

**PROJECT – PIXEL PERFECTION : TRANSFORMING YOUR PHOTOS  
WITH OUR CUTTING-EDGE IMAGE EDITING PLATFORM**

**A PROJECT REPORT**

*Submitted by*

**Team Id : NM2023TMID00055**

**Team Leader : PERUMAL RAJ A**

**Team Mem 01 : VISHAL P**

**Team Mem 02 : SANTHOSHKUMAR B**

**Team Mem 03 : GOVINDARAJAN S**

**Team Mem 04 : VIGNESH V**

# **1. INTRODUCTION**

## **1.1 Project Overview**

Pixel Perfection is an innovative image editing platform that allows users to transform their photos with ease and precision. Our cutting-edge software provides a wide range of tools and features that enable users to edit their images to achieve pixel-perfect results. Whether you're a professional photographer, graphic designer, or just someone who loves to take photos, Pixel Perfection is the perfect tool for enhancing your images. With its intuitive user interface and powerful editing tools, you can easily adjust or remove your image backgrounds, car image backgrounds, Bowl cut your face & Upscale your image, and more to create stunning images that are sure to impress. At Pixel Perfection, we understand the importance of high-quality images in today's digital age, and we're committed to providing our users with the tools they need to achieve pixel-perfect results. Whether you're looking to enhance your personal photos or create professional-quality images for your business or clients, Pixel Perfection has everything you need to get the job done.

## **1.2 Purpose**

The purpose of the Pixel Perfection project is to provide users with an innovative image editing platform that empowers them to transform their photos with ease and precision. The project aims to address the increasing demand for high-quality images in today's digital age by offering a comprehensive set of tools and features designed to achieve pixel-perfect results.

## **2. IDEATION & PROPOSED SOLUTION**

### **2.1 Problem Statement Definition**

In today's digital age, the demand for high-quality images has significantly increased across various domains, including photography, graphic design, and social media. However many individuals, including professional photographers and enthusiasts, face challenges when it comes to achieving pixel-perfect results in their image editing endeavors. The existing image editing software may lack intuitive user interfaces, advanced editing capabilities, and efficient tools, leading to frustration and suboptimal outcomes. Many image editing platforms are complex and require a steep learning curve, making it difficult for individuals with limited editing experience to achieve desired results. This restricts accessibility and prevents a wider range of users from effectively enhancing their images.

## 2.2 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users.


Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



## 2.3 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Template




### Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare  
👥 1 hour to collaborate  
👤 2-8 people recommended

[Share template feedback](#)



#### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

1

#### Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

PROBLEM

How to transform the images without losing original quality??  
Certain image file formats can't be uploaded



#### Key rules of brainstorming

To run an smooth and productive session

 Stay in topic.

 Encourage wild ideas.

 Defer judgment.

 Listen to others.

 Go for volume.

 If possible, be visual.

2

### Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

#### Perumal

Advanced color correction and grading tools to use

users may feel skeptical about the transformed images

some websites may not support certain file formats

#### Vishal

We give simple User-friendly interface

we need to think about the security of the image

users to easily post and share their edited photos

#### Santhosh

we need to provide Cloud storage

online making it easy to share and work on projects with others.

Allowing users to fine-tune colors to create unique visual styles.

#### Govindarajan

Integration with social media platforms

interface makes easy for both beginners and professionals to use.

Website is not responsive and doesn't have mobile version

#### Vignesh

fear of uploading image because of trust issue

reduce the image quality during uploading process

websites take long time to upload images

### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as they're within your mind.

#### Transforming Images

Advanced color correction and grading tools to use

Allowing users to fine-tune colors to create unique visual styles.

#### Problem Statement

websites take long time to upload images

some websites may not support certain file formats

Website is not responsive and doesn't have mobile version

reduce the image quality during uploading process

#### Fear of Security

we need to think about the security of the image

fear of uploading image because of trust issue

#### Easy to share

Integration with social media platforms

users to easily post and share their edited photos

4

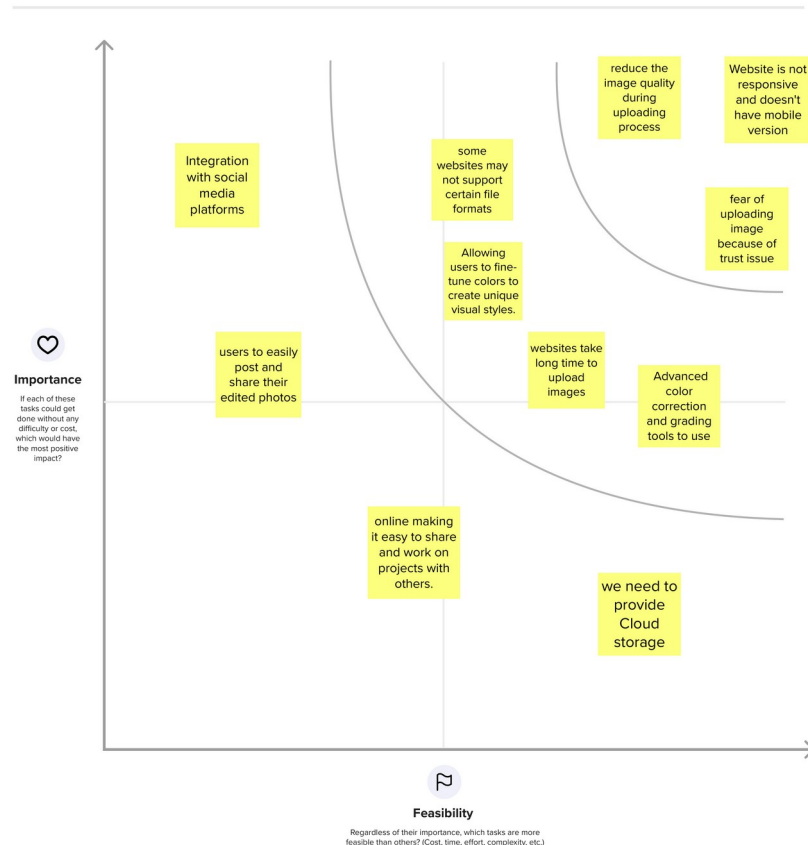
### Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.



## **2.4 Proposed Solution**

To address the challenges mentioned in the problem statement and provide an effective solution, the Pixel Perfection project aims to develop an innovative image editing platform with the following key features:

### **1. Advanced Editing Tools:**

The project will incorporate cutting-edge editing tools and features that enable precise adjustments and transformations. Users will have access to a wide range of tools for background adjustment and removal, car image background removal, creating a bowlcut hairstyle, image upscaling. These tools will provide users with the flexibility and capabilities needed to achieve pixel-perfect results.

### **2. Diverse Editing Capabilities:**

Pixel Perfection will offer a comprehensive set of editing options to cater to diverse user needs. The platform will include a variety of filters, effects, and enhancement tools to allow users to creatively enhance their images. Users will also have the ability to apply adjustments selectively to specific areas of their photos, providing them with fine-grained control over the editing process.

### 3. REQUIREMENT ANALYSIS

#### 3.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login via Username and Password Login via Gmail
FR-4	Upload Image	Upload Image through Drive Upload Image through Local Storage Upload Image through Url
FR-5	Transform Image	Choose Background remover Choose Cartoonize Choose Face Beauty Choose 3Dcartoon Choose Motion
FR-6	User Submit	Submit by Clicking Button

#### 3.2 Non Functional Requirements

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	<ol style="list-style-type: none"><li>1. Intuitive Interface: The user interface should be designed in a way that is easy to understand and navigate. Clear and concise labeling, well-organized menus, and intuitive icons can help users quickly locate and access the editing tools they need.</li><li>2. Simple and Consistent Controls: The editing controls should be straightforward and consistent</li></ol>



		<p>throughout the platform. Users should be able to easily adjust settings, apply effects, crop or resize images, and perform other editing actions without confusion or complexity.</p> <ol style="list-style-type: none"> <li>3. Visual Feedback: The platform should provide visual feedback to users when they perform actions. For example, when a user applies an effect or adjusts a setting, there should be clear visual cues indicating the changes made to the image. This helps users understand the impact of their actions and make adjustments accordingly.</li> </ol>
NFR-2	<b>Security</b>	<ol style="list-style-type: none"> <li>1. Secure Authentication: Implement a robust authentication mechanism to verify the identity of users before granting access to the platform. This can include features such as username/password authentication, two-factor authentication (2FA), or integration with third-party authentication providers.</li> <li>2. Encryption: Employ encryption techniques to protect sensitive data such as user credentials, edited photos, and any communication between the client and server. Use strong encryption algorithms and ensure that data at rest and in transit is properly encrypted.</li> <li>3. Authorization and Access Control: Implement role-based access control (RBAC) to ensure that users can only access the functionalities and features appropriate for their roles and permissions. This helps prevent unauthorized access and restricts users from tampering with other users' data.</li> </ol>

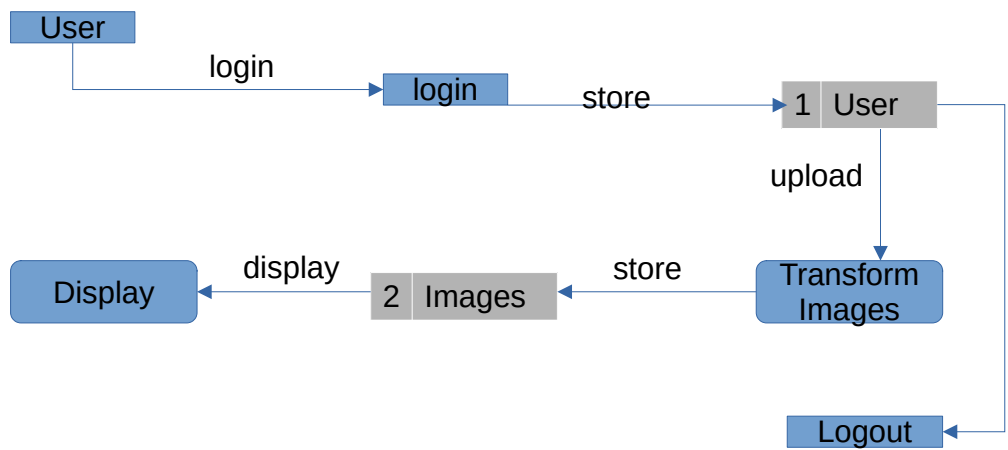
NFR-3	<b>Reliability</b>	<ol style="list-style-type: none"> <li>1. Error Handling: The platform should have robust error handling mechanisms in place to detect and handle errors gracefully. This includes providing informative error messages to users, logging errors for analysis and troubleshooting, and recovering from errors to maintain the stability of the system.</li> <li>2. Fault Tolerance: Pixel Perfection should be designed to withstand faults or failures without experiencing complete system breakdown. This can be achieved by implementing redundancy, such as backup servers or data replication, to ensure that the platform remains accessible even if certain components fail.</li> <li>3. Data Integrity: The platform should ensure the integrity of users' photos and edited images. This involves preventing data corruption, maintaining the accuracy of edits, and preserving the original files without unintended alterations or losses.</li> </ol>
NFR-4	<b>Performance</b>	<ol style="list-style-type: none"> <li>1. Speed and Responsiveness: Users expect real-time or near-real-time response when making edits to their photos. The platform should minimize processing delays and provide immediate feedback as users interact with the editing tools.</li> <li>2. Image Processing Efficiency: The platform should be able to handle image processing operations efficiently, especially when dealing with large image files or complex editing tasks. Optimizing algorithms and utilizing hardware acceleration (such as leveraging</li> </ol>

		<p>graphics processing units, if applicable) can help improve processing speed.</p> <ol style="list-style-type: none"> <li>3. File Loading and Saving: Loading and saving images should be quick and efficient. Users should not experience significant delays when importing images into the platform or saving their edited photos back to their devices or cloud storage.</li> </ol>
NFR-5	<b>Availability</b>	<ol style="list-style-type: none"> <li>1. Redundancy and Failover: Implement redundancy at various levels of the system architecture to minimize the impact of potential failures. This can include redundant servers, databases, and network infrastructure. Employ failover mechanisms to automatically switch to backup systems in case of failures, reducing downtime.</li> <li>2. Load Balancing: Distribute the user load across multiple servers to prevent any single server from becoming overwhelmed. Load balancing helps ensure that the system can handle a high volume of simultaneous user requests without degradation in performance.</li> </ol>
NFR-6	<b>Scalability</b>	<ol style="list-style-type: none"> <li>1. Horizontal Scalability: Pixel Perfection should be designed to scale horizontally, meaning that as the user base grows, the platform can handle the increased load by adding more servers or instances. This could involve deploying the platform across multiple servers or using cloud-based infrastructure that allows for easy scaling.</li> <li>2. Load Balancing: To efficiently distribute the incoming user requests and workload across</li> </ol>

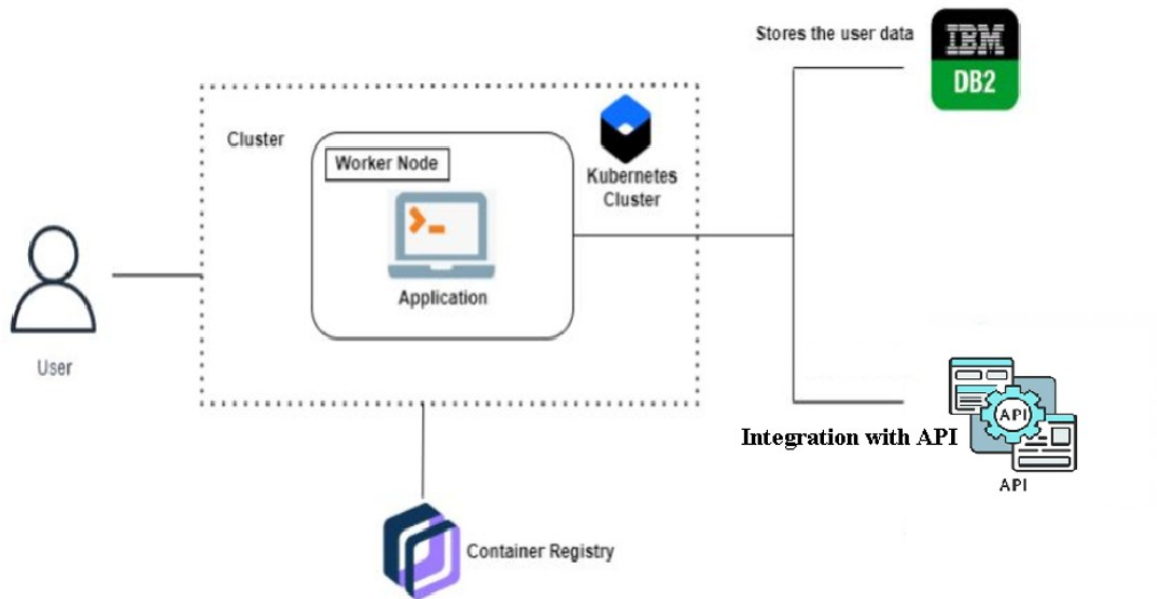
		multiple servers or instances, a load balancing mechanism should be implemented. This ensures that no single server is overwhelmed while others remain underutilized.
--	--	---

## 4. PROJECT DESIGN

### 4.1 Data Flow Diagrams



## 4.2 Solution & Technical Architecture



## 4.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Santhosh Kumar
		USN-2	As a user, I will easily logout the application	I can easily logout	Medium	Vignesh

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
		USN-3	As a user, I can register for the application through Gmail	I can register through my account	Medium	Perumal
	Login	USN-4	As a user, I can log into the application by entering email & password	I can access my account/ dashboard	High	Vishal
	Dashboard	USN-5	As a user, I can easily understand the user interface	I can easily access the interface	High	Govindarajan
Customer (Web user)	Registration	USN-6	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Santhosh Kumar
		USN-7	As a user, I will easily logout the application	I can easily logout	Medium	Vignesh
		USN-8	As a user, I can register for the application through Gmail	I can register through my account	Medium	Perumal
	Login	USN-9	As a user, I can log into the application by entering email & password	I can access my account/ dashboard	High	Vishal
	Dashboard	USN-10	As a user, I can easily understand the user interface	I can easily access the interface	High	Govindarajan

## 5. CODING & SOLUTIONING

### 5.1 Feature 1 – Bowlcut

```
#BowlCut
@app.route('/cart',methods=['POST','GET'])
def cart():
    if request.method=='POST':
        url,filename=getImageLink()
        output = replicate.run(
            "orpatashnik/styleclip:7af9a66f36f97fee2fece7dcc927551a951f0022cbdd23747b9212f23fc17021",
            input={"input":url}
        )
        data=requests.get(output).content
        file=open(filename,"wb")
        file.write(data)
        file.close()
        link=uploadAndClean(filename,"BOWL CUT")
        print(link)
    return render_template("upscale.html",link=output)
```

### 5.2 Feature 2 – Upscale

```
#upscale
@app.route('/upscale',methods=['POST','GET'])
def upscale():
    if request.method=='POST':
        url,filename=getImageLink()
        output = replicate.run(
            "tencentarc/gfpgan:9283608cc6b7be6b65a8e44983db012355fde4132009bf99d976b2f0896856a3",
            input={"img":url}
        )
        data=requests.get(output).content
        file=open(filename,"wb")
        file.write(data)
        file.close()
        link=uploadAndClean(filename,"UPSCALE")
        print(link)
    return render_template("upscale.html",link=link)
```

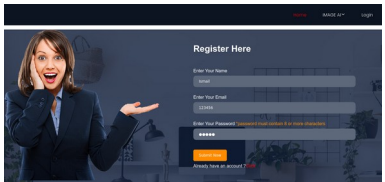
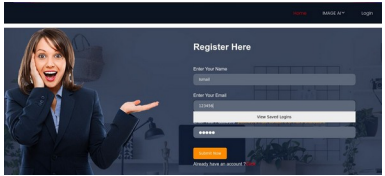
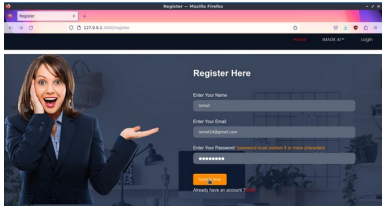
### 5.3 Database Schema

```
"""User can register"""
@app.route('/register_user',methods=['POST','GET'])
def register_user():
    msg='None'
    if request.method=='POST':
        Name=request.form['NAME']
        Email=request.form['EMAIL']
        Password=request.form['PASSWORD']
        sql="SELECT * FROM USER1 WHERE EMAIL=? AND PASSWORD=?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,Email)
        ibm_db.bind_param(stmt,2>Password)
        ibm_db.execute(stmt)
        acc=ibm_db.fetch_assoc(stmt)
```

```
#user login
@app.route('/login_user',methods=['POST','GET'])
def login_user():
    msg='None'
    if request.method=="POST":
        Email=request.form['EMAIL']
        Password=request.form['PASSWORD']
        sql="SELECT * FROM USER1 WHERE EMAIL=? AND PASSWORD=?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,Email)
        ibm_db.bind_param(stmt,2>Password)
        ibm_db.execute(stmt)
        acc=ibm_db.fetch_assoc(stmt)
```

## 6. RESULTS

### 6.1 Performance Metrics

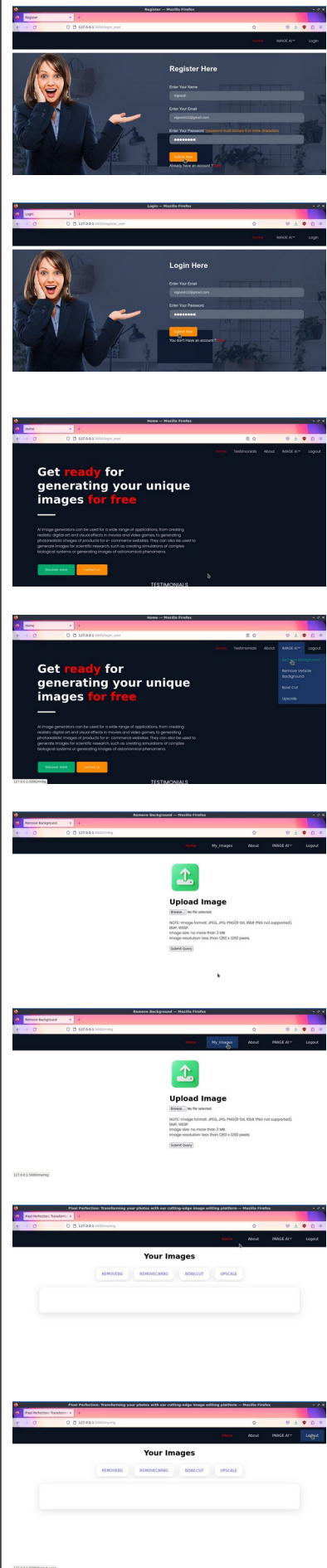
S.No.	Parameter	Values	Screenshot
4.	Form Validation	Login form, registration form and all other forms validation	  

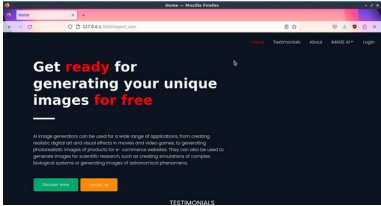
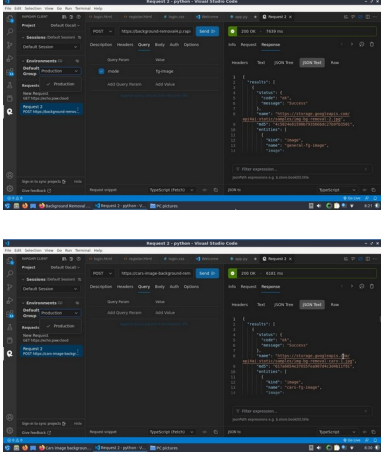
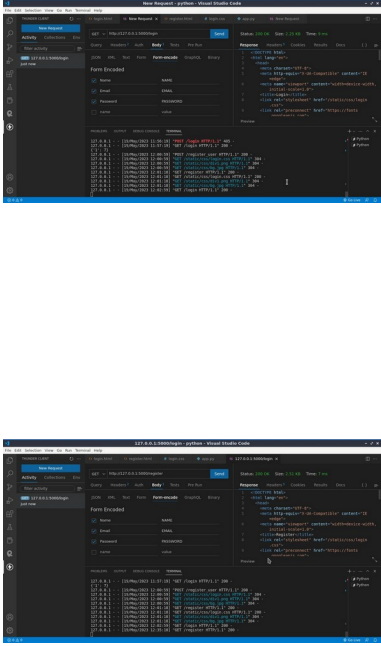


5.

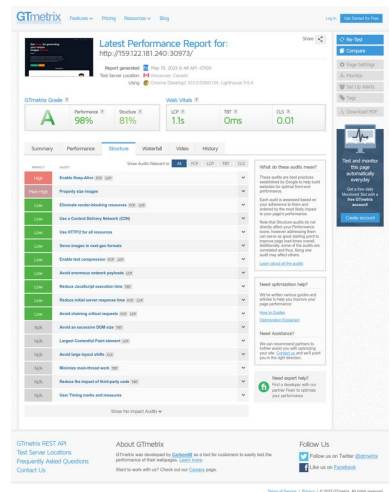
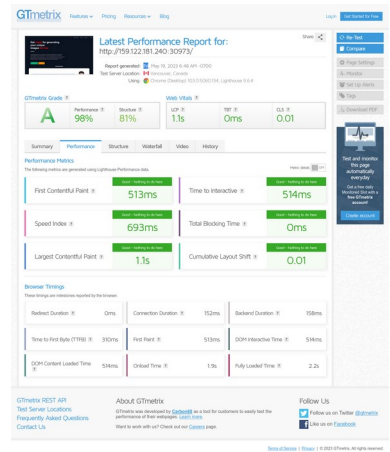
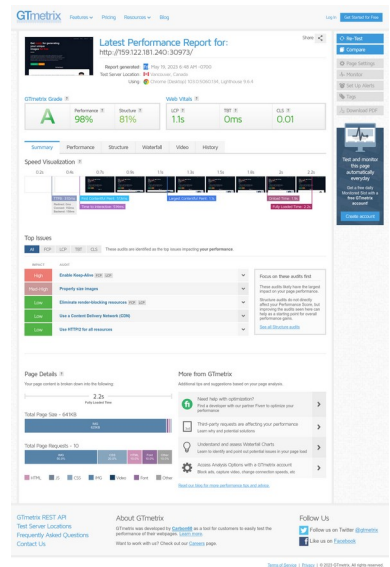
Project Flow

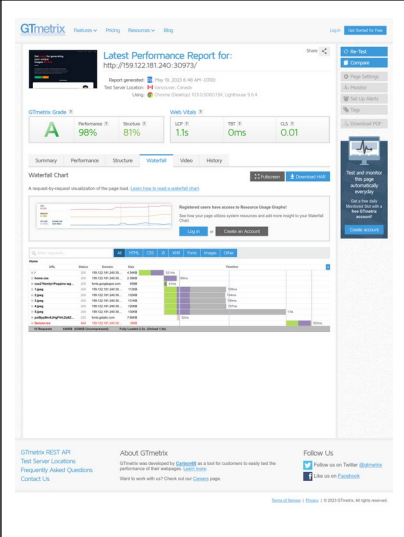
Redirections



			
3.	API Testing/Validation	Testing API	
4.	Database Schema Validation	Validating inputs as per schema	

5. Application performance testing using online tool (GTMetrix)



			
--	--	--	---

## 7. ADVANTAGES & DISADVANTAGES

### Advantages

1. Pixel Perfection allows a wider range of users to achieve professional quality image editing tools.
2. Users have granular control over various elements of their images, including backgrounds, objects and image resolution, allowing them to achieve pixel-perfect tools.
3. Pixel Perfection offers a comprehensive range of editing options, including specialized features like background removal, creating a bowlcut hairstyle and image upscaling.

### Disadvantages

1. Users may experience compatibility issues based on their devices, operating systems, or system specifications.
2. Users with highly specific editing needs may need to seek additional software or tools to fulfill those requirements.
3. Pixel Perfection is likely to be an online platform, requiring a stable internet connection to access and utilize its features fully.

## 8. CONCLUSION

In conclusion, the Pixel Perfection project aims to address the challenges faced by users in achieving pixel-perfect results in image editing. By developing an innovative image editing platform with an intuitive user interface, advanced editing tools, diverse capabilities, time efficiency, and a commitment to user satisfaction, Pixel Perfection offers a comprehensive solution for enhancing images with ease and precision.

## 9. FUTURE SCOPE

The integration of Artificial Intelligence technologies, such as machine learning and computer vision, could significantly enhance Pixel Perfection's capabilities. AI-powered features like automatic image enhancement, intelligent object recognition, and advanced background removal algorithms could streamline the editing process and offer users more efficient and accurate results.

## 10. APPENDIX

### Source Code:

```
from flask import *
import os
import base64
import requests
import ibm_boto3
import replicate
from ibm_botocore.client import Config, ClientError
import ibm_db
import re
import requests

app = Flask(__name__)

MODE='demo'
```

```
RAPIDAPI_KEY="3801acee72msh26906931aee075dp1640aejsn78b541076ce0"  
#rapid api key
```

```
OPTIONS = {  
  'demo': {  
    'url': 'https://demo.api4ai.cloud/img-bg-removal/v1/results',  
    'headers': {'A4A-CLIENT-APP-ID': 'sample'}  
  },  
  'rapidapi': {  
    'url': 'https://background-removal4.p.rapidapi.com/v1/results',  
    'headers': {'X-RapidAPI-Key': RAPIDAPI_KEY}  
  }  
}
```

```
COS_END_POINT="https://s3.us-south.cloud-object-storage.appdomain.cloud"  
COS_API_KEY_ID="Zilfo9px58o1ZpUqzICdwhwF6mV4ZbxKM86QAMUdVuUD"  
"
```

```
COS_RES_CRN="crn:v1:bluemix:public:cloud-object-storage:global:a/  
0c99b2c2ed6a41919b9d8b8ff36c9f3d:c56177ab-c237-4a12-9e31-913e74e58dc6::"
```

```
os.environ["REPLICATE_API_TOKEN"] =  
"r8_34nlmqzjOaNREnC4InS3jiwnlnvc13b2ci5Rd" #replicate api key
```

```
#ibm cloud object storage connection
```

```
cos=ibm_boto3.client("s3",ibm_api_key_id=COS_API_KEY_ID,ibm_service_instance_id=COS_RES_CRN,config=Config(signature_version="oauth"),endpoint_url=COS_END_POINT)
```

```
conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=8e359033-a1c9-4643-82ef-
```

```
8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30120;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=xhh29229;PWD=UGnlTT78369pGWHy",",")
```

```
print("Connection Successful",conn)
```

```
app.secret_key="vignesh"  
global user
```

```
@app.route("/")
```

```

def home():
    return render_template('home.html')

@app.route('/login')
def login():
    return render_template('login.html')

@app.route('/register')
def register():
    return render_template('register.html')

@app.route('/upscale_ui')
def upscale_ui():
    return render_template('upscale.html')

@app.route('/cartoon')
def cartoon():
    return render_template('cartoon.html')

@app.route('/beauty')
def beauty():
    return render_template('beauty.html')

@app.route('/vehrm')
def vehrm():
    return render_template('vehicleremove.html')

def getImageLink():    #To store image from form to local storage and get cos link
from it
    f=request.files['image']
    basepath=os.path.dirname(__file__)
    filepath=os.path.join(basepath,'uploads',f.filename)
    f.save(filepath)
    cos.upload_file(Filename=filepath,Bucket='vignesh-arch',Key=f.filename)
    link="https://vignesh-arch.s3.us-south.cloud-object-
storage.appdomain.cloud/"+f.filename
    return link,f.filename

```

```

def uploadAndClean(*img):    #method to store the output img to cos and to clean
the previously stored(unused images) from local storage and cos
    key_name=key_name=img[0].replace(".", "")+img[1]+"_out.jpg"
    cos.upload_file(Filename=img[0],Bucket='vignesh-arch',Key=key_name)
                                url="https://vignesh-arch.s3.us-south.cloud-object-
storage.appdomain.cloud/"+key_name
    storeHistory(url,img[1])
    basepath=os.path.dirname(__file__)
    os.remove(img[0])
    os.remove("/"+basepath+"/uploads/"+img[0])
    try:
        cos.delete_object(Bucket="vignesh-arch", Key=img[0])
        print("Item: {filename} deleted!\n".format(img[0]))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to delete object: {0}".format(e))
    return url

```

```

def storeHistory(*img): #To store the image url's in the ibm db2
    sql="INSERT INTO IMAGE_URL(USERD,"+img[1]+") VALUES(?,?)"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['USERD'])
    ibm_db.bind_param(stmt,2,img[0])
    ibm_db.execute(stmt)

```

```

def getHistory(): #To get the stored image url's from the ibm db2
    sql="SELECT * FROM IMAGE_URL WHERE USERD=?"
    stmt=ibm_db.prepare(conn,sql)
    try:
        ibm_db.bind_param(stmt,1,session['USERD'])
        ibm_db.execute(stmt)
        rmbg=[]
        carbg=[]
        upscale=[]
        boxcut=[]
        while True:
            data=ibm_db.fetch_assoc(stmt)
            if not data:

```



```

        break
    else:
        if data['REMOVEBG']!= None:
            rmbg.append(data['REMOVEBG'])
        if data['CARBG']!= None:
            carbg.append(data['CARBG'])
        if data['BOWLCUT']!= None:
            boxcut.append(data['BOWLCUT'])
        if data['UPSCALE']!= None:
            upscale.append(data['UPSCALE'])
    print(rmbg,carbg,upscale,boxcut)
    return rmbg,carbg,upscale,boxcut
except KeyError:
    return render_template('login.html')

```

#user login

```
@app.route('/login_user',methods=['POST','GET'])
```

```
def login_user():
```

```
    msg='None'
```

```
    if request.method=="POST":
```

```
        Email=request.form['EMAIL']
```

```
        Password=request.form['PASSWORD']
```

```
        sql="SELECT * FROM USER1 WHERE EMAIL=? AND PASSWORD=?"
```

```
        stmt=ibm_db.prepare(conn,sql)
```

```
        ibm_db.bind_param(stmt,1,Email)
```

```
        ibm_db.bind_param(stmt,2>Password)
```

```
        ibm_db.execute(stmt)
```

```
        acc=ibm_db.fetch_assoc(stmt)
```

```
        if acc:
```

```
            session['Loggedin']=True
```

```
            session['USERD']=acc['USERD']
```

```
            session['NAME']=acc['NAME']
```

```
            msg="logged"
```

```
            return render_template('home.html',log=msg)
```

```
        else:
```

```
            msg="Please check your Username and Password"
```

```
            return render_template('register.html',msg=msg)
```

```
    return render_template('login.html',msg=msg)
```

""""User can register""""

```
@app.route('/register_user',methods=['POST','GET'])
```

```
def register_user():
```

```
    msg='None'
```

```
    if request.method=='POST':
```

```
        Name=request.form['NAME']
```

```
        Email=request.form['EMAIL']
```

```
        Password=request.form['PASSWORD']
```

```
        sql="SELECT * FROM USER1 WHERE EMAIL=? AND PASSWORD=?"
```

```
        stmt=ibm_db.prepare(conn,sql)
```

```
        ibm_db.bind_param(stmt,1,Email)
```

```
        ibm_db.bind_param(stmt,2>Password)
```

```
        ibm_db.execute(stmt)
```

```
        acc=ibm_db.fetch_assoc(stmt)
```

```
        if acc:
```

```
            msg="You are already registered.Please Login using using your credentials"
```

```
            return render_template('login.html')
```

```
        elif not re.match(r'^[@]+\.[^@]+\.[^@]+',Email):
```

```
            msg="Please type valid Email"
```

```
        else:
```

```
            sql="SELECT COUNT(*) FROM USER1"
```

```
            stmt=ibm_db.prepare(conn,sql)
```

```
            ibm_db.execute(stmt)
```

```
            count=ibm_db.fetch_assoc(stmt)
```

```
            print(count)
```

```
            sql1="INSERT INTO USER1 VALUES(?,?,?,?)"
```

```
            stmt1=ibm_db.prepare(conn,sql1)
```

```
            ibm_db.bind_param(stmt1,1,Name)
```

```
            ibm_db.bind_param(stmt1,2,Email)
```

```
            ibm_db.bind_param(stmt1,3>Password)
```

```
            ibm_db.bind_param(stmt1,4,count['1']+1)
```

```
            ibm_db.execute(stmt1)
```

```
            msg="Successfully Registered"
```

```
            return render_template('login.html',msg=msg)
```

```
    return render_template('register.html',msg=msg)
```

#user logout

```
@app.route('/logout_user')
```

```

def logout_user():
    session.pop('loggedin',None)
    session.pop('USERID',None)
    return render_template('home.html')

@app.route('/myimg')
def myimg():
    try:
        rmbg,carbg,upscale,boxcut=getHistory()
    except ValueError:
        return render_template('login.html')
    return
render_template('my_images.html',rmbg=rmbg,carbg=carbg,upscale=upscale,boxcut
=boxcut)

#upscale
@app.route('/upscale',methods=['POST','GET'])
def upscale():
    if request.method=='POST':
        url,filename=getImageLink()
        output = replicate.run(
            "tencentarc/gfpgan:9283608cc6b7be6b65a8e44983db012355fde4132009bf99
d976b2f0896856a3",
            input={"img":url}
        )
        data=requests.get(output).content
        file=open(filename,"wb")
        file.write(data)
        file.close()
        link=uploadAndClean(filename,"UPSCALE")
        print(link)
    return render_template("upscale.html",link=link)

@app.route('/rmbg',methods=['POST','GET'])
def rmbg():
    link="None"
    if request.method=='POST':
        url,filename=getImageLink()
        response = requests.post(

```

```

        OPTIONS[MODE]['url'],
        headers=OPTIONS[MODE]['headers'],
        data={'url': url})
    img_b64 = response.json()['results'][0]['entities'][0]['image'].encode('utf8')
    path_to_image = os.path.join(filename)
    with open(path_to_image, 'wb') as img:
        img.write(base64.decodebytes(img_b64))
    link=uploadAndClean(path_to_image,"REMOVEBG")
    return render_template('removebg.html',link=link)

```

```

@app.route('/rmvehicle',methods=['POST','GET'])
def rmvehicle():
    link="None"
    if request.method=='POST':
        url,filename=getImageLink()
        response = requests.post(
            OPTIONS[MODE]['url'],
            headers=OPTIONS[MODE]['headers'],
            data={'url': url})
        img_b64 = response.json()['results'][0]['entities'][0]['image'].encode('utf8')
        path_to_image = os.path.join(filename)
        with open(path_to_image, 'wb') as img:
            img.write(base64.decodebytes(img_b64))
        link=uploadAndClean(path_to_image,"CARBG")
        return render_template('vehicleremove.html',link=link)

```

#BowlCut

```

@app.route('/cart',methods=['POST','GET'])
def cart():
    if request.method=='POST':
        url,filename=getImageLink()
        output = replicate.run(
            "orpatashnik/styleclip:7af9a66f36f97fee2fece7dcc927551a951f0022cbdd23747b9212f23fc17021",
            input={"input":url}
        )
        data=requests.get(output).content
        file=open(filename,"wb")
        file.write(data)

```

```
    file.close()
    link=uploadAndClean(filename,"BOWLCUT")
    print(link)
    return render_template("upscale.html",link=output)

if __name__=="__main__":
    app.run(debug=True,port=5000,host='0.0.0.0')
```

## **GITHUB LINK**

[Source Code](#)

## **PROJECT DEMO VIDEO LINK**

[Click Here](#)

## **PROJECT LIVE LINK**

[Click Here](#)