SYMBIOSIS INSTITUTE OF TECHNOLOGY (SIT)
Constituent of Symbiosis International (Deemed University), Pune
(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of india)
Re-Accredited by NAAC with 'A' Grade
॥वसुधैव कुटुम्बकम्॥

PROJECT REPORT

ON

PHOTO ENHANCER

UNDER THE GUIDANCE OF
PROF. KALYANI KADAM

SYMBIOSIS INSTITUTE OF TECHNOLOGY
(A CONSTITUENT OF
SYMBIOSIS INTERNATIONAL UNIVERSITY)
PUNE - 412115

2019-23

Prepared by:-
Patel Keny A - 19070122122
S Easwaran - 19070122143

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Photo Enhancer software. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external behavior.

Photo Enhancer is a convenient browser-based software for online image editing and enhancement.
It provides an easy-to-use user interface that allows you to quickly and painlessly edit or enhance images, photos or graphics.

1.2 Document Conventions

Requirement indicators used:
H= High priority requirements that must be met before the project is launched.
M= Medium priority requirements that should be met unless there are compelling reasons for the delay.

1.3 Intended audience and Reading Suggestions

This document is intended for analysts and developers of the system who will lead to the system with desired functionalities without any misinterpretation of actual requirements.
It is also an architectural baseline meant for testers and people involved in documentation for the purpose of software maintenance.

1.4 Product Scope

The key goal of the software is image editing, enhancement and processing. It aims at improvement in quality of a given image that boosts the sensitivity of information for human viewers or to develop an enhanced input for other crucial image processing procedures.

The software dramatically reduces the time and development efforts required to customize your graphical content, saving your time and money.

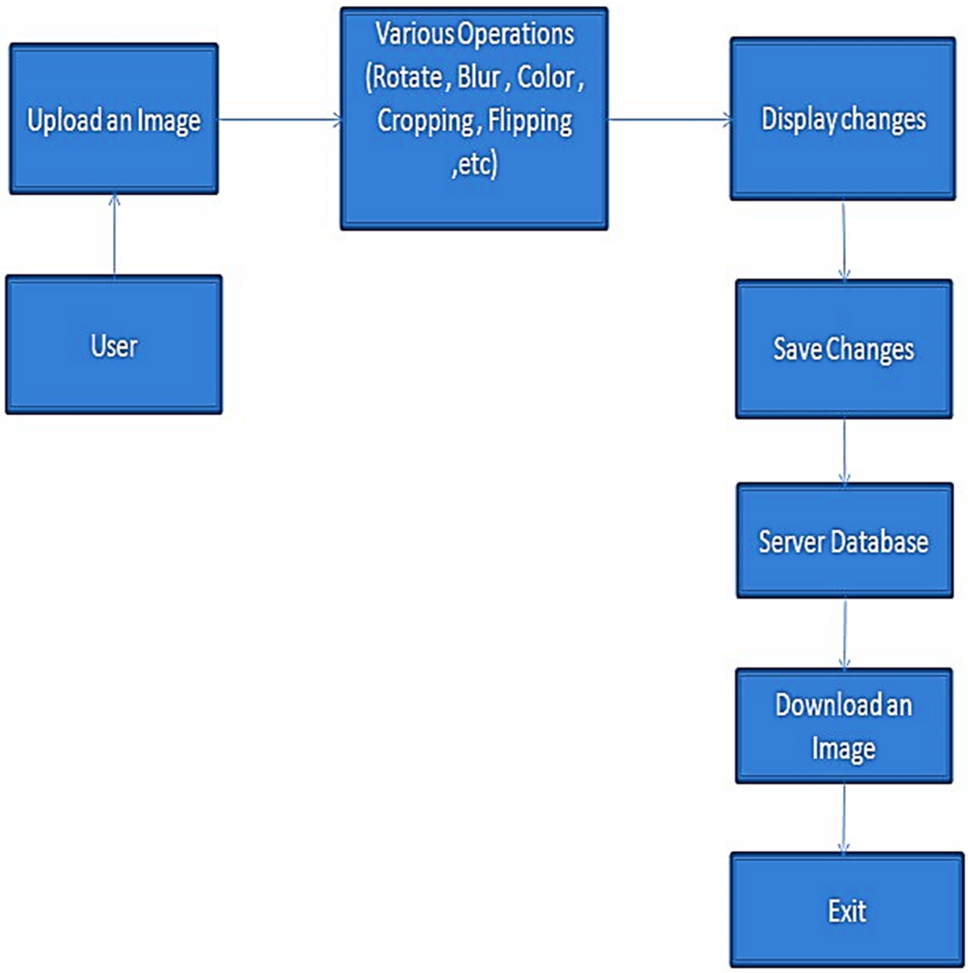
1.5 References

https://www.youtube.com/watch?v=UI6lqHOVHic
https://www.youtube.com/watch?v=zid-MVo7M-E
https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

2. Overall description

2.1 Product perspective

Photo Enhancer software is a dynamic web based product that is self-contained in nature. The following block diagram briefly represents the major components of the system.



2.2 Product Functions

The software provides the following major features to the user.

- Login (for existing user) and Sign up (for new user)
- Upload an image
- Crop an image
- Rotate the image
- Adjust contrast of the image and preset if required
- Adjust brightness of the image and preset if required
- Adjust or preset temperature and saturation of image
- Dehaze the image
- Download the modified image

2.3 User Classes and Characteristics

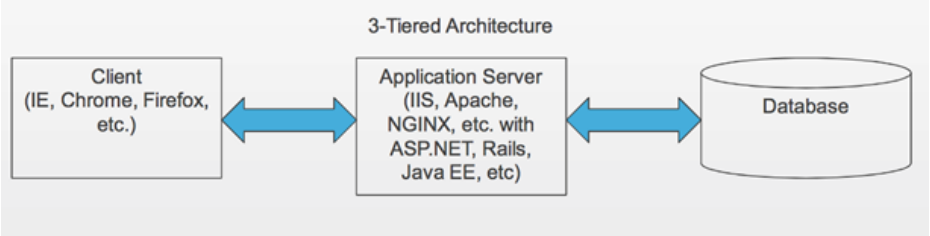
This application is for occasional photographers to improve the aesthetic visual quality of an image.
It also proves useful for digital data analysts.
The dehaze image feature can be a major help in exploratory data analysis and cleaning for the creation of dataset.
The user-interface is simple and doesn't require any inherent knowledge of detailed image processing.

2.4 Operating Environment

Operating System: Platform independent (Windows, Linux, etc.)
Specifications: Java-compatible web browser, Internet connection

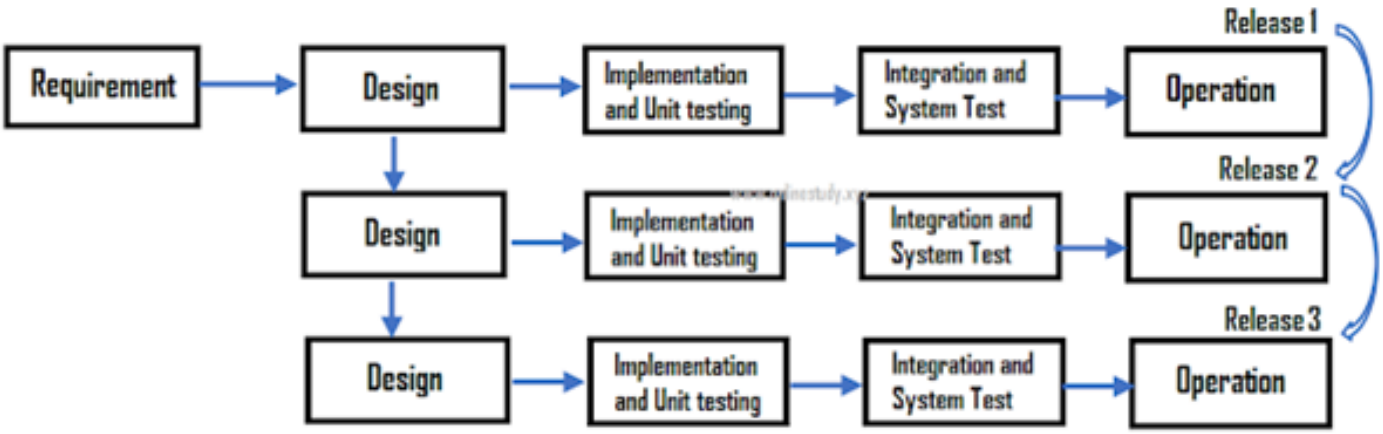
2.5 Design and Implementation Constraints

Preferred Hardware specifications: 64bit OS; i5 processor and above; 8GB installed RAM
Technology-stack used (for developers, testers): -
JavaEE(backend)
HTML+CSS, JavaScript(front-end)
MySQL(database)
Tools Used: MyEclipse IDE, Visual Studio Code
Version Control: Git



2.6 Process Model

Incremental Process Model is used to develop our mini project. The model combines the elements of the waterfall model with the iterative philosophy of prototyping. New features and maintenance of the software will be developed timely. We are using this process model as with every linear increment addition of new features to the software will help in development of the software. Hence, we are using this model as it is more flexible. Also, it is easy to test and debug the software through this model.



2.7 User Documentation

As a part of the development team, the last experience we want the user is to be contacting the support team after having failed to understand the product.
For ease of accessibility and understanding, we provide the following overview:

1.Perform signup if you are a new user. Existing users can login with their credentials.
2.Upload the image to be processed from your local device.
3.Apply the required features to edit and enhance the image from the available options
4.At each modification, you can save it or revert to the original state
5.Download the final image in the jpg format.

2.8 Assumptions and Dependencies

a. Working of software is dependent on the availability of Internet connection and a web browser

b. The software has access and required permissions to the user's cloud/ local device's storage (for uploading purpose)

3. System Features

3.1 LOGIN/ SIGNUP

3.1.1 Description and Priority

Priority = H

New User should be able to signup and create a new account.

Existing users can use their login credentials and operate their existing account.

3.1.2 Stimulus/Response Sequences

On entering valid username and password, login button is enabled and will take the user to the required page after displaying successfully logged in.

3.1.3 Functional Requirements

- Password must be of 6 characters minimum
- Display a message of invalid email-id /password if the above conditions do not satisfy.
- On successful login, take the user to the desired homepage.

3.2 Upload Image

3.2.1 Description and Priority

Priority = H

Users should be able to upload the image to the webpage from his/her local or cloud storage.

3.2.2 Stimulus/ResponseSequences

If the size exceeds or the format is unsupported, display an error message.

If the requirements are fulfilled, load the image in the work area.

3.2.3 Functional Requirements

- The maximum image size should be 10MB. Format supported (JPEG, PNG)
- Set the image in the work area according to the size of the work area on successful upload.
- If the load time exceeds the limit, display an error message
- After the image is properly loaded, enable the editing and enhancing options.
- If the user selects this option while there is already an image in the workspace, empty the workspace and load the newly selected image after displaying the confirmation message for the same.

3.3 CROP

3.3.1 Description and Priority

Priority = M

Users should be able to select the rectangular area of the image they want further.

The unselected area is cut off.

3.3.2 Stimulus/Response Sequences

Select a rectangular area of the image through mouse.

After selecting, crop the required area when user press the cropImage button.

3.3.3 Functional Requirements

- The selection area must have coordinates within the work area
- Defocus on the image area outside the selection.
- Enable save/revert option after first modification.

3.4 Rotate

3.4.1 Description & Priority

Priority = M

The user should be able to rotate the image at a specified angle.

3.4.2 Stimulus / Response sequences

The angle of rotation should be selected with the help of a slider.

On selection of the option, perform the corresponding functionality.

3.4.3 Functional Requirements

- The slider must be just outside the workspace.
- Enable save/revert option
- After the image is properly modified, enable the other editing and enhancing options

3.5 Tune Image

3.5.1 Description and Priority

Priority = M

Users should be able to select the options from the menu and the required change in the image should be seen.

The options include :

- Brightness
- Contrast
- Temperature
- Saturation
- Opacity

3.5.2 Stimulus/ResponseSequences

Display the menu and on selecting the option perform the corresponding actions.

The intensity of effects should be selected with the help of a slider.

On selection of the option, perform the corresponding functionality.

3.5.3 Functional Requirements

- The slider must be outside the workspace
- View the after effect of the option
- If the user uses the preset option, add the preset add the corresponding values to and update it in the database of the user.
- Enable save/revert option .
- After the image is properly modified, enable the other editing and enhancing options.

3.5A PRESET FEATURE

- Update and add the new presets to the database corresponding to the user.
- The values and related effects of the existing presets can be directly applied to various images.

3.6 DEHAZE

3.6.1 Description & Priority

Priority = M

User should be able to automatically get a dehazed image.

3.6.2 Stimulus/Response sequences

On selection of the option, perform the corresponding functionality

3.6.3 Functional Requirements

- Enable save/revert option after modification
- After the image is properly modified, enable the other editing and enhancing options

3.6 EXPORT IMAGE

3.6.1 Description and Priority

Priority = H

The user should be able to export the final resulting image in the desired jpg format to the desired location

3.6.2Stimulus/Response Sequences

On selection of the option, display the menu containing different formats.

Let the user provide the required location.

3.6.3 Functional Requirements

After the image is properly exported, display the message that the image is successfully exported.

4. Other Non-functional requirements

4.1 Performance Requirements

The website should load within 10 seconds at a normal condition (considering the internet speed and other things)

4.2 Security Requirements

The major security concern for any user is proper login in the account. For this purpose, a proper login mechanism should be used to avoid any form of hacking. Therefore, security is provided from the unwanted use of the software.

4.3 Safety Requirements

The transmission of information should be done securely and transmitted to the server without any change in the entered information.

4.4 Usability

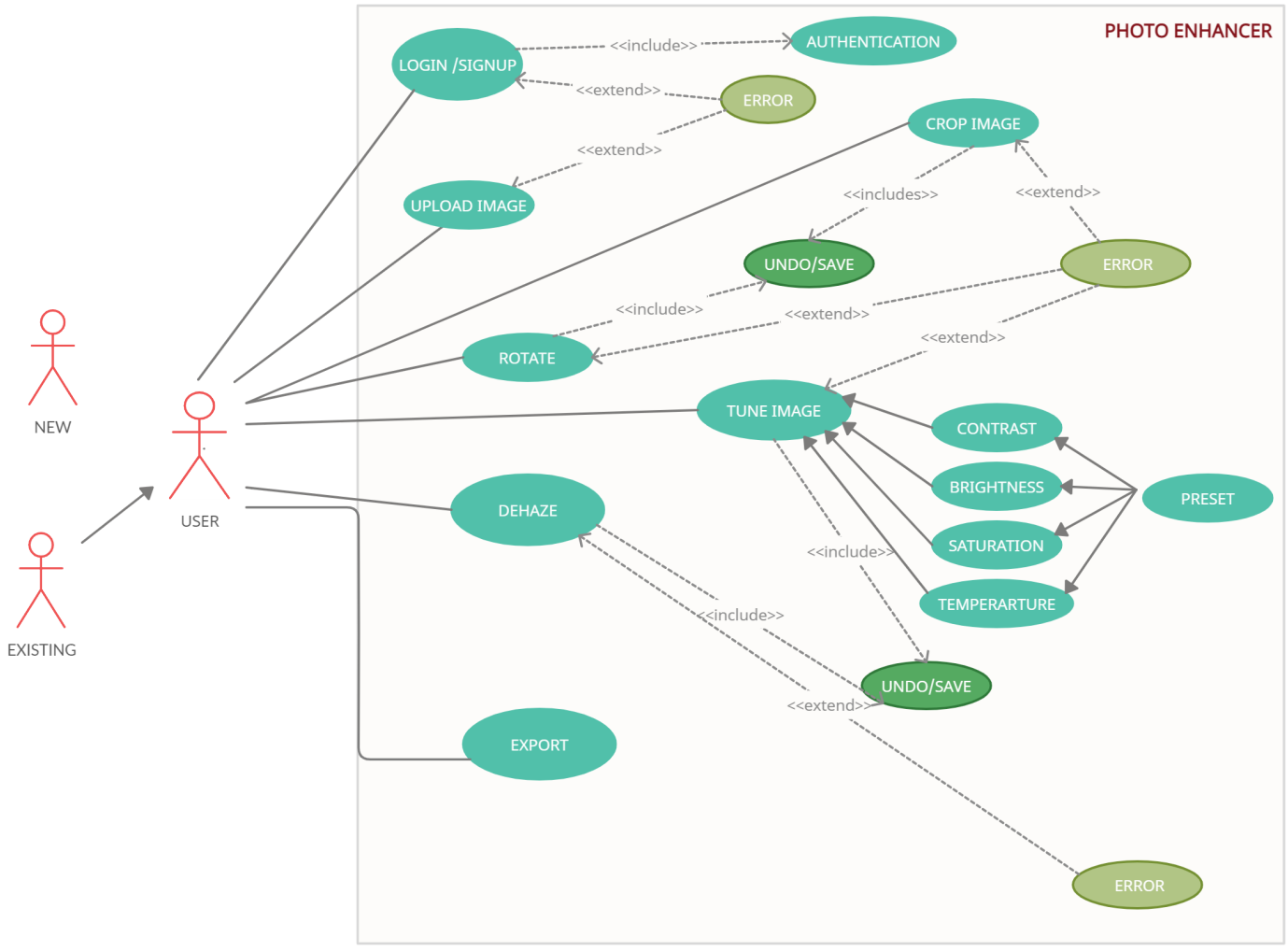
The UI of the webapp should be such that the colors do not harm the user's eyes negatively (contrast shouldn't be high) & also the webapp should be responsive

4.5 Availability

The webapp should be available 24*7 at any place.

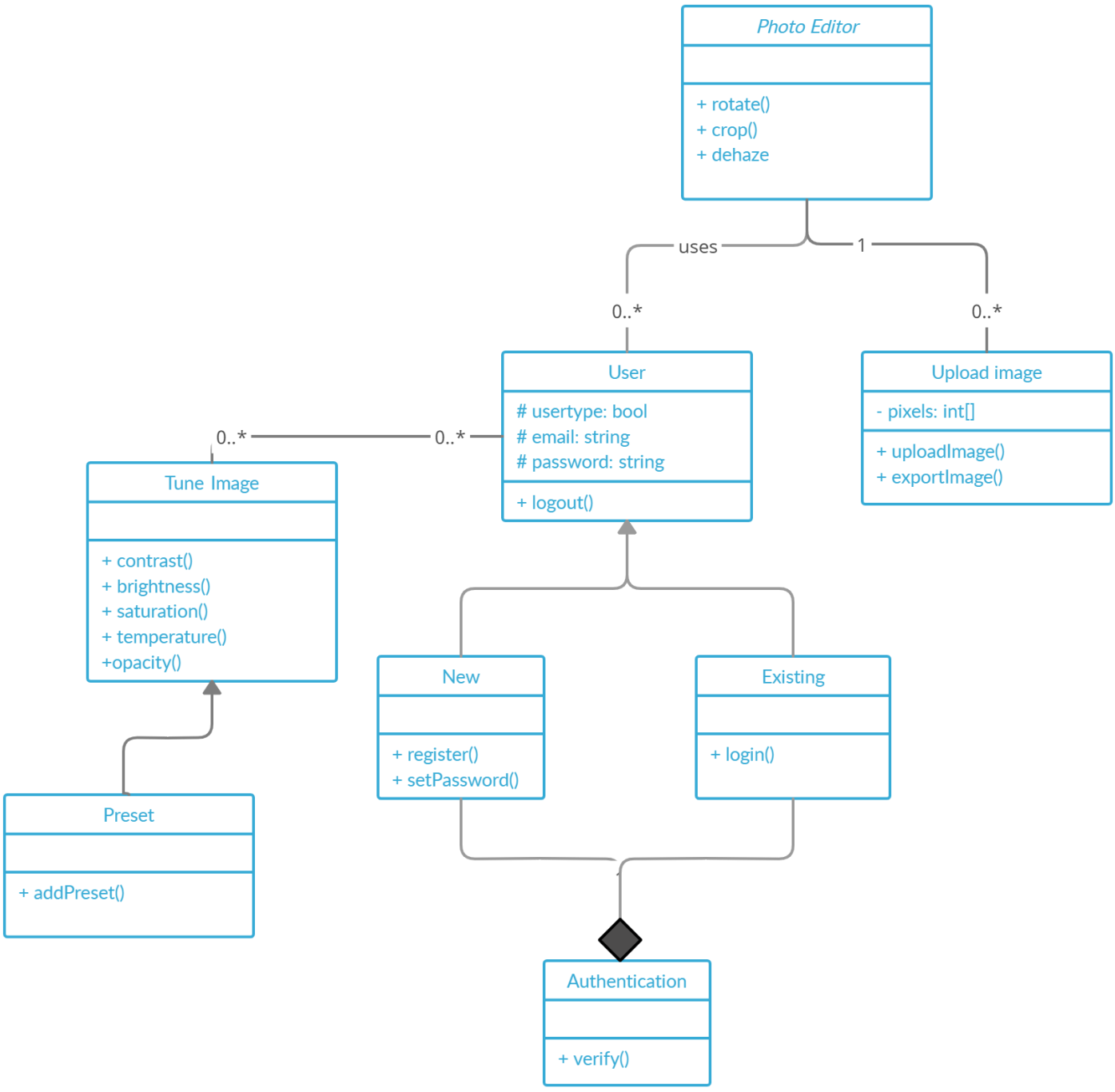
5. Analysis Models

5.1 Use-case Diagram

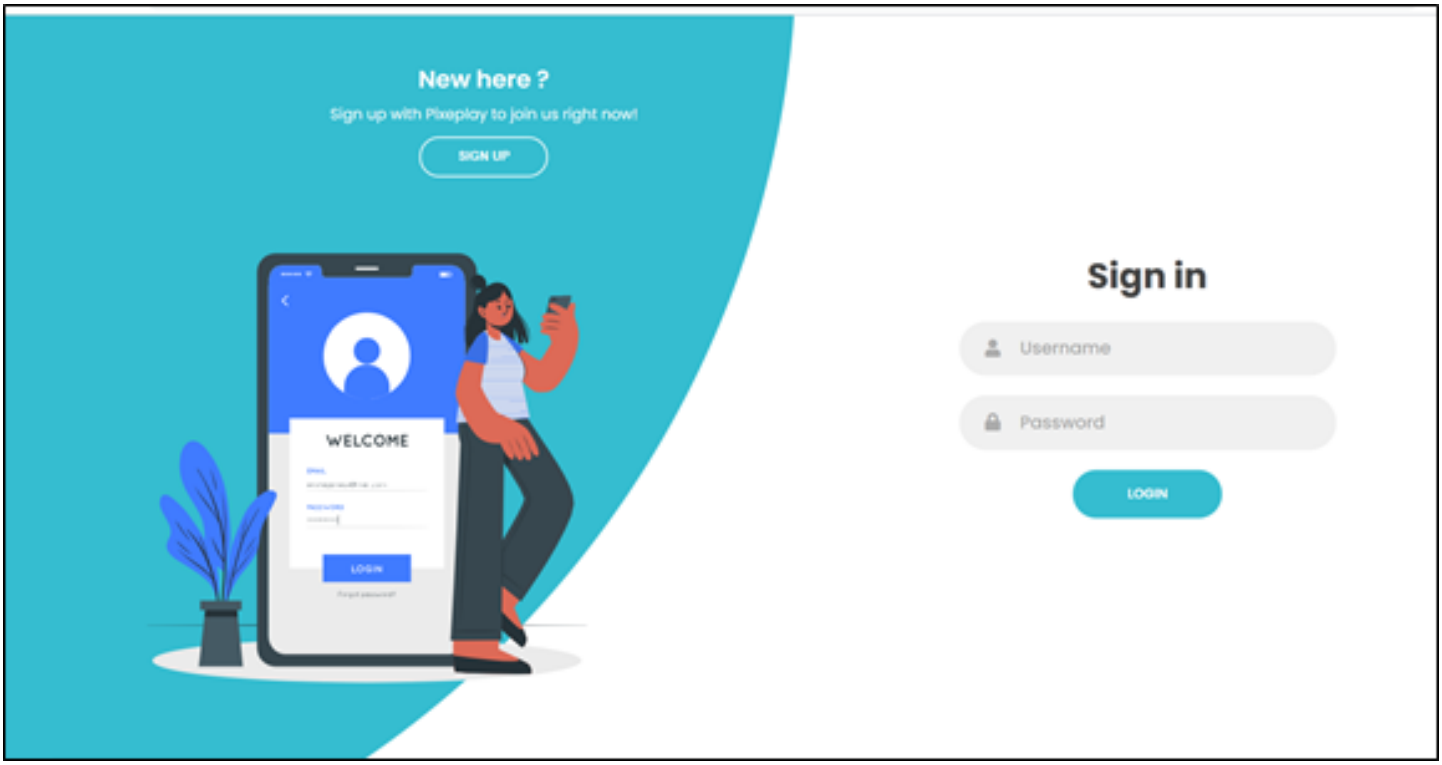


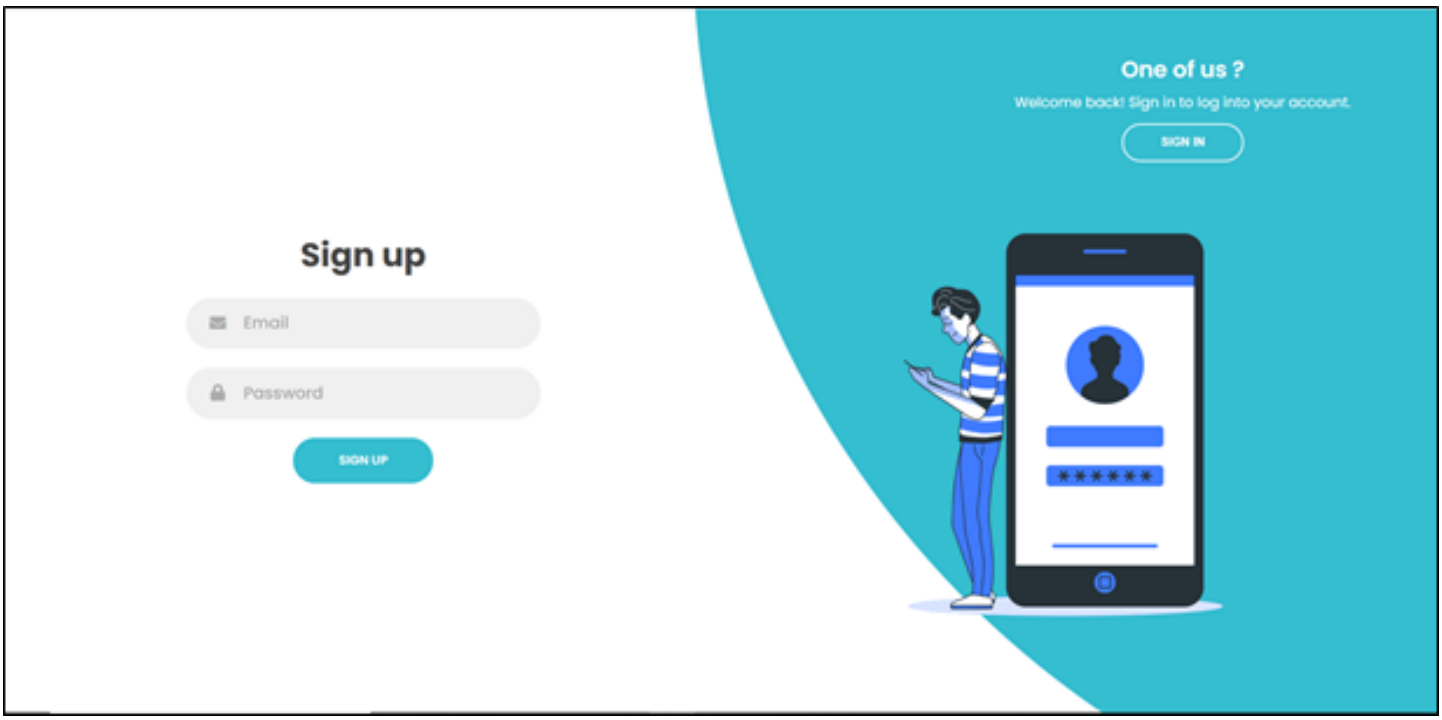
5. Analysis Models

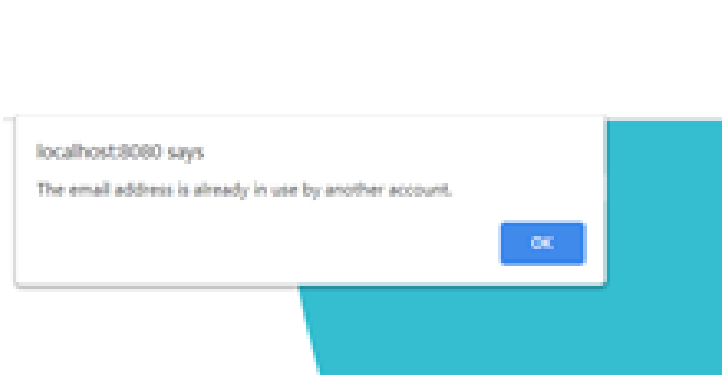
5.2 Class Diagram

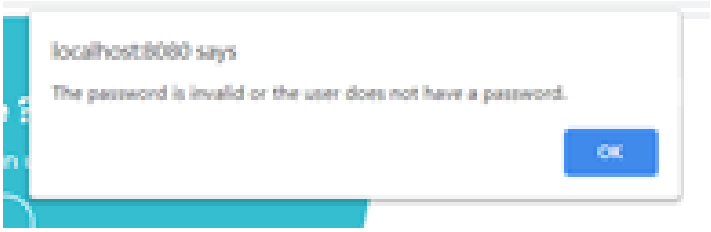


6. Implementation screenshots

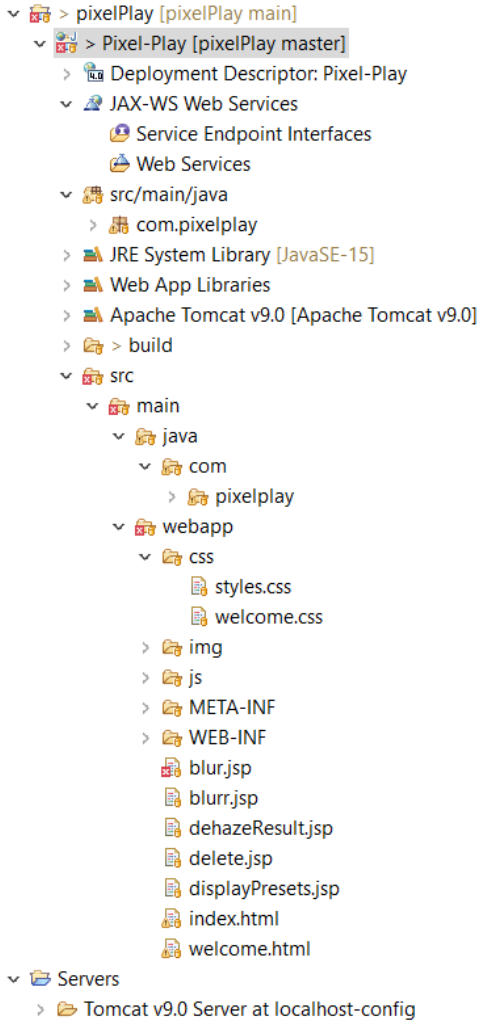








6. Implementation screenshots




```
    <script>
        // Your web app's Firebase configuration
        var firebaseConfig = {
            apiKey: "AIzaSyCPVbuGMZTknydebAoAmqwXH1JdbRI1AIg",
            authDomain: "pblproject-98357.firebaseapp.com",
            projectId: "pblproject-98357",
            storageBucket: "pblproject-98357.appspot.com",
            messagingSenderId: "1019786612975",
            appId: "1:1019786612975:web:13d4a0d629e0349c554b1a"
        };
        // Initialize Firebase
        firebase.initializeApp(firebaseConfig);
        const auth = firebase.auth();
    </script>
    <script src="./js/app.js"></script>
    <script src="./js/auth.js"></script>
</body>

</html>
```

6. Implementation screenshots

```
            </div>
        </div>

        <div class="panels-container">
            <div class="panel left-panel">
                <div class="content">
                    <h3>New here ?</h3>
                    <p>
                        Sign up with Pixeplay to join us right now!
                    </p>
                    <button class="btn transparent" id="sign-up-btn">
                        Sign up
                    </button>
                </div>
                <img src="img/log.svg" class="image" alt="" />
            </div>
            <div class="panel right-panel">
                <div class="content">
                    <h3>One of us ?</h3>
                    <p>
                        Welcome back! Sign in to log into your account.
                    </p>
                    <button class="btn transparent" id="sign-in-btn">
                        Sign in
                    </button>
                </div>
                <img src="img/register.svg" class="image" alt="" />
            </div>
```

```
@import url("https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600;700;800&display=swap

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body,
input {
  font-family: "Poppins", sans-serif;
}

.container {
  position: relative;
  width: 100%;
  background-color: #fff;
  min-height: 100vh;
  overflow: hidden;
}

.forms-container {
  position: absolute;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
}
```

6. Implementation screenshots

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

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <script src="https://kit.fontawesome.com/64d58efce2.js" crossorigin="anonymous"></script>
    <link rel="stylesheet" href="./css/styles.css" />
    <title>PixelPlay</title>
</head>

<body>
    <div class="container">
        <div class="forms-container">
            <div class="signin-signup">
                <form class="sign-in-form" id="signin-form">
                    <h2 class="title">Sign in</h2>
                    <div class="input-field">
                        <i class="fas fa-user"></i>
                        <input type="text" id="signin-email" placeholder="Username" />
                    </div>
                    <div class="input-field">
                        <i class="fas fa-lock"></i>
                        <input type="password" id="signin-password" placeholder="Password" />
                    </div>
                    <input type="submit" value="Login" class="btn solid" />
                </form>
                <form class="sign-up-form" id="signup-form">
```

```javascript
/*=========UPLOADING IMAGE TO CANVAS ELEMENT=============*/
var onload=function(e){
let imgInput = document.getElementById('imageInput');
  imgInput.addEventListener('change', function(e) {
    if(e.target.files) {
      let imageFile = e.target.files[0]; //here we get the image file
      var reader = new FileReader();
      reader.readAsDataURL(imageFile);
      reader.onloadend = function (e) {
       let myImage = new Image(); // Creates image object
        myImage.src = e.target.result; // Assigns converted image to image object
        myImage.onload = function(ev) {
          var myCanvas = document.getElementById("myCanvas"); // Creates a canvas object
          let myContext = myCanvas.getContext("2d"); // Creates a contect object
          myCanvas.width = myImage.width; // Assigns image's width to canvas
          myCanvas.height = myImage.height; // Assigns image's height to canvas
          myContext.drawImage(myImage,0,0); // Draws the image on canvas
          let imgData = myCanvas.toDataURL("image/jpeg",0.75); // Assigns image base64 string in jpeg forma
          ctx=myCanvas.getContext("2d");
          cPush();

          SessionImage.src=imgData;
```

6. Implementation screenshots

```
const signupForm = document.querySelector('#signup-form');
signupForm.addEventListener('submit',(e) => {
    e.preventDefault();

    const email = signupForm['signup-email'].value;
    const password = signupForm['signup-password'].value;

    auth.createUserWithEmailAndPassword(email,password).then(cred => {
        signupForm.reset();
        window.location.href="welcome.html";
    })
    .catch((error) => {
        var errorCode = error.code;
        var errorMessage = error.message;
        window.alert(errorMessage);
    })
});

const signinForm = document.querySelector('#signin-form');
signinForm.addEventListener('submit', (e) => {
    e.preventDefault();

    const email = signinForm['signin-email'].value;
    const password = signinForm['signin-password'].value;

    auth.signInWithEmailAndPassword(email,password).then(cred => {
        signinForm.reset();
        window.location.href="welcome.html";
    })
```

```
/*=========CROP IMAGE======*/
function crop()
{
  console.log("called");
  document.getElementById("crop").className.replace("","active");
  var section= document.getElementById("crop_section");
if (section.style.display == 'none') {
  section.style.display = 'block';
} else {
  section.style.display = 'none';
}

// initialize cropper by providing it with a target canvas and a XY ratio (height = width * ratio)
cropper.start(document.getElementById("myCanvas"), 1);
cropper.showImage(SessionImage.src);
cPush();
```

6. Implementation screenshots

```
let form1=$('preset_form');
$(document).on('submit','#preset_form',function(event){
  event.preventDefault();


    $.ajax({
      type: form1.attr('method'),
      url: form1.attr('action'),
      data:form1.serialize(),
      contentType:"json",

      success:function(data)
      {

        alert("Successfully added Preset!");
      },
      error:function()
      {
        alert("Failed to add preset");
      },

    })

});
```

```
(function(cropper, undefined) {
    "use strict"; // helps us catch otherwise tricky bugs

    /* DRAWING STUFF */
    var canvas;
    var context;

    var image;
    var restoreImage;
    var currentDimens = {};
    var cropping = false;

    var colors = {
        white: "#ffffff",
        black: "#000000",
        overlay: "rgba(0, 0, 0, 0.6)"
    };

    var overlay;
```

6. Implementation screenshots

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@page import="java.util.Base64"%>
<%@ page import="com.pixelplay.DarkChannelPriorDehazing" %>
<%@ page import="com.pixelplay.Filter" %>
<%@ page import="java.awt.image.*" %>
<%@ page import="javax.imageio.*" %>
<%@ page import="java.io.*" %>
<%@ page import="org.opencv.core.*" %>

<%@ page import=" org.opencv.imgcodecs.*,
 org.opencv.imgproc.*" %>
<%@ page import="com.pixelplay.DarkChannelPriorDehazing" %>

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Dehaze Effect</title>
</head>
<body>
<%

String path="";
path=(String)(request.getAttribute(path));
 final double krnlRatio = 0.01; // set kernel ratio
 final double eps = 0.000001;
 DarkChannelPriorDehazing d ;
//File f = new File(path);
```

```
String path="";
path=(String)(request.getAttribute(path));
 final double krnlRatio = 0.01; // set kernel ratio
 final double eps = 0.000001;
 DarkChannelPriorDehazing d ;
//File f = new File(path);
//BufferedImage b_in = ImageIO.read(f);
Mat image = Imgcodecs.imread(path, Imgcodecs.IMREAD_COLOR);
double minAtmosLight = 240.0; // set minimum atmospheric light
Mat outval = d.darkChannelDehazing(path, krnlRatio, minAtmosLight, eps);
BufferedImage bufImage = null;
int type = BufferedImage.TYPE_BYTE_GRAY;
if (outval.channels() > 1) {
    type = BufferedImage.TYPE_3BYTE_BGR;
}
int bufferSize = outval.channels() * outval.cols() * outval.rows();
byte[] b = new byte[bufferSize];
outval.get(0, 0, b); // get all the pixels
bufImage = new BufferedImage(outval.cols(), outval.rows(), type);
final byte[] targetPixels = ((DataBufferByte) bufImage.getRaster().getDataBuffer()).getData();
System.arraycopy(b, 0, targetPixels, 0, b.length);
ByteArrayOutputStream output = new ByteArrayOutputStream();
ImageIO.write(bufImage, "jpg", output);
String b64 = Base64.getEncoder().encodeToString(output.toByteArray());
```

6. Implementation screenshots

```
            context.save();
            context.fillStyle = colors.black;
            context.strokeStyle = colors.white;
            context.fillRect(x, y, w, h);
            context.strokeRect(x, y, w, h);
            context.restore();
        }
    }

    function drawOverlay() {
        // draw the overlay using a path made of 4 trapeziums (ahem)
        context.save();

        context.fillStyle = colors.overlay;
        context.beginPath();

        context.moveTo(0, 0);
        context.lineTo(overlay.x, overlay.y);
        context.lineTo(overlay.x + overlay.width, overlay.y);
        context.lineTo(canvas.width, 0);

        context.moveTo(canvas.width, 0);
        context.lineTo(overlay.x + overlay.width, overlay.y);
        context.lineTo(overlay.x + overlay.width, overlay.y + overlay.height);
```

```
        context.moveTo(canvas.width, 0);
        context.lineTo(overlay.x + overlay.width, overlay.y);
        context.lineTo(overlay.x + overlay.width, overlay.y + overlay.height);
        context.lineTo(canvas.width, canvas.height);

        context.moveTo(canvas.width, canvas.height);
        context.lineTo(overlay.x + overlay.width, overlay.y + overlay.height);
        context.lineTo(overlay.x, overlay.y + overlay.height);
        context.lineTo(0, canvas.height);

        context.moveTo(0, canvas.height);
        context.lineTo(overlay.x, overlay.y + overlay.height);
        context.lineTo(overlay.x, overlay.y);
        context.lineTo(0, 0);

        context.fill();

        context.restore();
    }
```

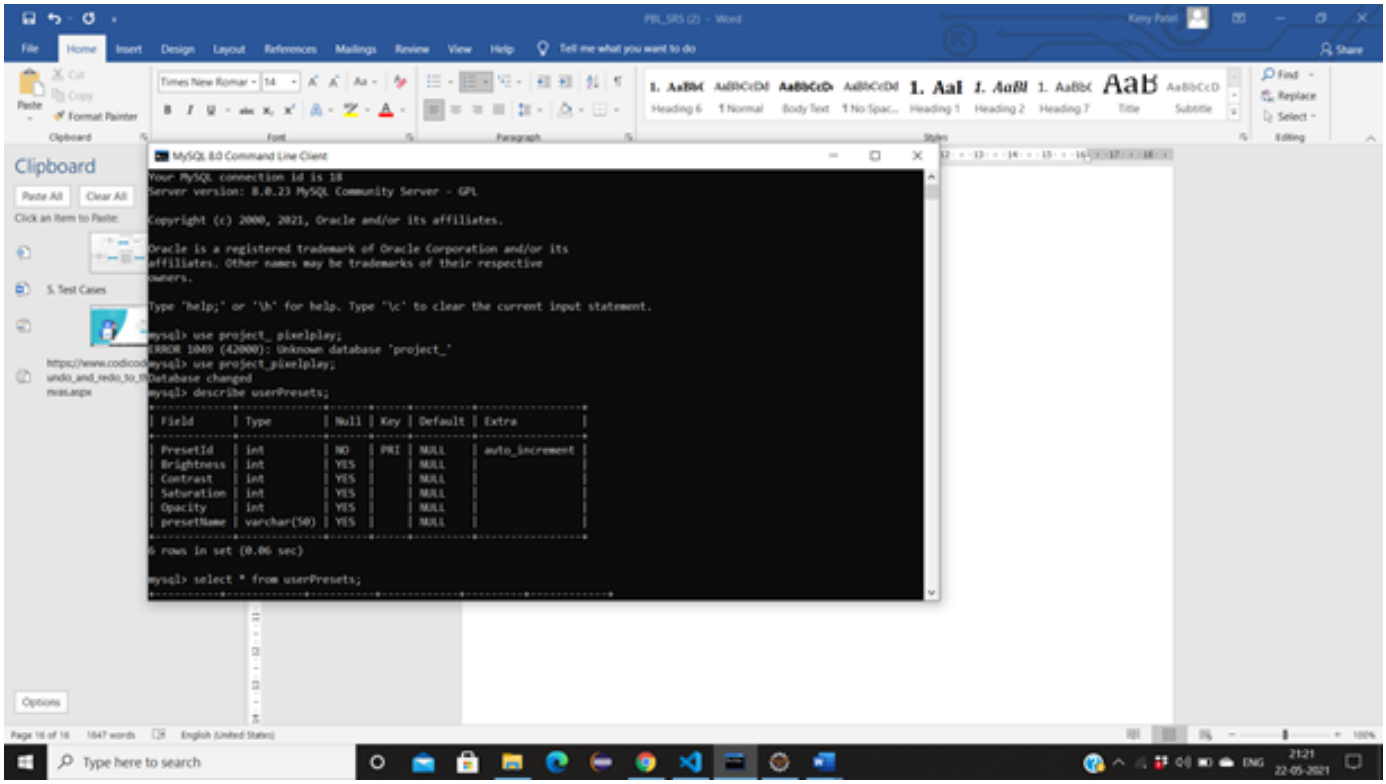
6. Implementation screenshots

```
/*============AJAX ADD IMAGE TO DB=================*/


let form = $('uploadForm');
$(document).on('submit', '#uploadForm', function(event){


    $.ajax({
type: form.attr('method'),
url: form.attr('action'),
data: form.serialize(),
success: function (responseText) {
var result=responseText;
$('#uploadResult').text(responseText.msg);
console.log("end");

}

});
 event.preventDefault();

});
```

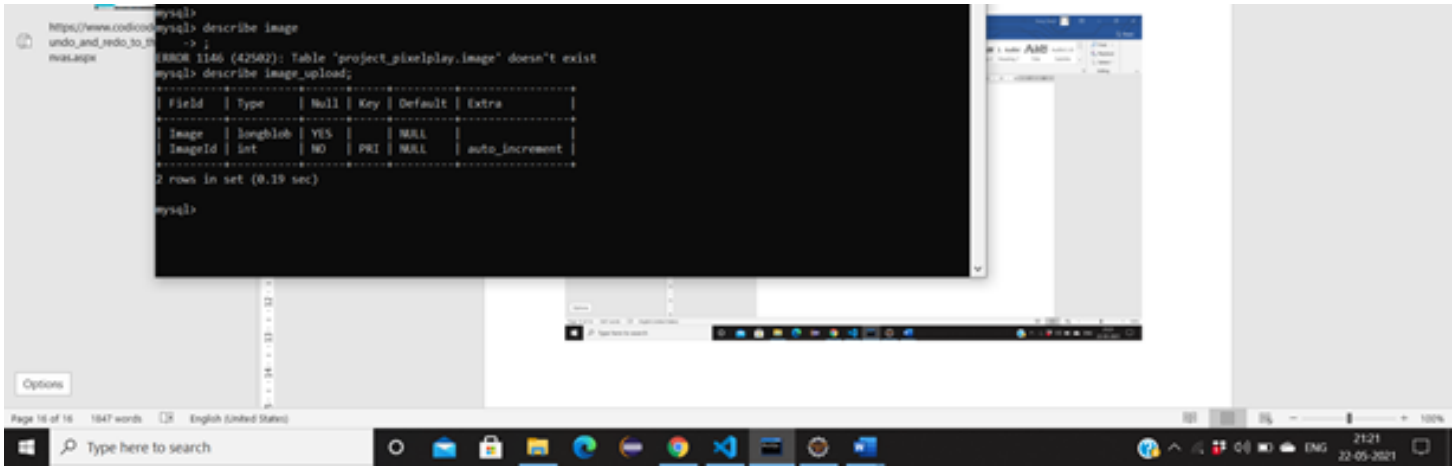
```
/*=========CROP IMAGE======*/
function crop()
{
  console.log("called");
  document.getElementById("crop").className.replace("","active");
  var section= document.getElementById("crop_section");
if (section.style.display == 'none') {
  section.style.display = 'block';
} else {
  section.style.display = 'none';
}

// initialize cropper by providing it with a target canvas and a XY ratio (height = width * ratio)
cropper.start(document.getElementById("myCanvas"), 1);
cropper.showImage(SessionImage.src);
cPush();
```

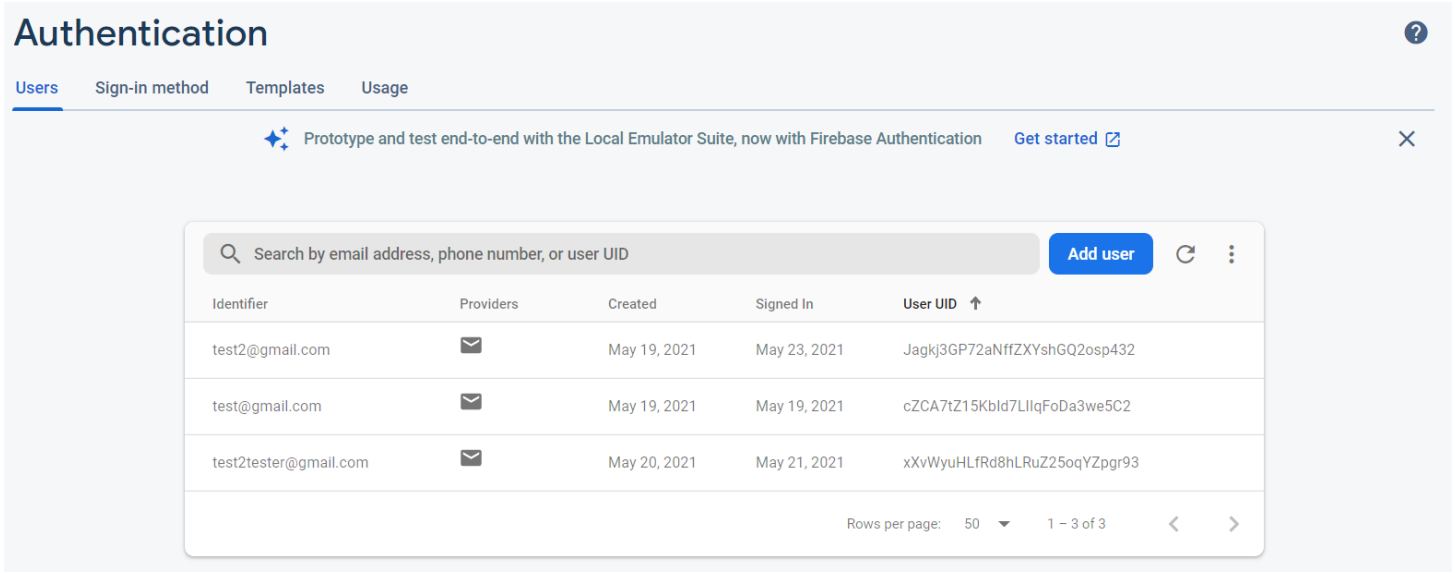
6. Implementation screenshots

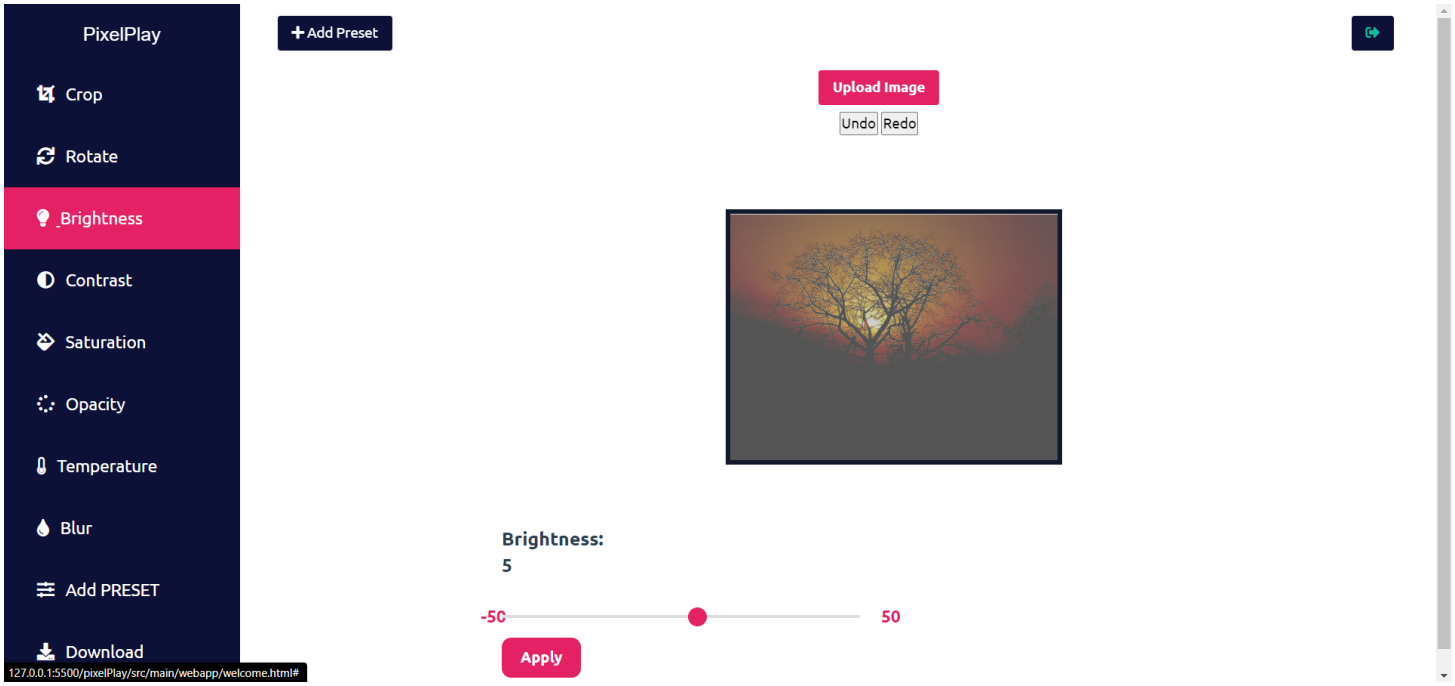
```
come.html   welcome.js   InsertImage.java   *addPreset.java   showPre
        {
            if(con != null){
                try{
                    con.close();
                }
                catch(Exception e){
                    e.printStackTrace();
                }
            }

        }
        String msg;
        if(result > 0){
            msg="Preset Added";
        }
        else{
            msg =" Some Error Occurred";
        }

        Gson gson=new Gson();
        JsonObject myObj = new JsonObject();
        JsonElement message= gson.toJsonTree(msg);
        myObj.add("msg", message);


        res.setContentType("application/json");

        res.setCharacterEncoding("utf-8");


        res.getWriter().write(myObj.toString());
```



6. Implementation screenshots







7. Project outcomes

Pixel Play provides a variety of features to users such as:

- Crop,
- Rotate Flip,
- Brightness,
- Contrast,
- Temperature,
- Saturation, etc.
- The preset option proves to be quite useful.
- Downloading edited image.

Learning outcomes

As a whole, it has been quite a good learning experience for us. Some of the major learning outcomes are as follows:

A basic understanding of image processing and what lies behind simple looking image manipulations.

The understanding of actual working of a web application in a client server architecture.

Functioning of client server communication integrated with database, HTTP protocols, servlets and glance of REST services.

Real- time application of algorithms and data structures.

Experiment with new tools and languages.

Major challenge faced:

Handling of images. It is very different from that of data in other formats.

Efficiently retrieving and maintaining data b/w client-server-database

8. Conclusion & Future Scope

Pixel Play is a basic photo enhancer web app aimed to improve the aesthetic and intricate quality by performing image manipulations.
The current implementation is at a quite basic level supporting primary features that satisfies the requirements of day-to-day image editing.
However, there remains quite a lot of scope for improvement.
As the project and its use grow, the 3-tier architecture can be shifted to MVC architecture for better efficiency.
Export of images in various formats can be added.
Other features like blur, filters, etc can also be implemented.
The project can be updated to a level higher by having extra features like searching presets and sharing of images.

References

https://www.programming-free.com/2012/08/ajax-with-jsp-and-servlet-using-jquery.html
https://www.youtube.com/watch?v=7TOmdDJc14s&list=PLsyeobzWxl7oGCz4k9VyxhfmQpSU1dV9b
https://www.codicode.com/art/undo_and_redo_to_the_html5_canvas.aspx
https://phg1024.github.io/image/processing/2014/02/24/ImageProcJS3.html

https://www.encodedna.com/html5/canvas/rotate-and-save-an-image-using-javascript-and-html5-canvas.htm
https://www.javatpoint.com/servlet-tutorial
https://towardsdatascience.com/all-in-one-image-dehazing-aod-paper-explanation-tensorflow-implementation-bb97f6a6f1ef