```
#include <WiFi.h>
#include <WebServer.h>

// Motor pins
const int MotorA1 = 12;
const int MotorA2 = 14;
const int MotorB1 = 22;
const int MotorB2 = 23;
const int MotorC1 = 0;
const int MotorC2 = 4;

// Wi-Fi credentials
const char* ssid = "aicli";
const char* password = "sweetpont370";

WebServer server(80);

// Tracking status and counts
String currentStatus = "Idle";
int shuffleCount = 0;
int distributeCount = 0;
int autoShuffleCount = 0;

void stopMotors() {
 digitalWrite(MotorA1, LOW);
 digitalWrite(MotorA2, LOW);
 digitalWrite(MotorB1, LOW);
 digitalWrite(MotorB2, LOW);
 digitalWrite(MotorC1, LOW);
 digitalWrite(MotorC2, LOW);
 currentStatus = "Stopped";
}

void handleRoot() {
 String html = R"rawliteral(
   <!DOCTYPE html>
   <html>
   <head>
     <title>card shuffler + distributor V1</title>
     <meta name="viewport" content="width=device-width, initial-scale=1">
     <style>
       body {
```

```
    font-family: Arial, sans-serif;
    text-align: center;
    background: #f2f2f2;
    margin-top: 40px;
  }
  h1 {
    font-size: 32px;
    color: #333;
  }
  button {
    background-color: #4CAF50;
    border: none;
    color: white;
    padding: 16px 32px;
    margin: 12px;
    font-size: 22px;
    cursor: pointer;
    border-radius: 8px;
    transition: background-color 0.3s ease;
  }
  button:hover {
    background-color: #45a049;
  }
  #stopBtn {
    background-color: #f44336;
  }
  #stopBtn:hover {
    background-color: #da190b;
  }
  #status {
    margin-top: 30px;
    font-size: 24px;
    font-weight: bold;
    min-height: 30px;
    color: #333;
  }
  #progressBar {
    width: 80%;
    height: 20px;
    background-color: #ddd;
    border-radius: 10px;
    margin: 20px auto;
```

```
        display: none;
      }
      #progress {
        height: 100%;
        width: 0%;
        background-color: #4CAF50;
        border-radius: 10px;
        transition: width 0.2s ease;
      }
      #counts {
        margin-top: 30px;
        font-size: 20px;
        color: #555;
      }
    </style>
    <script>
      function sendCommand(cmd) {
        document.getElementById("status").innerText = "Status: " +
cmd.charAt(0).toUpperCase() + cmd.slice(1) + "ing...";
        startProgressBar();

        fetch('/' + cmd).then(response => response.text()).then(text => {
          const [statusText, countsText] = text.split("||");
          document.getElementById("status").innerText = "Status: " + statusText;
          document.getElementById("counts").innerHTML = countsText;
          stopProgressBar();
        });
      }

      function startProgressBar() {
        const bar = document.getElementById("progressBar");
        const progress = document.getElementById("progress");
        bar.style.display = "block";
        progress.style.width = "0%";
        let width = 0;
        const interval = setInterval(() => {
          if (width >= 100) width = 0;
          width += 5;
          progress.style.width = width + "%";
        }, 100);
        bar.setAttribute("data-interval", interval);
      }
```

```
       function stopProgressBar() {
         const bar = document.getElementById("progressBar");
         const interval = bar.getAttribute("data-interval");
         clearInterval(interval);
         document.getElementById("progress").style.width = "100%";
         setTimeout(() => {
           bar.style.display = "none";
           document.getElementById("progress").style.width = "0%";
         }, 300);
       }
     </script>
   </head>
   <body>
     <h1>card shuffler + distributor V1</h1>
     <button onclick="sendCommand('shuffle')">Shuffle</button>
     <button onclick="sendCommand('distribute')">Distribute</button>
     <button id="stopBtn" onclick="sendCommand('stop')">Stop</button>
     <button onclick="sendCommand('autoshuffle')">Auto Shuffle (6s)</button>
     <div id="status">Status: Idle</div>
     <div id="progressBar"><div id="progress"></div></div>
     <div id="counts">
       Shuffle: 0 | Distribute: 0 | Auto Shuffle: 0
     </div>
   </body>
   </html>
 )rawliteral";

 server.send(200, "text/html", html);
}

void sendStatusAndCounts(const String& status) {
 String counts = "Shuffle: " + String(shuffleCount) +
                 " | Distribute: " + String(distributeCount) +
                 " | Auto Shuffle: " + String(autoShuffleCount);
 server.send(200, "text/plain", status + "||" + counts);
}

void handleShuffle() {
 Serial.println("Shuffle pressed");
 digitalWrite(MotorA1, LOW);
 digitalWrite(MotorA2, HIGH);
 digitalWrite(MotorB1, HIGH);
 digitalWrite(MotorB2, LOW);
 digitalWrite(MotorC1, LOW);
 digitalWrite(MotorC2, LOW);
 currentStatus = "Shuffling...";
 shuffleCount++;
 sendStatusAndCounts(currentStatus);
}

void handleDistribute() {
 Serial.println("Distribute pressed");
 digitalWrite(MotorA1, LOW);
 digitalWrite(MotorA2, LOW);
 digitalWrite(MotorB1, LOW);
 digitalWrite(MotorB2, LOW);
 digitalWrite(MotorC1, LOW);
 digitalWrite(MotorC2, HIGH);
 currentStatus = "Distributing...";
 distributeCount++;
 sendStatusAndCounts(currentStatus);
}

void handleAutoShuffle() {
 Serial.println("Auto Shuffle pressed");
 handleShuffle();  // Start shuffle
 autoShuffleCount++;

 // Schedule stop after 6 seconds
 delay(6000);
 stopMotors();
 currentStatus = "Auto Shuffle Complete";
 sendStatusAndCounts(currentStatus);
}

void handleStop() {
 Serial.println("Stop pressed");
 stopMotors();
 currentStatus = "Stopped";
 sendStatusAndCounts(currentStatus);
}

void setup() {
 Serial.begin(115200);

 pinMode(MotorA1, OUTPUT);
 pinMode(MotorA2, OUTPUT);
 pinMode(MotorB1, OUTPUT);
 pinMode(MotorB2, OUTPUT);
 pinMode(MotorC1, OUTPUT);
 pinMode(MotorC2, OUTPUT);
 stopMotors();

 WiFi.begin(ssid, password);
 Serial.print("Connecting to Wi-Fi");
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }
 Serial.println("\nConnected!");
 Serial.print("IP Address: ");
 Serial.println(WiFi.localIP());

 server.on("/", handleRoot);
 server.on("/shuffle", handleShuffle);
 server.on("/distribute", handleDistribute);
 server.on("/autoshuffle", handleAutoShuffle);
 server.on("/stop", handleStop);

 server.begin();
}

void loop() {
 server.handleClient();
}
```