**Kursi Plastic Object Identifier – Web-Based Conveyor Belt Classifier**

**NAME: AYESHA ISMAIL**

**ENROLLMENT NO: 02-235232-008**

**SEMESTER: BS-IT 4A**

**COURSE: CRITICAL THINKING**

**GitHub Link:**

https://github.com/Ayeshaismail5/kursi-object-identifier

**Objective**

The goal of this project is to simulate a real-world sorting system for plastic recycling by building a **web-based object classification tool**. The application classifies plastic objects as:

* **Black** → Sent to **Conveyor Belt A**
* **Transparent** → Sent to **Conveyor Belt B**
* **Colorful** → Sent to **Conveyor Belt C**

This directly supports **Kursi’s mission** of sustainable recycling through efficient and automated separation of materials.

**Approach & Logic**

We developed the system using **HTML, CSS, and JavaScript** instead of Python to make it user-friendly and easily deployable on the web.

**How It Works:**

1. **User uploads an image** of a plastic object.
2. The system **analyzes the image** using the Canvas API in JavaScript.
3. It calculates:
   * **Average brightness**
   * **Color diversity**
   * **Color saturation**
4. Based on defined thresholds, it **assigns the correct conveyor belt**.

**Detection Logic Used:**

| **Type** | **Condition** |
| --- | --- |
| **Black** | Average brightness < 60 AND color variation < 25 |
| **Transparent** | Brightness > 180 AND saturation < 15 |
| **Colorful** | Color variety > 100 |

These rules are derived using basic visual processing logic and simulate how a machine might classify objects based on how light or colorful they appear.

**Technologies Used**

* **HTML5** – Interface layout
* **CSS3** – Professional styling
* **JavaScript (Canvas API)** – Image analysis
* **No backend or frameworks** – Simple and deployable

**Dataset Structure**

We manually captured images for testing and organized them as:

-assets

* black
* transparent
* colorful

Each folder contains 5-10 sample images downloaded from royalty-free image sites.

**Screenshots**

1. **Code**

**Index.htm**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Kursi Object Identifier</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <div class="container">

    <h1>Kursi Object Identifier</h1>

    <p>Upload an image of a plastic object to detect and assign the conveyor belt.</p>

    <label class="upload-btn">

      Choose Image

      <input type="file" id="imageInput" accept="image/\*">

    </label>

    <canvas id="canvas" style="display:none;"></canvas>

    <div id="preview"></div>

    <div id="result"></div>

  </div>

  <script src="script.js"></script>

</body>

</html>

**style.css**

\* {

  box-sizing: border-box;

  margin: 0;

  padding: 0;

}

body {

  font-family: 'Segoe UI', sans-serif;

  background: linear-gradient(to right, #e3f2fd, #ffffff);

  min-height: 100vh;

  display: flex;

  justify-content: center;

  align-items: center;

  padding: 20px;

}

.container {

  background-color: #ffffff;

  padding: 40px 30px;

  border-radius: 16px;

  box-shadow: 0 12px 25px rgba(0, 0, 0, 0.1);

  text-align: center;

  max-width: 480px;

  width: 100%;

}

h1 {

  color: #1976d2;

  margin-bottom: 20px;

}

p {

  font-size: 16px;

  margin-bottom: 25px;

  color: #555;

}

.upload-btn {

  background-color: #1976d2;

  color: white;

  padding: 12px 20px;

  border-radius: 30px;

  cursor: pointer;

  font-size: 16px;

  display: inline-block;

  margin-bottom: 20px;

  transition: background 0.3s;

  position: relative;

  overflow: hidden;

}

.upload-btn input[type="file"] {

  opacity: 0;

  position: absolute;

  left: 0;

  top: 0;

  height: 100%;

  width: 100%;

  cursor: pointer;

}

.upload-btn:hover {

  background-color: #0d47a1;

}

#preview img {

  margin-top: 20px;

  max-width: 100%;

  border-radius: 10px;

  border: 1px solid #ccc;

}

#result {

  margin-top: 25px;

  font-size: 20px;

  color: #333;

  font-weight: bold;

}

**script.js**

document.getElementById('imageInput').addEventListener('change', function (event) {

  const file = event.target.files[0];

  if (!file) return;

  const img = new Image();

  const canvas = document.getElementById('canvas');

  const ctx = canvas.getContext('2d');

  const resultDiv = document.getElementById('result');

  const preview = document.getElementById('preview');

  const reader = new FileReader();

  reader.onload = function (e) {

    img.src = e.target.result;

    preview.innerHTML = `<img src="${e.target.result}" alt="Preview Image">`;

  };

  img.onload = function () {

    canvas.width = 100;

    canvas.height = 100;

    ctx.drawImage(img, 0, 0, 100, 100);

    const imageData = ctx.getImageData(0, 0, 100, 100);

    const data = imageData.data;

    let totalBrightness = 0;

    let colorCount = {};

    let totalSaturation = 0;

    for (let i = 0; i < data.length; i += 4) {

      const r = data[i];

      const g = data[i + 1];

      const b = data[i + 2];

      // Brightness

      const brightness = (r + g + b) / 3;

      totalBrightness += brightness;

      // RGB to HSL to get saturation

      const max = Math.max(r, g, b);

      const min = Math.min(r, g, b);

      const delta = max - min;

      const l = (max + min) / 2;

      let s = 0;

      if (delta !== 0) {

        s = delta / (255 - Math.abs(2 \* l - 255));

      }

      totalSaturation += s \* 100;

      const key = `${r >> 4}-${g >> 4}-${b >> 4}`;

      colorCount[key] = (colorCount[key] || 0) + 1;

    }

    const numPixels = data.length / 4;

    const avgBrightness = totalBrightness / numPixels;

    const avgSaturation = totalSaturation / numPixels;

    const numColors = Object.keys(colorCount).length;

    let belt = "";

    if (avgBrightness < 60 && numColors < 25) {

      belt = "A (Black)";

    } else if (avgBrightness > 180 && avgSaturation < 15) {

      belt = "B (Transparent)";

    } else if (numColors > 100) {

      belt = "C (Colorful)";

    } else {

      belt = "Uncertain – Needs Manual Check";

    }

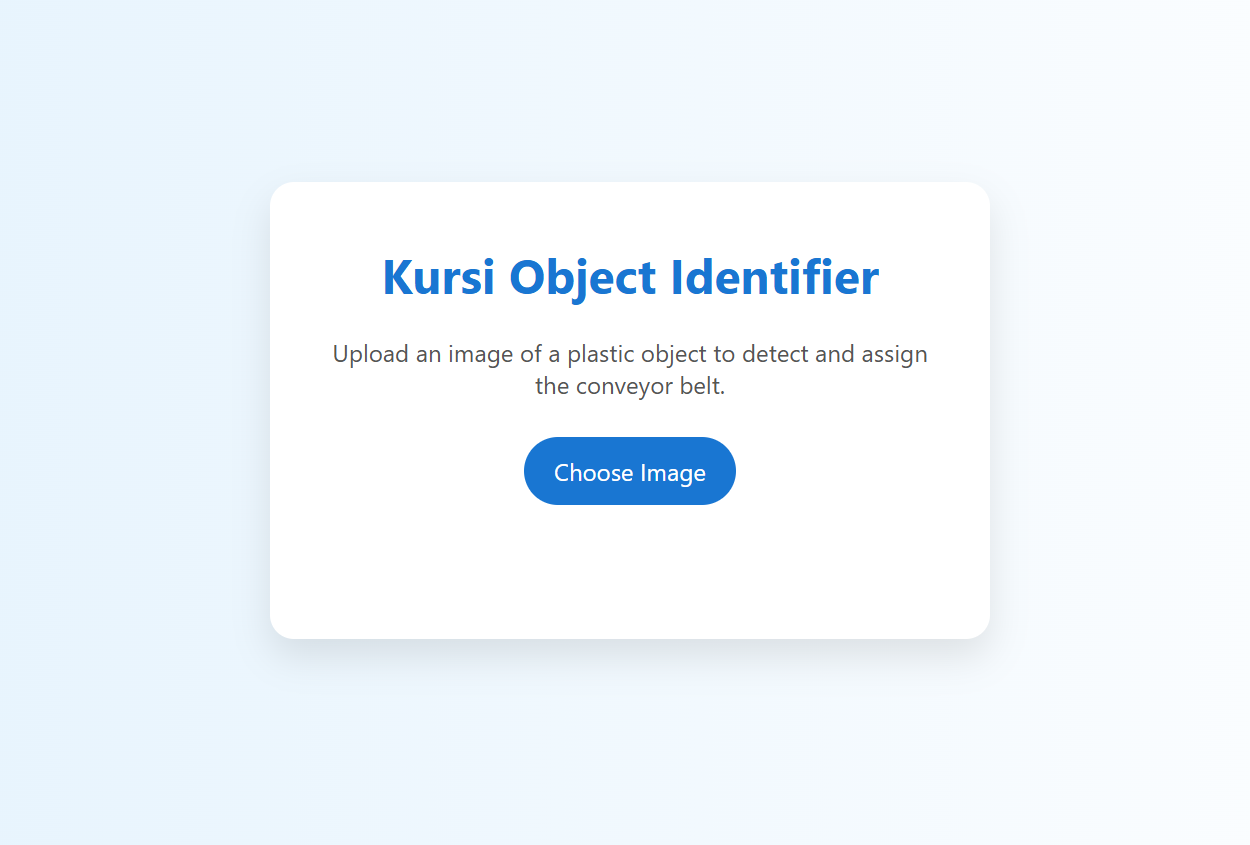
    resultDiv.innerHTML = `Assigned to Conveyor Belt: <strong>${belt}</strong>`;

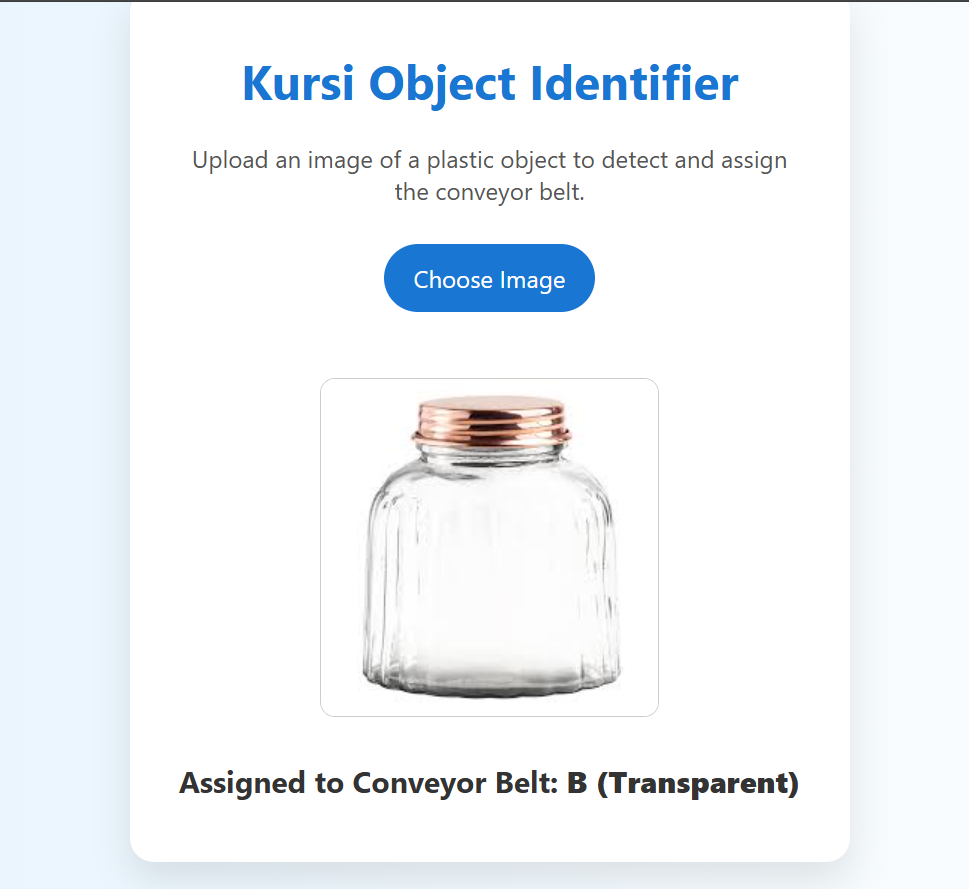
  };

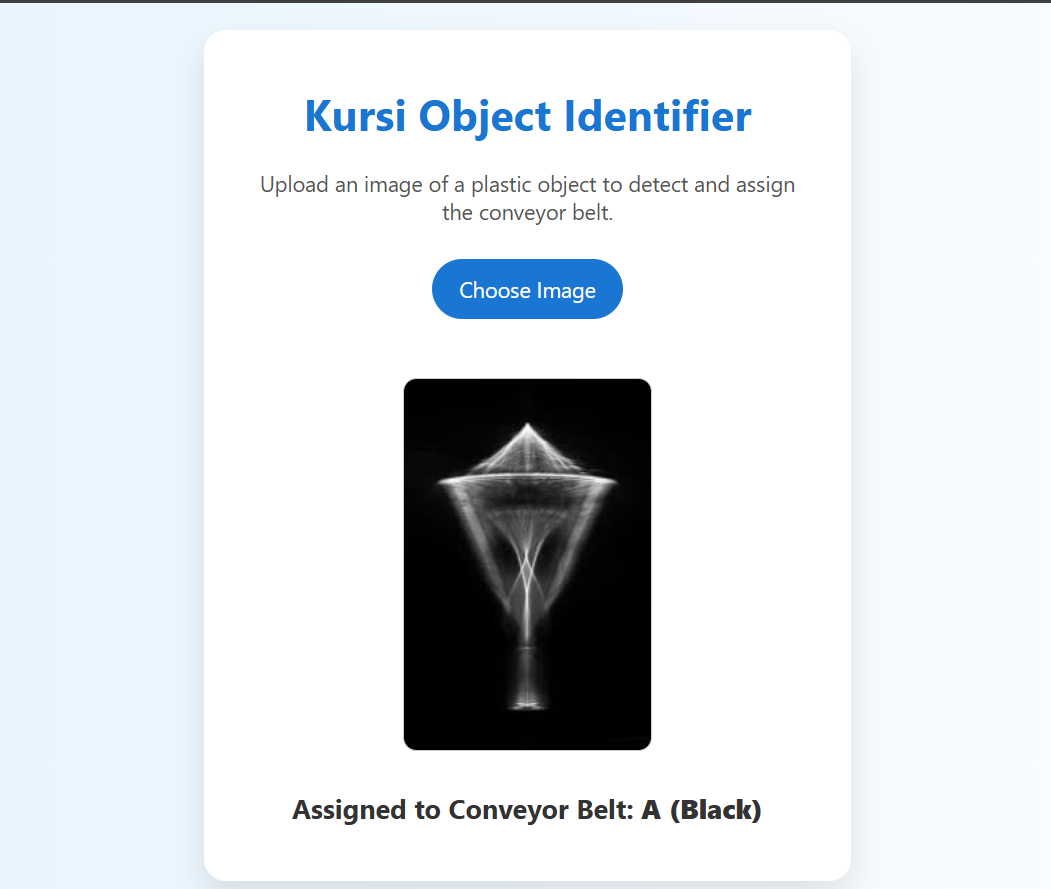
  reader.readAsDataURL(file);

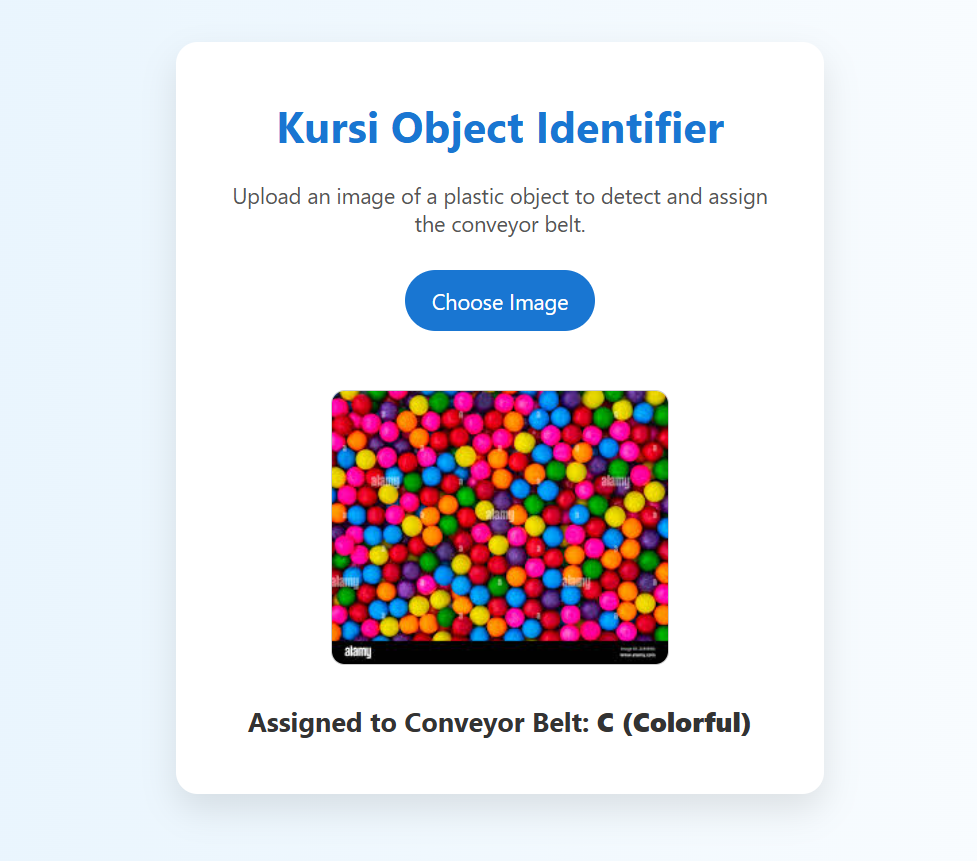
});

**2. Result Display**









**Challenges & Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Accurate detection of **black vs. colorful** when images have shadows or mixed pixels | Implemented precise thresholds using both brightness and color count |
| **Transparent objects** may look white on white background | Used brightness + saturation checks to distinguish true transparency |
| Keeping the interface **user-friendly and mobile responsive** | Designed with modern CSS and minimal inputs for accessibility |

**Outcomes**

* Developed a fully functional and visually appealing **web app** for object classification.
* Implemented **non-AI** logic that simulates basic machine vision behavior.
* Learned how **image data can be analyzed** with just browser tools, without needing Python/OpenCV.

**Conclusion**

This project demonstrates how rule-based logic can effectively address real-world sustainability problems like plastic recycling. With minimal tools, we created a simple, scalable, and interactive solution that aligns with Kursi’s mission: **turning waste into worth**.