# PicStream

## A Mini Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of

**MASTER OF COMPUTER APPLICATIONS**

**By**

**ROHIT KUMAR**

**( University Roll No. 2300290140148 )**

## RUPESH GUPTA

**( University Roll No. 2300290140150 )**

**Under the Supervision of**

**Dr. VIPIN KUMAR**

**Associate Professor**

**KIET Group of Institutions**



**Submitted to**

### DEPARTMENT OF COMPUTER APPLICATIONS

### KIET Group of Institutions, Ghaziabad

**Uttar Pradesh-201206**

**August 2024**

# DECLARATION

We hereby declare that the work presented in this report entitled **“Random Image Generator"**, was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, We shall be fully responsible and answerable.

**Rohit Kumar** (2300290140148)

**Rupesh Gupta** (2300290140150)

## ACKNOWLEDGEMENT

At the outset, we would like to thank our guide and advisor**, Dr. VIPIN KUMAR Associate Professor,** for giving us an opportunity to work on this challenging topic and providing us ample and valuable guidance throughout the Project.

Without his encouragement and constant guidance, we would not have able to finish this project. He has been always a source of inspiration and motivator for innovative ideas during the entire span of this work.

We are grateful **Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, KIET Group of Institutions, Ghaziabad**for providing all the necessary resources to carry out this Project work.

We will be failing in our duty if we don’t acknowledge the people behind this work to give us moral and psychological support. Our special thanks to our parents for their endless care and constant support.

**Rohit Kumar** (2300290140148)  **Rupesh Gupta (**2300290140150)

**CERTIFICATE**

Certified that **Rohit Kumar (2300290140148), Rupesh Gupta (2300290140150)** have carried out the project work having **“Random Image Generator”** for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself 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: Rohit Kumar (2300290140148)**

### Rupesh Gupta (2300290140150)

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

### Date: Dr. VIPIN KUMAR

### Associate Professor

### Department of Computer Applications

**KIET Group of Institutions, Ghaziabad**

**Signature of Internal Examiner Signature of External Examiner**

**Dr. Arun Kumar Tripathi**

**Head, Department of Computer Application**

**KIET Group of Institutions, Ghaziabad**

# ABSTRACT

Our Project Random Image Generator is an API based Image Generator, which is designed and kept under the huge scope of the audience i.e. students and faculty of KIET Group of Institute.

The **Random Image Generator** project aims to create a dynamic system that generates and displays random images, catering to various use cases such as inspiration, entertainment, and education. This application utilizes modern web technologies and APIs to fetch or create images on-demand, ensuring a seamless user experience.

The system provides functionalities for generating random images based on specific categories, themes, or completely random selections. Users can customize parameters like size, format, and filters to tailor the generated images to their needs. The application leverages efficient algorithms and integration with image-generation libraries or APIs to ensure high-quality results.

This project showcases skills in front-end and back-end development, API integration, and user interface design. It is scalable, making it suitable for applications in design tools, creative projects, or simply as a fun tool for random visual content exploration.

Through this project, we demonstrate the ability to blend creativity with technology to build a functional and engaging platform.

**TABLE OF CONTENT**

Chapter 1: Introduction 9

1.1 Project Description 9

1.2Project Scope 9

1.3 Future Scope 10

1.4Identification of need 10

1.5Problem Statement 10

1.6Software/Technology used in project 11

1.6.1 Non-Functional Requirement 11

1.6.2Functional Requirement 11

1.7Project schedule 12

1.7.1Pert Chart 12

1.7.2Gantt Chart 12

Chapter 2: Feasibility Study 13

2.1Introduction 13

2.2Main Aspects 13

2.2.1Technical feasibility 13

2.2.2Economical Feasibility 13

2.2.3Operational Feasibility 13

2.3Benefits 14

2.4SRS 14

Chapter 3:Design

|  |  |
| --- | --- |
| 3.1Introduction | 15 |
| 3.2Analysis | 15 |
| 3.3SDLC | 16 |
| 3.4Soft. Engg. Paradigm | 19 |
| 3.5Architecture of the system | 19 |
| 3.6Control Flow graph | 20 |

Chapter 4: Report

4.1Gist 21

4.2Some Snippets 21

4.2.1App 21

4.2.2 API Integeration 22

4.2.3Backend 24

Chapter 5: Coding 27

Chapter 6: Testing 84

6.1Introduction 84

6.1.1 Testing Objectives

6.1.2 Testing Principles

6.2Level of testing 85

6.2.1 Unit Testing

6.2.2 Integration Testing 85

6.2.3 System Testing 86

Chapter 7: Conclusion & Future scope

|  |  |
| --- | --- |
| 7.1Conclusion | 87 |
| 7.2Future Scope | 87 |
| 7.3 Bibliography | 89 |

**LIST OF FIGURES**

 **Figure 3.1: SDLC 16**

 **Figure 4.1: Login Page 22**

 **Figure 4.2: User Dashboard 23**

 **Figure 4.3: Generated Image Preview 24**

 **Figure 4.4: Generated Image 25**

 **Figure 4.5: About 26**

 **Figure6.1: Testing Pyramid 85**

**CHAPTER 1**

# INTRODUCTION

* 1. **Project Description**

In today's digital era, images play a significant role in enhancing communication, design, and information sharing. A Random Image Generator is a software application designed to fetch and display images randomly using APIs. These images can serve a variety of purposes, such as providing inspiration, serving as placeholders in web development, or simply offering an engaging and interactive experience for users.

The Random Image Generator uses modern web technologies like React.js for the front-end interface and Node.js for API integration. The application fetches high-quality random images from popular sources such as the Unsplash API. Users can specify categories, filters, or let the application surprise them with random visuals. This project aims to combine seamless user experience with robust backend communication, ensuring real-time fetching of images while maintaining flexibility and efficiency.

* 1. **Project Scope**

The scope of the Random Image Generator project includes:

1. Developing a responsive and interactive user interface using React.js.
2. Enabling users to fetch random images with minimal input or select categories for specific themes.
3. Integrating with a reliable image API such as Unsplash to ensure a large repository of high-quality visuals.
4. Providing options for users to download images or save preferences for future sessions.
5. Ensuring cross-platform compatibility to cater to desktop and mobile users.

Limitations:

* The application currently focuses only on image fetching and does not include additional features like editing or video generation.
* Offline functionality is not supported, as the app relies on real-time API requests.
  1. **Future Scope**

The following features are planned for future enhancements:

1. Authorization Services: Introduce user authentication for personalized experiences.
2. Database Integration: Save user preferences, histories, or favorites using services like Firebase.
3. Image Editing Options: Add simple editing capabilities like cropping, resizing, or applying filters.
4. Voice and Gesture Commands: Enable interaction using voice or gesture-based inputs.
5. Web and Mobile App Support: Extend the application to work seamlessly as a Progressive Web App (PWA) and native mobile application.
   1. **Identification of Need**

The internet is flooded with generic stock photos, making it challenging for users to find creative and diverse visuals. Developers and designers often require placeholder images or creative inspiration without spending excessive time searching. Furthermore, privacy concerns with certain image platforms can deter users from exploring their full potential.

Our solution offers:

* Randomized Creativity: Users get random images instantly, aiding creativity and saving time.
* End-to-End Privacy: Unlike other platforms, the Random Image Generator does not collect unnecessary user data.
* Ease of Use: A simple, user-friendly interface ensures accessibility for all user types.

This tool bridges the gap between creative demands and usability, offering a secure and efficient platform for image exploration.

* 1. **Problem Statement**

In creative projects and development processes, sourcing images can be time-consuming, repetitive, or limited by copyright constraints. Tools that lack variety or personalization often frustrate users.

The Random Image Generator addresses these problems by:

* Integrating with high-quality APIs like Unsplash to ensure vast and diverse image resources.
* Maintaining user privacy by avoiding unnecessary data collection or sharing.
* Offering a lightweight, intuitive, and responsive platform for hassle-free image fetching.

Key features include:

* Real-Time Fetching: High-speed, random image generation from robust sources.
* Customizability: Options for filtering based on user-defined categories.
* Enhanced Privacy: Ensuring end-to-end data security and minimal user information storage.
  1. **Software/Technology Used in Project**

A. React.js

React.js is a JavaScript library used for building user interfaces. It enables efficient and dynamic rendering of components, making it ideal for developing interactive applications.

B. Node.js

Node.js provides a robust backend runtime environment to handle API requests and real-time data fetching. It supports asynchronous programming, ensuring faster response times.

C. Unsplash API

The Unsplash API offers access to a vast collection of high-quality images. By integrating this API, the application ensures seamless random image fetching.

**1.6.1 Non-Functional Requirements**

The application must meet the following requirements:

1. Scalability: Capable of handling increasing API requests as user demand grows.
2. Portability: Usable across various devices and browsers without performance degradation.
3. Security: Ensures API keys and user data (if stored) are securely managed.
4. Reliability: Stable performance under different network conditions.
5. Maintainability: Code modularity for easier debugging and enhancements.

**1.6.2 Functional Requirements**

1. Image Fetching: Retrieve images based on user preferences or randomly.
2. Category Selection: Allow users to filter images by theme (e.g., nature, architecture).
3. Download Option: Provide users with the ability to download fetched images.
4. Error Handling: Notify users in case of API request failures or other errors.
5. Responsive Design: Ensure a seamless experience on all devices.

**1.7 Project Schedule**

Project planning is crucial for timely delivery and success. The timeline includes milestones for design, development, testing, and deployment.

**1.7.1 PERT Chart**

The PERT Chart helps evaluate the critical path and timeline for the project, ensuring efficient resource management.

**1.7.2 Gantt Chart**

The Gantt Chart provides a visual representation of the project schedule, detailing the duration and sequence of tasks.

**CHAPTER 2**

**Feasibility Study for Random Image Generator**

**2.1 Introduction**

The Random Image Generator system aims to provide users with the ability to generate unique and creative images using specified parameters or themes. The feasibility of such a system lies in leveraging advanced algorithms and APIs, ensuring secure and efficient operations, and providing a user-friendly interface. This document evaluates the technical, operational, and economic aspects of the project to ensure its success.

**2.2 Main Aspects of Feasibility**

Technical Feasibility

The project requires integration of advanced image-generation APIs (e.g., DALL·E or similar AI tools), robust server architecture, and a responsive user interface. Current technical resources, such as cloud computing platforms and programming frameworks (e.g., Python, JavaScript, Firebase), can easily support these requirements.

Operational Feasibility

The system will offer user-friendly options for generating images, selecting themes, and downloading results. End-users will include designers, educators, and enthusiasts, with minimal training required to operate the system. Staff will undergo basic training to maintain and troubleshoot the system.

Economic Feasibility

The system's cost-effectiveness lies in its use of open-source tools and pay-as-you-go cloud services, minimizing initial investment. The anticipated user base and subscription model ensure profitability.

**2.3 Benefits**

* Enhanced Creativity: Helps users visualize ideas effortlessly.
* Broad Accessibility: Allows use across devices and platforms.
* Secure System: Protects user inputs and generated outputs.
* Innovation Opportunity: Opens new avenues for AI-assisted design tools.

**2.4 System Requirements Specification**

Gathered Requirements

1. **Functional Requirements**
   * Users can input themes or keywords to generate images.
   * Save, download, and share generated images.
   * Support for multiple output formats (e.g., PNG, JPG).
2. **Non-Functional Requirements**
   * Scalability: Can handle multiple users simultaneously.
   * Security: Ensures user data and inputs are encrypted.
   * Maintainability: Easily updated with new features.

**2.5 Methods of Requirement Gathering**

* Document Review: Analyzed the current image generation tools to identify gaps.
* Interviews: Conducted with target users to understand their needs.
* Questionnaires: Distributed electronically to gather feedback on desired features.
* Observation: Studied user interactions with existing tools to improve interface design.

**CHAPTER 3**

# DESIGN

**3.1 INTRODUCTION**

The Random Image Generator system is designed to generate unique and creative images based on user input, such as themes, keywords, or preferences. The design phase focuses on outlining the system’s architecture, workflows, and technical aspects to ensure efficient implementation.

The primary goal is to create a user-friendly and technically robust application that leverages AI-based APIs for generating images. The design will incorporate all system requirements identified during the analysis phase and translate them into implementable system components**.**

**3.2 SYSTEM ANALYSIS**

Information Gathering

* Organization Needs: Gathered insights from users on current creative tools and identified gaps, such as limited customization or slow processing times.
* System Requirements: Collected functional and non-functional needs like scalability, real-time image rendering, and user-friendly interfaces.
* Tools Used: Document review, user interviews, and electronic data gathering using surveys.

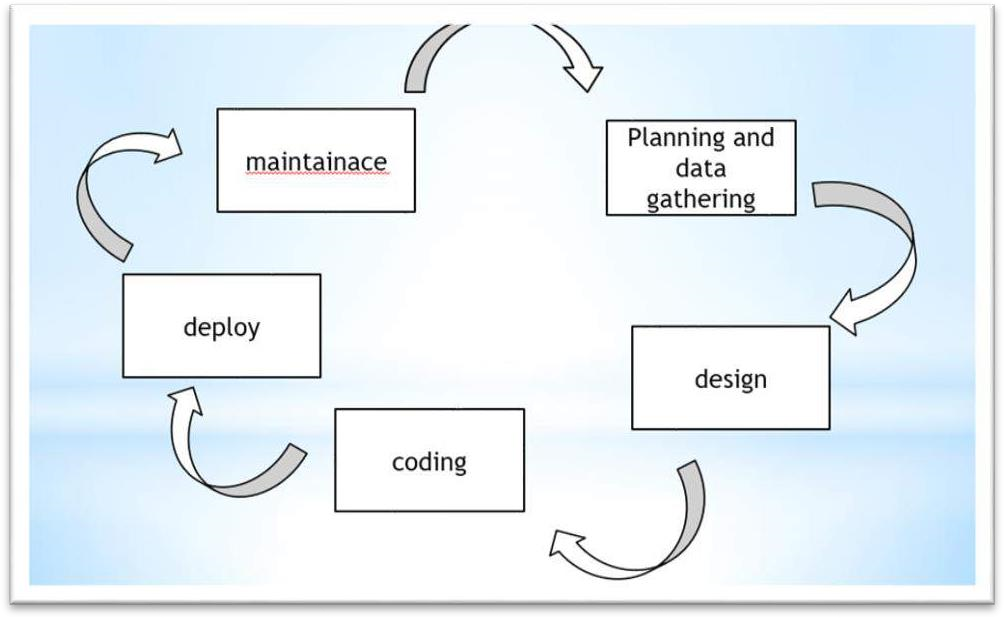
Preliminary Investigation

1. Request Clarification: Understand user requirements, e.g., themes, filters, and resolution options.
2. Feasibility Study: Confirmed the technical feasibility of integrating AI APIs for image generation.
3. Request Approval: Established that the system aligns with both technical and economic constraints.

**3.3 SDLC**

The Random Image Generator system follows the Waterfall Model, ensuring sequential development with clearly defined phases:

1. Requirement Gathering: Defined user input parameters like themes and filters.
2. Design: Developed architecture and flow diagrams for data and process handling.
3. Implementation: Coding to integrate user interfaces and AI-based image generation tools.
4. Testing: Ensured the system generates images as expected and meets performance benchmarks.
5. Deployment: Rolled out the system with support for web and mobile platforms.
6. Maintenance: Regular updates to enhance features and fix issues.



***Figure 3.1: Above image depicting the planning step***

### SDLC Phases

**Given below are the various phases:**

* Requirement gathering and analysis
* Design
* Implementation or coding
* Testing
* Deployment
* Maintenance
  1. **Requirement Gathering and Analysis**

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.

Business analyst and Project Manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.

**For Example,** A customer wants to have an application which involves money transactions. In this case, the requirement has to be clear like what kind of transactions will be done, how it will be done, in which currency it will be done, etc.

Once the requirement gathering is done, an analysis is done to check the feasibility of the development of a product. In case of any ambiguity, a call is set up for further discussion.

Once the requirement is clearly understood, the SRS (Software Requirement Specification) document is created. This document should be thoroughly understood by the developers and also should be reviewed by the customer for future reference.

1. **Design**

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

1. **Implementation or Coding**

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

1. **Testing**

Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

Retesting, regression testing is done until the point at which the software is as per the customer’s expectation. Testers refer SRS document to make sure that the software is as per the customer’s standard.

1. **Deployment**

Once the product is tested, it is deployed in the production environment or first [UAT (UserAcceptancetesting) i](https://www.softwaretestinghelp.com/what-is-user-acceptance-testing-uat/)s done depending on the customer expectation.

In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

1. **Maintenance**

After the deployment of a product on the production environment, maintenance of the product i.e., if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

## 3.4 SOFTWARE ENGG. PARADIGM APPLIED

Software engineering is a layered technology. The foundation for software engineering is the process layer. Software engineering processes the glue that holds the technology layers together and enables ratios and timely development of computer software. Process defines a framework for a set of key process areas that must be established for effective delivery of software engineering technology.

Software engineering methods provide the technical how-top’s for building software.

Methods encompass a broad array of tasks that include requirements analysis, design, program construction, testing and support. Software engineering tools provide automated or semi- automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another tool, a system for the support of software development, called computer-aided software engineering is established.

**The following paradigms are available:**

1. The Waterfall Model
2. The Prototyping Model
3. The Spiral model Etc.

## 3.5 ARCHIETECTURE OF THE SYSTEM

**System Architecture Overview**

The system is divided into the following layers:

1. **User Interface Layer**: Provides the front-end for users to input parameters and view results.
2. **Application Layer**: Handles user requests, processes inputs, and communicates with the API.
3. **AI Image Generation Layer**: Utilizes third-party AI APIs (e.g., DALL·E) to generate images.
4. **Storage Layer**: Stores user preferences and generated images temporarily for download.

## 3.6 CONTROL FLOW GRAPH

**Control Flow Explanation**

1. **User Input Node**: User provides input parameters.
2. **Input Validation Node**: Validates inputs like keywords and resolution.
3. **Image Generation Node**: Sends requests to AI-based APIs for image creation.
4. **Response Handling Node**: Receives generated images.
5. **Output Node**: Displays the images for download or further use.

**3.7 DIAGRAMS**

1. **System Architecture Diagram**: Graphical representation showing the relationship between different system components.
2. **Control Flow Diagram (CFD)**: Graph showing the flow of operations within the system.
3. **Data Flow Diagram (DFD)**: Represents data movement between modules (optional).

**CHAPTER 4**

**Report**

**4.1 GIST**

We have designed an Android-based Random Image Generator application to create unique images based on user-provided themes or keywords. This application leverages artificial intelligence for image generation, ensuring creativity and personalization in every result.

The application prioritizes user convenience and data security. Users can generate images, save them, and share them with others. The generated images are stored securely in cloud storage, ensuring easy accessibility and efficient use of resources.

For backend operations, the application utilizes Google Firebase, including authentication and Realtime Database. Firebase authentication ensures secure access for users through their Google accounts, and Realtime Database enables real-time saving and retrieval of user preferences and generated images.

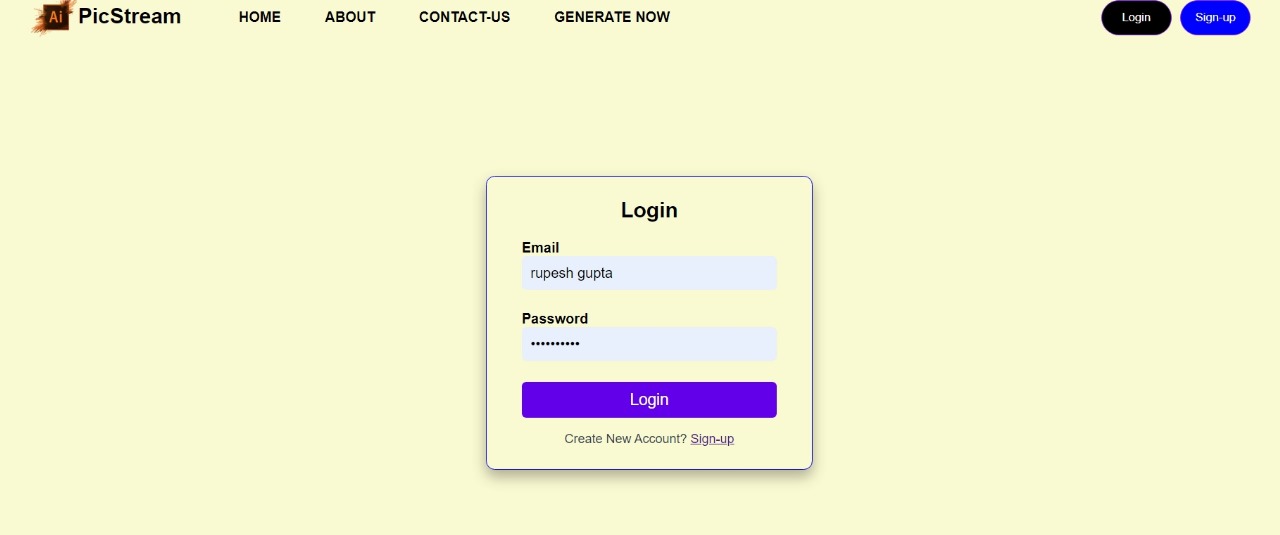
Users can also set preferences like themes, resolutions, and styles for the images. The application incorporates dynamic features such as:

* Real-time AI-based image generation.
* Options to save or share generated images.
* Secure storage for user-generated content.
* A user-friendly interface that caters to creativity and ease of use**.**

**4.2 SOME SNIPPETS**

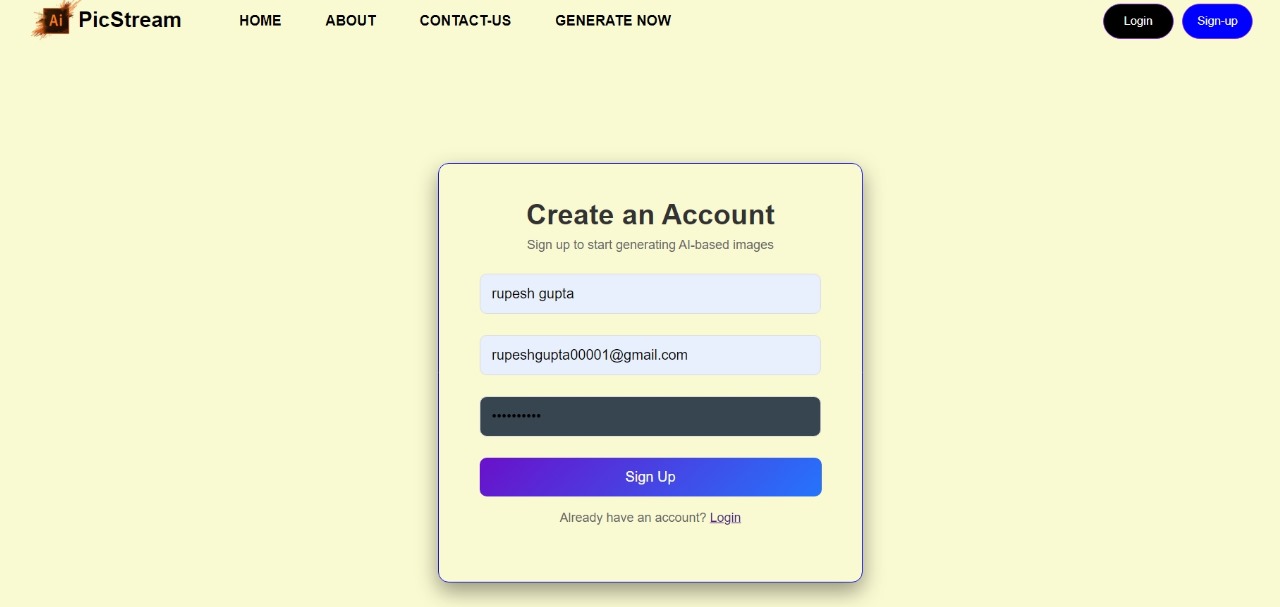
**4.2.1 APPLICATION**

Step 1: Login Page  
 The first page of the Random Image Generator application where the user logs in using Google authentication.



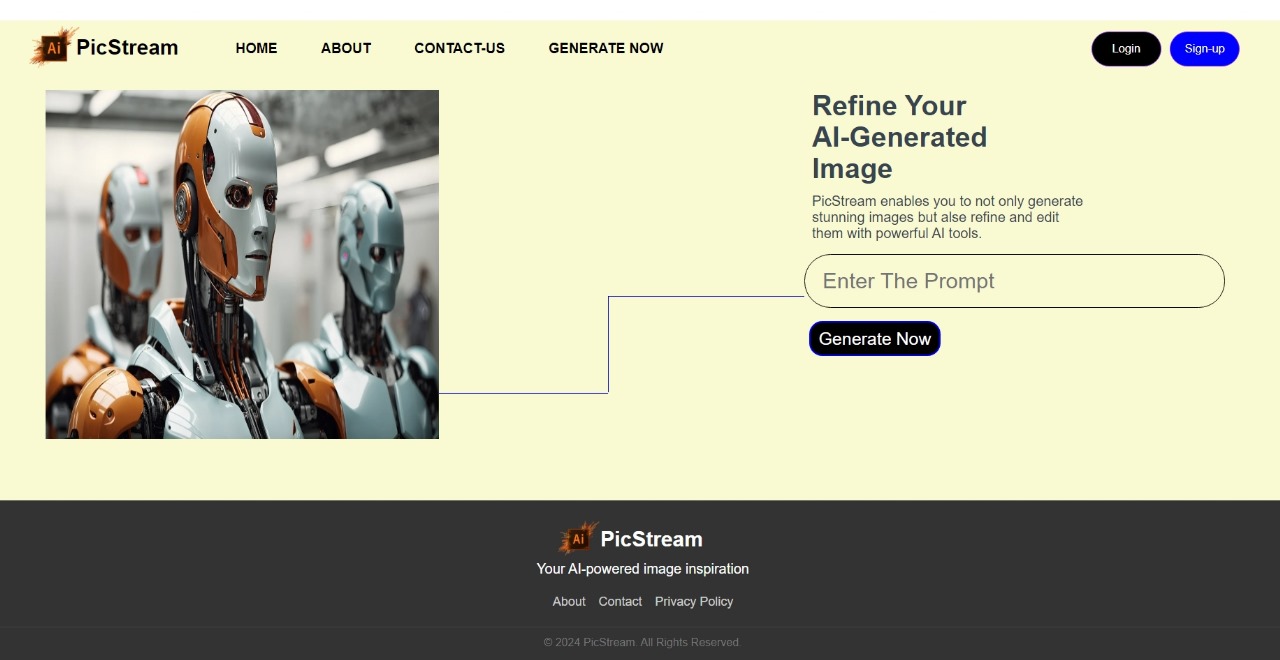
***Figure 4.1: Login Page***

**Step 2: User Dashboard** After login, the user accesses the dashboard, where they can input themes or keywords and customize image generation settings.



***Figure 4.2: User Dashboard***

**Step 3: Image Generation** Users can input parameters and generate images. They can view previews and either save or share the image.



***Figure 4.3: Generated Image Preview***

**4.2.2 FIREBASE**

Firebase Database  
 All data, including user preferences and generated images, is stored as JSON objects in Firebase Realtime Database. This ensures efficient data handling and accessibility**.**

Firebase Authentication  
 Firebase Authentication securely manages user login credentials, ensuring user identity verification through Google accounts.

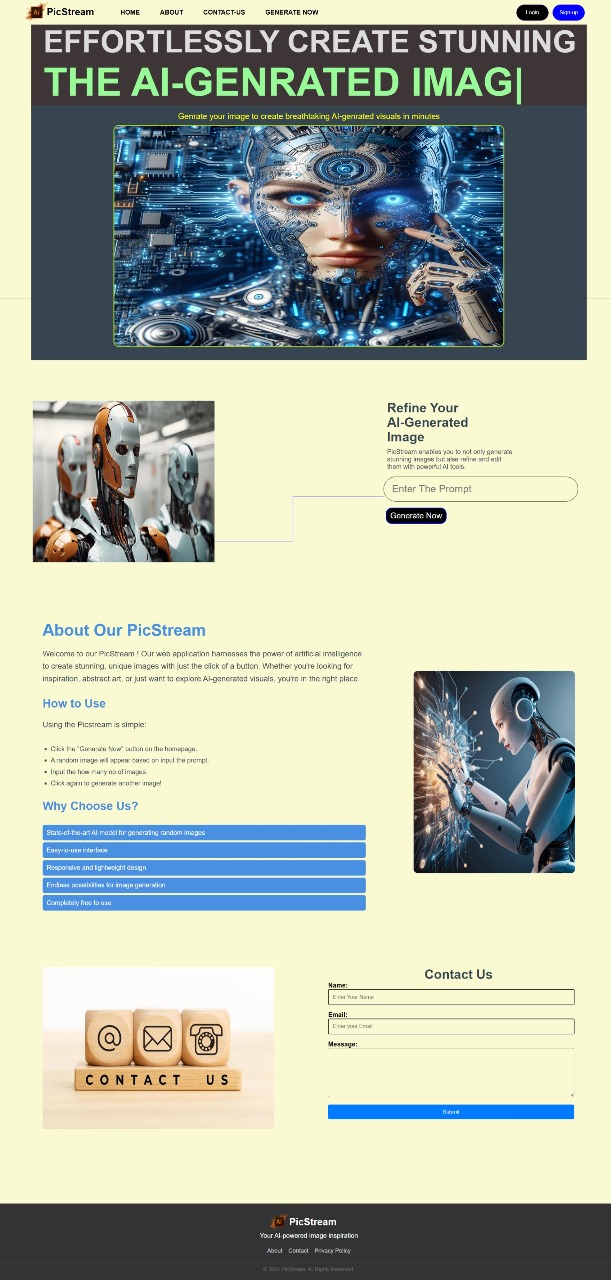


***Figure 4.4: Generated Image***

**4.2.3 ANDROID STUDIO**

Development Environment  
 The application was developed in Android Studio, utilizing its robust tools for coding, testing, and debugging. Below are some key components:

1. Toolbar: Provides options to run the app, debug, and access Android tools.
2. Navigation Bar: Helps traverse through project files.
3. Editor Window: Used for coding and designing application layouts.
4. Tool Window Bar: Displays expandable windows for logs, project structures, and more.



***Figure 4.5: About***

**CHAPTER 5**

# CODING

This chapter contains some codes of the project.The goal of the coding is to translate the design of the system into code in a given programming language. For a given design, the aim of this phase is to implement the design in the best possible manner. The coding phase affects both testing and maintenance profoundly.

**Some Codes are as Written below:**

## MainActivity

The main activity code is a MainActivity for picStream. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application. Following is the code generated of our Random Image Generator:

import React from 'react'

import './App.css'

import Navbar from './Navbar'

import Prompt from './Prompt'

import Header from './Header'

import Footer from './Footer'

import Contact from './Contact'

import About from './About'

import Policy from './Policy'

import Login from './Login'

import Sign from './Sign'

import Pagenotfound from './pagenotfound'

import { BrowserRouter,Routes,Route } from 'react-router-dom'

import { Toaster } from 'react-hot-toast'

import PrivacyPolicy from './Policy'

const App = () => {

return (

<>

<BrowserRouter>

<Navbar/>

<Routes>

<Route path='/' element={<><Header/><Prompt/><About/><Contact/></>}/>

<Route path='/About' element={<About/>}/>

<Route path='/Contact' element={<Contact/>}/>

<Route path='/Prompt' element={<Prompt/>}/>

<Route path='/Footer' element={<Footer/>}/>

<Route path='/Policy' element={<Policy/>}/>

<Route path='/Login' element={<Login/>}/>

<Route path='/Sign' element={<Sign/>}/>

<Route path='/\*' element={<Pagenotfound/>}/>

</Routes>

<Footer/>

</BrowserRouter>

<Toaster/>

</>

)

}

export default App

import React from 'react'

import { Link ,useNavigate} from 'react-router-dom'

const Navbar = () => {

let data=localStorage.getItem('user')

let navigate=useNavigate()

let logout=()=>{

localStorage.clear()

navigate('/Sign')

}

return (

<>

<div className="main">

<img src="logo.png" alt="" />

<h2 className='logo'>PicStream</h2>

<ul className='item'>

<li><Link to='/'>HOME</Link></li>

<li><Link to='/About'>ABOUT</Link></li>

<li><Link to='/Contact'>CONTACT-US</Link></li>

<li><Link to='/Prompt'>GENERATE NOW</Link></li>

</ul>

<div className="right">

{

data?<Link onClick={logout} to='/Sign'><button>Logout</button></Link>:

<><Link to='/Login'><button>Login</button></Link><Link to='/Sign'><button id='signupbtn'>Sign-up</button></Link></>

}

{/\* <Link to='/Login'><button>Login</button></Link>

<Link to='/Sign'><button>Sign-up</button></Link> \*/}

</div>

</div>

</>

)

}

export default Navbar

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

import "./Sign.css";

import { Link, useNavigate } from "react-router-dom";

function SignUp() {

const [username, setUsername] = useState("");

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

let navigate=useNavigate()

let cancelname=useRef("")

let cancelemail=useRef("")

let cancelpswd=useRef("")

const handleSubmit = (e) => {

e.preventDefault();

cancelname.current.value=""

cancelemail.current.value=""

cancelpswd.current.value=""

collectdata()

};

let collectdata=async()=>{

let result= await fetch("http://localhost:5000/register",{

method: 'post',

body: JSON.stringify({username,email,password}),

headers:{

'Content-Type':'application/json'

},

})

let data= await result.json()

localStorage.setItem("user",JSON.stringify(data))

navigate('/')

}

useEffect(()=>{

let data=localStorage.getItem('user')

if(data){

navigate('/')

}

})

return (

<div className="mainsign">

<div className="signup-page">

<div className="signup-container">

<h2>Create an Account</h2>

<p>Sign up to start generating AI-based images</p>

<form onSubmit={handleSubmit}>

<div className="input-group">

<input

type="text"

placeholder="Enter Username"

value={username}

onChange={(e) => setUsername(e.target.value)}

required

ref={cancelname}

/>

</div>

<div className="input-group">

<input

type="email"

placeholder="Enter your Email"

value={email}

onChange={(e) => setEmail(e.target.value)}

required

ref={cancelemail}

/>

</div>

<div className="input-group">

<input

type="password"

placeholder="Enter your Password"

value={password}

onChange={(e) => setPassword(e.target.value)}

required

ref={cancelpswd}

/>

</div>

<button type="submit" className="signup-button">

Sign Up

</button>

</form>

<p className="login-link">

Already have an account? <Link to='/Login'>Login</Link>

</p>

</div>

</div>

</div>

);

}

export default SignUp;

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

import './Login.css';

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

function LoginPage() {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

let cancelemail=useRef("")

let cancelpswd=useRef("")

let navigate=useNavigate()

const handleLogin = (e) => {

e.preventDefault();

fetchdata()

cancelemail.current.value=""

cancelpswd.current.value=""

};

let fetchdata=async()=>{

let result= await fetch("http://localhost:5000/login",{

method: 'post',

body: JSON.stringify({email,password}),

headers:{

'Content-Type':'application/json'

},

})

result= await result.json()

if(result.email){

localStorage.setItem("user",JSON.stringify(result))

navigate('/')

} else{

alert("please enter valid details")

}

}

useEffect(()=>{

let data=localStorage.getItem('user')

if(data){

navigate('/')

}

})

return (

<div className="login-container" >

<div className="login-box">

<h2 id='loginheading'> Login </h2>

<form onSubmit={handleLogin}>

<div className="input-group">

<label htmlFor="email">Email</label>

<input

type="email"

id="loginemail"

value={email}

onChange={(e) => setEmail(e.target.value)}

required

ref={cancelemail}

/>

</div>

<div className="input-group">

<label htmlFor="password">Password</label>

<input

type="password"

id="loginpassword"

value={password}

onChange={(e) => setPassword(e.target.value)}

required

ref={cancelpswd}

/>

</div>

<button type="submit" className="login-button">Login</button>

</form>

<p className="signup-link">

Create New Account? <Link to='/Sign'>Sign-up</Link>

</p>

</div>

</div>

);

}

export default LoginPage;

.about-container {

display: flex;

flex-direction: column;

align-items: center;

padding:55px;

background-color: lightgoldenrodyellow;

color: #333;

font-family: 'Arial', sans-serif;

text-align: center;

padding-top: 75px;

}

.about-content {

max-width: 800px;

margin-bottom: 30px;

}

.about-content h1 {

font-size: 2.5em;

color: #4a90e2;

margin-bottom: 20px;

}

.about\_h2 {

color: white;

font-size: 28px;

margin-top: 30px;

}

li {

font-size: 16px;

color: #36454f;

margin: 10px 0;

}

.about-content p {

font-size: 1.2em;

line-height: 1.6;

margin-bottom: 20px;

color: #36454f;

margin-top: 20px;

}

.about-content h2 {

font-size: 1.8em;

color: #4a90e2;

margin-top: 30px;

}

ol {

margin-left: 20px;

font-size: 16px;

color: white;

padding-top: 5px;

list-style-type: square;

}

.about-content ul {

list-style-type: none;

padding: 0;

margin-top: 30px;

}

.about-content ul li {

background-color: #4a90e2;

color: white;

padding: 10px;

margin: 5px 0;

border-radius: 5px;

}

.about-image img {

width: 100%;

height: auto;

max-width: 400px;

border-radius: 10px;

/\* box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); \*/

animation: fadeIn 0.8s ease-in-out;

}

@media (min-width: 768px) {

.about-container {

flex-direction: row;

justify-content: space-around;

text-align: left;

}

.about-content {

margin-right: 20px;

}

}

.about-image img{

height: 500px;

width: 550px;

}

.about-content h1:hover {

color: yellowgreen;

}

.choose ul li:hover {

background-color: #333;

transition: background-color 0.3s ease;

}

\*{

padding: 0;

margin: 0;

font-family: Arial, sans-serif;

}

.main{

background-color: lightgoldenrodyellow;

height: 60px;

max-width: 100%;

display: flex;

position: sticky;

top: 0;

text-align: center;

}

.logo{

margin-left: -20px;

padding: 15px;

font-weight: bold;

}

.item {

display: flex;

list-style: none;

text-align: center;

gap: 50px;

margin-left: 50px;

align-items: center;

cursor: pointer;

}

.item li a{

color: black;

text-decoration: none;

font-weight: bold;

}

.main img{

margin-left: 60px;

}

.item li a:hover{

color: darkblue;

}

.right{

color: white;

padding: 10px;

margin-left:470px;

}

.right button{

height: 40px;

width: 80px;

border-radius: 20px;

margin-left: 10px;

border: 1px solid blueviolet;

background-color: black;

color: white;

cursor: pointer;

}

#signupbtn{

background-color: blue;

}

#signupbtn:hover{

background-color: black;

}

.header{

background-color: lightgoldenrodyellow;

height: 850px;

max-width: 100%;

}

.heading{

font-size: 75px;

color: gainsboro;

margin-left: 30px;

}

.paragraph{

color: yellow;

text-align: center;

padding-top: 15px;

font-size: 20px

}

.content{

max-width: 90%;

height: 200px;

background-color: #3e3636;

margin: auto;

}

.image{

height: 630px;

width: 90%;

background-color:#36454f;

margin: auto;

}

.spn{

font-size: 97px;

color:palegreen;

}

.btn{

height: 30px;

width: 190px;

background-color: black;

color: white;

margin: 10px;

border-radius: 18px;

text-align: center;

padding-top: 15px;

cursor: pointer;

font-weight: bold;

border: 2px solid rgb(136, 255, 0);

}

.generate{

display: flex;

text-align: center;

justify-content: center;

}

.inside\_img{

width: 70%;

height: 545px;

border-radius: 15px;

border: 3px solid yellowgreen;

margin: auto;

display: flex;

margin-top: 10px;

}

.main\_prompt{

max-height: 100%;

max-width: 100%;

background-color: lightgoldenrodyellow;

display: flex;

padding-top: 80px;

}

.left\_content{

height: 450px;

width: 60%;

padding-left: 80px;

display: flex;

}

.line{

width: 25%;

height: 0.1px;

margin-top: 40%;

background-color: blue;

}

.line2{

width: 0.1%;

height: 110px;

margin-top: 236px;

background-color: blue;

}

.line3{

width: 250px;

height: 0.1px;

margin-top: 236px;

background-color: blue;

}

.left\_content img{

height: 400px;

width: 450px;

}

.right\_content{

height:450px;

width: 40%;

text-align: left;

}

.promptcontent{

float:right;

margin-right: 260px;

}

.right\_content h1{

color: #36454f;

}

.right\_content p{

color: #36454f;

margin-top: 10px;

font-weight: lighter;

}

#prompt{

width: 450px;

height: 40px;

margin-top: 15px;

border-radius: 30px;

border: 1px solid black;

outline: none;

font-size: 25px;

padding-left: 20px;

}

.submit{

height: 40px;

width: 150px;

font-size: 20px;

border-radius: 15px;

border: 2px solid blue;

background-color: black;

color: white;

cursor: pointer;

margin-left: 1%;

margin-top: 15px;

}

#number\_img{

height: 30px;

width: 85px;

margin-left: 2%;

border: none;

outline: none;

padding: 5px;

font-size: 15px;

}

.main\_contact{

background-color: lightgoldenrodyellow;

width: 100%;

display: flex;

height: 605px;

padding-top: 30px;

}

.contact {

width: 50%;

margin: 0 auto;

height: 480px;

padding-left: 95px;

animation: fadeIn 0.8s ease-in-out;

padding-top: 20px;

}

.contact\_img{

width: 50%;

height: 500px;

}

.contact\_img img{

height: 400px;

width: 80%;

padding-top: 20px;

padding-left: 105px;

}

#contact\_us{

text-align: center;

color: #36454f;

margin-right: 70px;

}

.contact-form {

display: flex;

flex-direction: column;

}

.form-group {

margin-bottom: 15px;

}

label {

margin-bottom: 5px;

font-weight: bold;

color: black;

}

input, textarea {

width: 82%;

padding: 10px;

border-radius: 4px;

outline: none;

background-color:lightgoldenrodyellow;

}

textarea {

height: 100px;

background-color: lightgoldenrodyellow;

outline: none;

}

#contact\_btn{

padding: 10px 15px;

background-color: #007BFF;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

width: 85%;

}

button:hover {

background-color: #0056b3;

}

.form-status {

margin-top: 15px;

color: green;

text-align: center;

}

.mainerror{

background-color: black;

height: 345px;

padding-top: 100px;

}

.mainerror h1{

color: white;

text-align: center;

font-size: 120px;

font-weight: lighter;

}

.mainerror p{

color: white;

text-align: center;

}

.promptimage{

background-color: lightgoldenrodyellow;

padding-bottom: 20px;

display: flex;

flex-wrap: wrap;

}

.generated\_image img{

height: 300px;

width: 450px;

margin-left: 40px;

padding-top: 10px;

}

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

import './App.css';

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

import toast from 'react-hot-toast';

function Contact() {

let [name,setname] = useState(" ")

let [email,setemail]=useState("")

let [msg,setmsg]=useState(" ")

let cancelname=useRef("")

let cancelemail=useRef("")

let cancelmsg=useRef("")

const [imageUrl, setImageUrl] = useState(null);

const [loading, setLoading] = useState(false);

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

const query = async (data) => {

setLoading(true);

setError(null);

try {

const response = await fetch(

"https://api-inference.huggingface.co/models/ZB-Tech/Text-to-Image",

{

headers: {

Authorization: "Bearer hf\_vymfolkdROTRUpAqWOPZQdKYuYraRHLYTh",

"Content-Type": "application/json",

},

method: "POST",

body: JSON.stringify(data),

}

);

if (!response.ok) {

throw new Error('Network response was not ok');

}

const result = await response.blob();

const imageObjectURL = URL.createObjectURL(result);

setImageUrl(imageObjectURL);

console.log( imageObjectURL);

} catch (err) {

setError(err.message);

} finally {

setLoading(false);

}

};

const handleGenerateImage = () => {

query({ "inputs": "Astronaut riding a horse" });

};

let item=localStorage.getItem('user')

let navigate=useNavigate()

let handleSubmit=(e)=>{

e.preventDefault()

cancelname.current.value=""

cancelemail.current.value=""

cancelmsg.current.value=""

if(!item){

navigate('/Sign')

}

contactdata()

}

let contactdata=async()=>{

let result= await fetch("http://localhost:5000/contacts",{

method: 'post',

body: JSON.stringify({name,email, message: msg }),

headers:{

'Content-Type':'application/json'

},

})

let data= await result.json()

toast.success("Your message has been sent")

}

return (

<>

<div className="main\_contact">

<div className="contact\_img">

<img src="contact\_img5.jpg" alt="" />

</div>

<div className="contact">

<h1 id='contact\_us'>Contact Us </h1>

<form onSubmit={handleSubmit} className="contact-form">

<div className="form-group">

<label htmlFor="name">Name:</label> <br />

<input

type="text"

id="name"

name="name"

placeholder='Enter Your Name'

required

onChange={(e)=>setname(e.target.value)}

ref={cancelname}

autoComplete='off'

/>

</div>

<div className="form-group">

<label htmlFor="email">Email:</label> <br />

<input

type="email"

id="email"

name="email"

placeholder='Enter your Email'

onChange={(e)=>setemail(e.target.value)}

required

ref={cancelemail}

autoComplete='off'

/>

</div>

<div className="form-group">

<label htmlFor="message">Message:</label> <br />

<textarea

id="message"

name="message"

onChange={(e)=>setmsg(e.target.value)}

required

ref={cancelmsg}

></textarea>

</div>

<button type="submit" id='contact\_btn'>Submit</button>

</form>

</div>

</div>

</>

);

}

export default Contact;

[1:46 PM, 12/9/2024] Rupesh KIET: .footer {

background-color: #333;

color: #fff;

padding: 20px 0;

text-align: center;

}

.footer-content {

display: flex;

flex-direction: column;

align-items: center;

gap: 20px;

}

.footer-logo h2 {

font-size: 1.5em;

}

.footer-logo{

display: flex;

align-items: center;

justify-content: center;

margin-left: -30px;

}

.footer-logo img{

height: 50px;

width: 50px;

}

.footer-logo p {

font-size: 0.9em;

color: #ccc;

}

.footer-links {

display: flex;

gap: 15px;

}

.footer-links a {

color: #ccc;

text-decoration: none;

font-size: 0.9em;

}

.footer-links a:hover {

color: blue;

}

.footer-social {

display: flex;

gap: 15px;

}

.footer-social a {

color: #ccc;

font-size: 1.2em;

}

.footer-social a:hover {

color: #fff;

}

.footer-bottom {

border-top: 1px solid #444;

margin-top: 20px;

padding-top: 10px;

}

.footer-bottom p {

font-size: 0.8em;

color: #777;

}

[1:46 PM, 12/9/2024] Rupesh KIET: import React from 'react';

import './Footer.css';

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

const Footer = () => {

return (

<footer className="footer">

<div className="footer-logo">

<img src="logo.png" alt="" />

<h2>PicStream</h2>

</div>

<div className="footer-content">

<p>Your AI-powered image inspiration</p>

<div className="footer-links">

<Link to="/About">About</Link>

<Link to="/Contact">Contact</Link>

<Link to="/Policy">Privacy Policy</Link>

</div>

{/\* <div className="footer-social">

<a href="https://facebook.com" target="\_blank" rel="noopener noreferrer">

<i className="fab fa-facebook-f"></i>

</a>

<a href="https://twitter.com" target="\_blank" rel="noopener noreferrer">

<i className="fab fa-twitter"></i>

</a>

<a href="https://instagram.com" target="\_blank" rel="noopener noreferrer">

<i className="fab fa-instagram"></i>

</a>

</div> \*/}

</div>

<div className="footer-bottom">

<p>&copy; {new Date().getFullYear()} PicStream. All Rights Reserved.</p>

</div>

</footer>

);

};

export default Footer;

[1:46 PM, 12/9/2024] Rupesh KIET: import React from 'react'

import { ReactTyped } from 'react-typed'

const Header = () => {

return (

<>

<div className="header">

<div className="content">

<h1 className='heading'>EFFORTLESSLY CREATE STUNNING <br />

<ReactTyped className='spn' strings={ ["THE AI-GENRATED IMAGES"]} typeSpeed={70} backSpeed={50} loop={true}/> </h1>

</div>

<div className="image">

<p className='paragraph'>Genrate your image to create breathtaking AI-genrated visuals in minutes</p>

<div className="inside\_img">

<img src="image.jpg" alt="" style={{height:'545px', width:'100%', backgroundRepeat:'no-repeat', borderRadius:'15px'}}/>

</div>

</div>

</div>

</>

)

}

export default Header

[1:47 PM, 12/9/2024] Rupesh KIET: outline: none;a

[1:47 PM, 12/9/2024] Rupesh KIET: .login-container {

display: flex;

justify-content: center;

align-items: center;

height: 635px;

background-size: cover;

background-position: center;

background-repeat: no-repeat;

color: #fff;

background-color: lightgoldenrodyellow;

}

.login-box {

background: lightgoldenrodyellow;

padding:25px 40px;

border-radius: 10px;

box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);

border: 1px solid blue;

animation: fadeIn 0.8s ease-in-out;

}

#loginheading {

text-align: center;

margin-bottom: 20px;

color:black;

}

.input-group {

margin-bottom: 15px;

}

.label {

color: white;

}

#loginemail {

width: 93%;

padding: 10px;

border-radius: 5px;

border: none;

outline: none;

font-size: 16px;

background-color: #36454f;

}

#loginpassword{

width: 93%;

padding: 10px;

border-radius: 5px;

border: none;

outline: none;

font-size: 16px;

background-color: #36454f;

}

.login-button {

width: 100%;

padding: 10px;

background-color: #6200ea;

color: #fff;

border: none;

border-radius: 5px;

font-size: 18px;

cursor: pointer;

transition: background-color 0.3s;

}

.login-button:hover {

background-color: #3700b3;

}

.signup-link{

margin-top: 1rem;

font-size: 0.9rem;

color: #36454f;

text-align: center;

}

[1:47 PM, 12/9/2024] Rupesh KIET: import { StrictMode } from 'react'

import { createRoot } from 'react-dom/client'

import App from './App.jsx'

import './index.css'

import { Auth0Provider } from '@auth0/auth0-react';

createRoot(document.getElementById('root')).render(

<>

<App />

</>

)

[1:47 PM, 12/9/2024] Rupesh KIET: import React from 'react'

const Pagenotfound = () => {

return (

<>

<div className="mainerror">

<h1>404</h1>

<p>OOPS! NOTHING WAS FOUND</p>

</div>

</>

)

}

export default Pagenotfound

[1:48 PM, 12/9/2024] Rupesh KIET: .mainpolicy{

height: 100%;

width: 100%;

background-color: lightgoldenrodyellow;

display: flex;

}

.privacy-policy {

max-width: 800px;

margin: 0 auto;

padding: 20px;

font-family: Arial, sans-serif;

line-height: 1.6;

color: #36454f;

background-color: lightgoldenrodyellow;

}

.privacy-policy h1 {

text-align: center;

margin-bottom: 20px;

font-size: 2em;

color: darkblue;

}

.privacy-policy h1:hover{

color: yellow;

}

.privacy-policy h2 {

font-size: 1.5em;

color: darkblue;

margin-top: 20px;

}

.privacy-policy p {

margin: 10px 0;

}

.privacy-policy ul {

list-style: disc;

padding-left: 20px;

}

.privacy-policy ul li {

margin-bottom: 8px;

color: #36454f;

}

.policyimg img{

height: 500px;

width: 420px;

border-radius: 15px;

}

.policyimg{

margin: 80px;

margin-top: 380px;

}

[1:48 PM, 12/9/2024] Rupesh KIET: import React from 'react';

import './Policy.css';

const PrivacyPolicy = () => {

return (

<div className="mainpolicy">

<div className="privacy-policy">

<h1>Privacy Policy</h1>

<p>Last updated: {new Date().toLocaleDateString()}</p>

<section>

<h2>1. Introduction</h2>

<p>Welcome to PicStream. This Privacy Policy outlines how we collect, use, and protect your information when you use our AI-based image generator application.</p>

</section>

<section>

<h2>2. Information We Collect</h2>

<p>We may collect the following types of information:</p>

<ul>

<li><strong>Personal Information:</strong> such as your name, email address, and any other information you provide when you contact us.</li>

<li><strong>Usage Data:</strong> details on how you interact with our app, including pages visited, clicks, and image preferences.</li>

</ul>

</section>

<section>

<h2>3. How We Use Your Information</h2>

<p>We use the information we collect for the following purposes:</p>

<ul>

<li>To improve the quality and functionality of our app</li>

<li>To provide customer support</li>

<li>To analyze and understand user preferences</li>

<li>To ensure the security of our application</li>

</ul>

</section>

<section>

<h2>4. Sharing Your Information</h2>

<p>We do not sell or rent your personal information to third parties. We may share your information only in the following situations:</p>

<ul>

<li>With service providers who help us operate the application</li>

<li>To comply with legal obligations or respond to valid legal requests</li>

</ul>

</section>

<section>

<h2>5. Data Security</h2>

<p>We implement reasonable security measures to protect your information. However, please note that no data transmission over the internet is completely secure.</p>

</section>

<section>

<h2>6. Your Rights</h2>

<p>You have the right to access, correct, or delete your personal information. If you wish to exercise these rights, please contact us at [Your Contact Email].</p>

</section>

{/\* <section>

<h2>7. Changes to This Privacy Policy</h2>

<p>We may update our Privacy Policy periodically. Any changes will be posted on this page with an updated "Last Updated" date.</p>

</section> \*/}

</div>

<div className="policyimg">

<img src="policy1.jpg" alt="" />

</div>

</div>

);

};

export default PrivacyPolicy;

[1:48 PM, 12/9/2024] Rupesh KIET: import React, { useEffect, useRef } from 'react';

import { useState } from 'react';

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

const Prompt = () => {

let [prompts,setprompt]=useState("")

let [count, setcount]=useState(" ")

let [images,setimages]=useState([])

let canceldata=useRef("")

// let cancelcount=useRef("")

let dataitem=localStorage.getItem('user')

let navigate=useNavigate()

let handlesubmit=(e)=>{

e.preventDefault()

canceldata.current.value=""

if(!dataitem){

navigate('/Sign')

}

fetchdata()

}

let fetchdata=async()=>{

let api= await fetch(https://api.unsplash.com/search/photos?page=1&query=${prompts}&client\_id=88xDbvrMCGvU4vfkFz6yWUpJpp4l1YbbYgIQvMl2qfw)

let response= await api.json()

setimages(response.results)

console.log(response.results);

}

let specefic=async()=>{

let data= await fetch("https://api-inference.huggingface.co/models/ZB-Tech/Text-to-Image",{

headers:{

Authorization: "Bearer hf\_vymfolkdROTRUpAqWOPZQdKYuYraRHLYTh",

"Content-Type": "application/json",

},

method: "POST",

body: JSON.stringify(${prompts}),

})

const result = await response.blob();

console.log(result);

query().then((res)=>{

let url=URL.createObjectURL(res)

img.src=url

})

}

return (

<>

<div className="main\_prompt">

<div className="left\_content">

<img src="promptimg.webp" alt="" />

<span className='line'></span>

<span className='line2'></span>

<span className='line3'></span>

</div>

<div className="right\_content">

<div className="promptcontent">

<h1>Refine Your <br />AI-Generated <br />Image</h1>

<p>PicStream enables you to not only generate <br />stunning images but alse refine and edit <br /> them with powerful AI tools.</p>

</div>

<form action="" onSubmit={handlesubmit}>

<input type="text"

name="text" id="prompt"

placeholder='Enter The Prompt'

autoComplete='off'

required

onChange={(e)=>setprompt(e.target.value)}

ref={canceldata}

/> <br />

{/\* <input type="text"

name="text" id="number\_img"

placeholder='No Of Image'

required

autoComplete='off'

onChange={(e)=>setcount(e.target.value)}

/> \*/}

{/\* <select name="select"

id="number\_img"

onChange={(e)=>setcount(e.target.value)}

ref={cancelcount}

>

<option value="select">select</option>

<option value=" 1"> 1</option>

<option value=" 2"> 2</option>

<option value=" 3"> 3</option>

<option value=" 4"> 4</option>

<option value=" 5"> 5</option>

<option value=" 6"> 6</option>

<option value=" 7"> 7</option>

<option value="8"> 8</option>

<option value=" 9"> 9</option>

<option value="10"> 10</option>

</select> \*/}

<button type="submit" class="submit">Generate Now</button>

</form>

</div>

</div>

<div className="promptimage">

{images.map((value, index) => (

<div class="generated\_image" key={index}>

<img src={value.urls.small} alt="generated" download/>

</div>

))}

</div>

</>

)

}

export default Prompt

[1:48 PM, 12/9/2024] Rupesh KIET: .mainsign{

font-family: "Arial", sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

background: linear-gradient(135deg, #0f2027, #203a43, #2c5364);

background-size: cover;

box-sizing: border-box;

}

.signup-page {

display: flex;

justify-content: center;

align-items: center;

width: 100%;

height: 100vh;

box-sizing: border-box;

background-color: lightgoldenrodyellow;

}

.signup-container {

background: rgba(255, 255, 255, 0.9);

background: lightgoldenrodyellow;

padding: 2.5rem;

border: 1px solid blue;

border-radius: 12px;

max-width: 400px;

width: 100%;

text-align: center;

box-shadow: 0 10px 25px rgba(0, 0, 0, 0.4);

animation: fadeIn 0.8s ease-in-out;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.signup-container h2 {

font-size: 2rem;

margin-bottom: 0.5rem;

color: #333;

}

.signup-container p {

color: #666;

font-size: 0.9rem;

margin-bottom: 1.5rem;

}

.input-group {

position: relative;

margin-bottom: 1.5rem;

}

.input-group input {

width: 90%;

padding: 0.8rem;

border: 1px solid #ddd;

border-radius: 8px;

font-size: 1rem;

transition: border-color 0.3s, box-shadow 0.3s;

background-color: #36454f;

}

.input-group input:focus {

border-color: #6a11cb;

box-shadow: 0 4px 12px rgba(106, 17, 203, 0.2);

outline: none;

}

.signup-button {

width: 97%;

padding: 0.8rem;

background: linear-gradient(135deg, #6a11cb, #2575fc);

color: white;

font-size: 1rem;

border: none;

border-radius: 8px;

cursor: pointer;

transition: background 0.3s, transform 0.2s;

}

.signup-button:hover {

background: linear-gradient(135deg, #2575fc, #6a11cb);

transform: translateY(-2px);

}

.login-link {

margin-top: 1rem;

font-size: 0.9rem;

color: #36454f;

}

.mainsign{

font-family: "Arial", sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

background: linear-gradient(135deg, #0f2027, #203a43, #2c5364);

background-size: cover;

box-sizing: border-box;

}

.signup-page {

display: flex;

justify-content: center;

align-items: center;

width: 100%;

height: 100vh;

box-sizing: border-box;

background-color: lightgoldenrodyellow;

}

.signup-container {

background: rgba(255, 255, 255, 0.9);

background: lightgoldenrodyellow;

padding: 2.5rem;

border: 1px solid blue;

border-radius: 12px;

max-width: 400px;

width: 100%;

text-align: center;

box-shadow: 0 10px 25px rgba(0, 0, 0, 0.4);

animation: fadeIn 0.8s ease-in-out;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.signup-container h2 {

font-size: 2rem;

margin-bottom: 0.5rem;

color: #333;

}

.signup-container p {

color: #666;

font-size: 0.9rem;

margin-bottom: 1.5rem;

}

.input-group {

position: relative;

margin-bottom: 1.5rem;

}

.input-group input {

width: 90%;

padding: 0.8rem;

border: 1px solid #ddd;

border-radius: 8px;

font-size: 1rem;

transition: border-color 0.3s, box-shadow 0.3s;

background-color: #36454f;

}

.input-group input:focus {

border-color: #6a11cb;

box-shadow: 0 4px 12px rgba(106, 17, 203, 0.2);

outline: none;

}

.signup-button {

width: 97%;

padding: 0.8rem;

background: linear-gradient(135deg, #6a11cb, #2575fc);

color: white;

font-size: 1rem;[1:48 PM, 12/9/2024] Rupesh KIET: border: none;

border-radius: 8px;

cursor: pointer;

transition: background 0.3s, transform 0.2s;

}

.signup-button:hover {

background: linear-gradient(135deg, #2575fc, #6a11cb);

transform: translateY(-2px);

}

.login-link {

margin-top: 1rem;

font-size: 0.9rem;

color: #36454f;

}

[1:49 PM, 12/9/2024] Rupesh KIET: {

"name": "picstream",

"private": true,

"version": "0.0.0",

"type": "module",

"scripts": {

"dev": "vite",

"build": "vite build",

"lint": "eslint .",

"preview": "vite preview"

},

"dependencies": {

"@auth0/auth0-react": "^2.2.4",

"react": "^18.3.1",

"react-dom": "^18.3.1",

"react-hot-toast": "^2.4.1",

"react-router-dom": "^6.26.2",

"react-typed": "^2.0.12"

},

"devDependencies": {

"@eslint/js": "^9.11.1",

"@types/react": "^18.3.10",

"@types/react-dom": "^18.3.0",

"@vitejs/plugin-react": "^4.3.2",

"eslint": "^9.11.1",

"eslint-plugin-react": "^7.37.0",

"eslint-plugin-react-hooks": "^5.1.0-rc.0",

"eslint-plugin-react-refresh": "^0.4.12",

"globals": "^15.9.0",

"vite": "^5.4.8"

}

}

[1:49 PM, 12/9/2024] Rupesh KIET: backend

[1:50 PM, 12/9/2024] Rupesh KIET: let mongoose=require('mongoose')

mongoose.connect("mongodb://localhost:27017/picstream")

[1:50 PM, 12/9/2024] Rupesh KIET: let express=require('express')

require('./database/config')

let cors=require('cors')

let User=require('./database/user')

let contact=require('./database/contact')

let interview\_kracker=require('./database/user\_interview\_kracker')

let app=express()

app.use(express.json())

app.use(cors())

app.post("/register",async(req,resp)=>{

let user= new User(req.body)

let result= await user.save()

result= result.toObject()

// delete result.password

resp.send(result)

})

app.post("/contacts",async(req,resp)=>{

let contactdata=new contact(req.body)

let con\_details=await contactdata.save()

resp.send(con\_details)

})

app.post("/login",async(req,resp)=>{

if(req.body.password && req.body.email){

let logindata= await User.findOne(req.body).select("-passwo…

[1:51 PM, 12/9/2024] Rupesh KIET: let mongoose=require('mongoose')

let userschema=new mongoose.Schema({

username:String,

email:String,

password:String

})

module.exports=mongoose.model("user\_interview\_krackers",userschema)

[1:51 PM, 12/9/2024] Rupesh KIET: {

"name": "backend",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node index.js"

},

"author": "",

"license": "ISC",

"dependencies": {

"cors": "^2.8.5",

"express": "^4.21.1",

"mongoose": "^8.8.0"

}

}

**CHAPTER 6**

# TESTING

## 6.1 INTRODUCTION

### 6.1.1 Testing Objectives

### The testing objectives for the Random Image Generator application are as follows:

### Ensure that the system generates images accurately based on user inputs.

### Verify that all features work seamlessly without errors or crashes.

### Validate the security and privacy of user data stored in the cloud.

### Test the application’s performance under different workloads.

### 6.1.2 Testing Principles

### The following principles guide the testing process:

### All tests should trace back to the customer’s requirements.

### Test cases should be planned during the design phase of the application.

### The Pareto Principle applies, where 80% of issues are likely caused by 20% of the components.

### Testing progresses from smaller units to larger integrations.

### Exhaustive testing is impossible; strategic test planning is critical.

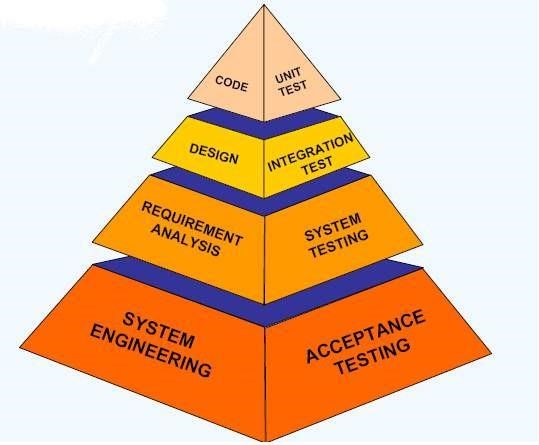
### 6.2 LEVEL OF TESTING

There are different levels of testing

->Unit Testing

->Integration Testing

->System Testing



***Figure 6.1: Testing pyramid***

### 6.2.1Unit testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The important control parts are tested to uncover with in the boundary of the module. The module interface is tested to ensure that the information properly flows into and out of the program unit and boundary conditions are tested to ensure that the modules operate properly at boundaries established to limit or restrict processing. Test date is provided through testing screens.

### 6.2.2Integration testing

Integrating testing is a systematic technique for constructing Program structure while conducting tests to uncover error associates with interfacing. The objective is to take unit modules and built a program structure that has been directed by design.

* Integration Testing will test whether the modules work well together.
* This will check whether the design is correct.

### 6.2.3System testing

System testing is the process of testing the completed software as a part of the environment it was created for. It is done to ensure that all the requirements specified by the customer are met. System testing involves functional testing and performance testing.

System Testing will contain the following testing:

* + **Functional Testing**

Function Testing will test the implementation of the business needs.

* + **Performance Testing**

Performance Testing will test the non-functional requirements of the system like the speed, load etc.

**CHAPTER 7**

# CONCLUSION AND FUTURE SCOPE

**7.1 CONCLUSION**

**The Random Image Generator successfully meets its objectives by providing:**

* Dynamic and creative image generation based on user inputs.
* Secure storage and sharing of generated images.
* A seamless user experience leveraging Firebase for real-time operations and security.

The application demonstrates extendibility, maintainability, and portability. By adhering to modern development practices and technologies, the project ensures high quality and reliability.

**7.2 FUTURE SCOPE**

The application has potential for the following enhancements:

* Authorization Service: Introduce role-based access control for advanced features.
* Improved Database Management: Expand cloud storage to handle a larger user base and more extensive datasets.
* Voice Command Integration: Enable users to generate images through voice input.
* Web Support: Extend the application for web platforms to reach a broader audience.
* AI Enhancements: Include style transfer or deep learning models for more artistic image generation options.

### BIBLIOGRAPHY

 **Li Ma et al** (2014)

*Research and Development of Mobile Application for Android Platform.* International Journal of Multimedia and Ubiquitous Engineering, 9(4):187-198.

 **Nikhil M. Dongre** (2017)

*Journal of Computer Engineering (IOSR-JCE), Volume 19, Issue 2, Ver. I (Mar.-Apr. 2017), PP 65-77.*

 **Javed Ahmad Shaheen et al** (2017)

*Android OS with its Architecture and Android Application with Dalvik Virtual Machine Review.* International Journal of Multimedia and Ubiquitous Engineering, Vol. 12, No. 7, pp. 19-30.

 **Sajid Nabi Khan, IkhlaqUl Firdous** (2017)

*Review on Android App Security.* International Journal of Advanced Research in Computer Science and Software Engineering, Volume 7, Issue 4.

 **LazarelaLazareska, Kire Jakimoski et al** (2017)

*Analysis of the Advantages and Disadvantages of Android and iOS Systems and Converting Applications from Android to iOS Platform and Vice Versa.* American Journal of Software Engineering and Applications, 6(5): 116-120.

 **Bin Peng et al** (2012)

*The Android Application Development College Challenge.* IEEE 14th International Conference on High Performance Computing and Communication & IEEE 9th International Conference on Embedded Software and Systems.

 **Shao Guo-Hong** (2015)

*Application Development Research Based on Android Platform.* 7th International Conference on Intelligent Computation Technology and Automation.

 **S Karthick** (2017)

*Android Security Issues and Solutions.* 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA).

 **PravinAuti, Sangam Mahale, Vikram Zanjad, Madhuri Dangat** (n.d.)

*An Android-Based Random Image Generator Application.*

 **S, A. K.** (n.d.)

*Mastering Firebase for Android Development: Build real-time, scalable, and cloud-enabled Android apps with Firebase.*

 **Professional Android Application Development** by Reto Meier.

 **The Busy Coder's Guide to Advanced Android Development** by Mark L. Murphy.