```
 function opening(app, event)
    if size(app.img, 3) == 3
        grayImage = rgb2gray(app.img);
    else
        grayImage = app.img;
    end
    se = strel('disk', 1);
    % Opening
    imageData = imopen(grayImage, se);
    imshow(imageData, [], 'Parent', app.UIAxes2);
 end


 function closing(app, event)
    if size(app.img, 3) == 3
        grayImage = rgb2gray(app.img);
    else
        grayImage = app.img;
    end
    se = strel('disk', 1);
    % Closing
    imageData = imclose(grayImage, se);
imshow(imageData, [], 'Parent', app.UIAxes2);
    end


 function histogram(app, event)
 grayImage = rgb2gray(app.img); % Ensure grayImage
  % Convert image to double
    if ~isa(grayImage, 'double')
  grayImage = im2double(grayImage); % Convert to double for processing
    end
    figure;
% Histogram of the original image
subplot(1, 2, 1);
imhist(grayImage);
```

Added the Histogram of the original image
<— —

```
title('Histogram of Original Image');
% Histogram of the equalized image
subplot(1, 2, 2);
imhist(equalizedImage);
title('Histogram of Equalized Image');
 end


 function Log(app, event)
    if size(app.img, 3) == 3
        grayImage = rgb2gray(app.img);
    else
        grayImage = app.img;
    end
     grayImage = im2double(grayImage);
    logImage = log(1 + grayImage);
    logImage = mat2gray(logImage);
    logImage = im2uint8(logImage);
    imshow(logImage, 'Parent', app.UIAxes2);
    title(app.UIAxes2, 'Logarithmic Filter');
    end



        function negative(app, event)
            if size(app.img, 3) == 3
                grayImage = rgb2gray(app.img);
            else
                grayImage = app.img;
            end
            negativeImage = 255 - grayImage;
            imshow(negativeImage, 'Parent', app.UIAxes2);
            title(app.UIAxes, 'Negative Filter');
        end
```

```
        function opening(app, event)

    if size(app.img, 3) == 3
        grayImage = rgb2gray(app.img);
    else
        grayImage = app.img;
    end

    se = strel('disk', 1);

    % Opening
    imageData = imopen(grayImage, se);
    imshow(imageData, [], 'Parent', app.UIAxes2);

        end
```

```
        function closing(app, event)

    if size(app.img, 3) == 3
        grayImage = rgb2gray(app.img);
    else
        grayImage = app.img;
    end

    se = strel('disk', 1);

    % Closing
    imageData = imclose(grayImage, se);
imshow(imageData, [], 'Parent', app.UIAxes2);

        end
```

```
        function histogram(app, event)
            grayImage = rgb2gray(app.img); % Ensure grayscale
     % Convert image to double precision if necessary
     if ~isa(grayImage, 'double')
         grayImage = im2double(grayImage); % Convert to double for processing
     end
     figure;

% Histogram of the original image
subplot(1, 2, 1);
imhist(grayImage);
title('Histogram of Original Image');

% Histogram of the equalized image
subplot(1, 2, 2);
imhist(equalizedImage);
title('Histogram of Equalized Image');
        end
```

```
        function Log(app, event)

    if size(app.img, 3) == 3
        grayImage = rgb2gray(app.img);
    else
        grayImage = app.img;
    end

    grayImage = im2double(grayImage);

    logImage = log(1 + grayImage);

    logImage = mat2gray(logImage);
    logImage = im2uint8(logImage);

    imshow(logImage, 'Parent', app.UIAxes2);
    title(app.UIAxes2, 'Logarithmic Filter');
        end
```

```
        function negative(app, event)

            if size(app.img, 3) == 3
                grayImage = rgb2gray(app.img);
            else
                grayImage = app.img;
            end
            negativeImage = 255 - grayImage;
            imshow(negativeImage, 'Parent', app.UIAxes2);
            title(app.UIAxes, 'Negative Filter');
        end
```