteeing content producers maintain control over their creations.

What truly sets DataFortress apart is its solid commitment to data security in the digital realm. An intrusion detection system represents a strong defense against cyberattacks. In a time when data is frequently at the front of innovation, DataFortress is a pillar of stability and credibility.

As we look towards the future, DataFortress has the potential to make a lasting impact on society by changing how collaborative content creation takes place. Its user-friendly interface and emphasis on security contribute to a more efficient and transparent environment, promoting creativity and collaboration. DataFortress represents more than just a tool; it promises a secure, collaborative, and innovative future for content creators and editors alike.

5.2 Future Scope

- Integration of advanced AI capabilities [20] for automated content approval and organization.
- Collaboration with educational institutions to tailor the platform for academic use, incorporating virtual classrooms and collaborative project spaces.
- Continuous updates and enhancements to the security infrastructure to stay ahead of evolving cyber threats.

REFERENCES

- [1] Tedersoo, L., Küngas, R., Oras, E., Köster, K., Eenmaa, H., Leijen, Ä., ... & Sepp, T. (2021). Data sharing practices and data availability upon request differ across scientific disciplines. *Scientific data*, 8(1), 192.
- [2] Shyam, A., & Mukesh, N. (2020). A Django based educational resource sharing website: Shreic. *Journal of scientific research*, 64(1), 138-152.
- [3] Gore, H., Singh, R. K., Singh, A., Singh, A. P., Shabaz, M., Singh, B. K., & Jagota, V. (2021). Django: Web development simple & fast. *Annals of the Romanian Society for Cell Biology*, 25(6), 4576-4585.
- [4] Reddy, P. M., Manjula, S. H., & Venugopal, K. R. (2017). Secure data sharing in cloud computing: a comprehensive review. *International Journal of Computer (IJC)*, 25(1), 80-115.
- [5] Sun, Y. (2022). Research on the method of digital media content creation based on the internet of things. *Computational Intelligence and Neuroscience*, 2022.
- [6] Ghimire, D. (2020). Comparative study on Python web frameworks: Flask and Django.
- [7] Ghelani, D. (2022). Cyber security, cyber threats, implications and future perspectives: A Review. *Authorea Preprints*.
- [8] Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1), 1-22.
- [9] Schwarz, J. A. (2013). *Online file sharing: Innovations in media consumption*. Routledge.
- [10] Code, V. S. (2019). Visual studio code. *línea*. Available: <https://code.visualstudio.com>.
- [11] Duisebekova, K., Khabirov, R., & Zholzhan, A. (2021). Django as Secure Web-Framework in Practice. *Вестник Казахской академии транспорта и коммуникаций им. М. Тынышпаева*, (1), 275-281.

- [12] Nielson, S. J., & Monson, C. K. Practical Cryptography in Python.
- [13] Chen, S., Ahmmed, S., Lal, K., & Deming, C. (2020). Django Web Development Framework: Powering the Modern Web. *American Journal of Trade and Policy*, 7(3), 99-106.
- [14] Fitzgerald, B., & Stol, K. J. (2014, June). Continuous software engineering and beyond: trends and challenges. In *Proceedings of the 1st International Workshop on rapid continuous software engineering* (pp. 1-9).
- [15] Usmonov, M. T. O. G. L. (2021). Autentification, authorization and administration. *Science and Education*, 2(7), 233-242.
- [16] Holovaty, A., Kaplan-Moss, J., Holovaty, A., & Kaplan-Moss, J. (2008). The django template system. *The Definitive Guide to Django: Web Development Done Right*, 31-58.
- [17] Hakim, A. R., Rinaldi, J., & Setiadji, M. Y. B. (2020, June). Design and Implementation of NIDS Notification System Using WhatsApp and Telegram. In *2020 8th International Conference on Information and Communication Technology (ICoICT)* (pp. 1-4). IEEE.
- [18] Petrie, H., & Kheir, O. (2007, April). The relationship between accessibility and usability of websites. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 397-406).
- [19] Heimbach, I., & Hinz, O. (2018). The impact of sharing mechanism design on content sharing in online social networks. *Information Systems Research*, 29(3), 592-611.
- [20] Stocco, A. (2019, April). How artificial intelligence can improve web development and testing. In *Companion Proceedings of the 3rd International Conference on the Art, Science, and Engineering of Programming* (pp. 1-4).