PROJECT REPORT:

CREATIVE APP

IN

DJANGO – CREATIVE HUB

BY

MOHAMMED ALEEMUDDIN

MST01-0013

Abstract

The project "CreativeHub" aims to establish a dynamic platform that fosters creativity, collaboration, and innovation across diverse artistic domains. CreativeHub seeks to connect artists, designers, writers, and creators of all kinds in a vibrant ecosystem that encourages idea exchange, skill development, and the realization of creative projects.

Table of Contents:

1. Introduction:

The CreativeHub project is a platform for various creative tools website developed using Django framework. It incorporates various sections such as Home page, Register Page, Login Page, Art Page, Write Page, Music Page, Puzzle page, Profile Page, and About Us Page to provide users with a comprehensive creative tools related information.

2. Project Overview:

The project integrates Django framework along with Bootstrap for frontend styling and responsiveness. It features an enhanced admin interface using Django crispy forms for easy customization.

Upon signing up, you'll receive a warm confirmation email, signaling the beginning of your journey with CreativeHub. As you log in, you'll discover a wealth of possibilities at your fingertips, with access to four distinct pages dedicated to art, notes, music, and puzzles.

On the art page, unleash your inner artist as you paint, draw, and create to your heart's content. With powerful tools and a canvas limited only by your imagination, you'll bring your visions to life and transform blank spaces into vibrant masterpieces. And with the ability to save your creations as images, your artistic legacy will endure long after the strokes of your brush fade away.

Turn to the notes page to capture your thoughts, ideas, and stories with ease. Whether you're jotting down a poem, drafting a short story, or simply making a to-do list, our intuitive interface makes it simple to organize your thoughts and unleash your creativity in written form.

For those with a musical inclination, the music page offers a symphony of possibilities. Write, compose, and listen to music with our user-friendly tools, and let the melodies flow freely as you explore the endless possibilities of sound.

And when you're in need of a mental challenge, turn to the puzzle page for a dose of stimulating fun, test your wits and sharpen your mind with an array of engaging puzzles that are sure to keep you entertained for hours on end.

At CreativeHub, the possibilities are as limitless as your imagination. Join us today and unlock a world of creativity, connection, and endless inspiration. Welcome to your new creative home. Welcome to CreativeHub.

3. Technologies Used:

- Django 5.0.1
- Bootstrap 4
- Django Crispy forms
- H.T.M.L
- C.S.S
- JavaScript

Django 5.0.1: Django is a high-level Python web framework that facilitates rapid development of secure and maintainable websites and web applications. It follows the model-view-template (MVT) architectural pattern.

Bootstrap 4: Bootstrap is a popular front-end framework for building responsive and mobile-first web projects. It provides pre-designed templates and components using HTML, CSS, and JavaScript for creating user interfaces.

Django Crispy Forms: Django Crispy Forms is a Django application that helps developers create elegant, crispy, and DRY (Don't Repeat Yourself) forms effortlessly by providing a clean, uniform layout and styling through templates and CSS classes. It simplifies the process of designing and rendering forms in Django projects, enhancing both aesthetics and user experience.

HTML: HTML (Hypertext Markup Language) is the standard markup language used to create the structure and content of web pages. It defines the elements and tags used to display text, images, links, and other multimedia content on the web.

CSS: CSS (Cascading Style Sheets) is a stylesheet language used to style the presentation and layout of HTML documents. It defines the styles, such as colors, fonts, margins, and padding, to enhance the visual appearance of web pages.

JavaScript: JavaScript is a high-level programming language used to add interactivity, dynamic behavior, and functionality to web pages. It allows developers to manipulate the HTML and CSS of a webpage, handle user interactions, and perform asynchronous operations.

4. System Architecture:

1. User Interface Layer (Presentation Layer):
   - This layer represents the user interface components of your Django application.
   - It includes HTML templates, CSS stylesheets, and client-side JavaScript files for rendering the user interface.
   - Django templates will be used to generate dynamic HTML content based on data from the backend.

2. Web Server:
   - Django will be deployed on a web server like Apache or Nginx using WSGI (Web Server Gateway Interface).
   - The web server will handle incoming HTTP requests and route them to the appropriate Django views.

3. Django Application Layer:
   - This layer consists of Django apps defined in the `INSTALLED_APPS` setting.
   - Key components include:
     - User app (user.apps.UserConfig): This app manages user authentication, registration, and user-related functionalities.
     - Crispy Forms and Crispy Bootstrap4: These apps integrate with Django forms to provide a clean, stylish layout for forms using Bootstrap 4.
     - Admin app (django.contrib.admin): This app provides an administration interface for managing site content and users.
   - Each Django app contains models, views, forms, and templates specific to its functionality.

4. Middleware Layer:
   - Middleware components defined in the `MIDDLEWARE` setting intercept and process HTTP requests and responses.
   - Key middleware components include:
     - SecurityMiddleware: Enhances security by setting various security-related HTTP headers.
     - SessionMiddleware: Manages user sessions.
     - CommonMiddleware: Performs common operations like URL rewriting and content-type detection.

- CsrfViewMiddleware: Protects against Cross-Site Request Forgery (CSRF) attacks.
- AuthenticationMiddleware: Handles user authentication.
- MessageMiddleware: Enables passing messages between views and templates for displaying notifications.
- XFrameOptionsMiddleware: Protects against Clickjacking attacks.

5. Database Layer:
- Django supports various relational databases like PostgreSQL, MySQL, SQLite, etc.
- Models defined in Django apps interact with the database to store and retrieve data.
- Django's ORM (Object-Relational Mapping) simplifies database operations by allowing developers to work with database objects using Python classes and methods.

6. Static and Media Files:
- Static files (e.g., CSS, JavaScript, images) are served by Django's static file serving mechanism.
- Media files (e.g., user-uploaded files) are stored and served using Django's media file handling capabilities.

7. Security Layer:
- Django provides built-in security features like authentication, authorization, CSRF protection, and secure session management.
- Additional security measures can be implemented at the web server level (e.g., SSL/TLS encryption, firewall configurations).

This architecture provides a high-level overview of how your Django project is structured and how various components interact to deliver the desired functionality to users.

5. Implementation Details:

Home Page:

The home page serves as the initial landing page for your website or application.

It typically provides an overview of the site's content, navigation options, and may feature highlights or announcements.

The home page aims to engage visitors and direct them to other sections or features of the site.

Art Page:

The art page is a section of your website or application dedicated to artistic expression.

It provides users with tools and features for creating visual art, such as painting and drawing.

Users can unleash their creativity, explore different artistic techniques, and save their creations for future reference or sharing.

Write Page:

The write page offers a platform for users to engage in written expression.

It may include features such as text editors or note-taking tools for writing stories, poems, essays, or any other form of written content.

Users can draft, edit, and save their writings on the write page, facilitating creative writing and documentation.

Music Page:

The music page caters to users interested in music composition, listening, or exploration.

It provides tools for composing music, including features for writing musical scores or creating digital compositions.

Users can also listen to pre-existing music, explore different genres, and discover new artists or tracks.

Puzzle Page:

The puzzle page offers a collection of interactive puzzles or brain teasers for users to solve.

It may include various types of puzzles such as crosswords, Sudoku, word searches, or logic puzzles.

Users can challenge themselves, exercise their problem-solving skills, and enjoy recreational entertainment on the puzzle page.

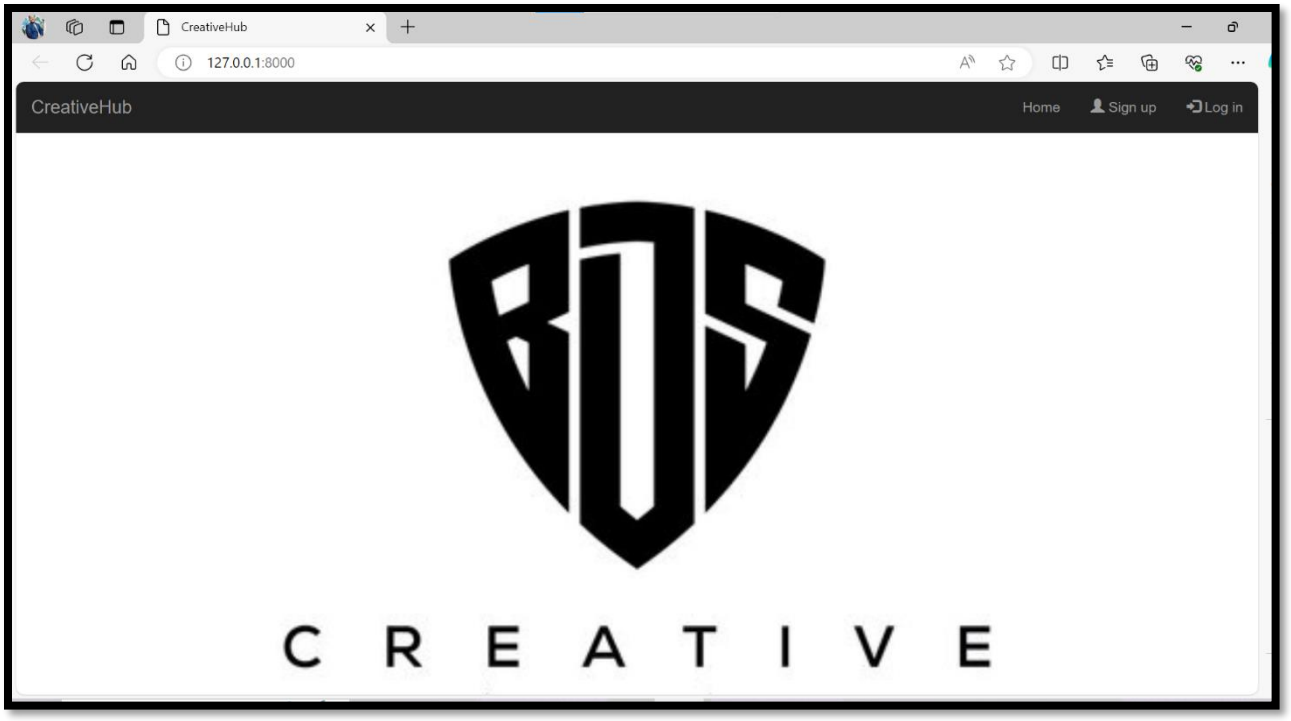

Fig 1: Login/Register Page

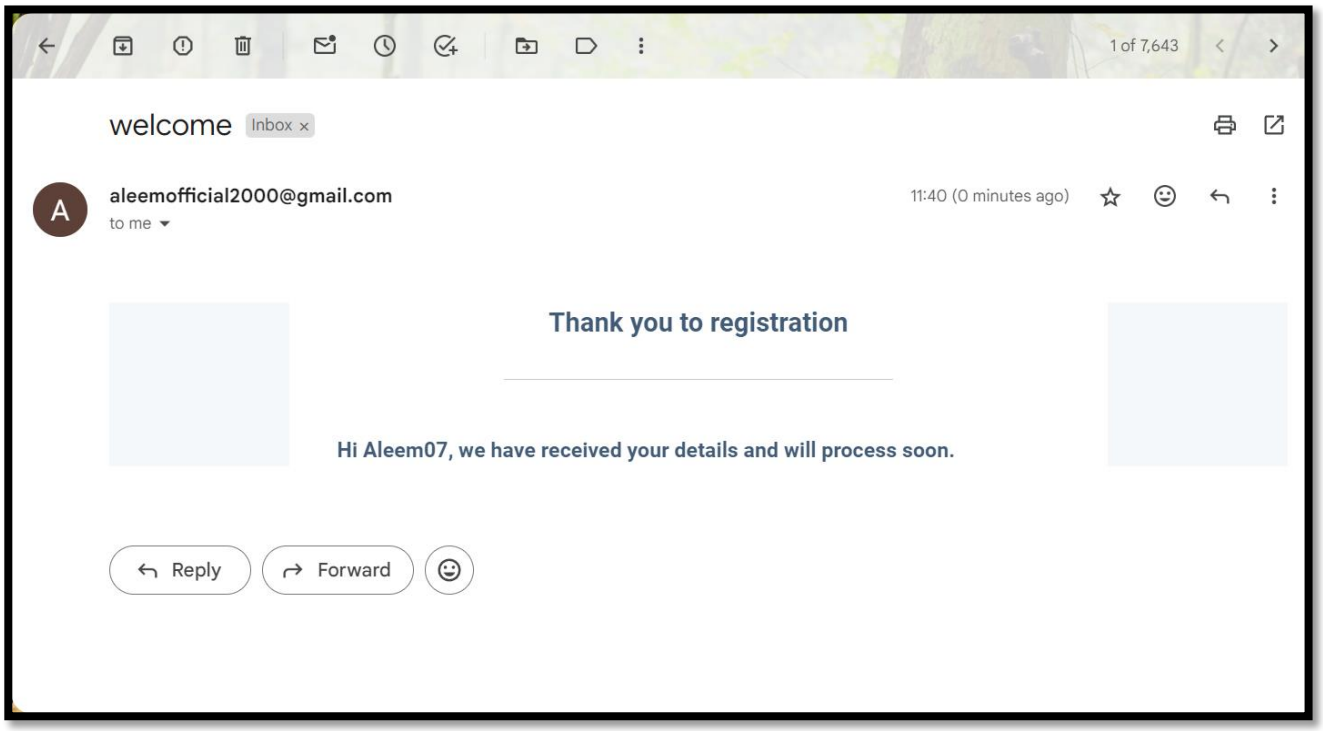

Fig2. Registration Confirmation Email Screenshot

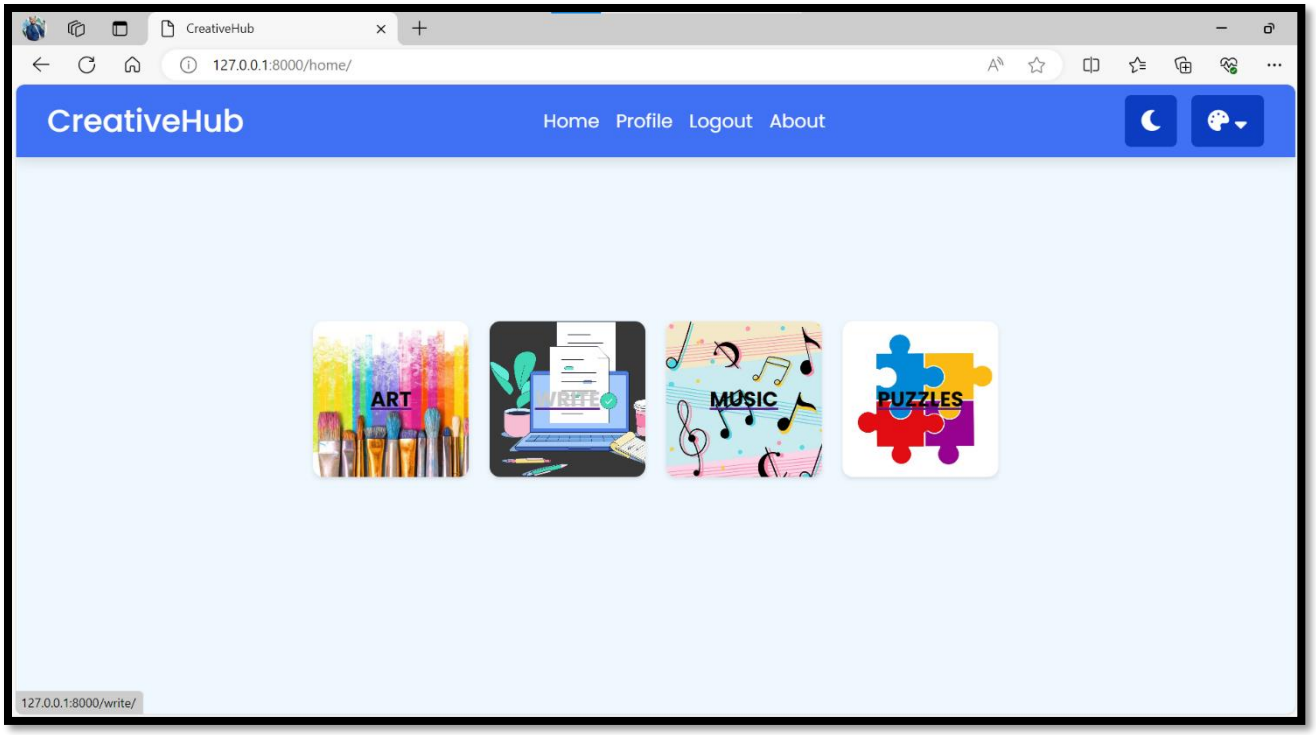

Fig3.Home Page

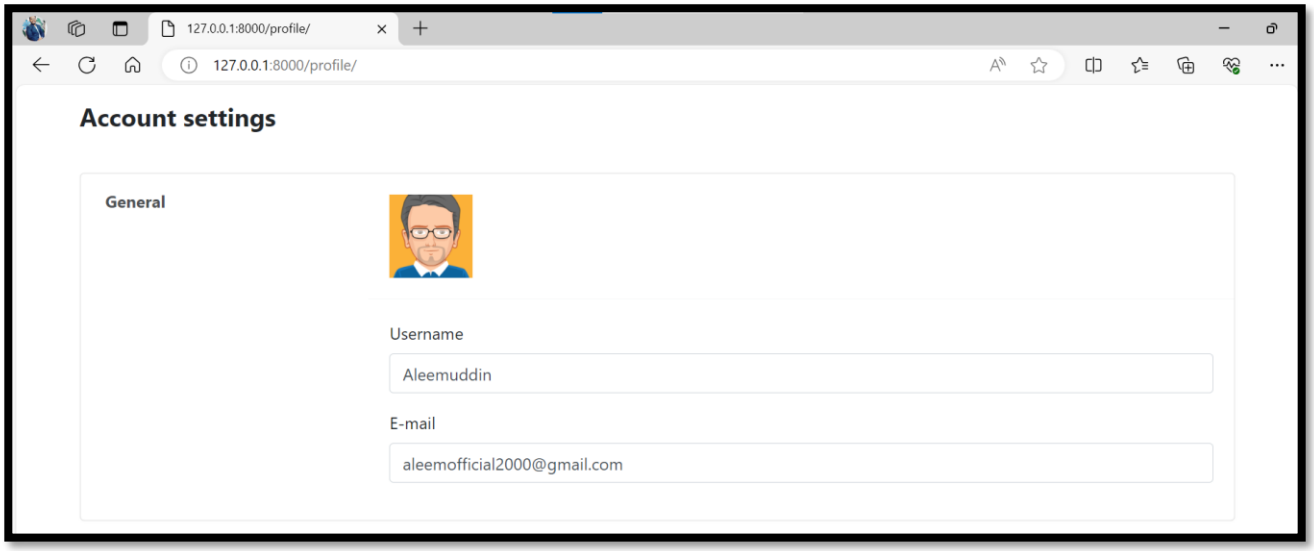

Fig4.Profile Page

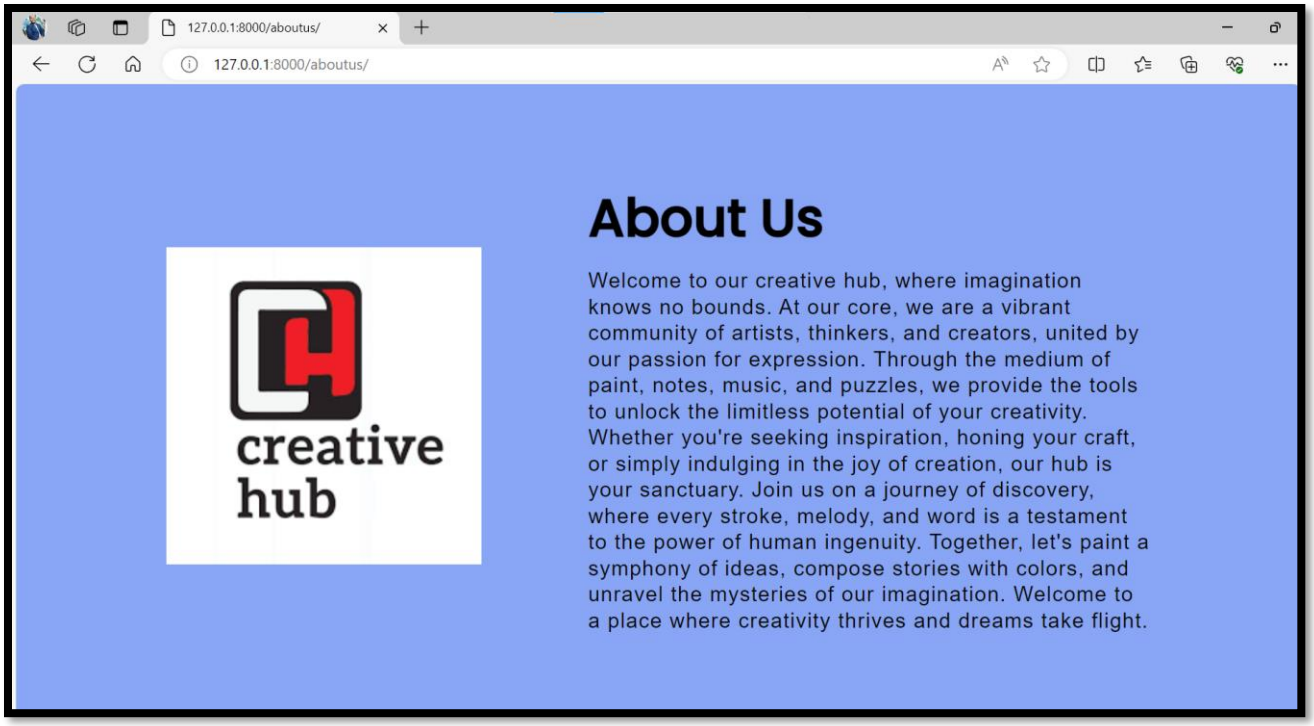

Fig5.About us page

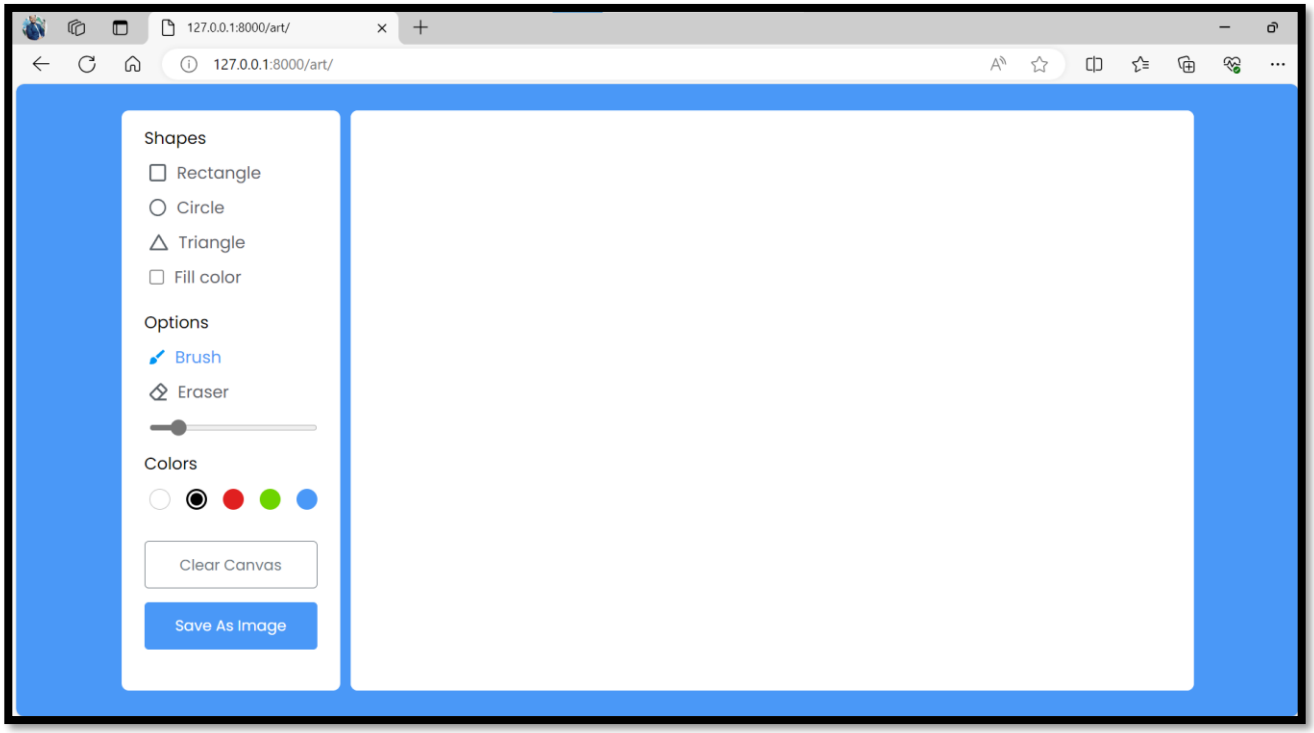

Fig6. Art Page

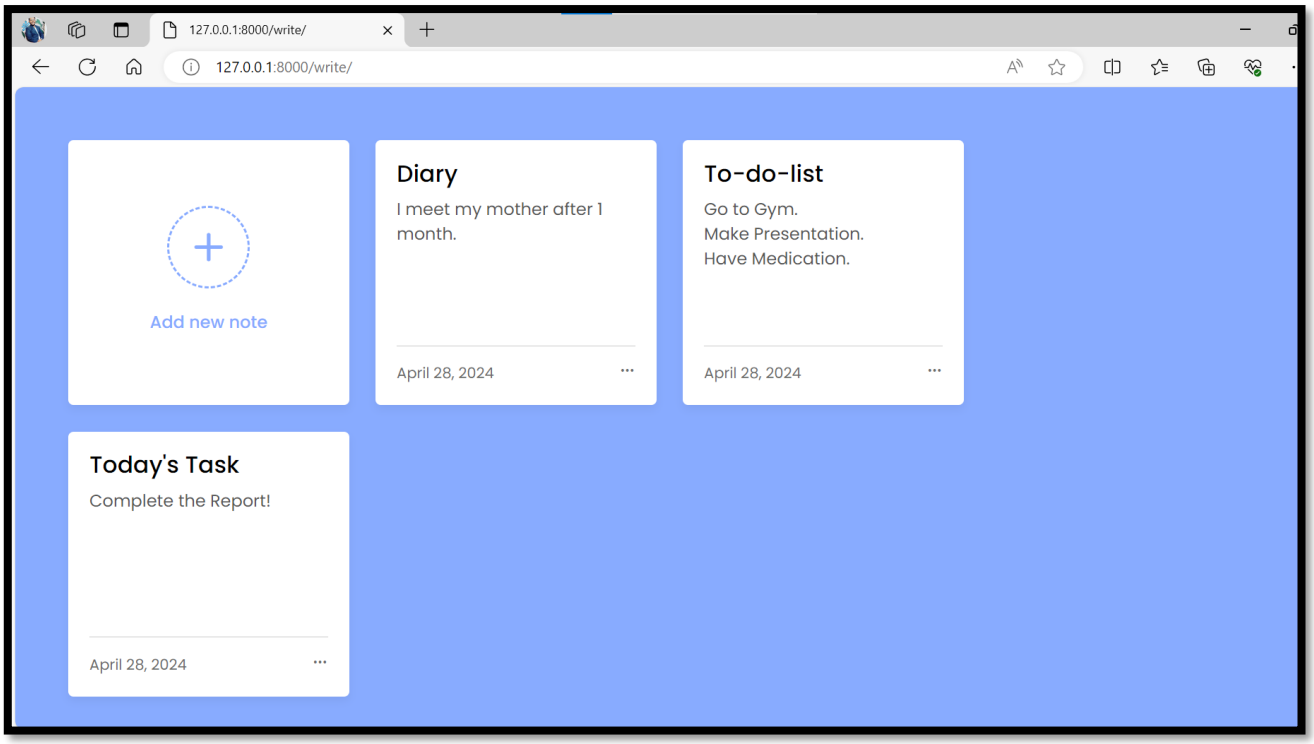

Fig7. Write Page

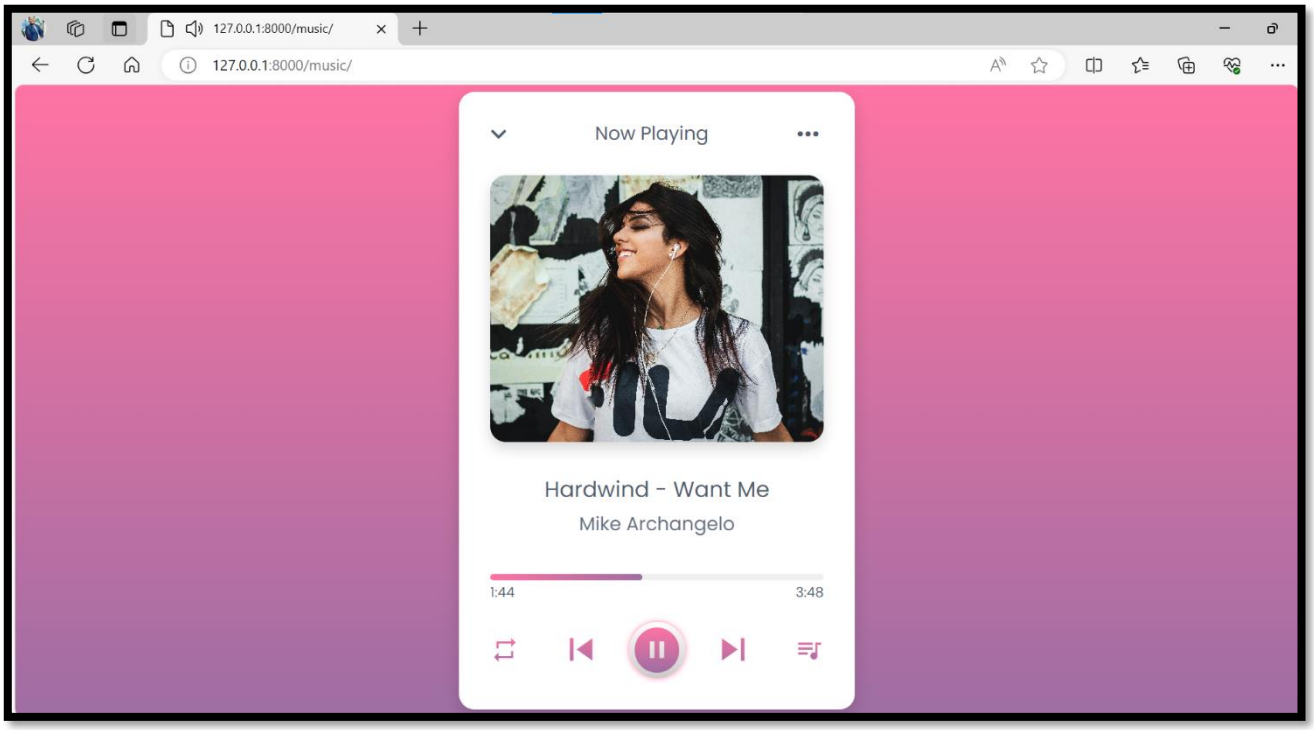

Fig 8. Music Page

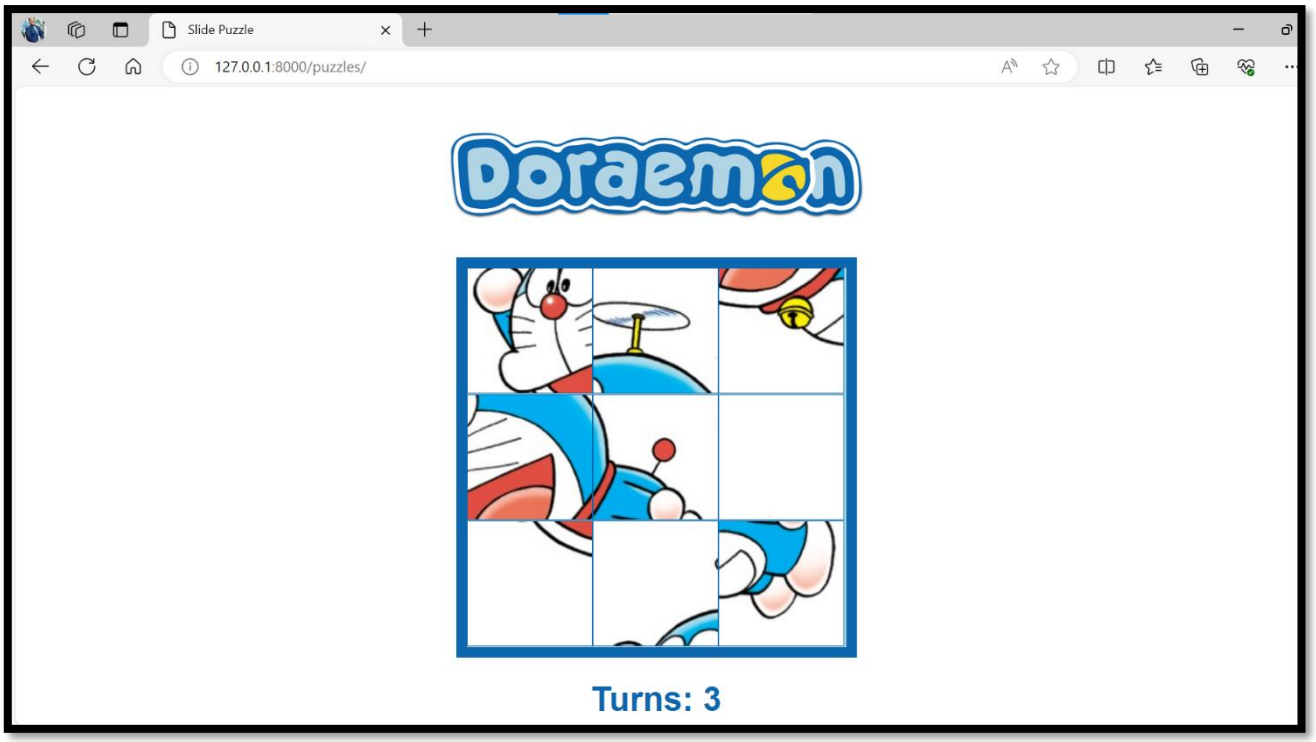

Fig 9. Puzzle Page

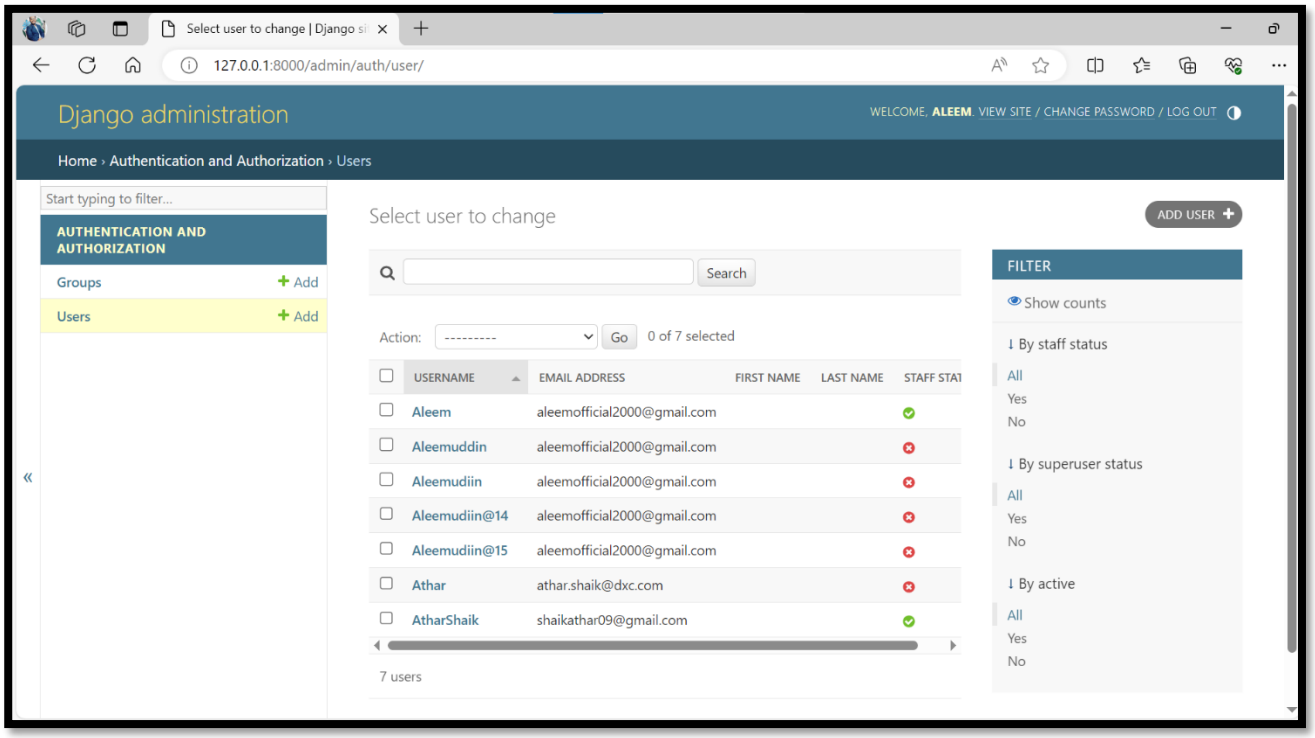

Fig 10. Admin Page

Login/Register Page:

Functionality:

The login/register page enables users to sign in or create accounts securely. It validates input, handles errors, and upon successful registration, sends a confirmation email to the user's provided address. The login/register page ensures secure access and account creation. Upon successful login, users are redirected to the home page for further exploration.

Technologies used:

Frontend: HTML, CSS, Bootstrap4

Backend: Django

Middleware: 'django.middleware.security.SecurityMiddleware'

'django.contrib.auth.middleware.AuthenticationMiddleware'

Views.py

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth import authenticate, login, update_session_auth_hash
from django.contrib.auth.decorators import login_required
from django.contrib.auth.forms import AuthenticationForm

from .forms import UserRegisterForm
from django.core.mail import send_mail
from django.core.mail import EmailMultiAlternatives
from django.template.loader import get_template
from django.template import Context


#################### index#########################################
def index(request):
    return render(request, 'user/index.html', {'title':'CreativeHub'})

########### register here #################################
def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
```

```
            username = form.cleaned_data.get('username')
            email = form.cleaned_data.get('email')
            ######################### mail system
####################################
            htmly = get_template('user/Email.html')
            d = { 'username': username }
            subject, from_email, to = 'welcome', 'your_email@gmail.com', email
            html_content = htmly.render(d)
            msg = EmailMultiAlternatives(subject, html_content, from_email,
[to])
            msg.attach_alternative(html_content, "text/html")
            msg.send()
            ########################################################################
            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, 'user/register.html', {'form': form,
'title':'register here'})

################ login
forms####################################################
def Login(request):
    next_page = request.POST.get('next', request.GET.get('next', 'home'))
    if request.method == 'POST':
        form = AuthenticationForm(request, request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(username=username, password=password)
            if user is not None:
                login(request, user)
                messages.success(request, f'Thankyou {username}!')
                return redirect(next_page)
            else:
                messages.error(request, 'Invalid username or password.')
    else:
        form = AuthenticationForm()
    return render(request, 'user/login.html', {'form': form, 'title': 'Log
in', 'next': next_page})
```

Home Page:

Functionality:

The homepage acts as the first page users encounter when they visit your website or app. Its primary function is to offer a snapshot of what the site offers, including navigation options and possibly showcasing important content or announcements. Essentially, the homepage's purpose is to captivate visitors and guide them to explore other parts or functionalities of the site.

Technologies used:

Frontend: HTML, CSS, Bootstrap4, Javascript

Backend: Django

Middleware: 'django.middleware.csrf.CsrfViewMiddleware'

Views.py

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

Urls.py

```
from django.urls import path, include
from django.conf import settings
from . import views
from .views import home, profile, aboutus
from django.conf.urls.static import static

urlpatterns = [
        path('', views.index, name ='index'),
        path('home/', home, name='home'),
        path('profile/', profile, name='profile'),
        path('aboutus/', aboutus, name='aboutus'),
        # Add new URLs for Art, Write, Music, and Puzzles pages
        path('art/', views.art_page, name='art_page'),
        path('write/', views.write_page, name='write_page'),
        path('music/', views.music_page, name='music_page'),
        path('puzzles/', views.puzzles_page, name='puzzles_page'),
]
```

Art Page:

Functionality:

The art page offers a platform for artistic expression within your website or app. It equips users with tools and features for creating visual art, including painting and drawing functionalities. Users can unleash their creativity, experiment with various artistic techniques, and save their artworks for future reference or sharing.

Technologies used:

Frontend: HTML, CSS, Javascript

Backend: Django

Middleware: 'django.contrib.sessions.middleware.SessionMiddleware'

Views.py

```
def art_page(request):
    # Logic to fetch data or perform any other operations
    return render(request, 'user/art_page.html')
```

Calling Script.js

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="{% static 'css/art_work.css' %}">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="{% static 'js/script.js' %}" defer></script>
  </head>
```

Write Page:

Functionality:

The write page resembles a note-taking platform where users can jot down their thoughts or ideas. It provides a simple text editor for users to write notes, which are automatically saved along with the timestamp of creation or modification. Users can organize their notes into categories or tags for easy retrieval and management. Additionally, they can quickly search through their notes based on keywords or dates.

Technologies used:

Frontend: HTML, CSS, Javascript

Backend: Django

Middleware: 'django.contrib.sessions.middleware.SessionMiddleware'

Views.py

```
def write_page(request):
    # Logic to fetch data or perform any other operations
    return render(request, 'user/write_page.html')
```

Calling Write.js

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="{% static "css/write_work.css" %}">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="{% static 'js/writing.js' %}" defer></script>
    <link rel="stylesheet"
href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">
  </head>
```

Music Page:

Functionality:

The music page grants users access to the music present in the database, enabling them to listen to tracks at their convenience. Users can loop their preferred songs for continuous playback and easily navigate through the available music library. Additionally, they have the option to skip forward to the next track as desired, enhancing their listening experience.

Technologies used:

Frontend: HTML, CSS, Javascript

Backend: Django

Middleware: 'django.contrib.sessions.middleware.SessionMiddleware'

Views.py

```
def music_page(request):
    # Logic to fetch data or perform any other operations
    return render(request, 'user/music_page.html')
```

Calling Music.js and music-list.js:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="{% static 'css/music_work.css' %}">
  <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
  <script src="{% static 'js/music-list.js' %}" defer></script>
  <script src="{% static 'js/music.js' %}" defer></script>
  <script>
    const staticUrl = "{% static '' %}";
</script>
</head>
```

Puzzle Page:

Functionality:

The Doraemon puzzle page presents users with an interactive jigsaw puzzle featuring images from the popular series.
Users can drag and drop puzzle pieces to solve the Doraemon-themed puzzle.
Once completed, users can view the entire image and receive a congratulatory message.
Additionally, users can access hints or guidance if they encounter difficulty in solving the puzzle, enhancing the overall user experience.

Technologies used:

Frontend: HTML, CSS, Javascript

Backend: Django

Middleware: 'django.contrib.sessions.middleware.SessionMiddleware'

Views.py

```
def puzzles_page(request):
    # Logic to fetch data or perform any other operations
    return render(request, 'user/puzzles_page.html')
```

Calling puzzle.js:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Slide Puzzle</title>
        <link rel="stylesheet" href="{% static 'css/puzzle.css' %}">
        <script src="{% static 'js/puzzle.js' %}" defer></script>
        <script>
            const staticUrl = "{% static '' %}";
        </script>
    </head>
```

Profile Page:

Functionality:

The profile page allows users to view and edit their username and email ID. It provides basic information management while prioritizing user privacy. Additionally, users can log out from the profile page.

Technologies used:

Frontend: HTML, CSS, Bootstrap4

Backend: Django

Middleware: 'django.middleware.common.CommonMiddleware'

Views.py

```
def profile(request):
    user = request.user
    return render(request, 'user/profile.html', {'user': user})
```

Calling Bootstrap4:

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="{% static "css/profile.css" %}">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
```

Aboutus Page:

Functionality:

The "About Us" page specifically focuses on detailing the tools and uses of CreativeHub. It provides an overview of the platform's features, functionalities, and how users can utilize them for their creative endeavors. Additionally, it may highlight unique aspects or benefits of CreativeHub that set it apart from other similar platforms. Users can gain insights into the capabilities and potential applications of CreativeHub through this page.

Technologies used:

Frontend: HTML, CSS, Bootstrap4

Backend: Django

Middleware: 'django.middleware.common.CommonMiddleware'

Views.py

```
def aboutus(request):
    # Logic to handle settings page
    return render(request, 'user/aboutus.html')
```

Calling aboutus.css:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="{% static "css/aboutus.css" %}">
</head>
```

6.Challenges Faced:

1. Music Page - Uploading Music Data via JavaScript:

- **Challenge**: Difficulty in uploading music data dynamically using JavaScript.

- **Explanation**: Implementing a feature to allow users to upload music files and store them in the database posed challenges due to the complexities of handling file uploads via JavaScript.

- **Resolution**: Overcame the challenge by researching and implementing appropriate JavaScript libraries or frameworks for handling file uploads, ensuring seamless integration with the backend database.

2. Puzzle Page - Managing Puzzle Images:

- **Challenge**: Managing puzzle images and integrating them into the puzzle page interface.

- **Explanation**: Obtaining and displaying puzzle images dynamically on the page while ensuring optimal performance and user experience proved to be challenging.

- **Resolution**: Addressed the challenge by optimizing image loading techniques, such as lazy loading or preloading, and implementing efficient image management strategies to enhance page responsiveness and usability.

7.Conclusion:

In conclusion, the CreativeHub project represents a significant endeavor aimed at providing users with a platform to explore and express their creativity. Through the development process, several key functionalities were successfully implemented, including the music page and puzzle page, despite facing various challenges along the way. Despite these challenges, the project showcases the dedication and perseverance of the development team in creating a robust and user-friendly platform. Moving forward, continual improvements and enhancements will be essential to further refine the user experience and expand the platform's capabilities. Overall, CreativeHub stands as a testament to the power of innovation and collaboration in fostering a community of creative individuals.

8.References:

1. Crispy Forms Documentation: [https://django-crispy-forms.readthedocs.io/en/latest/]
2. Crispy Bootstrap4 Documentation: [https://django-crispy-forms.readthedocs.io/en/latest/]
3. Django Admin Documentation: [https://docs.djangoproject.com/en/stable/ref/contrib/admin/]
4. Django Documentation: [https://docs.djangoproject.com/en/5.0/]