

ArtGallery

**A PROJECT REPORT
for
Mini Project (KCA353)
Session (2023-24)**

Submitted by

Bittu Jaiswal

University Roll No : 2200290140048

**Submitted in partial fulfillment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

**Under the Supervision of
Dr. Vipin Kumar
Associate Professor**



Submitted to

**DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206**

(February 2024)

CERTIFICATE

Certified that **Bittu Jaiswal (2200290140048)** has carried out the project work having “**ArtGallery**” (**Mini Project KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Bittu Jaiswal (2200290140048)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Vipin Kumar
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

This project aims to develop an innovative image search application leveraging API technology, designed to revolutionize the way users explore and discover visual content. The application will harness the power of an API, acting as a gateway to an extensive database of images, enabling users to query and retrieve relevant visual data swiftly and efficiently through a user-friendly interface.

This application will offer seamless interaction, allowing users to input keywords, phrases, or visual cues to initiate searches. The integration of an API will facilitate real-time communication with a vast repository of images, enabling rapid retrieval and display of results matching the user's query.

The project's key focus lies to enhance search accuracy and relevancy. Implementation of advanced search functionalities, such as filters based on image attributes, metadata, or categories, will enable users to precisely narrow down their search parameters.

Additionally, the application will prioritize scalability and robustness by ensuring compatibility with various devices and operating systems. It will embrace a modular and adaptable architecture, facilitating potential expansion and integration with future updates or additional APIs for enriched functionalities.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Vipin Kumar** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Bittu Jaiswal

TABLE OF CONTENTS

	Certificate	i
	Abstract	ii
	Acknowledgements	iii
	Table of Contents	iv-v
	List of Figures	vi
1	Introduction	1-2
1.1	Background	1
1.2	Project overview	1
1.3	Objective	1
1.4	Key features	1
1.5	Scope of the project	2
2	Problem identification & feasibility study	3-4
2.1	Problem Identification	3
2.2	Feasibility Study	3
2.2.1	Technical Feasibility	3
2.2.2	Operational Feasibility	4
2.2.3	Economic Feasibility	4
3	Requirement Analysis	5-6
3.1	Introduction	5
3.2	Functional Requirements	5
3.2.1	User Module	5
3.3	Non-Functional Requirements	5
3.3.1	Performance	5
3.3.2	Security	5
3.3.3	Scalability	6
3.3.4	Usability	6
4	Project Planning and Scheduling	7-8
4.1	Pert Chart	7
5	Hardware and Software Specification	9-10
5.1	Hardware Specification	9
5.2	Software Specification	10
6	Choice of Tools & Technology	11-14
6.1	React	11
6.2	Data Flow Diagram	11
6.3	Context Level Diagram	12

7	ER-Diagram	15-16
7.1	Entity-relationship model	15
7.2	Class Diagram	16
8	Form Design	17-22
8.1	Home	17
8.2	Signup	18
8.3	Signin	19
8.4	Search Module	20
8.5	Result	21
9	Coding	23-67
10	Testing	68-69
9.1	Introduction	68
9.2	Types of Testing	69
	Future Scope and Further Enhancement of the Project	70
	Conclusion & References	71

LIST OF FIGURES

Figure No.	Name of Figure	Page No.
4.1	Pert Chart	8
6.1	0 Level DFD	13
6.2	1 Level DFD	13
6.3	2 nd Level DFD	14
7.1	Entity-relationship Model	15
8.1	Home	17
8.2	Signup	18
8.3	Signin	19
8.4	Search module	20
8.5	Result	21

CHAPTER 1

INTRODUCTION

1.1 Background

This project aims to develop an innovative image search application leveraging API technology, designed to revolutionize the way users explore and discover visual content. The application will harness the power of an API, acting as a gateway to an extensive database of images, enabling users to query and retrieve relevant visual data swiftly and efficiently.

1.2 Project Overview

ArtGallery is a web-based application designed to fetches image data from Api about the searched thing and displays all the images in beautiful layout. This system employs a robust technology stack, primarily utilizing React and Node.js. This system provides a user-friendly and dynamic platform for users.

1.3 Objective

The primary objectives of ArtGallery include:

Efficiency: Develop an application that utilizes an API to swiftly and accurately retrieve images based on user queries, ensuring a seamless and responsive experience.

User Convenience: Providing end users with a smooth and intuitive interface to search image and download

1.4 Key Features

ArtGallery systems include a number of features, including:

Image Search: Intuitive interface for users to Search Image.

Download Image: In ArtGallery you can also download the image.

Zoom Image: In ArtGallery you can also zoom the image.

1.5 Scope of the project

The scope of ArtGallery extends to the following:

User Interface: A responsive and user-friendly interface for end users.

Scalability: The Application is designed to adapt and scale as per the evolving needs of the user.

This chapter sets the stage for a detailed exploration of the ArtGallery. The subsequent chapters will highlight the technical aspects, functionalities and implementation details, providing a comprehensive understanding of this innovative solution

CHAPTER 2

PROBLEM IDENTIFICATION & FEASIBILITY STUDY

2.1 Problem Identification

The motivation behind developing the ArtGallery arises from the acknowledgment of enduring issues in conventional manual image retrieval processes. Manual systems frequently result in inefficiencies, imprecise image recognition, delayed search processing, and a dearth of transparency in retrieving visual content. These challenges not only hinder the operational flow of image searches but also contribute to a suboptimal user experience

2.2 Feasibility Study

Prior to initiating the development of the ArtGallery, an exhaustive feasibility study was undertaken to evaluate the viability and practicability of the envisioned system. This study comprehensively covers technical, operational, and economic feasibility aspects pertinent to the implementation of an advanced image retrieval platform.

2.2.1 Technical Feasibility

The technical feasibility examination guaranteed that the envisaged ArtGallery could be constructed utilizing the chosen technologies- React. It scrutinized the accessibility of necessary software and hardware resources, the adaptability of the system with the current infrastructure, and the requisite technical proficiency for the triumphant implementation of the image retrieval platform.

2.2.2 Operational Feasibility

Operational feasibility scrutinized the degree to which the ArtGallery aligns with the current operations of image retrieval. This entailed evaluating the seamless integration into daily routines, the adaptability of personnel to the new system, and the overarching influence on operational processes within the context of image search functionality.

2.2.3 Economic Feasibility

Economic feasibility delved into the cost-effectiveness of implementing the ArtGallery. This encompassed evaluating development costs, continual maintenance expenses, and potential savings or revenue generation attributed to enhanced efficiency and user satisfaction in image retrieval processes. A meticulous cost-benefit analysis played a pivotal role in ascertaining the financial viability of the image search project.

In conclusion, the problem identification phase brought to light the deficiencies inherent in existing image retrieval systems, paving the way for a focused solution. The subsequent feasibility study, likewise, affirmed that the envisioned ArtGallery is not merely a theoretically sound concept but also a pragmatic and economically viable undertaking.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Introduction

Requirement analysis serves as a pivotal phase in the development lifecycle of the ArtGallery. This chapter aims to meticulously gather, document, and analyze the functional and non-functional requirements that will shape the design and implementation of the ArtGallery.

3.2 Functional Requirements

3.2.1 User Module

The User Module is fundamental to the ArtGallery, encompassing functionalities such as user registration, login, and profile management. End-users should be able to search image, download image, and zoom image seamlessly.

3.3 NON-FUNCTIONAL REQUIREMENTS

3.3.1 PERFORMANCE

The ArtGallery should be responsive, ensuring swift image search, download image, and zoom image. Response times should be optimized to enhance user satisfaction.

3.3.2 SECURITY

Data security is paramount. User authentication, secure transmission of sensitive information, and access controls must be implemented to safeguard user data.

3.3.3 SCALABILITY

The system should be designed to accommodate an increasing number of users.

3.3.4 USABILITY

A user-friendly interface is essential for end-users. The ArtGallery should be intuitive, should be easy to use

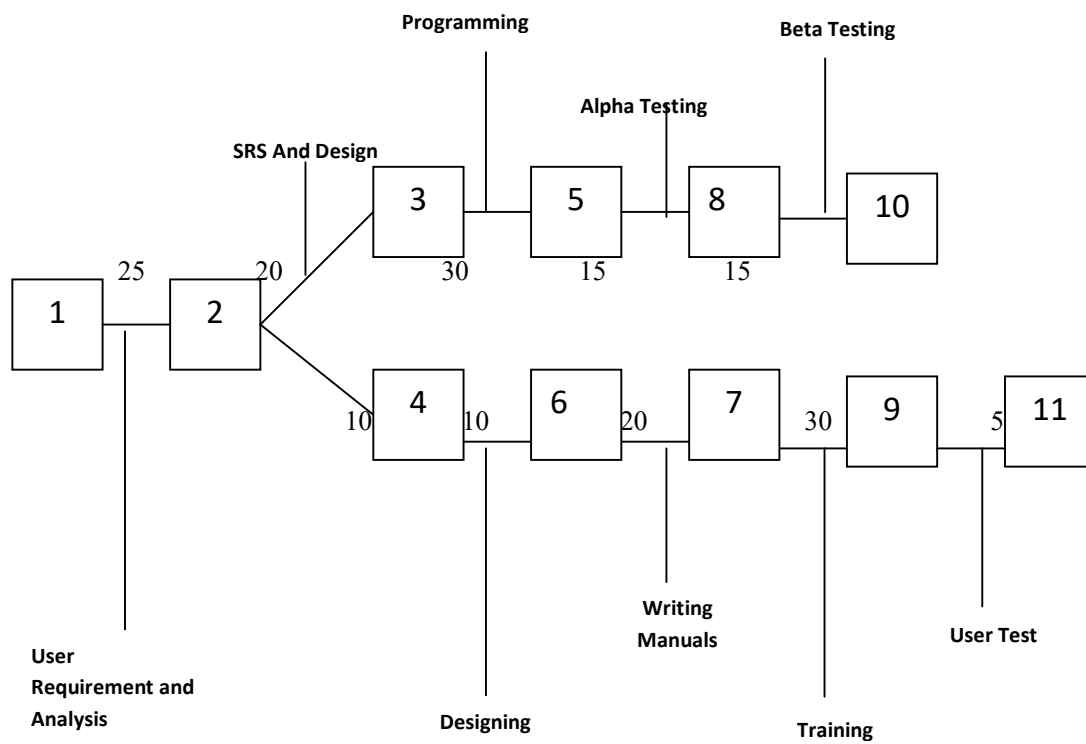
CHAPTER 4

PROJECT PLANNING AND SCHEDULING

4.1 Pert Chart:

A PERT chart is a project management tools used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique. A PERT chart presents a graphic illustration of a project as network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the project linked by labelled vectors (directional lines) representing tasks in the project.

The direction of the arrows on the lines indicates the sequence of tasks.



CHAPTER 5

HARDWARE & SOFTWARE SPECIFICATION

5.1 Hardware Specification

The ArtGallery will be designed and deployed on a hardware infrastructure that guarantees optimal performance and reliability. The recommended hardware specifications are as follows:

Server:

Processor: Intel Core i5 or equivalent

RAM: 8 GB or higher

Storage: 256 GB SSD or higher

Database Server:

Processor: Intel Core i5 or equivalent

RAM: 8 GB or higher

Storage: 256 GB SSD or higher

Network Interface: Gigabit Ethernet

Client Machines:

Processor: Intel Core i3 or equivalent

RAM: 4 GB or higher

Storage: 128 GB SSD or higher

Network Interface: 100 Mbps Ethernet or Wi-Fi

5.2 Software Specification

The ArtGallery will be developed using a combination of server-side and client-side technologies. The development and deployment environment will be facilitated by MERN, which provides a comprehensive stack for web application development. The software specifications include:

Server-Side Technologies:

Operating System: Windows Server 2016 or later

Web Server: Node.js

Database Management System: MongoDB

Server-Side Scripting Language: JavaScript (Node.js)

Client-Side Technologies:

Web Browser: Latest versions of Chrome, Firefox, Safari, or Edge

Client-Side Scripting: JavaScript

Development Tools:

Integrated Development Environment (IDE): Visual Studio Code.

Version Control:

Git: Version control for collaborative development

Security:

SSL/TLS: Ensure secure data transmission over the network

Firewall: Implement firewall rules to restrict unauthorized access

Anti-malware Software: Regularly updated anti-malware software on server and client machines

CHAPTER 6

CHOICE OF TOOLS & TECHNOLOGY

6.1 React

React is a JavaScript library used for building user interfaces. React allows developers to describe the UI's appearance at any point in time, and it automatically updates and renders the right components when the data changes.

React organizes the UI into reusable components, making it easier to manage and maintain complex applications. React uses a virtual DOM to optimize and update the actual DOM efficiently, improving performance by minimizing unnecessary re-rendering.

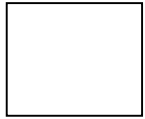
React uses JSX, a syntax extension that looks similar to XML or HTML, to describe the structure of UI components in a more concise and readable way. React components can manage their internal state, and data can be passed to components through props, allowing for dynamic and interactive UIs

6.2 Data Flow Diagram

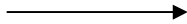
The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It's also known as BUBBLE CHART. The purpose of DFD is major transformation that will become in system design symbols used in DFD: -

In the DFD, four symbols are used and they are as follows.

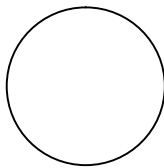
1. A square defines a source (originator) or destination of system data.



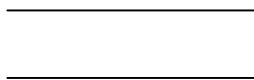
2. An arrow identifies data flow-data in motion. It is a pipeline through which information flows.



3. A circle or a “bubble” (Some people use an oval bubble) represents a process that transfers incoming data flows into outgoing data flows.



4. An open rectangle is a data store-data at rest, or a temporary repository of data.



6.4 Context Level Diagram

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. ArtGallery is shown as one process in the context diagram; which is also known as zero level DFD, shown below. The context diagram plays an important role in understanding the system and determining the boundaries. The main process can be broken into sub-processes and the system can be studied with more detail; this is where 1st level DFD comes into play.



Fig 6.1

A Level 0 Data Flow Diagram (DFD) is the most abstract and high-level representation of a system's processes, data flows, and external entities. It provides a bird's-eye view of the entire system and illustrates the major processes involved. At this level, the focus is on the overall functionality of the system without delving into the detailed processes within each function.

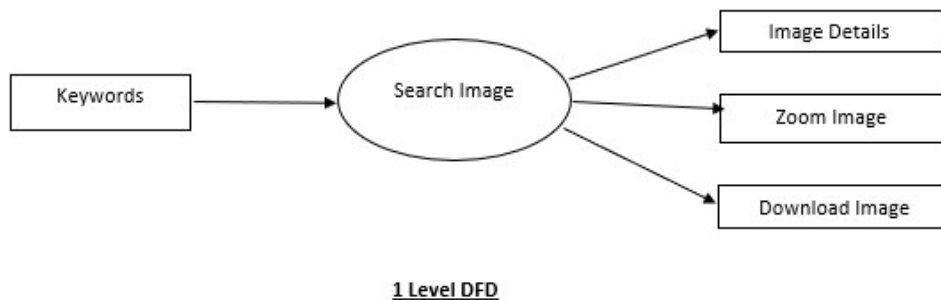
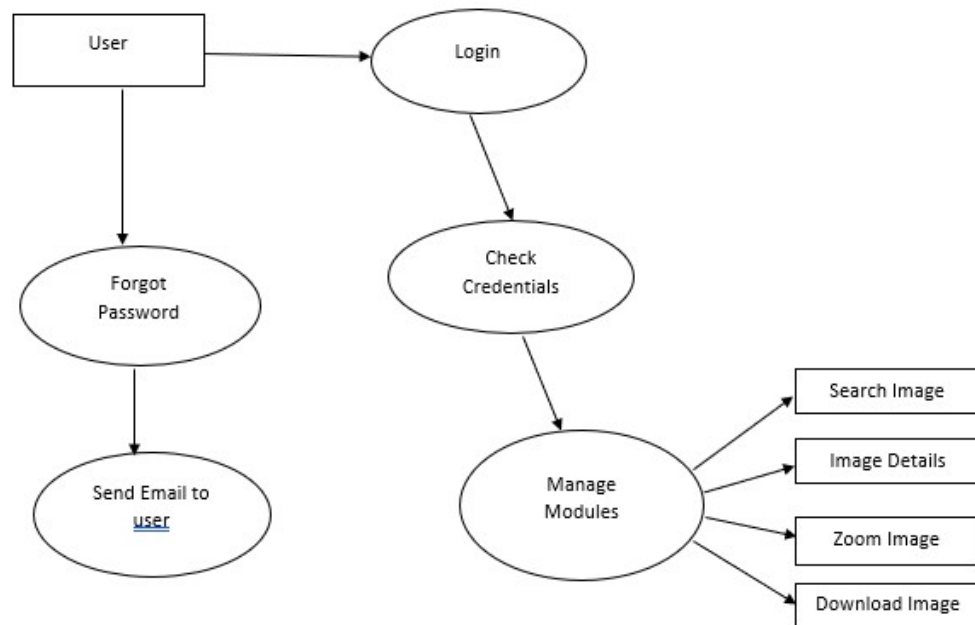


Fig 6.2

A Level 1 Data Flow Diagram (DFD) is a more detailed representation of a system compared to the Level 0 DFD. It breaks down the major processes identified in the Level 0 DFD into sub-processes, providing a more granular view of the system's functionality. Each process from the Level 0 DFD is decomposed into further subprocesses, making it easier to understand the detailed interactions within the system.



2nd Level DFD

Fig 6.3

A Level 2 Data Flow Diagram (DFD) is a further detailed refinement of a Level 1 DFD. It continues the process of decomposition, breaking down the subprocesses identified in the Level 1 DFD into even more detailed activities or tasks. At this level, the focus is on capturing a more comprehensive and intricate understanding of how specific processes function within the system.

CHAPTER 7

ER-DIAGRAM

7.1 Entity-relationship model: -

The entity-relationship model or entity-relationship diagram (ERD) is a data model or diagram for high-level descriptions of conceptual data model, and it provides a graphical notation for representing such data models in the form of entity-relationship diagrams.

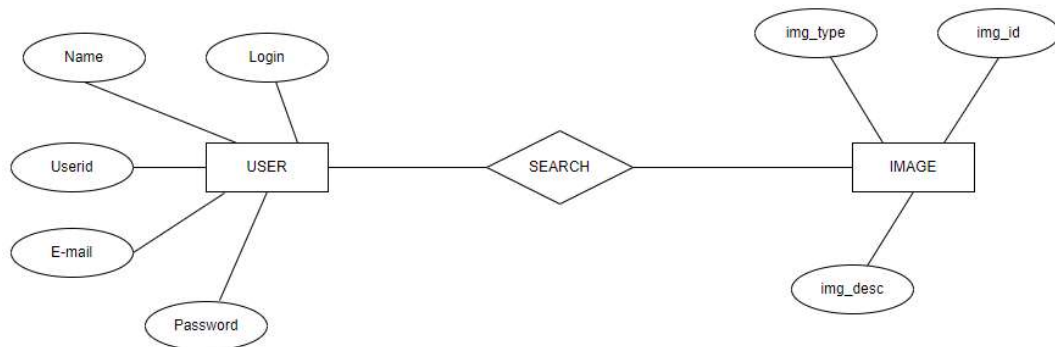


Fig 7.1

7.2 Class Diagram: -

Authentication:

- Classification: Weak Class
- Description: Represents user authentication details, including username and password. This class is responsible for user login functionality.

User:

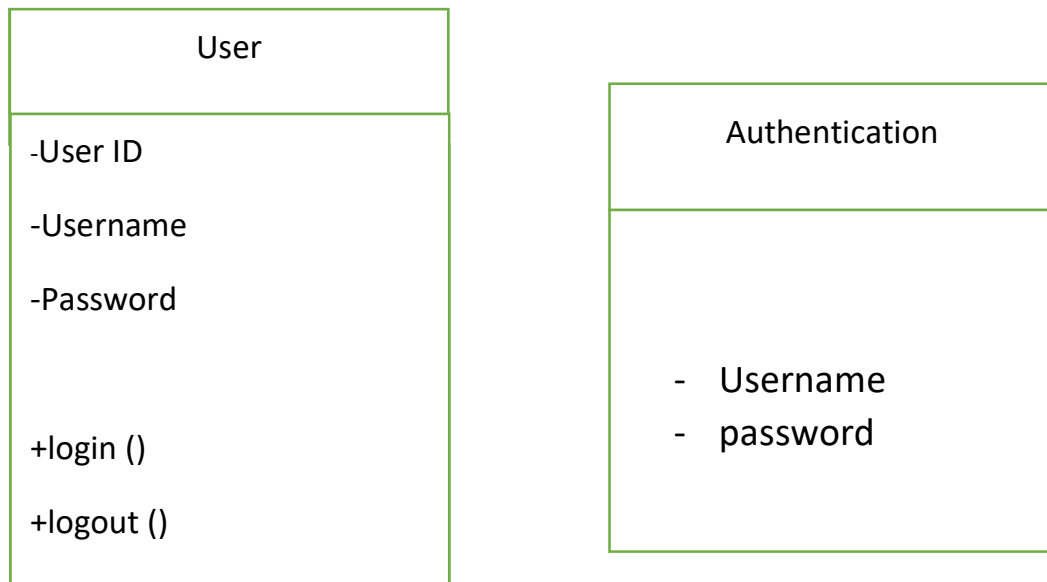
- Classification: Strong Class
- Description: Represents the users of the system.

Canteen Item:

- Classification: Strong Class
- Description: Represents the items available in the canteen, including details such as item ID, name, price, and stock quantity.

Stock:

- Classification: Strong Class
- Description: Represents the stock of items in the canteen, including details such as item ID, quantity, and reorder level.



CHAPTER 8

FORM DESIGN

8.1 Home

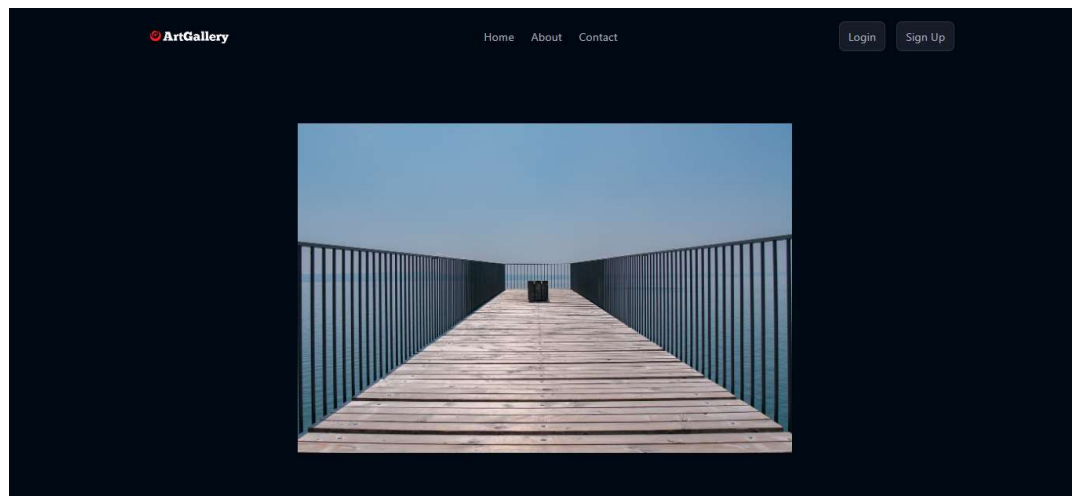


Fig. 8.1

The home page of an image search application serves as the initial point of interaction for users and typically aims to provide a user-friendly and intuitive experience. A navigation menu present, offering links to different sections of the application, such as trending images, categories, user account settings.

8.2 Signup

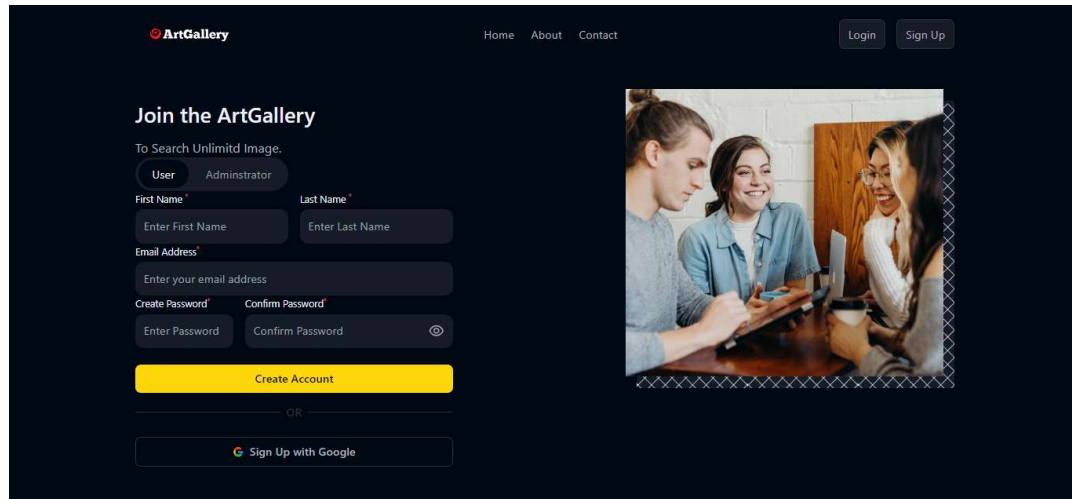
The image shows a web page for 'ArtGallery' with a dark blue background. At the top, there is a navigation bar with links for 'Home', 'About', and 'Contact', along with 'Login' and 'Sign Up' buttons. The main heading is 'Join the ArtGallery'. Below it, a subheading says 'To Search Unlimited Image.' There are two tabs: 'User' (selected) and 'Administrator'. The form includes fields for 'First Name', 'Last Name', 'Email Address', 'Create Password', and 'Confirm Password'. A prominent yellow 'Create Account' button is at the bottom of the form. Below this is an 'OR' separator and a 'Sign Up with Google' button. To the right of the form is a photograph of three young women smiling and looking at a tablet together.

Fig. 8.2

The primary element of the signup page is the registration form, where users provide essential information to create an account. This typically includes fields for a username, email address, password, and possibly additional details for personalization.

A prominent "Sign Up" or "Create Account" button allows users to submit their information and proceed with the registration process.

Upon successful registration, a confirmation message is displayed, welcoming users to the platform and providing any additional instructions, such as logging in to access the application.

8.3 Signin

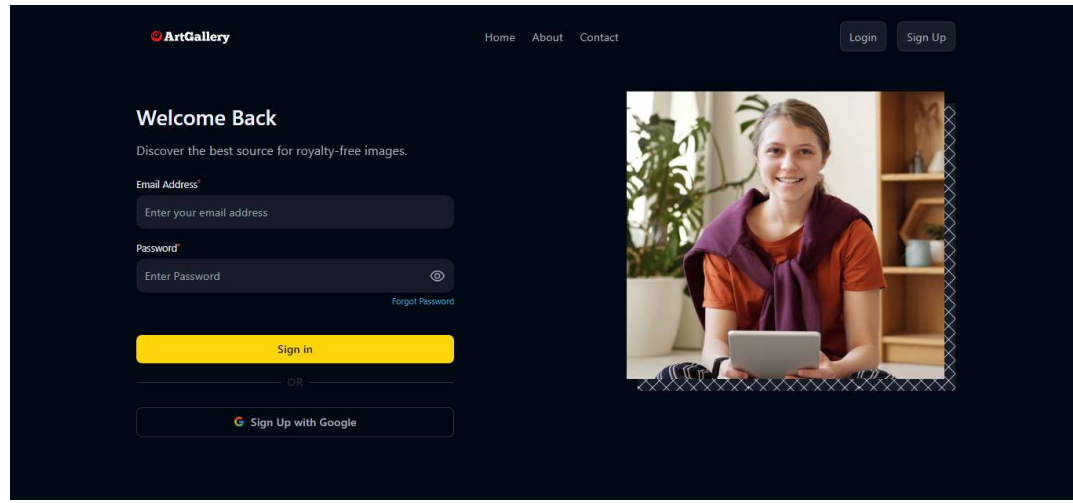


Fig 8.3

The login module in the ArtGallery is a pivotal component responsible for securely authenticating users and regulating access to the system. This module verifies user identity by validating entered credentials, typically consisting of a username and password. Following successful authentication user can search image. Session management is initiated to maintain user states throughout their interactions within the ArtGallery.

Security measures, including password hashing and salting, are implemented to safeguard user credentials. The login module may also include features such as an account lockout mechanism to counter multiple failed login attempts, logging and auditing for monitoring user activities, and password recovery options for forgotten passwords.

Overall, the login module is of paramount importance as it serves as the initial checkpoint, ensuring that only authorized users gain access to the ArtGallery while upholding security and user-friendly practices.

8.4 Search module

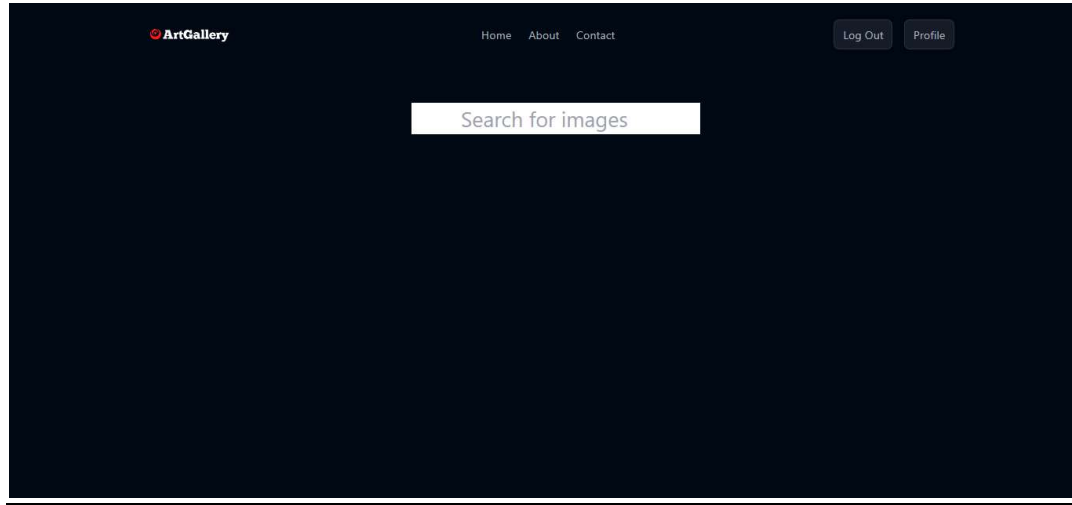


Fig 8.4

Search in an image search application typically refers to the functionality or component within the application that allows users to input their search queries and retrieve relevant results.

8.5 Result

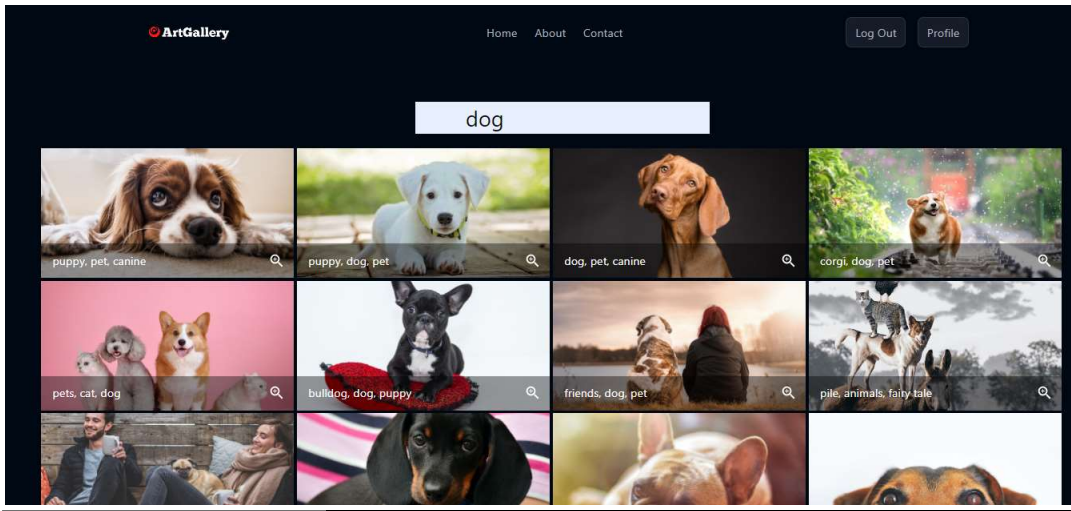


Fig 8.5

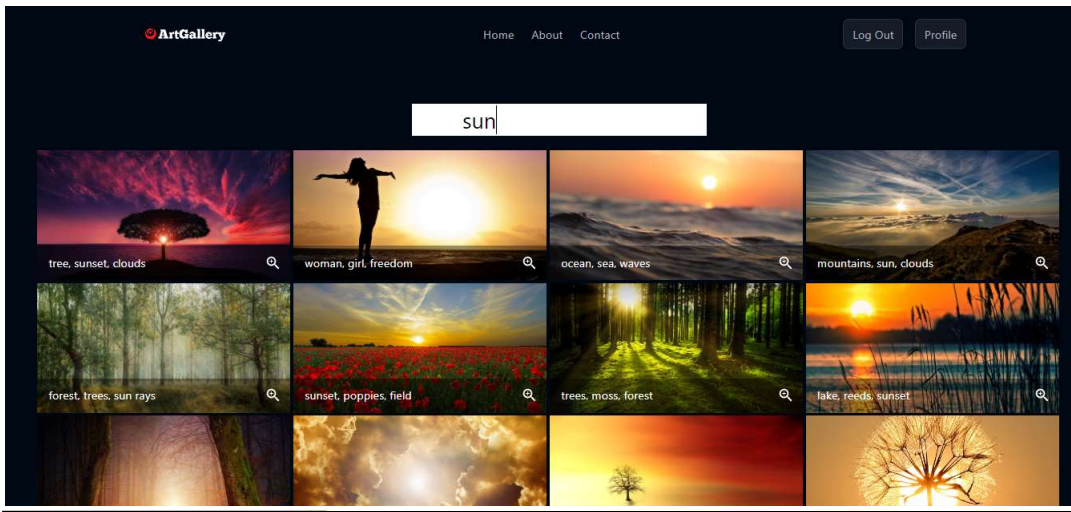


Fig 8.6

When users initiate a search in an image search application, the system employs sophisticated algorithms to analyze and match the query against a vast database of indexed images. The results are then displayed to the user in the form of a visually appealing grid or list of images.

Relevant metadata, such as image file name, size, and additional information, may be provided alongside the results. This helps users assess the suitability of each image before clicking on it.

Image search applications often feature a responsive design, enabling users to view results seamlessly across various devices, from smartphones to desktops, enhancing accessibility and user experience.

CHAPTER 9

CODING

Package.json

```
{
  "name": "router-project",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@material-ui/core": "^4.12.4",
    "@mui/icons-material": "^5.14.15",
    "@mui/material": "^5.14.15",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.6.0",
    "material-ui": "^0.20.2",
    "prop-types": "^15.8.1",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-hot-toast": "^2.4.1",
```

```

    "react-icons": "^4.8.0",
    "react-router-dom": "^6.9.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "tailwindcss": "^3.2.7"
  }
}

```

Index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { BrowserRouter } from "react-router-dom";
import { Toaster } from "react-hot-toast";

const root =
ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <BrowserRouter>
    <App />
    <Toaster />
  </BrowserRouter>
);
```


App.js

```
import './App.css';
import Navbar from './Components/Navbar';
import Login from './Components/Login';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Signup from './Components/Signup';
import Dashboard from './Components/Search';
import PrivateRoute from './Components/PrivateRoute';
import { Route, Routes } from 'react-router-dom';
import React, { useState } from 'react';
import MuiThemeProvider from 'material-
ui/styles/MuiThemeProvider'

function App() {
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  return (
    <MuiThemeProvider>
      <div className="w-screen h-screen bg-richblack-900 flex
flex-col ">
        <Navbar isLoggedIn={isLoggedIn}
setIsLoggedIn={setIsLoggedIn} />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/About" element={<About />} />
          <Route path="/Contact" element={<Contact />} />
          <Route
            path="/login"
```

```

        element={<Login setIsLoggedIn={setIsLoggedIn} />}
      />
    <Route
      path="/signup"
      element={<Signup setIsLoggedIn={setIsLoggedIn} />}
    />

    <Route isLoggedIn={isLoggedIn}>
      <Route path="/dashboard" element={<Dashboard />} />
    </Route>

  </Routes>
</div>
</MuiThemeProvider>
);
}

export default App;

```

App.css

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;  
  
html,  
body {  
  overflow-x: hidden;  
}
```

Home.js

```
import logo from "../assets/1280.jpg";

function Home() {
  return (
    <div className='grid place-items-center text-richblack-100
text-3xl h-full'>
      <img src={logo} height={200} width={700} loading="lazy"
alt="Logo" />
    </div>
  );
}

export default Home;
```

Navbar.jsx

```
import React from "react";
import logo from "../assets/Logo1.png";
import { Link } from "react-router-dom";
import { toast } from "react-hot-toast";

const Navbar = (props) => {
  const isLoggedIn = props.isLoggedIn;
  const setIsLoggedIn = props.setIsLoggedIn;
  return (
    <div className="flex justify-between items-center w-11/12
max-w-[1160px] py-4 mx-auto">
      <Link to="/">
        <img src={logo} height={32} width={180} loading="lazy"
alt="Logo" />
      </Link>

      <nav>
        <ul className="flex gap-x-6 text-richblack-100">
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/About">About</Link>
          </li>
          <li>
            <Link to="/Contact">Contact</Link>
          </li>
        </ul>
      </nav>
    </div>
  );
};
```

```

    { /* Button - Login = Signup = Logout = Dashboard */ }

    <div className="flex items-center gap-x-4 text-richblack-
100">
      { !isLoggedIn && (
        <Link to="/login">
          <button className="bg-richblack-800 py-[8px] px-
[12px] rounded-[8px] border border-richblack-700">
            Login
          </button>
        </Link>
      ) }
      { !isLoggedIn && (
        <Link to="/signup">
          <button className="bg-richblack-800 py-[8px] px-
[12px] rounded-[8px] border border-richblack-700">
            Sign Up
          </button>
        </Link>
      ) }
      { isLoggedIn && (
        <Link to="/">
          <button
            onClick={() => {
              setIsLoggedIn(false);
              toast.success("Logout Sucessfully");
            }}
            className="bg-richblack-800 py-[8px] px-[12px]
rounded-[8px] border border-richblack-700"
          >
            Log Out
          </button>
        </Link>
      ) }
    </div>
  ) }
}

```

```

    })
    {isLoggedIn && (
      <Link to="/dashboard">
        <button className="bg-richblack-800 py-[8px] px-
[12px] rounded-[8px] border border-richblack-700">
          Profile
        </button>
      </Link>
    })
  </div>
</div>
);
};

export default Navbar;

```

LoginForm.jsx

```
import React from "react";
import { useState } from "react";
import { toast } from "react-hot-toast";
import { AiOutlineEyeInvisible, AiOutlineEye } from "react-
icons/ai";
import { Link, useNavigate } from "react-router-dom";

const LoginForm = (props) => {
  const setIsLoggedIn = props.setIsLoggedIn;

  const navigate = useNavigate();

  const [showPassword, setShowPassword] = useState(false);

  const [formData, setFormData] = useState({
    email: "",
    password: "",
  });

  function changeHandler(event) {
    setFormData([
      (prev) => [
        {
          ...prev,
          [event.target.name]: event.target.value,
        },
      ],
    ]);
  }
}
```



```

    }

    function submitHandler(e) {
        e.preventDefault();
        setIsLoggedIn(true);
        toast.success("Login Success");
        navigate("/dashboard");
    }

    return (
        <form
            onSubmit={submitHandler}
            className="flex flex-col w-full gap-y-4 mt-6"
        >
            <label className="w-full">
                <p className="text-[0.875rem] text-richblack-5 mb-1 leading-[1.375rem]">
                    Email Address
                    <sup className="text-pink-200">*</sup>
                </p>

                <input
                    type="email"
                    required
                    value={formData.email}
                    placeholder="Enter your email address"
                    onChange={changeHandler}
                    name="email"
                    className="bg-richblack-800 rounded-[0.75rem] w-full p-[12px] text-richblack-5"
                />
            </label>

```

```

        <label className="w-full relative">
            <p className="text-[0.875rem] text-richblack-5
mb-1 leading-[1.375rem]">
                Password
                <sup className="text-pink-200">*</sup>
            </p>

            <input
                type={showPassword ? "text" : "password"}
                required
                value={formData.password}
                placeholder="Enter Password"
                onChange={changeHandler}
                name="password"
                className="bg-richblack-800 rounded-
[0.75rem] w-full p-[12px] text-richblack-5"
            />

            <span onClick={() =>
setShowPassword(!showPassword)} className="absolute right-3 top-
[38px] cursor-pointer ">
                {showPassword ? <AiOutlineEyeInvisible
fontSize={24} fill='#AFB2BF' /> : <AiOutlineEye fontSize={24}
fill='#AFB2BF' />}
            </span>

            <Link to="#">
                <p className="text-xs mt-1 text-blue-100 max-w-
max ml-auto">Forgot Password</p>
            </Link>
        </label>

        <button className="bg-yellow-50 py-[8px] px-[12px]
rounded-[8px] mt-6 font-medium text-richblack-900">Sign in</button>
    </form>
    );
};

```

```
export default LoginForm
```

SignupForm.jsx

```
import React from "react";
import { useState } from "react";
import { toast } from "react-hot-toast";
import { AiOutlineEyeInvisible, AiOutlineEye } from "react-
icons/ai";
import { useNavigate } from "react-router-dom";

const SignupForm = (props) => {
  const setIsLoggedIn = props.setIsLoggedIn;

  const navigate = useNavigate();

  const [showCreatePass, setShowCreatePass] = useState(false);
  const [showConfirmPass, setShowConfirmPass] = useState(false);
  const [accountType, setAccountType] = useState("student");

  const [formData, setFormData] = useState({
    firstName: "",
    lastName: "",
    email: "",
    password: "",
    confirmPassword: "",
  });

  function changeHandler(event) {
    setFormData([
      (prev) => [
```

```

        {
            ...prev,
            [event.target.name]: event.target.value,
        },
    ],
]);
}

function submitHandler(event) {
    event.preventDefault();
    if(formData.password !== formData.confirmPassword) {
        toast.error("Passwords do not match");
        return;
    }

    setIsLoggedIn(true);
    toast.success("Account Create");
    const accountData = {
        ...formData,
    };
    console.log(accountData);

    navigate("/dashboard");
}

return (
    <div>
        <div className="flex bg-richblack-800 p-1 gap-x-1 rounded-
full max-w-max">
            <button
                onClick={() => setAccountType("Student")}
                className={` ${
                    accountType === "student"

```

```

        ? "bg-richblack-900 text-richblack-5"
        : "bg-transparent text-richblack-200 "
    } py-2 px-5 rounded-full transition-all`}
>
    User
</button>

<button
    onclick={() => setAccountType("instructor")}
    className={` ${
        accountType === "instructor"
            ? "bg-richblack-900 text-richblack-5"
            : "bg-transparent text-richblack-200 "
        } py-2 px-5 rounded-full transition-all`}
>
    Adminstrator
</button>
</div>

<form onSubmit={handleSubmit}>
    <div className="flex gap-x-4">
        <label htmlFor="" className="w-full">
            <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
                First Name <sup className="text-pink-200">*</sup>
            </p>
            <input
                type="text"
                required
                placeholder="Enter First Name"
                onChange={changeHandler}
                value={FormData.firstName}
                name="firstName"
            />
        </label>
    </div>
    <button type="submit" className="bg-richblack-900 text-richblack-5 py-2 px-5 rounded-full transition-all">
        Submit
    </button>
</form>

```

```

        className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
    />
</label>

```

```

<label htmlFor="" className="w-full">
    <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
        Last Name <sup className="text-pink-200">*</sup>
    </p>
    <input
        type="text"
        required
        placeholder="Enter Last Name"
        onChange={changeHandler}
        value={FormData.lastName}
        name="lastName"
        className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
    />
</label>
</div>

```

```

<label htmlFor="" className="w-full">
    <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
        Email Address
        <sup className="text-pink-200">*</sup>
    </p>

    <input
        type="email"
        required

```

```

        placeholder="Enter your email address"
        value={formData.email}
        onChange={changeHandler}
        className="bg-richblack-800 rounded-[0.75rem] w-full
p-[12px] text-richblack-5"
        name="email"
    />
</label>

```

```

<div className="flex gap-x-4">
    <label htmlFor="w-full relative">
        <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
            Create Password
            <sup className="text-pink-200">*</sup>
        </p>

```

```

        <input
            type={showCreatePass ? "text" : "password"}
            required
            placeholder="Enter Password"
            onChange={changeHandler}
            value={formData.password}
            name="password"
            className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
        />
        {/* <span
            onClick={() => setShowCreatePass(!showCreatePass)}
            className="absolute right-3 top-[38px] cursor-
pointer z-10"
        >
            {showCreatePass ? (

```



```

        <AiOutlineEyeInvisible fontSize={24}
fill="#AFB2BF" />
      ) : (
        <AiOutlineEye fontSize={24} fill="#AFB2BF" />
      )}
    </span> */}
  </label>

  <label htmlFor="" className="w-full relative">
    <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
      Confirm Password
      <sup className="text-pink-200">*</sup>
    </p>

    <input
      type={showConfirmPass ? "text" : "password"}
      required
      placeholder="Confirm Password"
      onChange={changeHandler}
      value={formData.confirmPassword}
      name="confirmPassword"
      className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
    />

    <span
      onClick={() =>
setShowConfirmPass(!showConfirmPass)}
      className="absolute right-3 top-[38px] cursor-
pointer z-10"
    >
      {showConfirmPass ? (

```

```

        <AiOutlineEyeInvisible fontSize={24}
fill="#AFB2BF" />
      ) : (
        <AiOutlineEye fontSize={24} fill="#AFB2BF" />
      )}
    </span>
  </label>
</div>

  <button className="bg-yellow-50 py-[8px] px-[12px]
rounded-[8px] mt-6 font-medium text-richblack-900 w-full">
    Create Account
  </button>
</form>
</div>
);
};

export default SignupForm;

```

Login.jsx

```
import React from 'react'
import loginImg from "../assets/login.png";
import Template from './Template';

function Login({ setIsLoggedIn }) {
  return (
    <Template
      title="Welcome Back"
      desc1="Discover the best source for royalty-free images."
      image={loginImg}
      formType="login"
      setIsLoggedIn={setIsLoggedIn}
    />
  );
}

export default Login
```

Signup.jsx

```
import React from 'react'
import signupImg from "../assets/signup.png";
import Template from './Template';

function Signup({ setIsLoggedIn }) {
  return (
    <Template
      title="Join the ArtGallery"
      desc1="To Search Unlimitd Image."
      image={signupImg}
      formType="signup"
      setIsLoggedIn={setIsLoggedIn}
    />
  );
}

export default Signup
```

Navbar.jsx

```
import React from "react";
import logo from "../assets/Logo1.png";
import { Link } from "react-router-dom";
import { toast } from "react-hot-toast";

const Navbar = (props) => {
  const isLoggedIn = props.isLoggedIn;
  const setIsLoggedIn = props.setIsLoggedIn;
  return (
    <div className="flex justify-between items-center w-11/12
max-w-[1160px] py-4 mx-auto">
      <Link to="/">
        <img src={logo} height={32} width={180} loading="lazy"
alt="Logo" />
      </Link>

      <nav>
        <ul className="flex gap-x-6 text-richblack-100">
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/About">About</Link>
          </li>
          <li>
            <Link to="/Contact">Contact</Link>
          </li>
        </ul>
      </nav>

      { /* Button - Login = Signup = Logout = Dashboard */ }
```

```

    <div className="flex items-center gap-x-4 text-richblack-
100">
      {!isLoggedIn && (
        <Link to="/login">
          <button className="bg-richblack-800 py-[8px] px-
[12px] rounded-[8px] border border-richblack-700">
            Login
          </button>
        </Link>
      )}
      {!isLoggedIn && (
        <Link to="/signup">
          <button className="bg-richblack-800 py-[8px] px-
[12px] rounded-[8px] border border-richblack-700">
            Sign Up
          </button>
        </Link>
      )}
      {isLoggedIn && (
        <Link to="/">
          <button
            onClick={() => {
              setIsLoggedIn(false);
              toast.success("Logout Sucessfully");
            }}
            className="bg-richblack-800 py-[8px] px-[12px]
rounded-[8px] border border-richblack-700"
          >
            Log Out
          </button>
        </Link>
      )}

```

```

        {isLoggedIn && (
            <Link to="/dashboard">
                <button className="bg-richblack-800 py-[8px] px-
[12px] rounded-[8px] border border-richblack-700">
                    Profile
                </button>
            </Link>
        )}
    </div>
</div>
);
};

export default Navbar;

```

SignupForm.jsx

```
import React from "react";
import { useState } from "react";
import { toast } from "react-hot-toast";
import { AiOutlineEyeInvisible, AiOutlineEye } from "react-
icons/ai";
import { useNavigate } from "react-router-dom";

const SignupForm = (props) => {
  const setIsLoggedIn = props.setIsLoggedIn;

  const navigate = useNavigate();

  const [showCreatePass, setShowCreatePass] = useState(false);
  const [showConfirmPass, setShowConfirmPass] = useState(false);
  const [accountType, setAccountType] = useState("student");

  const [formData, setFormData] = useState({
    firstName: "",
    lastName: "",
    email: "",
    password: "",
    confirmPassword: "",
  });

  function changeHandler(event) {
    setFormData([
      (prev) => [
        {

```



```

        ...prev,
        [event.target.name]: event.target.value,
      },
    ],
  ]);
}

function submitHandler(event) {
  event.preventDefault();
  if(formData.password !== formData.confirmPassword) {
    toast.error("Passwords do not match");
    return;
  }

  setIsLoggedIn(true);
  toast.success("Account Create");
  const accountData = {
    ...formData,
  };
  console.log(accountData);

  navigate("/dashboard");
}

return (
  <div>
    <div className="flex bg-richblack-800 p-1 gap-x-1 rounded-
full max-w-max">
      <button
        onclick={() => setAccountType("Student")}
        className={` ${
          accountType === "student"
            ? "bg-richblack-900 text-richblack-5"

```

```

        : "bg-transparent text-richblack-200 "
      } py-2 px-5 rounded-full transition-all`}
    >
      User
    </button>

    <button
      onClick={() => setAccountType("instructor")}
      className={` ${
        accountType === "instructor"
          ? "bg-richblack-900 text-richblack-5"
          : "bg-transparent text-richblack-200 "
        } py-2 px-5 rounded-full transition-all`}
    >
      Adminstrator
    </button>
  </div>

  <form onSubmit={handleSubmit}>
    <div className="flex gap-x-4">
      <label htmlFor="" className="w-full">
        <p className="text-[0.875rem] text-richblack-5 mb-1 leading-[1.375rem]">
          First Name <sup className="text-pink-200">*</sup>
        </p>
        <input
          type="text"
          required
          placeholder="Enter First Name"
          onChange={changeHandler}
          value={FormData.firstName}
          name="firstName"

```

```

        className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
    />
</label>

```

```

<label htmlFor="" className="w-full">
    <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
        Last Name <sup className="text-pink-200">*</sup>
    </p>
    <input
        type="text"
        required
        placeholder="Enter Last Name"
        onChange={changeHandler}
        value={FormData.lastName}
        name="lastName"
        className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
    />
</label>
</div>

```

```

<label htmlFor="" className="w-full">
    <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
        Email Address
        <sup className="text-pink-200">*</sup>
    </p>

    <input
        type="email"
        required

```

```

        placeholder="Enter your email address"
        value={formData.email}
        onChange={changeHandler}
        className="bg-richblack-800 rounded-[0.75rem] w-full
p-[12px] text-richblack-5"
        name="email"
    />
</label>

```

```

<div className="flex gap-x-4">
    <label htmlFor="w-full relative">
        <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
            Create Password
            <sup className="text-pink-200">*</sup>
        </p>

```

```

        <input
            type={showCreatePass ? "text" : "password"}
            required
            placeholder="Enter Password"
            onChange={changeHandler}
            value={formData.password}
            name="password"
            className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
        />
        {/* <span
            onClick={() => setShowCreatePass(!showCreatePass)}
            className="absolute right-3 top-[38px] cursor-
pointer z-10"
        >
            {showCreatePass ? (

```

```

        <AiOutlineEyeInvisible fontSize={24}
fill="#AFB2BF" />
      ) : (
        <AiOutlineEye fontSize={24} fill="#AFB2BF" />
      )}
    </span> */}
  </label>

  <label htmlFor="" className="w-full relative">
    <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
      Confirm Password
      <sup className="text-pink-200">*</sup>
    </p>

    <input
      type={showConfirmPass ? "text" : "password"}
      required
      placeholder="Confirm Password"
      onChange={changeHandler}
      value={formData.confirmPassword}
      name="confirmPassword"
      className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
    />

    <span
      onClick={() =>
setShowConfirmPass(!showConfirmPass)}
      className="absolute right-3 top-[38px] cursor-
pointer z-10"
    >
      {showConfirmPass ? (

```

```

        <AiOutlineEyeInvisible fontSize={24}
fill="#AFB2BF" />
      ) : (
        <AiOutlineEye fontSize={24} fill="#AFB2BF" />
      )}
    </span>
  </label>
</div>

  <button className="bg-yellow-50 py-[8px] px-[12px]
rounded-[8px] mt-6 font-medium text-richblack-900 w-full">
    Create Account
  </button>
</form>
</div>
);
};

export default SignupForm;

```

ImageResults.jsx

```
import React,{Component} from 'react';
import PropTypes from 'prop-types';
import {GridList,GridTile} from 'material-ui/GridList';
import IconButton from 'material-ui/IconButton';
import ZoomIn from 'material-ui/svg-icons/action/zoom-in';
import Dialog from 'material-ui/Dialog';
import FlatButton from 'material-ui/FlatButton';

class ImageResults extends Component{
  state={
    open:false,
    currentImg:''
  }
  handleOpen=img=>{
    this.setState({open:true,currentImg:img})
  }
  handleClose={()=>{
    this.setState({open:false});
  }}
  render()
  {
    let imageList;
    const {images}=this.props

    if(images)
    {
      imageList=(
        <GridList cols={4}>
          { images.map(img=>(
            <GridTile
```

```

        title={img.tags}
        key={img.id}
        actionIcon={
            <IconButton
onClick={(()=>this.handleOpen(img.largeImageURL))>
            <ZoomIn color="white" />
            </IconButton>
        }
    >
    <img src={img.largeImageURL} alt="" />
    </GridTile>
    ))
    }
    </GridList>
    )
}
else{
    imageList=null;
}
const actions=[
    <FlatButton label="Close" primary={true}
onClick={this.handleClose}/>
]
return(
    <div
style={{marginLeft:50,marginRight:50,marginTop:20}}>
        {imageList}
        <Dialog
            actions={actions}
            modal={false}
            open={this.state.open}
            onRequestClose={this.handleClose}
        >

```



```

        <img src={this.state.currentImg} alt=""
style={{width:'100%'}} />
      </Dialog>
    </div>
  )
}
}
ImageResults.propTypes={
  images:PropTypes.array.isRequired
}

export default ImageResults;

```

Template.jsx

```
import React from "react";
import frame from "../assets/frame.png";
import SignupForm from "./SignupForm.jsx";
import LoginForm from "./LoginForm.jsx";
import { FcGoogle } from "react-icons/fc";

const Template = ({ title, desc1, desc2, image, formType,
setIsLoggedIn }) => {
  return (
    <div className="flex w-11/12 max-w-[1160px] py-12 mx-auto
gap-y-0 gap-x-12 justify-between">
      <div className="w-11/12 max-w-[450px] mx-0 text-white">
        <h1 className="text-richblack-5 font-semibold text-
[1.875rem] leading-[2.375rem]">{title}</h1>
        <p className="text-[1.125rem] mt-4 leading-[1.625rem]">
          <span className="text-richblack-100">{desc1}</span>
          <span className="text-blue-100 italic">{desc2}</span>
        </p>

        {formType === "signup" ? <SignupForm
setIsLoggedIn={setIsLoggedIn} /> : <LoginForm
setIsLoggedIn={setIsLoggedIn} />}

      <div className="flex w-full items-center my-4 gap-x-2">
        <div className="h-[1px] w-full bg-richblack-
700"></div>
        <p className="text-richblack-700 font-medium leading-
[1.375rem]">OR</p>
    </div>
```

```

        <div className="h-[1px] w-full bg-richblack-700"></div>
      </div>

      <button className="w-full flex items-center justify-center rounded-[8px] font-medium text-richblack-100 border-richblack-700 border px-[12px] py-[8px] gap-x-2 mt-6">
        <FcGoogle />
        <p>Sign Up with Google</p>
      </button>
    </div>

    <div className="relative w-11/12 max-w-[450px]">
      <img
        src={frame}
        alt="patter"
        width={558}
        height={504}
        loading="lazy"

      />
      <img
        src={image}
        alt="patter"
        width={558}
        height={504}
        loading="lazy"
        className="absolute -top-4 right-4 "
      />
    </div>
  </div>
);
};

```

```
export default Template;
```

Search.jsx

```
import React,{Component} from 'react';
import axios from 'axios';
import ImageResults from "../pages/imageResults";
class Search extends Component{
  state={
    searchText:'',
    apiUrl:'https://pixabay.com/api',
    apiKey:'17241914-90da7b93c0ccceb734849dcd1',
    images:[]
  };
  onTextChange=(e)=>{
    const val=e.target.value;
    this.setState({[e.target.name]:val},()=>{
      if(val==='')
      {
        this.setState({images:[]});
      }
      else{
        axios
          .get(
            `${this.state.apiUrl}?key=${this.state.apiKey}&q=${
              this.state.searchText
            }&image_type=photo&safesearch=true`
          )
          .then(res=>this.setState({images:res.data.hits}))
          .catch(err=>console.log(err));
      }
    });
  };
```

```

};
render(){
  console.log(this.state.images);
  return(
    <div>
      <input type="text"
        style=
        {{backgroundColor: 'white',
        color:
        'Black',
        marginLeft:50,
        marginTop:50,
        paddingTop:0,
        paddingLeft:70,
        fontSize:30,
        borderTopStyle:"hidden",
        borderRightStyle:"hidden",
        borderLeftStyle:"hidden",
        outline:"none",
        borderBottomStyle:"groove",
      }}
      placeholder="Search for images"
      name="searchText"
      value={this.state.searchText}
      onChange={this.onTextChange}
    />
    <br />
    {this.state.images.length>0?(<ImageResults
      images={this.state.images}/>):null}
    </div>
  )
}

```

```
}
```

```
export default Search;
```

ImageResults.jsx

```
import React,{Component} from 'react';
import PropTypes from 'prop-types';
import {GridList,GridTile} from 'material-ui/GridList';
import IconButton from 'material-ui/IconButton';
import ZoomIn from 'material-ui/svg-icons/action/zoom-in';
import Dialog from 'material-ui/Dialog';
import FlatButton from 'material-ui/FlatButton';
class ImageResults extends Component{
  state={
    open:false,
    currentImg:''
  }
  handleOpen=img=>{
    this.setState({open:true,currentImg:img})
  }
  handleClose=()=>{
    this.setState({open:false});
  }
  render()
  {
    let imageList;
    const {images}=this.props

    if(images)
    {
      imageList=(
        <GridList cols={4}>
          { images.map(img=>(
            <GridTile
```



```

        title={img.tags}
        key={img.id}
        actionIcon={
            <IconButton
onClick={(()=>this.handleOpen(img.largeImageURL))>
            <ZoomIn color="white" />
            </IconButton>
        }
    >
    <img src={img.largeImageURL} alt="" />
    </GridTile>
    ))
    }
    </GridList>
    )
}
else{
    imageList=null;
}
const actions=[
    <FlatButton label="Close" primary={true}
onClick={this.handleClose}/>
]
return(
    <div
style={{marginLeft:50,marginRight:50,marginTop:20}}>
        {imageList}
        <Dialog
            actions={actions}
            modal={false}
            open={this.state.open}
            onRequestClose={this.handleClose}
        >

```

```

        <img src={this.state.currentImg} alt=""
style={{width:'100%'}} />
      </Dialog>
    </div>
  )
}
}
ImageResults.propTypes={
  images:PropTypes.array.isRequired
}

export default ImageResults;

```

CHAPTER 10

TESTING

9.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionalities of components, sub-assemblies, and/or a finished product it is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

9.2 Types of Testing

9.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have is white box oriented and some modules the steps are conducted in parallel.

9.2.2 Integration Testing

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus, the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

9.2.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT

- Further enhancements of this project will be user can also upload Image
- Another enhancement would be user give feedback.

CONCLUSION & REFERNCES

In conclusion, developing an image search application using React and integrating it with an API offers a powerful and user-friendly solution for retrieving and displaying images. React's component-based architecture enables a modular and organized development process, allowing for the creation of a dynamic and responsive user interface.

Utilizing an API for image retrieval enhances the application's capabilities, providing access to vast databases of images and ensuring real-time updates. This integration allows for a seamless user experience, with the application fetching relevant images based on user queries.

The use of React's state management and lifecycle methods facilitates efficient handling of data and UI updates, contributing to a smooth and interactive experience for the end-users. React's virtual DOM enhances performance by minimizing unnecessary re-rendering, ensuring that the application remains fast and responsive even when dealing with large sets of image data.

References

Coding phase: -

1. React

Referenced Sites:

www.w3school.com

www. <https://react.dev/learn>