```
from PIL import Image

def width_to_char(width):
    """Maps bar width to corresponding character (1-indexed)."""
    if width == 1:
        return " "  # A width of 1 corresponds to a space
    elif 2 <= width <= 27:
        # Calculate and return the corresponding uppercase character
        return chr(width + ord('A') - 2)  # Convert width to a character (A-Z)
    else:
        return ''  # Return an empty string for invalid widths

def format_decoded_string(decoded_str):
    """Formats the decoded string to capitalize the first letter of each word."""
    # Split the decoded string into individual words
    words = decoded_str.split(' ')
    # Capitalize the first letter of each word
    formatted_words = [word.capitalize() for word in words]
    # Join the capitalized words back into a single string with spaces
    return ' '.join(formatted_words)

def decode_barcode(image_path):
    # Step 1: Open the specified image (output.png)
    image = Image.open(image_path)
    width, height = image.size  # Get the dimensions of the image
    print(width,height)
    # Step 2: Define the row to scan for barcode data
    row_to_scan = 200
    pixels = image.load()  # Load pixel data for manipulation

    # Initialize variables for decoding
    Str = ""  # String to hold the decoded characters
    current_width = 0  # Width of the current bar
    in_bar = False  # Flag to indicate if currently in a bar

    # Step 3: Loop through each pixel in the specified row to calculate bar widths
    for x in range(width):
        # Check the color of the current pixel
        if pixels[x, row_to_scan] == (0, 0, 0):  # If the pixel is black (part of a
bar)
            if not in_bar:
                # Start of a new bar
                in_bar = True
                current_width = 1  # Reset current width for the new bar
            else:
                # Increment the width of the current bar
                current_width += 1
        else:  # If the pixel is white (indicating a space)
            if in_bar:
                # End of the current bar
                char = width_to_char(current_width)  # Map width to character
                Str += char  # Append the character (or space) to the result string
                in_bar = False  # Reset the flag for the next bar
                current_width = 0  # Reset the current width for the next bar

    # Check if the last bar ends at the end of the image
    if in_bar:
        char = width_to_char(current_width)  # Map the last bar width to character
        Str += char  # Append the last character (or space)
```

```
    return Str  # Return the fully decoded string

# Example usage of the decoding function
decoded_string = decode_barcode('output.png')  # Decode the barcode from the image

#\\\\\\\\\\\\\\\This is the function to format the string if we dont want this then
it prints the output as 'ABBAS CHEDDAD'
formatted_string = format_decoded_string(decoded_string)  # Format the decoded
string
print("Decoded String:", decoded_string)
print("Fromatted and Decoded String:", formatted_string)
 # Print the formatted decoded string
```