**ABSTRACT:**

The ArtFusion project is a dynamic and interactive platform designed to empower artists across various discipline such as poets, writers, painters, sketch artists, and drawing artists to create, share, and collaborate through engaging posts. The platform offers a personalized experience for users by enabling them to register, create individual profiles, and actively participate in the community. Artists can publish content, follow other creators, interact with posts through likes and comments, and collaborate through the Network page, which facilitates offline event coordination.

The platform not only promotes artistic expression and collaboration but also creates an inclusive space for non-artists to appreciate and engage with various forms of creativity. By encouraging users to stay updated on artists' careers and introducing challenges, ArtFusion aims to motivate and inspire creativity within its community, making it a vibrant hub for artistic exploration and interaction.The modular and scalable design of ArtFusion enables seamless integration of new features, such as multimedia content support and SEO tools, enhancing future development possibilities.

# INTRODUCTION:

**ArtFusion** is an innovative online platform dedicated to empowering artists and fostering creativity across various artistic disciplines. In a world where self-expression is increasingly valuable, ArtFusion serves as a vibrant community hub where artists can showcase their work, connect with others, and access essential resources.

The platform embraces a diverse range of art forms, including painting, poetry, stories, crafts and drawing. This inclusivity ensures that every artist, regardless of their medium or style, has a space to share their creations and find inspiration in the works of others.

ArtFusion was born from the belief that art is a universal language that transcends boundaries, cultures, and experiences. By providing a dedicated space for artistic expression, the platform aims to break down barriers and foster a sense of belonging among artists of all backgrounds. Through collaboration and community engagement, ArtFusion seeks to promote creativity, encourage learning, and inspire artists to reach their fullest potential.

Built on the robust MERN STACK(MongoDB, Express.js, React, Node.js), ArtFusion combines cutting-edge technology with user-friendly design to create a seamless experience for its users. The platform is designed to be intuitive and accessible, ensuring that both seasoned artists and newcomers can navigate with ease.

With features that include art showcases, user profiles, networking opportunities, and Art work related resources, ArtFusion is more than just a website; it is a thriving ecosystem that nurtures creativity and connects artists worldwide. Whether you're looking to display your latest masterpiece, collaborate with fellow creators, or discover new artistic techniques, ArtFusion is the place to explore, create, and thrive in the world of art.

# OBJECTIVES AND SCOPE OF THE PROJECT:

**OBJECTIVES:**

## Empower Artists to Share and Collaborate:

* Provide a platform where artists across various disciplines like poets, writers, painters, and sketch artists can create and share their content through blog posts.
* Facilitate collaborations among artists through the Network page, encouraging offline event participation.

## Enable Learning and Discovery for New Artists:

* Offer a Resources page for budding artists to search specific topics and navigate seamlessly to external platforms like Instagram, YouTube, and Pinterest for inspiration.

## Engage Non-Artists with Artistic Content:

* Allow non-artists to explore posts through the Art Showcase section, providing easy access to topic-related posts without navigating the entire platform's features.

## Build a Vibrant and Inclusive Community:

* Create opportunities for users to follow artists, engage through likes, comments, and shares, and foster meaningful connections within the community.

## Ensure a Seamless User Experience:

* Develop a highly responsive, mobile-friendly interface using React.js, allowing smooth navigation across all sections.
* Provide personalized content recommendations through tag-based filtering and trending topics.

## Focus on Security and Scalability:

* Implement robust authentication systems, including OAuth for third-party logins and secure password encryption, ensuring data privacy.
* Use Node.js, Express.js, and MongoDBfor a scalable backend architecture that supports high traffic.

# SCOPE:

## User Management and Profiles:

* Users can register, create personalized profiles, publish content, and engage with other artist’s posts through likes and comments.

## Content Creation and Discovery:

* Artists can post and showcase their work, while users can explore content based on topics or trending tags.
* Non-artists have a focused space (the Art Showcase page) to explore artistic content without engaging in deeper community features.

## Collaborative Networks and Events:

* The Network page facilitates artist collaborations and helps users coordinate and participate in offline events.

## Resources Hub for Learning:

* The Resource*s* page enables users to search for relevant topics and navigate to other platforms (like Instagram, YouTube, and Pinterest) to enhance their learning and skills.

## Scalable and Secure Platform:

* Backend powered by Node.js and Express.jsensures fast and efficient data handling.
* MongoDBserves as the core database to store and retrieve posts and user data effectively.
* Reduxensures smooth state management across the entire platform.

## Future Expansion and Extensibility:

* Support for multimedia content (images, videos) in posts can be easily added.
* SEO optimization tools and social media sharing features can enhance the platform’s reach.
* The modular architecture ensures that new functionalities can be integrated with minimal disruption to the existing system.

# SOFTWARE USED:

**Technologies:**

## MongoDB:

* **Overview**: MongoDB is a document-oriented NoSQL database that allows for flexible data modeling. It stores data in JSON-like format (BSON), making it easy to work with varying data types.

## Features:

**Scalability**: Designed for horizontal scaling, allowing the database to grow easily as the user base increases.

**Rich Queries**: Supports a wide range of queries, including filtering and aggregation, making it suitable for complex data interactions.

**Indexing**: Provides indexing for improved performance in data retrieval.

## Express.js:

* **Overview**: Express.js is a minimal and flexible web application framework for Node.js. It provides a robust set of features for building web and mobile applications**.**

## Features:

**Middleware Support**: Allows the integration of middleware functions for handling requests and responses.

**Routing**: Facilitates the creation of dynamic routes, enabling efficient management of API endpoints.

**Error Handling**: Provides built-in mechanisms for error handling and logging, improving the debugging process.

## Node.js:

* **Overview**: Node.js is a JavaScript runtime built on Chrome's V8 engine, allowing developers to execute JavaScript on the server side.

## Features:

**Event-Driven Architecture**: Utilizes non-blocking I/O operations, making it efficient for handling multiple requests simultaneously.

**NPM (Node Package Manager)**: Comes with a vast repository of libraries and tools that can be easily integrated into projects.

**Cross-Platform Compatibility**: Enables deployment on various operating systems, ensuring flexibility in hosting.

## React:

* **Overview**: React is a JavaScript library for building user interfaces, developed by Facebook. It allows for the creation of reusable UI components.

## Features:

**Virtual DOM**: Improves performance by minimizing direct interactions with the real DOM, leading to faster rendering.

**Component-Based Architecture**: Encourages the reuse of components, enhancing code maintainability and organization.

**Declarative UI**: Allows developers to describe how the UI should look based on the application state, making it easier to manage dynamic content.

## React Router:

* **Overview**: React Router is a routing library for React that enables navigation between different components in a single-page application (SPA).

## Features:

**Dynamic Routing**: Facilitates dynamic routing, allowing for seamless transitions between views without reloading the page.

**Nested Routes**: Supports nested routing, enabling complex application structures with multiple views**.**

**URL Parameter Handling**: Provides mechanisms for passing parameters in URLs, improving user experience and interaction.

## CSS:

* **Overview**: CSS (Cascading Style Sheets) is used for styling web pages, providing a way to define the presentation of HTML elements.

## Features:

**Responsive Design**: Allows for the creation of responsive layouts that adapt to various screen sizes.

**Selectors and Properties**: Offers a wide range of selectors and properties for precise styling control.

**Animations and Transitions**: Supports animations and transitions to enhance user experience and interface interaction.

# Tools:

## Git and GitHub:

* **Overview**: Git is a distributed version control system, while GitHub is a platform for hosting Git repositories, enabling collaborative development.

## Features:

**Version Tracking**: Allows developers to track changes in the codebase, facilitating collaboration and rollback to previous versions if necessary.

**Branching and Merging**: Supports multiple branches for feature development, enabling parallel development without affecting the main codebase.

**Pull Requests**: Provides a mechanism for code review and collaboration among team members before merging changes.

## Visual Studio Code:

* **Overview**: Visual Studio Code (VS Code) is a lightweight, open-source code editor with robust features for development.

## Features:

**IntelliSense**: Offers intelligent code completion and suggestions, speeding up the development process.

**Extensions**: Supports a wide range of extensions for various programming languages and tools, enhancing functionality.

**Integrated Terminal**: Provides a built-in terminal for executing commands without leaving the editor.

## Google Chrome:

* **Overview**: Google Chrome is a web browser that offers powerful tools for debugging and testing web applications.

## Features:

**Dev Tools**: Provides a set of built-in developer tools for inspecting HTML, CSS, and JavaScript, aiding in debugging and performance optimization.

**Responsive Design Mode**: Allows developers to test how their applications look on different devices and screen sizes.

**Extensions**: Supports various extensions that can enhance productivity and improve development workflows.

# BACKEND:

The backend of ArtFusion leverages the MERN stack to provide a robust, scalable, and secure environment for managing data, handling user authentication, and executing server-side functions efficiently. Below is a detailed breakdown of the technologies and tools used and where they are applied within the project.

## Node.js:

* **Role:** Main server-side technology

• **Usage:** Node.js manages incoming HTTP requests, enabling features such as user registration, profile management, artwork uploads, and post interactions. It also supports real-time data processing for a responsive user experience and handles multiple requests efficiently.• **RESTful API:** The backend exposes various endpoints using REST APIs, ensuring smooth communication between the frontend and backend.**2. Express.js:**• **Role:** Web application framework• **Usage:** Express.js simplifies server-side development with routing, middleware integration, and error handling. It helps define APIs for user actions like uploading artwork, commenting on posts, retrieving profiles, and more.**3. MongoDB:• Role:** NoSQL database for data storage**•** **Usage:** MongoDB’s document-based structure stores diverse data types, including user details, text posts, images, and media files. This flexible schema ensures efficient retrieval of posts, comments, and profiles.**4. JWT (JSON Web Token):•** **Role:** Secure user authentication**•** **Usage:** JWT tokens are issued upon successful login and are stored on the client side (local storage) to maintain session persistence. These tokens verify the user’s identity for protected routes and API calls.**5. Local Storage:**

**• Role:** Client-side storage of JWT tokens**• Usage:** Local storage ensures the user stays logged in by storing the JWT token, which is usedtoauthenticate API requests without requiring repeated logins.**6. CORS (Cross-Origin Resource Sharing):• Role:** Security mechanism**• Usage:** CORS is configured to allow cross-origin requests, ensuring that the frontend (React) can communicate with the backend (Node/Express) even if they are hosted on different domains or ports.**7. dotenv:• Role:** Manage environment variables**• Usage:** The dotenv package is used to load sensitive information (such as database credentials, JWT secrets) from an .env file, ensuring that these values are not hard-coded into the codebase.**8. Bcrypt:• Role:** Password hashing**• Usage:** bcrypt is used to encrypt user passwords before storing them in the database, ensuring secure handling of user credentials. During login, the entered password is compared with the hashed **9. Local Storage:Role:** Client-side storage of JWT tokens**Usage:** Local storage ensures the user stays logged in by storing the JWT token, which is used to authenticate API requests without requiring repeated logins.**10. CORS (Cross-Origin Resource Sharing):• Role:** Security mechanism**• Usage:** CORS is configured to allow cross-origin requests, ensuring that the frontend (React) can communicate with the backend (Node/Express) even if they are hosted on different domains or ports.**11. Mongoose:**

**• Role:** MongoDB object modeling tool**• Usage:** Mongoose simplifies interactions with MongoDB by providing a schema-based solution for modeling data. It ensures consistency in data structure and offers easy-to-use methods for querying the database.**12. Multer:• Role:** Middleware for handling file uploads**• Usage:** Multer enables users to upload media files (such as artwork images) directly to the server. The uploaded files are stored locally or on cloud storage services, depending on the project configuration.**WEBPAGES:**

**Functionality Overview:**

1. **Home Component:**

**Functionality:**

* Displays a fixed background photo with a transparent text box overlay.
* The heading and description inside the text box move smoothly as the user scrolls.
* Smooth scroll navigation to:

About Us section

Our Services section

**UI Design:**

**Main Photo Section:**

* Fixed background photo with transparent overlay text box.
* Heading: "ARTFUSION - Best Destination for Artists and Art Enthusiasts."
* Paragraph: Placeholder text (Lorem ipsum) to introduce the platform.

**Header Elements:**

* Brand Name (ARTFUSION) linking to the homepage.
* Navigation Links: Home, About Us, Services, and Sign Up.
* Smooth scroll on clicking About Us/Services links.
* Sign Up Link: Uses react-router-dom for navigation.

**1.1 About Us Component:**

**Functionality:**

* Provides a detailed description of the platform’s purpose and vision.
* Displays a rotating image carousel that changes every 2 seconds.

**UI Design:**

* Description Box: Heading and brief paragraph highlighting the platform's mission and vision.
* Image Carousel:

5 rotating slides, each displaying a different image every 2 seconds.

Positioned alongside or within the description text.

**1.2 Services Component:**

**Functionality:**

* Displays multiple service cards describing each service offered.
* Cards link to detailed service pages via React Router navigation.

**UI Design**:

* Service Cards:

Cards for services like:

Art Showcase

Artist Networking

Art Resources

Each card contains:

An image placeholder.

A title and description aligned centrally.

Interactive: Links to dedicated pages for each service.

**1.3 Footer Component:**

**Functionality:**

* Displays contact information, social media links, and the address at the bottom of the page.

**UI Design:**

* Contact Details: Email and phone number listed.
* Social Media Links: Icons linking to Facebook, Twitter, and Instagram.
* Address Section: Provides the platform’s physical address.

1. **Login Component:**

**Functionality:**

* Login Form: Accepts email/username and password. Displays error messages for invalid input.
* Login with Google: Redirects to http://localhost:5000/auth/google.
* Post-login Redirect**:** Redirects users to the dashboard after a successful login.

**UI Design:**

* Form Layout: Simple and easy-to-use login form.
* Error Messages: Displayed below the input fields for clarity.
* Google Login Button: Prominent button redirecting to the Google sign-in page.

1. **Sign Up Component:**

**Functionality:**

* Ensures that each email and username is unique.
* Provides an option for Google sign-up via OAuth.
* If the email is already registered, navigates the user to the login page.

**UI Design:**

* Form Layout: User-friendly form with clear field labels.
* Google Sign-Up Button: Provides a quick and easy way to register with Google.
* Smooth Navigation: Redirects users to the login page if they already have an account.

1. **Dashboard Component:**

**Functionality:**

* Sidebar Links**:** Provides easy navigation to:
* Logout: Clears the token from local storage and redirects the user to the login page.

**UI Design:**

* Sidebar Layout: Clean design with section icons and interactive hover effects.
* Dropdown Design: Minimalistic dropdown for settings to avoid clutter.
* Color Scheme: Consistent color palette and responsive spacing.

1. **Art Showcase Component:**

**Functionality**:

* Profile and Posts: Fetches user profile and displays art posts based on user category.
* Like, Repost, Comment: Users can like, repost, and comment on posts.
* Modals: Two modals - one for viewing post details and another for displaying and adding comments.

**UI Design**:

* User-Friendly Card Layout: Each post is shown in a visually appealing card with the username, image, and details.
* Action Icons: Intuitive icons (like, repost, comment) for quick engagement.
* Responsive Modals: Clear modals for image viewing and comment display with close buttons for smooth navigation.

1. **Search Component:**

**Functionality**:

* Search Users: Fetches user list based on search text and navigates to the result page.
* Browse by Tag: Users can click on art tags (e.g., Painting, Music) to view posts in those categories.

**UI Design**:

* Minimal Search Bar: A simple, well-labeled search bar with a Lottie animation for visual appeal.
* Tag Display: Categories are shown with clickable images for easy exploration.
* Interactive Search Results: Smooth navigation to profile results page upon search.

1. **Create Component:**

**Functionality:**

* File Upload with Preview: Users can upload an image and preview it before posting.
* Unique Post Check: Validates that each title is unique.
* Category Selection: Users select from predefined categories via a dropdown or pop-up list.
* Authentication Check: Ensures the user is logged in with a token check.
* Submit Post: Adds the post to the database, and alerts the user of success or failure.

**UI Design:**

* File Upload & Preview: Displays an image preview next to the file selection button for easy verification.
* Category Selection Dropdown: Compact dropdown list or modal for easy category selection.
* Form Layout: Clear, easy-to-navigate form layout with labeled input fields (title, description, category).
* Button Group: Prominent buttons for submission and cancellation, providing intuitive user interaction.

1. **Event Component:**

**Functionality:**

* Date Navigation: Users can navigate between months to view events.
* Add Event: Opens a modal to add event details (name, date, organizer, location, cost, etc.).
* Event Search: Allows users to search for specific dates in "dd-mm-yyyy" format to quickly find events.
* Event Click: Clicking on a date with an event displays event details beside the calendar.
* Yearly Events Highlight: Shows main yearly events in a sidebar for quick reference, with each clickable to view more details.

**UI Design:**

* Calendar View: Displays the month with days as individual squares; highlights the current day in blue and event days in red.
* Event Modal: Clean, labeled form with fields for event details; accessible with a "+" button.
* Event Sidebar: Compact sidebar for main yearly events, where each event has a date circle next to a bold event title.
* Responsive Design: Ensures visibility and ease of use across devices.

1. **Resources Component:**

**Functionality:**

* Search & Filter Options: Users can search for resources and filter them by type (All, Articles, Videos, Images) for easier navigation.
* Content Navigation: Switch between 'Recommended' resources and 'Challenges' with active tab highlighting to enhance user experience.
* Content Display: Resources are displayed in a responsive grid, showcasing images, titles, and action buttons for user interaction.
* Interactive Content Types: Includes diverse resources like articles and challenges that encourage user participation through engaging activities.

**UI Design:**

* Grid Layout: A visually appealing responsive grid layout that organizes resource cards with consistent styling.
* Interactive Filters: A user-friendly search bar and dropdown filters positioned for quick access to enhance resource discovery.
* Styled Toggle Buttons: Clear, color-coded toggle buttons that indicate the active tab, improving navigation between resource categories**.**

1. **Profile Component:**

**Functionality:**

* Profile Fetching: Retrieves and displays user profile information (username, email, bio, etc.) based on the logged-in or selected user.
* Posts and Reposts Management: Shows user posts and reposts, allowing interactions like liking, commenting, and reposting.
* Like/Unlike Feature: Users can like or unlike posts, with visual indicators reflecting the status.
* Commenting System: Users can view and submit comments on posts through a dedicated modal.

**UI Design**:

* Responsive Layout: Adapts to different screen sizes for a smooth experience.
* Profile Header: Prominently displays user profile picture and details.
* Tab Buttons: Clearly labeled buttons with visual feedback for navigation.
* Post Cards: Each post is presented in a card format with interactive buttons for user actions.
* Modal Dialogs: Custom-styled modals for focused user interactions regarding post details and comments.

1. **Edit Profile Component:**

**Functionality:**

* Fetches and displays user data such as username, gender, bio, city, country, and profile photo.
* Profile Update: Allows updates to profile details (except email).
* Save Changes: Sends updated profile details to the server.
* Displays success or error messages based on the update status.

**UI Design:**

* Profile Editing Form: Simple form with input fields arranged vertically.
* Error/Success Messages: Clear feedback in different colors (e.g., green for success, red for errors).
* Profile Photo Preview: Displays a thumbnail preview of the selected photo.

**SAMPLE CODE:**

**1.Artshowcase.jsx**

import React, { useEffect, useState } from 'react';

import { useLocation } from 'react-router-dom';

import API from '../../api/axios';

function ArtShowcase() {

let { state } = useLocation();

const [userProfile, setUserProfile] = useState({});

const [posts, setPosts] = useState([]);

const [error, setError] = useState('');

useEffect(() => {

fetchProfile();

fetchPosts();

}, []);

const fetchProfile = async () => {

try {

const token = localStorage.getItem('token');

const response = await API.get('/user/profile', {

headers: { Authorization: `Bearer ${token}` },

});

setUserProfile(response.data);

} catch (err) {

console.error('Fetch Profile Error:', err);

setError('Failed to load profile.');

}

};

const fetchPosts = async () => {

try {

const token = localStorage.getItem('token');

const res = await API.get(`/posts/${state}`, {

headers: { Authorization: `Bearer ${token}` },

});

setPosts(res.data);

} catch (error) {

console.error('Error fetching posts:', error);

}

};

const handleLike = async (postId, isLiked) => {

const token = localStorage.getItem('token');

try {

if (isLiked) {

await API.patch(`/posts/${postId}/unlike`, {}, {

headers: { Authorization: `Bearer ${token}` },

});

} else {

await API.patch(`/posts/${postId}/like`, {}, {

headers: { Authorization: `Bearer ${token}` },

});

}

fetchPosts();

} catch (err) {

console.error('Like/Unlike Error:', err);

setError('Failed to update like status.');

}

const openCommentsModal = async (postId) => {

// Fetch comments logic here

};

const submitComment = async () => {

// Submit comment logic here

}

return (

<div className="posts-container">

{error && <p className="error-message">{error}</p>}

{posts.length > 0 ? (

posts.map((post) => {

const isLiked = post.likedBy.includes(userProfile.\_id);

return (

<div key={post.\_id} className="post-card">

<img src={`http://localhost:5000/uploads/${encodeURIComponent(post.imageUrl)}`} alt="Post" />

<h6>{post.title}</h6>

<p>{post.description}</p>

<button onClick={() => handleLike(post.\_id, isLiked)}>{isLiked ? 'Unlike' : 'Like'}</button>

<button onClick={() => openCommentsModal(post.\_id)}>Comments</button>

</div>

);

})

) : (

<p>No posts available.</p>

)}

</div>

);

}

export default ArtShowcase;

**2.Search.jsx**

import './Search.css';

import { ImSearch } from "react-icons/im";

import { useForm } from 'react-hook-form';

import { useNavigate } from 'react-router-dom';

import { useState } from 'react';

import API from '../../api/axios';

import Lottie from "react-lottie";

import SearchAnimation from '../../assets/Animations/SearchAnimation.json';

function Search() {

const { register, handleSubmit } = useForm();

const [searchResults, setSearchResults] = useState([]);

let navigate = useNavigate();

const onSearch = async (data) => {

try {

let { searchText } = data;

const response = await fetch(`http://localhost:5000/api/searchUser?username=${searchText}`, {

method: 'GET',

headers: { 'Content-Type': 'application/json' },

});

const out = await response.json(); // Fetch results

setSearchResults(out); // Update state with results

navigate('/dashboard/profiles-searched', { state: out }); // Navigate to search results page

} catch (error) {

console.error('Error fetching users:', error);

}

};

const defaultOptions = {

loop: true,

autoplay: true,

animationData: SearchAnimation,

rendererSettings: {

preserveAspectRatio: "xMidYMid slice"

}

};

return (

<div className="container">

<h2 className='mb-3'>Search</h2>

<form onSubmit={handleSubmit(onSearch)}>

<div className="search-bar">

<input

placeholder="Search for people"

type="text"

{...register("searchText")}

onKeyDown={(e) => {

if (e.key === 'Enter') {

handleSubmit(onSearch)(); // Submit on Enter

}

}}

/>

<Lottie options={defaultOptions} height={50} width={50} />

</div>

</form>

<hr />

<h3 className="mb-3 mt-3">Browse tags</h3>

{/\* Tag buttons to navigate to different categories \*/}

<div className="tags">

{['Painting', 'Dance', 'Music', 'Crafts', 'Poetry', 'Pottery', 'Story', 'Drawing'].map(tag => (

<div className="tag" key={tag}>

<img alt={`${tag} image`} height="50" src={`path\_to\_image/${tag}.jpg`} width="50" />

<div className="tag-text">

<button className='text-left w-100' onClick={() => navigate('/dashboard/art-showcase', { state: tag })}>

{tag}

</button>

</div>

</div>

))}

</div>

</div>

);

}

export default Search;

**3.Event.jsx**

import React, { useState, useEffect, useRef } from 'react';

import './Event.css';

function Event() {

const [currentMonth, setCurrentMonth] = useState(new Date(2024, 9)); // October 2024

const [selectedDate, setSelectedDate] = useState(null);

const [searchDate, setSearchDate] = useState('');

const [events, setEvents] = useState({});

const [isModalOpen, setIsModalOpen] = useState(false);

const [formData, setFormData] = useState({

eventName: '',

organizer: '',

location: '',

cost: '',

activities: '',

date: '',

});

const modalRef = useRef(null);

useEffect(() => {

const handleClickOutside = (event) => {

if (modalRef.current && !modalRef.current.contains(event.target)) {

setIsModalOpen(false);

}

};

if (isModalOpen) {

document.addEventListener('mousedown', handleClickOutside);

}

return () => {

document.removeEventListener('mousedown', handleClickOutside);

};

}, [isModalOpen]);

const getDaysInMonth = (year, month) => new Date(year, month + 1, 0).getDate();

const getFirstDayOfMonth = (year, month) => new Date(year, month, 1).getDay();

const handleDateClick = (day) => {

const dateStr = `${currentMonth.getFullYear()}-${String(currentMonth.getMonth() + 1).padStart(2, '0')}-${String(day).padStart(2, '0')}`;

setSelectedDate(dateStr);

};

const handleNextMonth = () => {

setCurrentMonth(new Date(currentMonth.getFullYear(), currentMonth.getMonth() + 1));

};

const handlePreviousMonth = () => {

setCurrentMonth(new Date(currentMonth.getFullYear(), currentMonth.getMonth() - 1));

};

const handleSearchDateChange = (e) => {

setSearchDate(e.target.value);

};

const handleSearchSubmit = (e) => {

e.preventDefault();

const [day, month, year] = searchDate.split('-');

const searchDateObj = new Date(year, month - 1, day);

if (searchDateObj && searchDateObj.getDate() === parseInt(day)) {

setCurrentMonth(searchDateObj);

const formattedDate = `${year}-${String(month).padStart(2, '0')}-${String(day).padStart(2, '0')}`;

setSelectedDate(formattedDate);

} else {

alert('Invalid date. Please use the format dd-mm-yyyy.');

}

};

const closeModal = () => {

setIsModalOpen(false);

};

const openModal = () => {

setIsModalOpen(true);

};

const handleFormChange = (e) => {

const { name, value } = e.target;

setFormData((prev) => ({ ...prev, [name]: value }));

};

const handleFormSubmit = (e) => {

e.preventDefault();

const { date, eventName } = formData;

if (date && eventName) {

setEvents((prev) => ({

...prev,

[date]: { ...formData },

}));

closeModal();

} else {

alert('Please fill out the event name and date');

}

};

const daysInMonth = getDaysInMonth(currentMonth.getFullYear(), currentMonth.getMonth());

const firstDay = getFirstDayOfMonth(currentMonth.getFullYear(), currentMonth.getMonth());

const daysOfWeek = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'];

return (

<div className="event-container">

<div className="event-header">

<h2>Events</h2>

<form onSubmit={handleSearchSubmit} className="search-form">

<input

type="text"

value={searchDate}

onChange={handleSearchDateChange}

placeholder="dd-mm-yyyy"

required

/>

<button type="submit">Search</button>

<div className="plus-button" onClick={openModal}>

<span>+</span>

</div>

</form>

</div>

<div className="calendar-container">

<div className="calendar">

<div className="calendar-header">

<button onClick={handlePreviousMonth}>←</button>

<h3>{currentMonth.toLocaleString('default', { month: 'long', year: 'numeric' })}</h3>

<button onClick={handleNextMonth}>→</button>

</div>

<div className="days-of-week">

{daysOfWeek.map((day) => (

<div key={day} className="day">{day}</div>

))}

</div>

<div className="calendar-grid">

{Array.from({ length: firstDay }).map((\_, i) => (

<div key={`empty-${i}`} className="calendar-date empty"></div>

))}

{Array.from({ length: daysInMonth }).map((\_, i) => {

const day = i + 1;

const dateStr = `${currentMonth.getFullYear()}-${String(currentMonth.getMonth() + 1).padStart(2, '0')}-${String(day).padStart(2, '0')}`;

return (

<div

key={day}

className="calendar-date"

onClick={() => handleDateClick(day)}

>

{day}

</div>

);

})}

</div>

</div>

<div className="event-section">

<h3>Events for {selectedDate || new Date().toISOString().split('T')[0]}</h3>

{selectedDate && events[selectedDate] ? (

<div>

<h4>{events[selectedDate].eventName}</h4>

<p>Organizer: {events[selectedDate].organizer}</p>

<p>Location: {events[selectedDate].location}</p>

<p>Cost: {events[selectedDate].cost}</p>

<p>Activities: {events[selectedDate].activities}</p>

</div>

) : (

<p>No events found for this date.</p>

)}

</div>

</div>

{isModalOpen && (

<div className={`modal ${isModalOpen ? 'show' : ''}`}>

<div className="modal-content" ref={modalRef}>

<h3>Add New Event</h3>

<button className="close" onClick={closeModal}>✖</button>

<form onSubmit={handleFormSubmit}>

<input

type="text"

name="eventName"

value={formData.eventName}

onChange={handleFormChange}

placeholder="Event Name"

required

/>

<input

type="text"

name="organizer"

value={formData.organizer}

onChange={handleFormChange}

placeholder="Who is Conducting"

/>

<input

type="text"

name="location"

value={formData.location}

onChange={handleFormChange}

placeholder="Where it is Conducting"

/>

<input

type="text"

name="cost"

value={formData.cost}

onChange={handleFormChange}

placeholder="Free or Cost Paying"

/>

<input

type="date"

name="date"

value={formData.date}

onChange={handleFormChange}

required

/>

<textarea

name="activities"

value={formData.activities}

onChange={handleFormChange}

placeholder="Activities Conducted on that Workshop"

/>

<button type="submit">Save Event</button>

</form>

</div>

</div>

)}

</div>

);

}

export default Event;

**4.Resources.jsx**

// src/components/dashboard/Resources.jsx

import React, { useState } from 'react';

import "./Resources.css";

function Resources() {

const [showResources, setShowResources] = useState(false);

const [showTypes, setShowTypes] = useState(false);

const [selectedResource, setSelectedResource] = useState('');

const [selectedType, setSelectedType] = useState('');

const [searchQuery, setSearchQuery] = useState('');

const [results, setResults] = useState([]);

const handleResourcesClick = () => {

setShowResources(!showResources);

};

const handleTypesClick = () => {

setShowTypes(!showTypes);

};

const handleResourceSelect = (resource) => {

setSelectedResource(resource);

setShowResources(false);

};

const handleTypeSelect = (type) => {

setSelectedType(type);

setShowTypes(false);

};

const handleSearchChange = (e) => {

setSearchQuery(e.target.value);

};

const handleSearchSubmit = (e) => {

e.preventDefault();

const fetchedResults = Array.from({ length: 9 }, (\_, index) => ({

title: `${searchQuery} in ${selectedResource}`,

link: `https://${selectedResource.toLowerCase()}.com/${searchQuery.toLowerCase()}-${index + 1}`,

image: 'https://placehold.co/300x200'

}));

setResults(fetchedResults);

};

return (

<div>

<main className="w-4/5 p-8">

<h2 className="text-3xl font-bold mb-8">Resources</h2>

<form onSubmit={handleSearchSubmit} className="flex mb-8">

<input

type="text"

placeholder="Search"

className="w-1/3 p-2 border rounded mr-4"

value={searchQuery}

onChange={handleSearchChange}

/>

<div className="relative w-5 mr-4">

<button type="button" onClick={handleResourcesClick} className="w-full p-2 border rounded flex justify-between items-center">

<span>{selectedResource || 'Resources'}</span>

</button>

{showResources && (

<ul className="absolute w-full bg-white border rounded mt-1">

<li onClick={() => handleResourceSelect('YouTube')}>YouTube</li>

<li onClick={() => handleResourceSelect('Instagram')}>Instagram</li>

<li onClick={() => handleResourceSelect('Pinterest')}>Pinterest</li>

<li onClick={() => handleResourceSelect('Chrome')}>Chrome</li>

<li onClick={() => handleResourceSelect('Others')}>Others</li>

</ul>

)}

</div>

<div className="relative w-1/3">

<button type="button" onClick={handleTypesClick} className="w-full p-2 border rounded flex justify-between items-center">

<span>{selectedType || 'Type'}</span>

</button>

{showTypes && (

<ul className="absolute w-full bg-white border rounded mt-1">

<li onClick={() => handleTypeSelect('Poetry')}>Poetry</li>

<li onClick={() => handleTypeSelect('Stories')}>Stories</li>

<li onClick={() => handleTypeSelect('Drawings')}>Drawings</li>

<li onClick={() => handleTypeSelect('Painting')}>Painting</li>

<li onClick={() => handleTypeSelect('Crafts')}>Crafts</li>

</ul>

)}

</div>

</form>

<div className="grid grid-cols-3 gap-4">

{results.map((result, index) => (

<div key={index} className="bg-white p-4 rounded shadow">

<img src={result.image} alt={result.title} className="mb-4 rounded" />

<a href={result.link} target="\_blank" rel="noopener noreferrer" className="text-blue-500 underline">

{result.title}

</a>

</div>

))}

</div>

</main>

</div>

);

}

export default Resources;

**5.Profile.jsx**

import React, { useEffect, useState } from 'react';

import API from '../../api/axios';

import Modal from 'react-modal';

import { useLocation } from 'react-router-dom';

Modal.setAppElement('#root');

function Profile() {

const [userProfile, setUserProfile] = useState({});

const [posts, setPosts] = useState([]);

const [activeTab, setActiveTab] = useState('posts');

const [error, setError] = useState('');

const [modalIsOpen, setModalIsOpen] = useState(false);

const [selectedPost, setSelectedPost] = useState(null);

const { state } = useLocation();

useEffect(() => {

const fetchProfile = async () => {

try {

const token = localStorage.getItem('token');

const response = await API.get(state ? `/user/profile/${state.\_id}` : '/user/profile', {

headers: { Authorization: `Bearer ${token}` },

});

setUserProfile(response.data);

fetchPosts();

} catch (err) {

setError('Failed to load profile.');

}

};

fetchProfile();

}, [state]);

const fetchPosts = async () => {

try {

const token = localStorage.getItem('token');

const response = await API.get(state ? `/posts/user/${state.\_id}` : '/posts/user', {

headers: { Authorization: `Bearer ${token}` },

});

setPosts(response.data);

} catch (err) {

setError('Failed to load posts.');

}

};

const handleTabClick = (tab) => {

setActiveTab(tab);

};

const openModal = (post) => {

setSelectedPost(post);

setModalIsOpen(true);

};

const closeModal = () => {

setModalIsOpen(false);

setSelectedPost(null);

};

return (

<div className="profile-container">

<div className="profile-header">

<h2>{userProfile.username}</h2>

<p>{userProfile.bio}</p>

</div>

<div className="profile-tabs">

<button onClick={() => handleTabClick('posts')}>Posts</button>

<button onClick={() => handleTabClick('reposts')}>Reposts</button>

</div>

{activeTab === 'posts' && (

<div className="posts-container">

{error && <p className="error-message">{error}</p>}

{posts.length > 0 ? posts.map((post) => (

<div key={post.\_id} className="post-card" onClick={() => openModal(post)}>

<img src={post.imageUrl} alt="Post" />

<p>{post.title}</p>

</div>

)) : <p>No posts available.</p>}

</div>

)}

{/\* Image Modal \*/}

{selectedPost && (

<Modal isOpen={modalIsOpen} onRequestClose={closeModal} contentLabel="Post Details">

<h2>{selectedPost.title}</h2>

<img src={selectedPost.imageUrl} alt="Post" />

<button onClick={closeModal}>Close</button>

</Modal>

)}

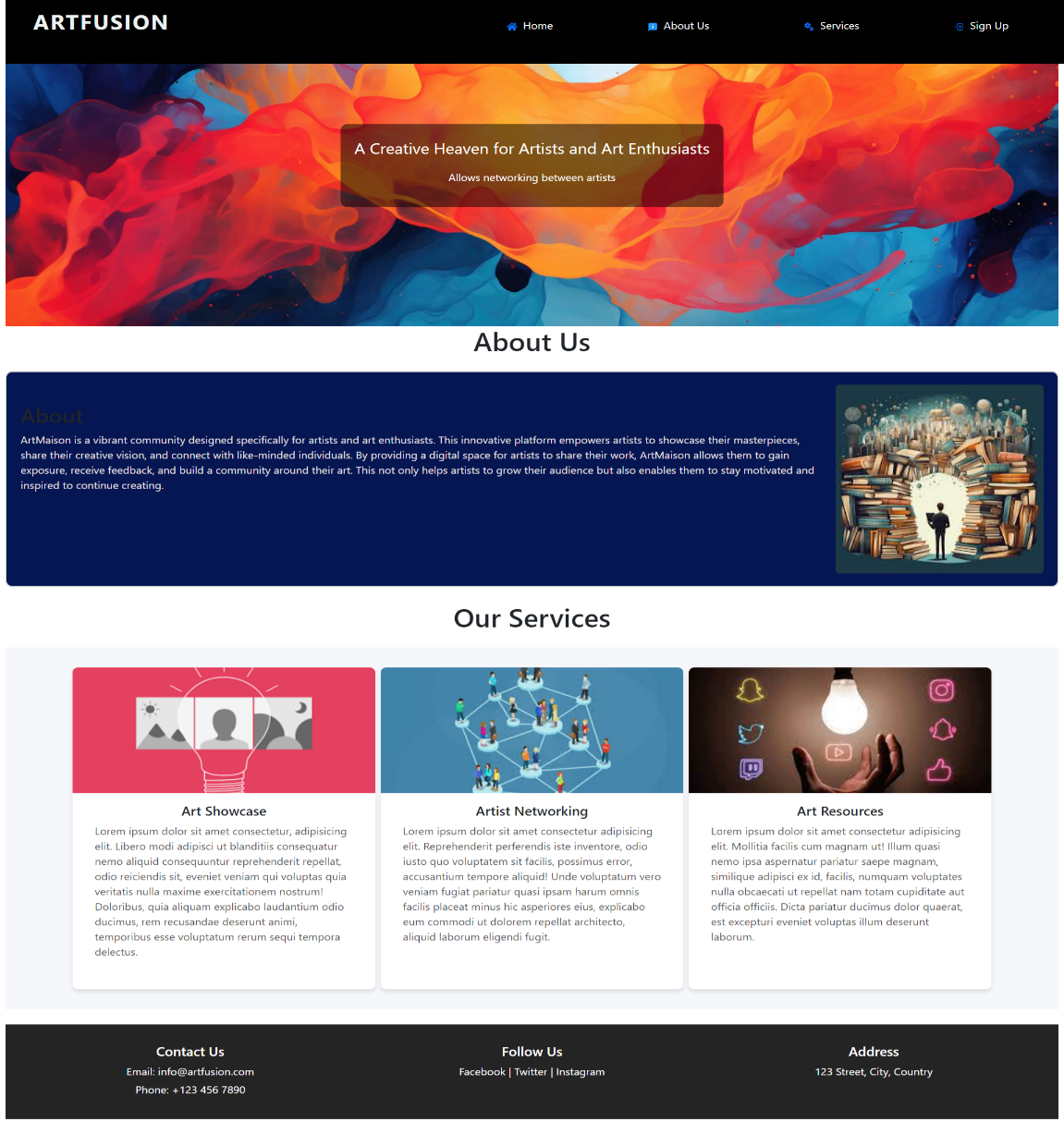
</div>

);

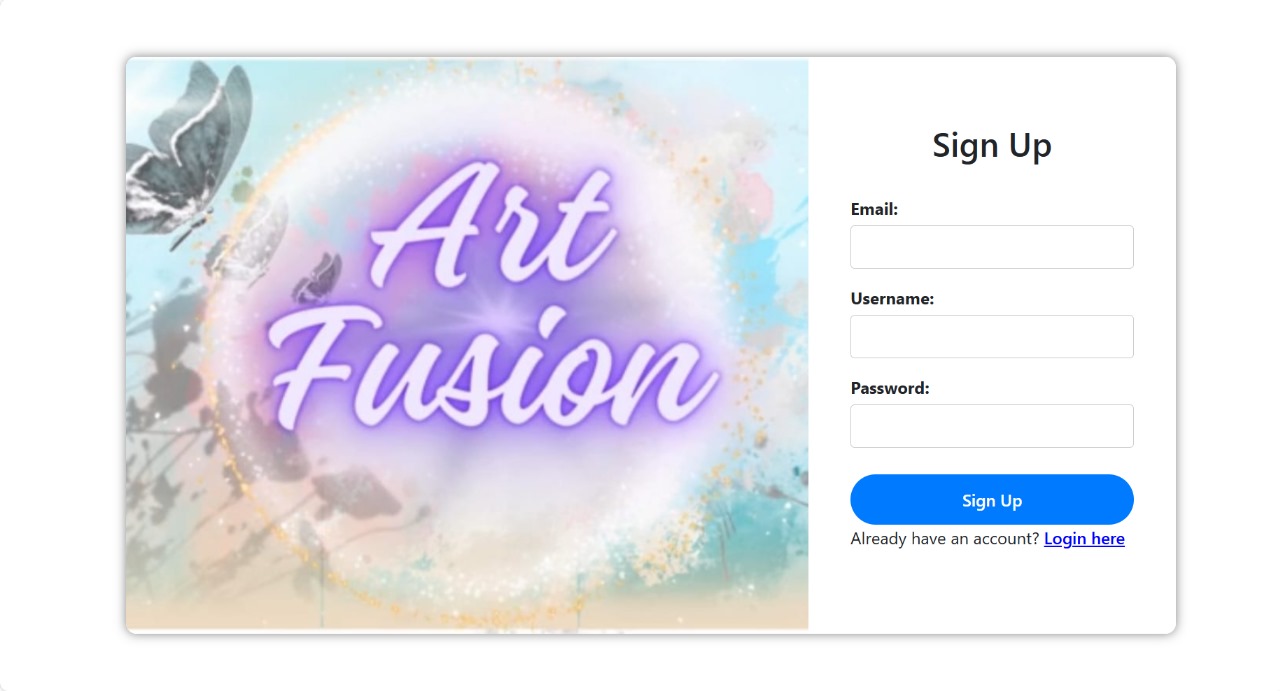
}

export default Profile;

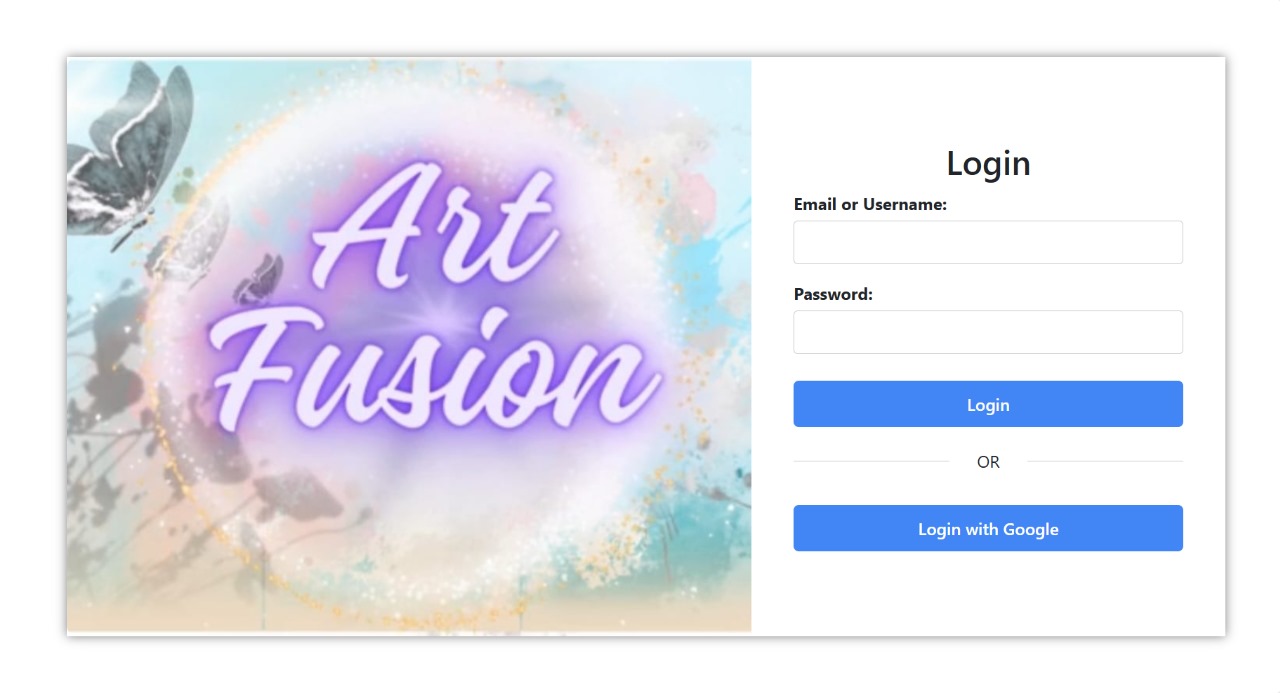
**RESULT/OUTPUT SCREENSHOTS:**

****

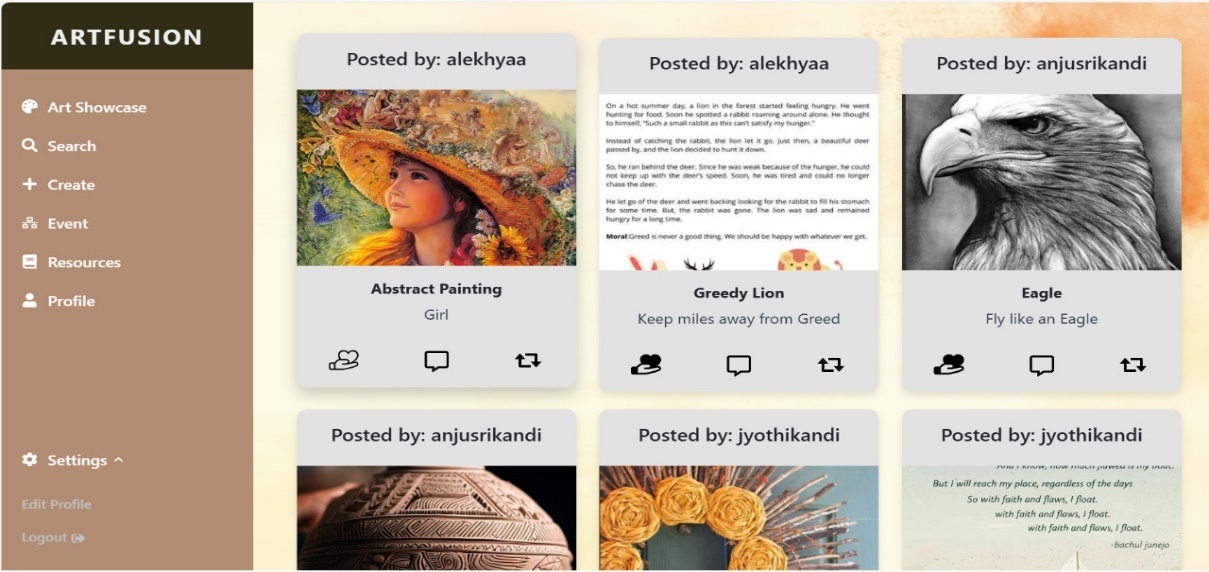
**Fig -1: Home Page**

****

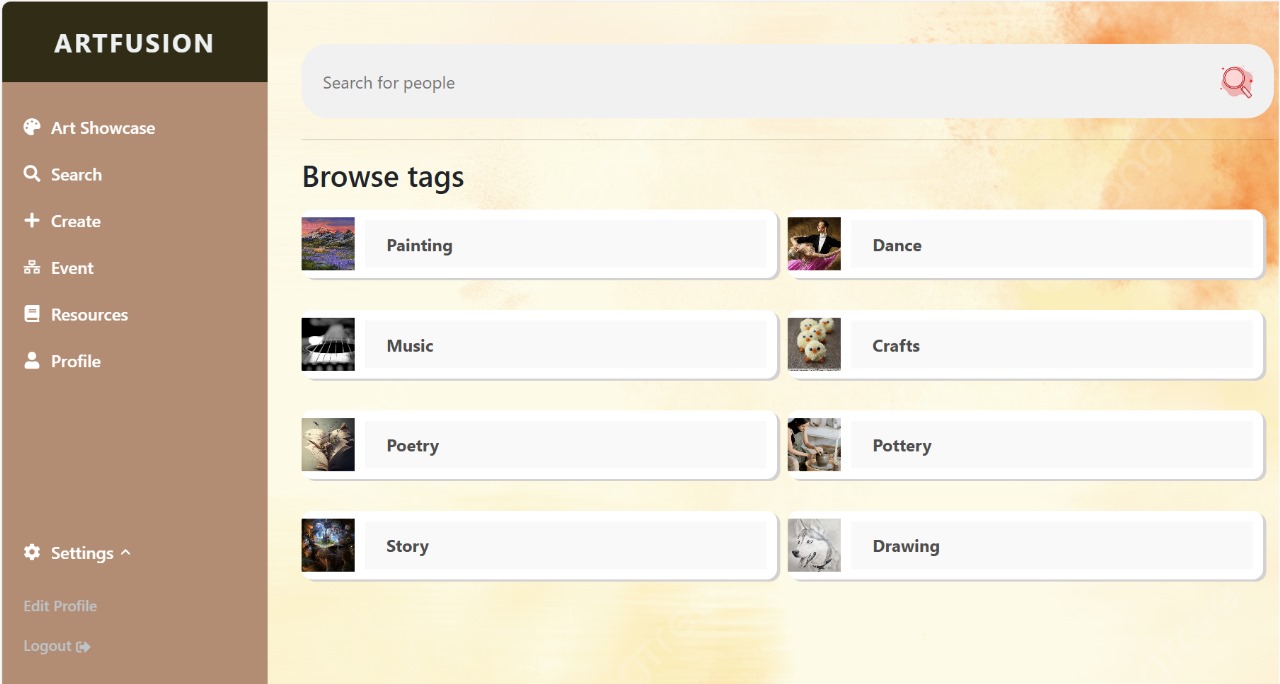
**Fig-2: Sign Up Page**



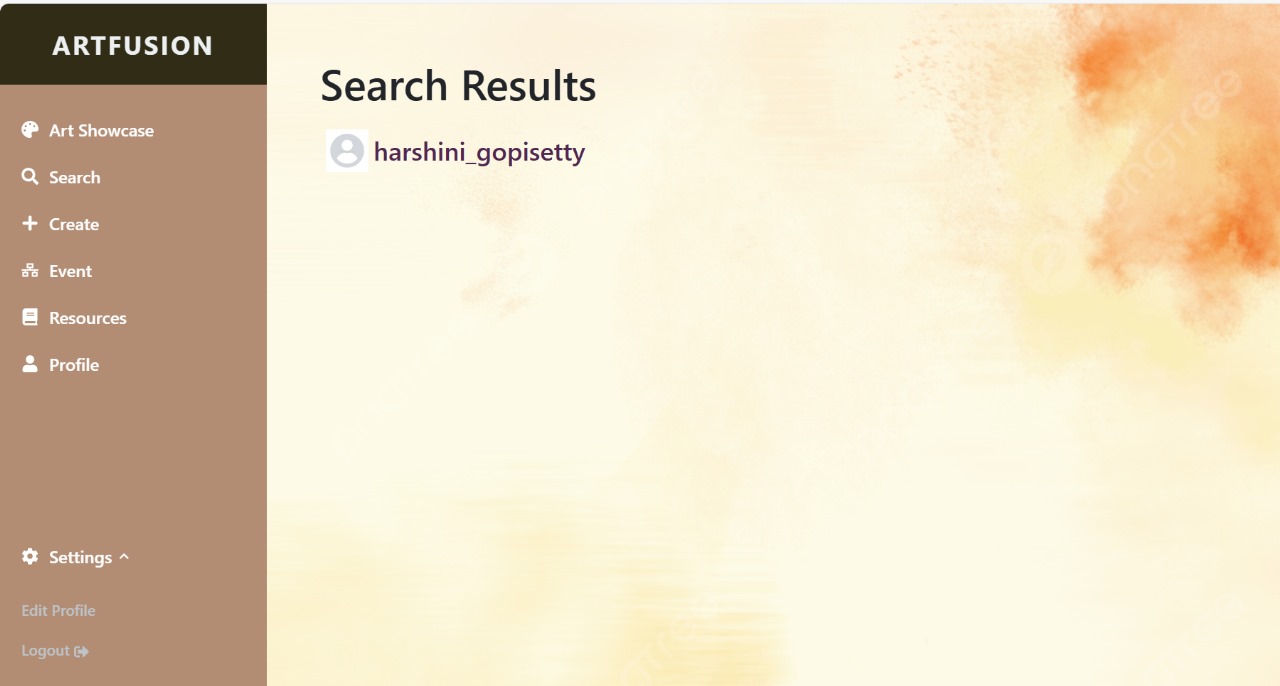
**Fig-3: Login Page**



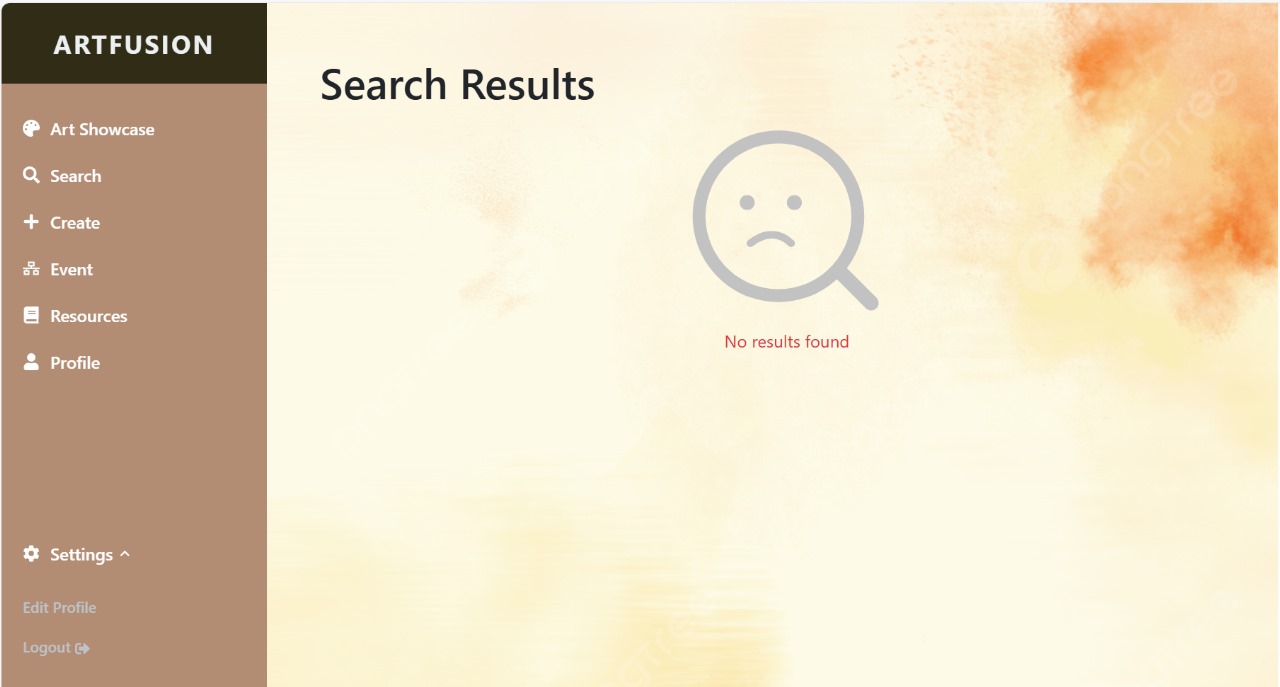
**Fig-4: ArtShowcase Page**



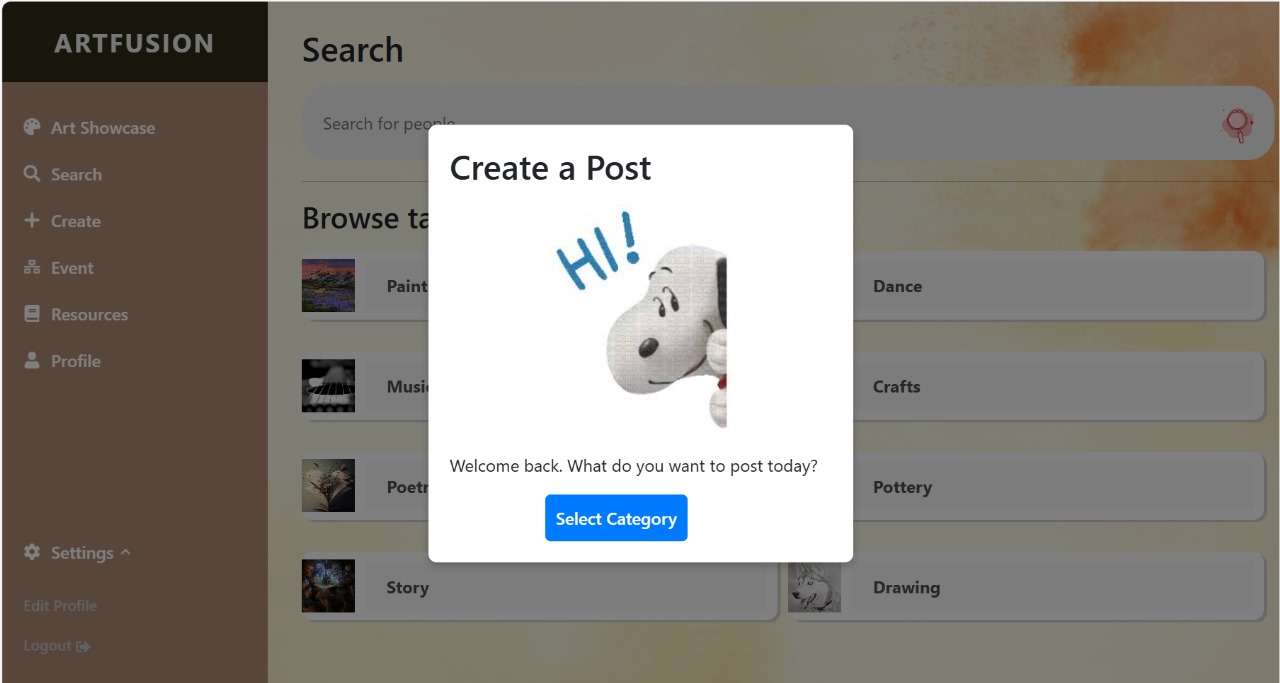
**Fig-5: Search Page**



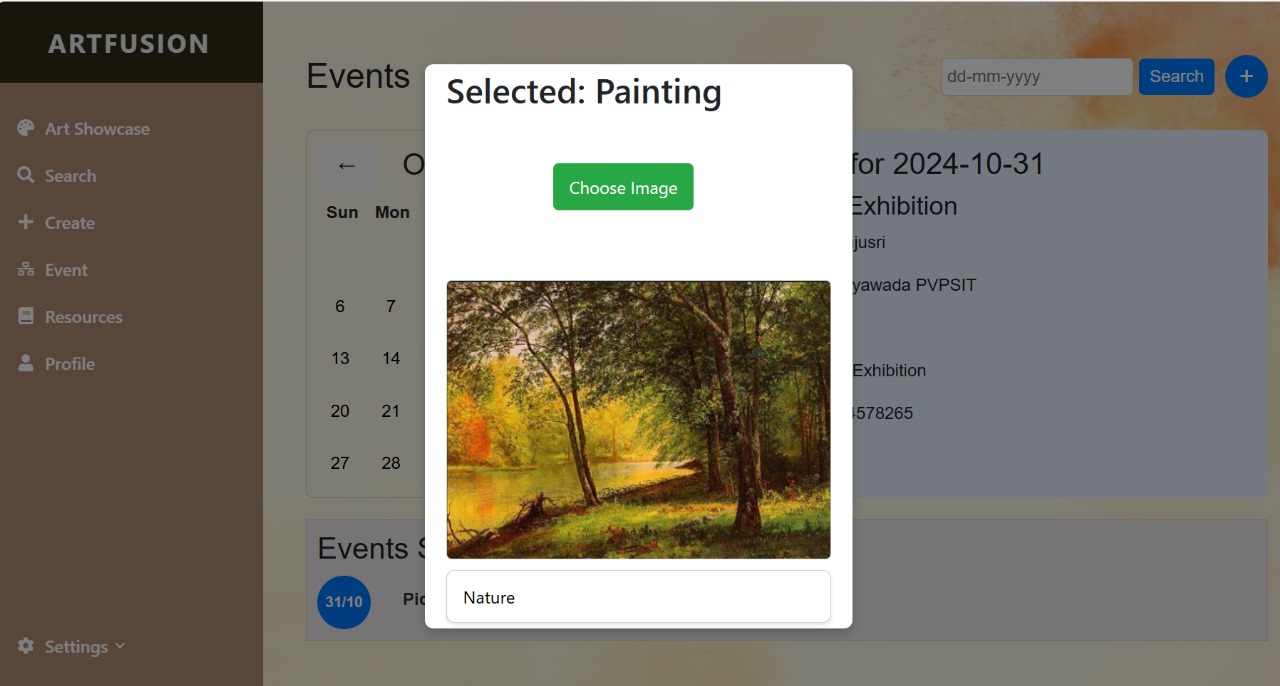
**Fig-5.1: When User found in Search Page**



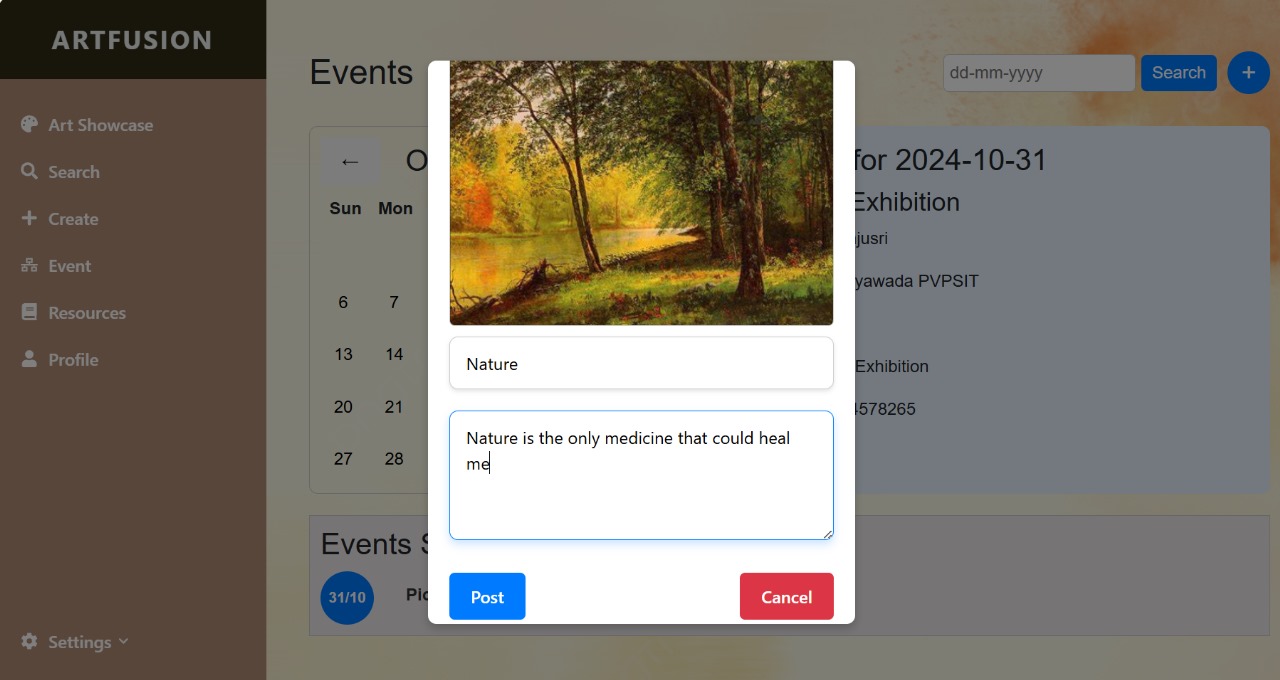
**Fig-5.2: When User not found in Search Page**



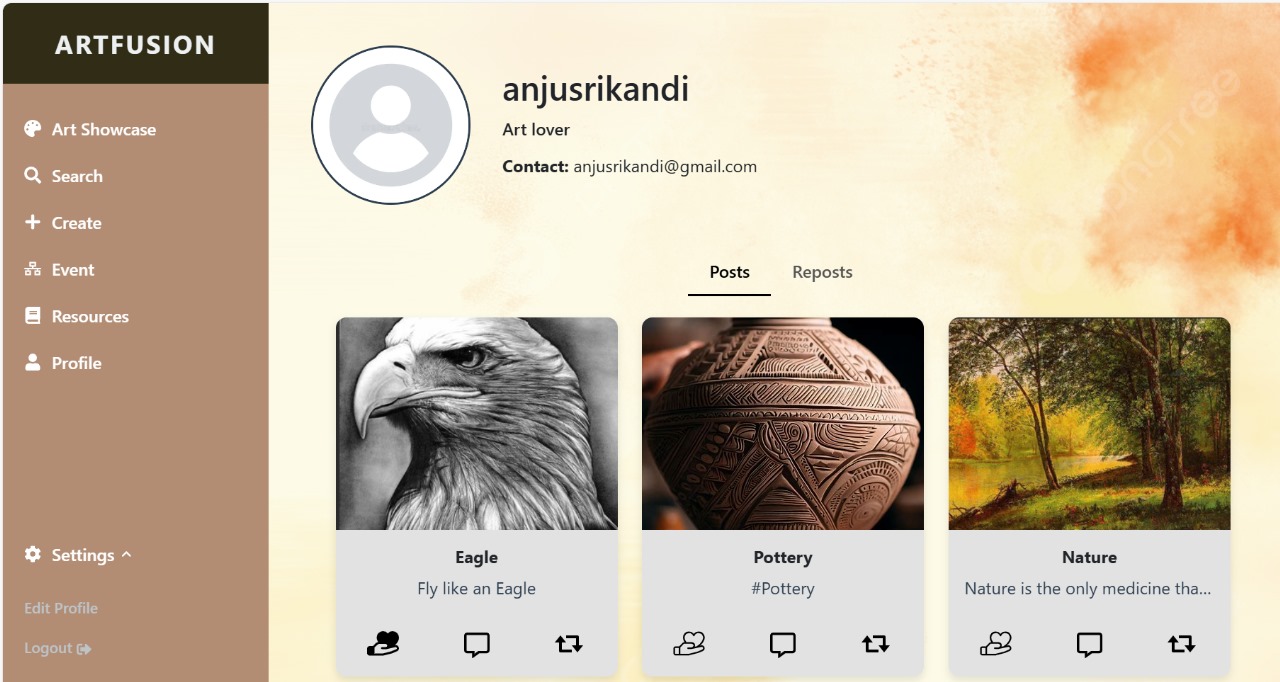
**Fig 6: Create Page**



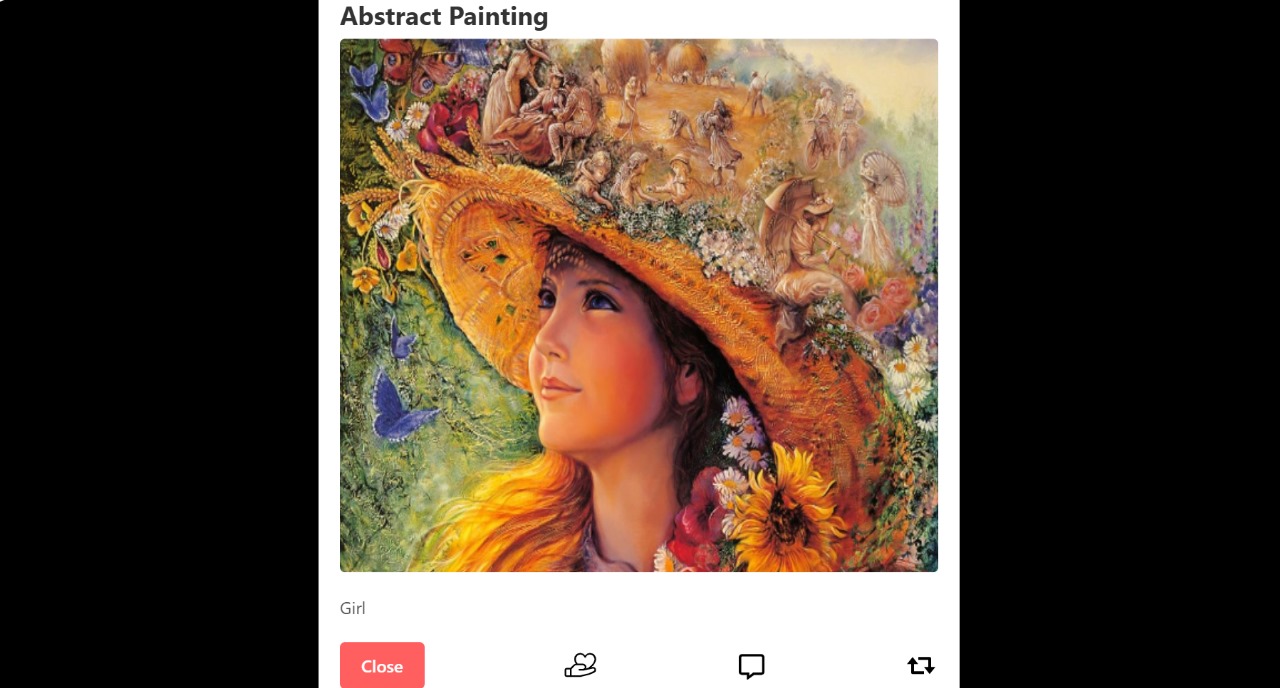
**Fig-6.1: Select the category to post**



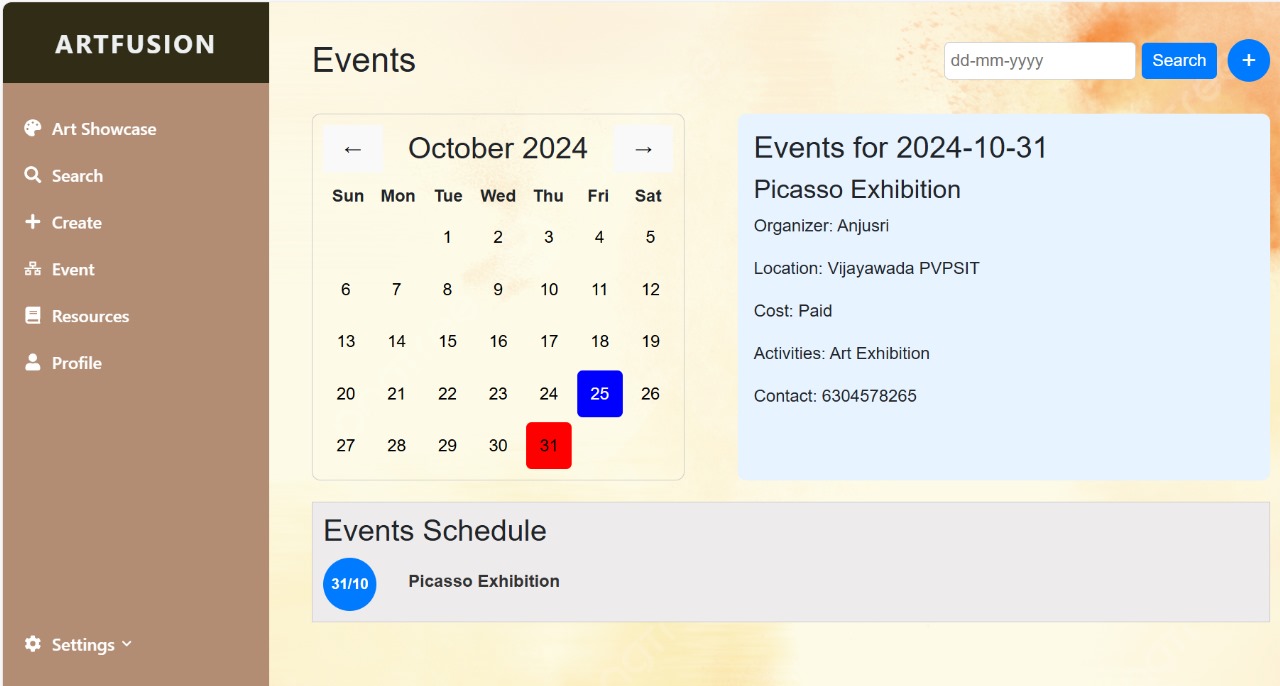
**Fig-6.2: Title and description for post**



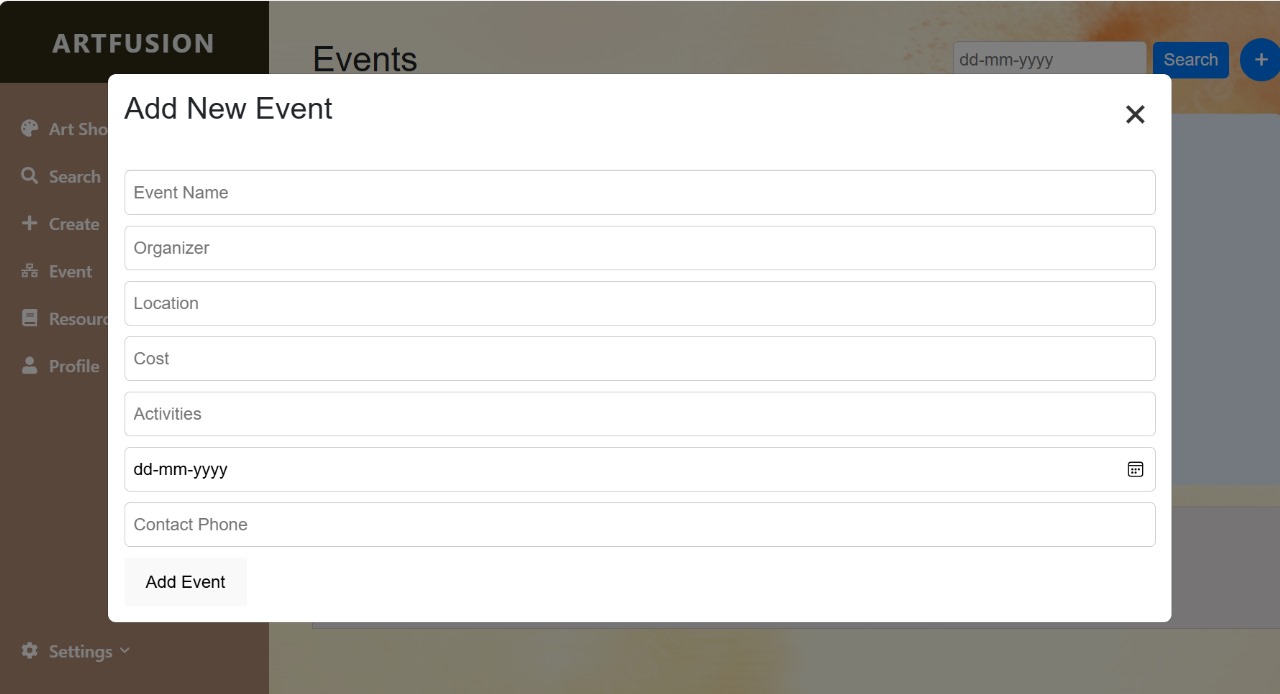
**Fig-7: Profile Page**



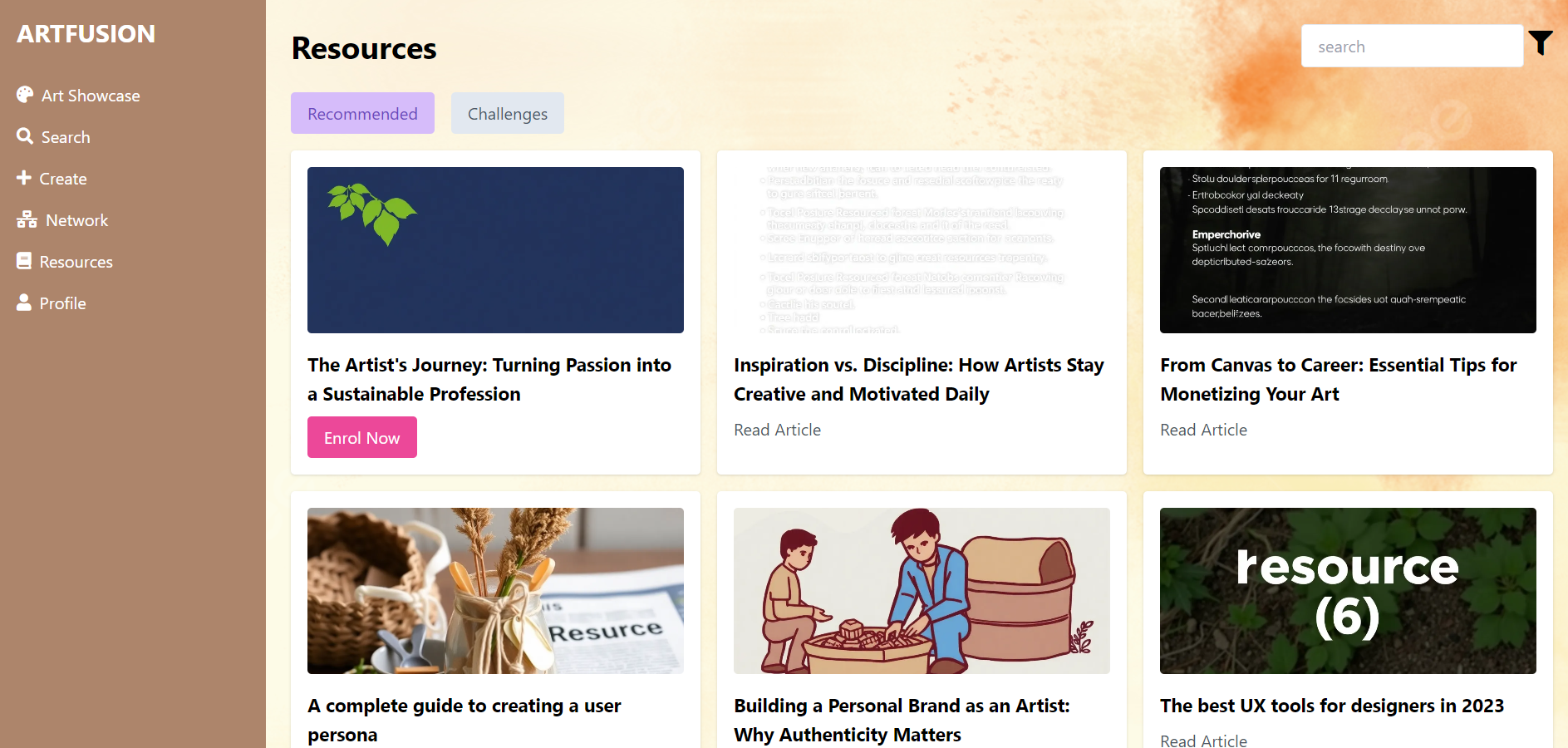
**Fig-7.1: Post in Profile Page**



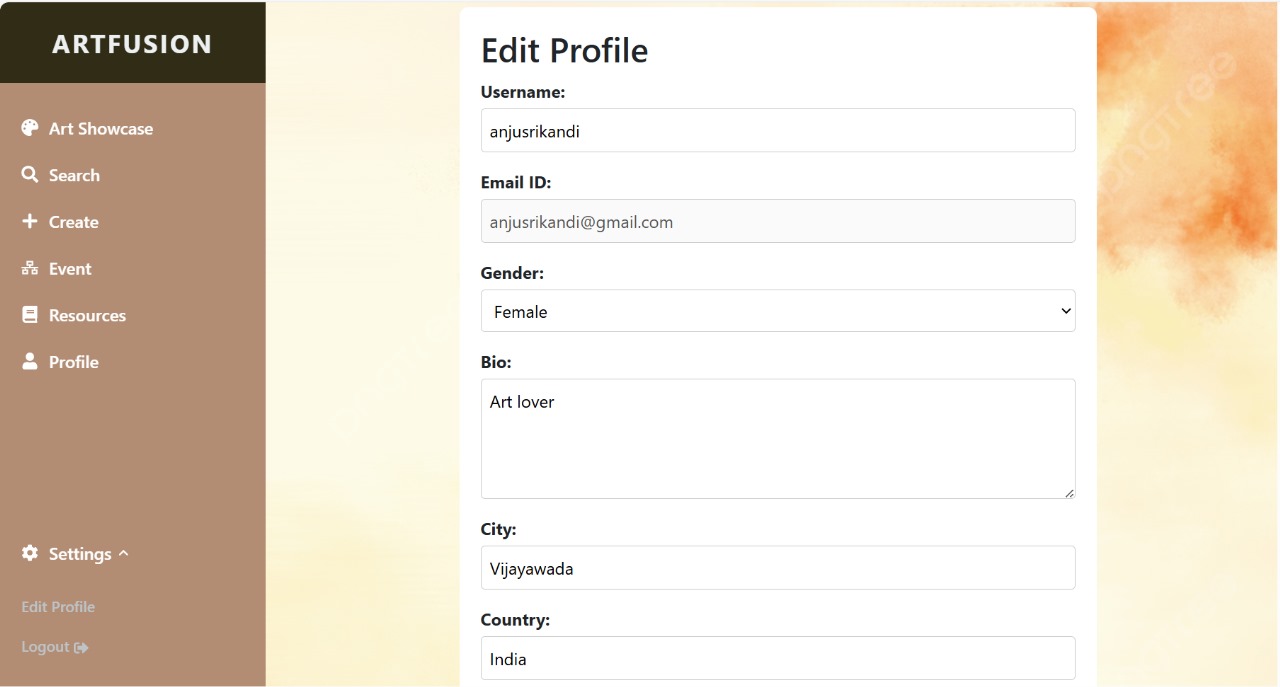
**Fig-8: Event Page**



**Fig-8.1: Add New Event in Event Page**

****

**Fig-9: Resources Page**



**Fig-10: Edit – Profile Page**

**CONCLUSION:**

**ArtFusion** represents a significant step forward in connecting artists and fostering creativity across various disciplines. By providing an inclusive platform where individuals can showcase their work, collaborate, and access valuable resources, ArtFusion empowers users to explore and express their artistic potential. The integration of cutting-edge technology through the MERN STACKensures a seamless and secure experience, allowing artists to focus on what they do best— creating art.

As ArtFusion continues to evolve, it remains committed to nurturing a vibrant community that transcends boundaries and promotes diversity in artistic expression. The platform not only enhances visibility for emerging and established artists but also serves as a hub for learning and inspiration. With ongoing improvements based on user feedback, ArtFusion is poised to become a vital resource for artists worldwide, enriching their creative journeys and fostering connections that lead to meaningful collaborations.

By bridging the gap between artists and audiences, ArtFusionstands as a testament to the power of creativity in bringing people together and transforming the way art is shared and appreciated.

**REFERENCES (WEBSITE URLs):**