```
% Phase 1: Load the input image and convert it to grayscale
input_image = imread('dog.jpg');
grayscale_image = rgb2gray(input_image);  % Convert to grayscale if it's RGB

% Convert image to double precision for calculations
grayscale_image = double(grayscale_image);

% Get image size
[rows, cols] = size(grayscale_image);

% Phase 2: Apply the Fourier Transform
F = fft2(grayscale_image);  % Compute the 2D Fourier Transform
F_shifted = fftshift(F);  % Shift the zero frequency component to the center

% Create meshgrid for distance calculations in frequency domain
[u, v] = meshgrid(1:cols, 1:rows);
u = u - floor(cols/2);
v = v - floor(rows/2);
D = sqrt(u.^2 + v.^2);  % Distance from the center

% Phase 3: Apply Butterworth Filter
D0 = 50;  % Cutoff frequency
n = 2;    % Filter order
butterworth_filter = 1 ./ (1 + (D ./ D0).^(2*n));  % Butterworth filter
formula

% Apply Butterworth filter to the Fourier Transform of the image
F_butterworth = F_shifted .* butterworth_filter;

% Phase 4: Apply Gaussian Filter
sigma = 30;  % Standard deviation (controls cutoff frequency)
gaussian_filter = exp(-(D.^2) / (2 * sigma^2));  % Gaussian filter formula

% Apply Gaussian filter to the Fourier Transform of the image
F_gaussian = F_shifted .* gaussian_filter;

% Phase 5: Perform Inverse Fourier Transform to get the filtered images
% Butterworth Filtered Image
F_butterworth_shifted = ifftshift(F_butterworth);  % Shift back
image_butterworth = real(ifft2(F_butterworth_shifted));  % Inverse FFT

% Gaussian Filtered Image
F_gaussian_shifted = ifftshift(F_gaussian);  % Shift back
image_gaussian = real(ifft2(F_gaussian_shifted));  % Inverse FFT

% Phase 6: Display the Results
figure;
subplot(2, 2, 1), imshow(uint8(grayscale_image)), title('Original Image');
subplot(2, 2, 2), imshow(log(1 + abs(F_shifted)), []), title('Fourier
Transform');
```

```
subplot(2, 2, 3), imshow(uint8(image_butterworth)), title('Butterworth
Filtered Image');
subplot(2, 2, 4), imshow(uint8(image_gaussian)), title('Gaussian Filtered
Image');
```