



# **QPoint** **Robotic Solutions**

Web Based User  
Interface

Menu Structure

Any robot + Our products = **INSTANT AUTOMATION**



We didn't make the robot...  
**We made the robot EASY.**

# GOAL

- Replace the current code based menu structure with a web page based menu that allows
  - cleaner operator interaction
  - more attractive, and current appearance
  - Full functionality with main supervisory application
  - Requires ZERO connectivity with outside world (local machine operation, no www access)
- Menu structure is likely available from various code examples, don't want to pay any ongoing royalties, but one time purchases are not out of the question for a code base that is useful
- Web page must send and receive messages from supervisory application for data, and control response using java script. Examples provided
- System will need to run OFFLINE on a windows based PC, .Net container is CHROME based

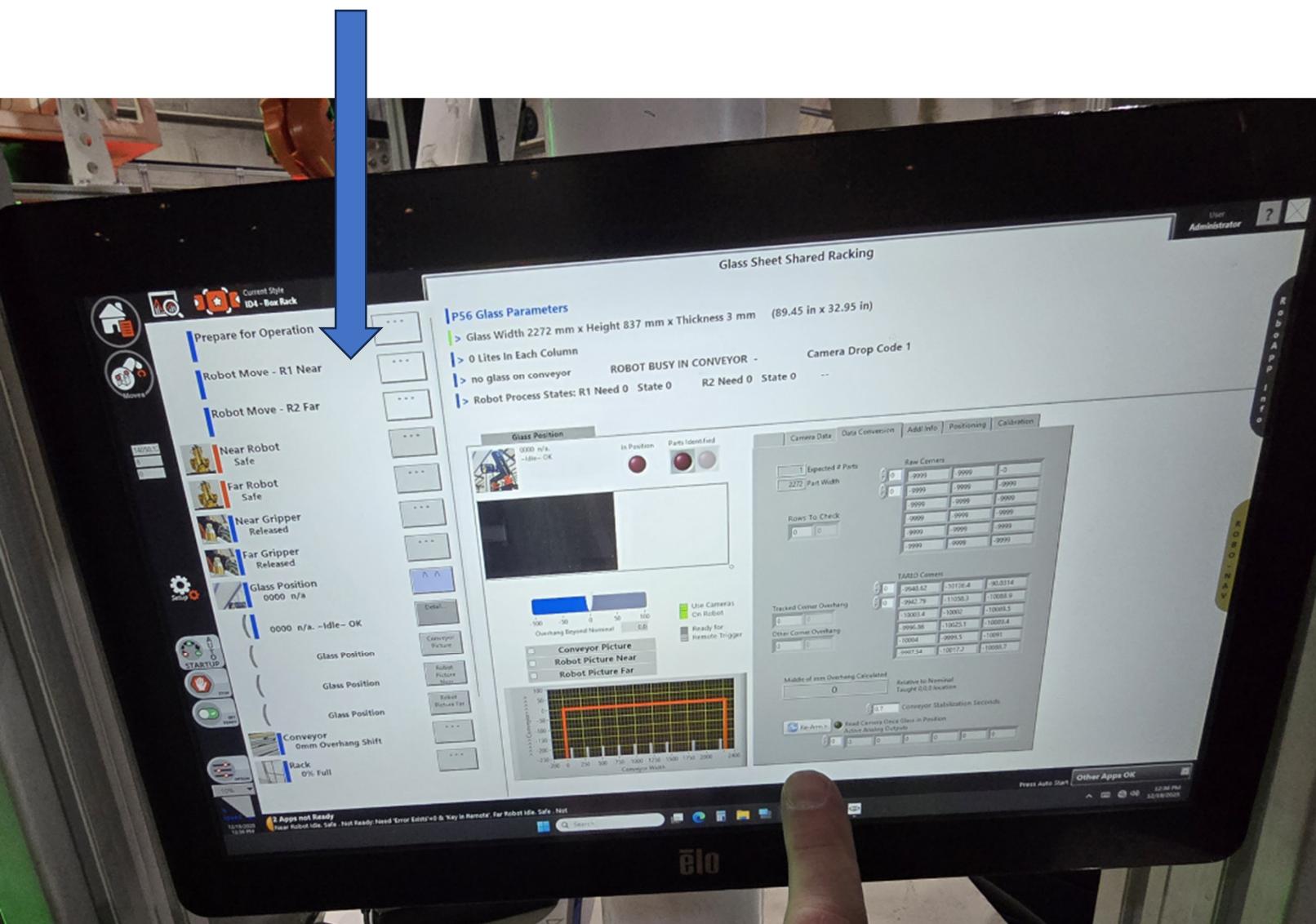
Any robot + Our products = **INSTANT AUTOMATION**

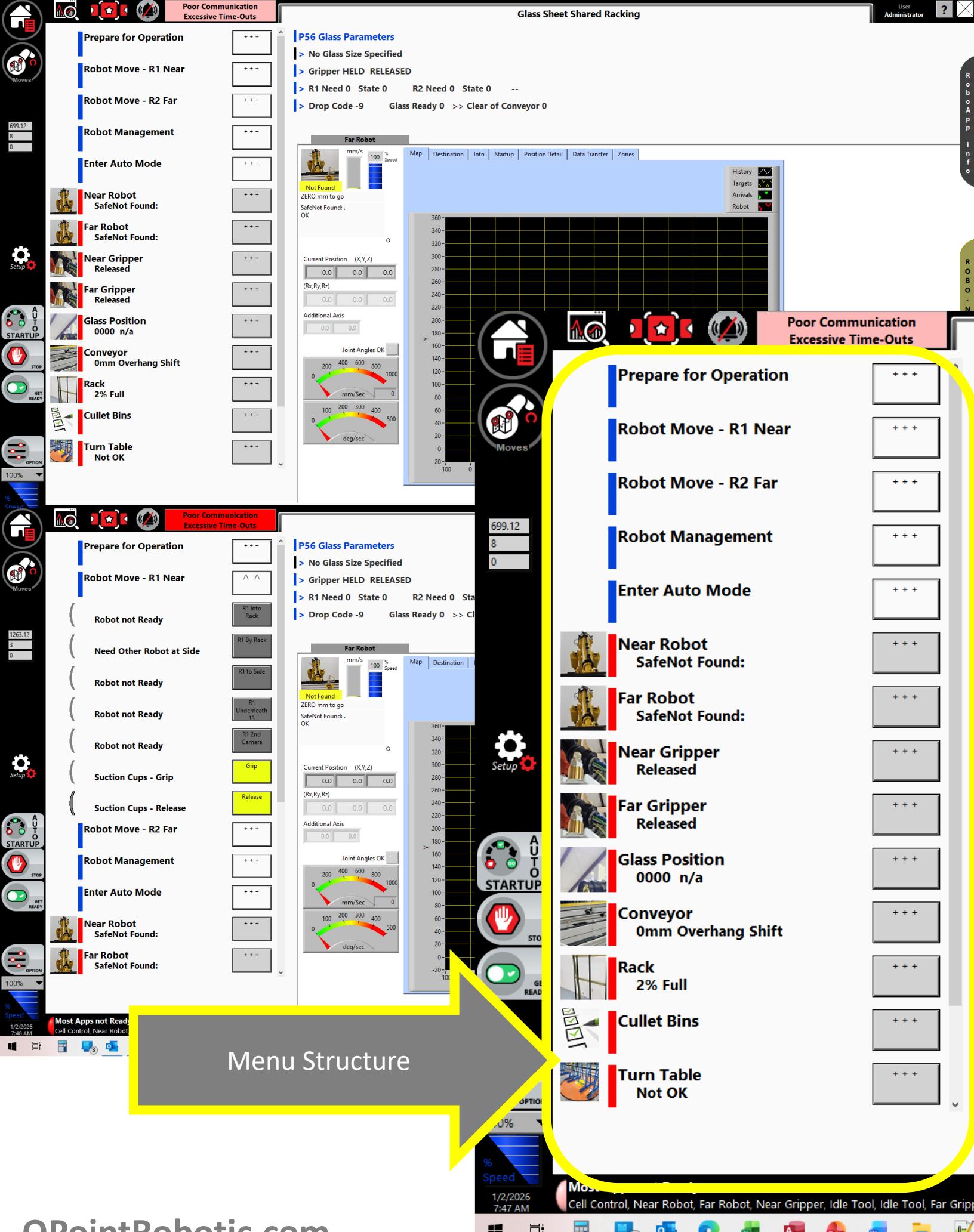
**QPoint**  
Robotic Solutions

We didn't make the robot...  
**We made the robot EASY.**

# Current Functionality

- Left hand menu structure

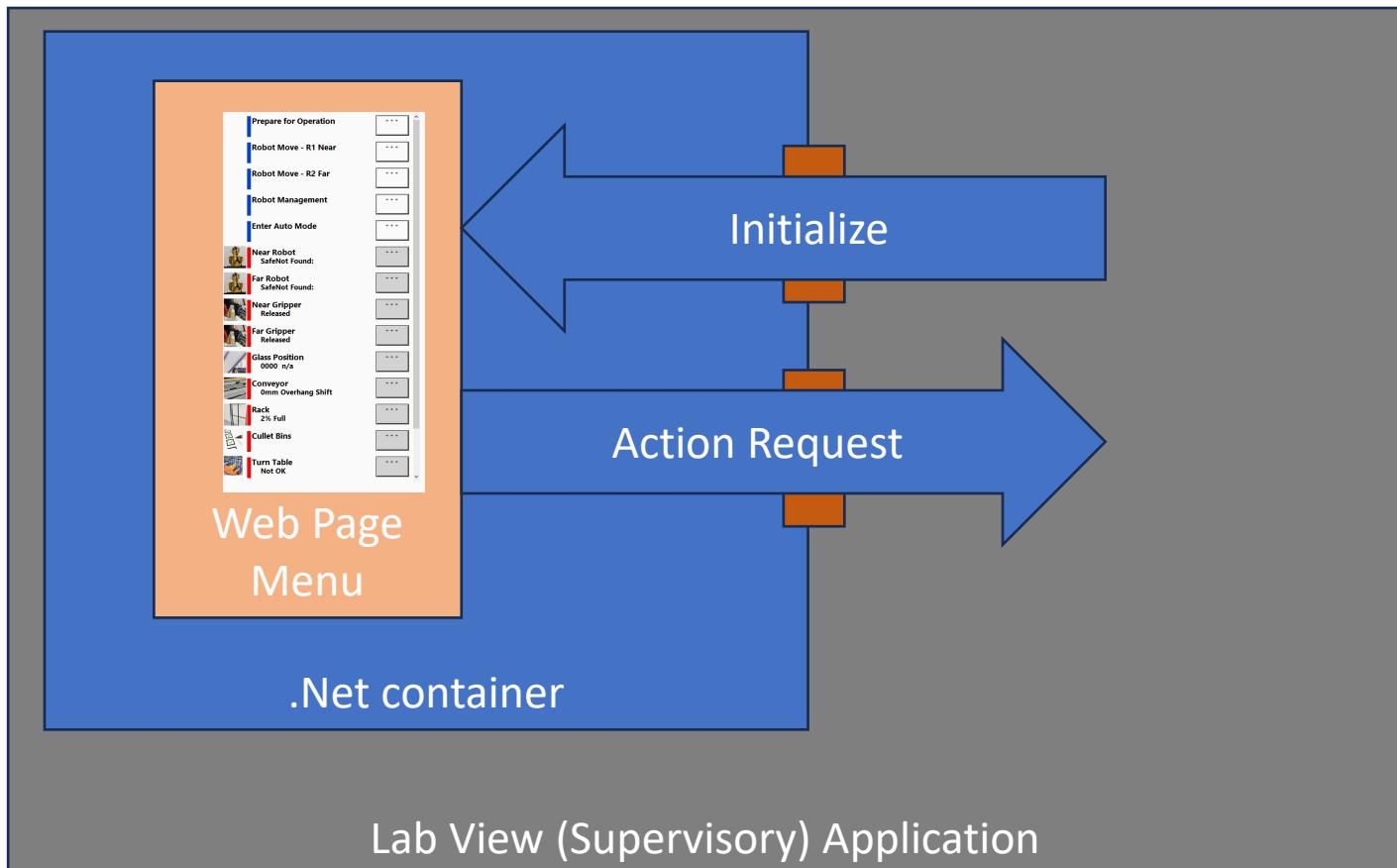




Any robot + Our products = **INSTANT AUTOMATION**

# Technical Structure

- The web page will reside inside of a .Net container in a LabView (windows based) application.
- The web page will reference a WebView2 code base, running a visual studio developed .Net application to house the interaction handling between the webpage and the LabView application.
- The .Net interactions are tested
- Required focus in on the web page, and establishing the rules for the interaction subroutines
- \*NO CODE\* is needed to be developed for the .Net, or LabView portion. We will handle anything needed for that.
- Sample web page provided



Any robot + Our products = **INSTANT AUTOMATION**

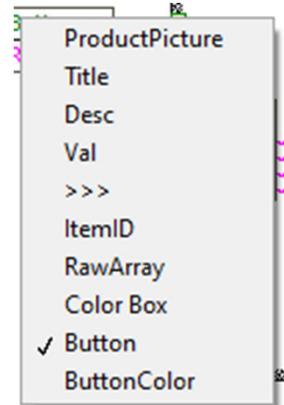


We didn't make the robot...  
**We made the robot EASY.**

# Vertical Menu

## Elements for an item Appearance

- Heading
- Sub Text
- Button
  - Action Type (designates coloring/appearance)
    - Menu Expansion, Detail view of Item, Device control, Robotic movement
    - Button text
    - Enabled
  - Image



Any robot + Our products = **INSTANT AUTOMATION**



We didn't make the robot...  
**We made the robot EASY.**

# Page Size

- 460 pixels wide (approx.)
- Should have room for at least 10 rows in vertical of 874 pixels or so
- Vertical scrolling ok, NO horizontal scrolling
- Menu should show ONLY active items in category, not a full expanding list (like a phone menu where only the active group is shown, and a back or up button to return to the previous category)

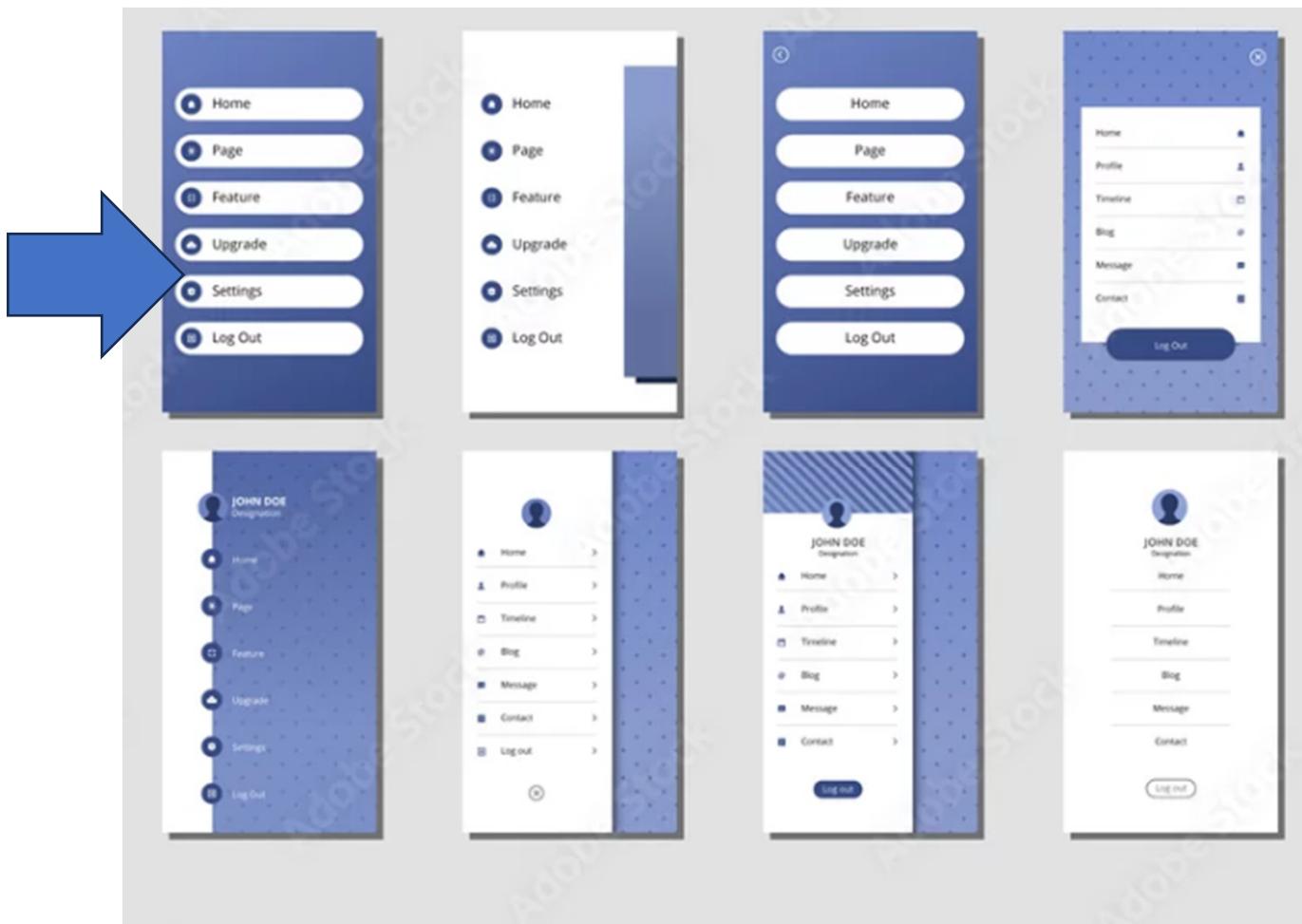
Any robot + Our products = **INSTANT AUTOMATION**

**QPoint**  
Robotic Solutions

We didn't make the robot...  
**We made the robot EASY.**

# Appearance

- Rounded buttons as upper left is consistent with theme of application



Any robot + Our products = **INSTANT AUTOMATION**

# QPoint

## Robotic Solutions

We didn't make the robot...  
**We made the robot EASY.**

# Sample web code

```
<!DOCTYPE html>
<html>
<head>
<title>Interactive Tree</title>
<style>
svg {
border: 1px solid #ccc;
background-color: #f9f9f9;
}
text {
font-family: sans-serif;
font-size: 12px;
pointer-events: none; /* So clicks go to the circle */
}
</style>
</head>
<body>
<h2>Interactive Tree Structure</h2>
<svg id="tree" width="800" height="400"></svg>
<script>

//-----
// RECEIVE messages from C# - was not able to get it to prouce activity on web page side
// -----
//window.chrome.webview.addEventListener('message', event => {
//    console.log("C# sent:", event.data);
//}

// -----
// Show it on the page
document.getElementById("status").innerText =
"Received from C#: " + event.data;

// -----
// Optional popup
alert("C# sentt: " + event.data);

//----- Fuctions related to the tree view
const treeData = {
name: "Root",
children: [
{
name: "Child 1",
children: [
{name: "Grandchild 1.1"},
{name: "Grandchild 1.2"}
],
},
{
name: "Child 2",
children: [
{name: "Grandchild 2.1"}
]
}
];
};

const svg = document.getElementById("tree");
const nodeRadius = 20;
let ySpacing = 80;
let xSpacing = 100;
let currentX = 50;
function drawNode(node, depth, parentX, parentY) {
let x = currentX;
let y = depth * ySpacing + 50;

// Draw line from parent to current node
if (parentX != null && parentY != null) {
const line = document.createElementNS("http://www.w3.org/2000/svg", "line");
line.setAttribute("x1", parentX);
line.setAttribute("y1", parentY);
line.setAttribute("x2", x);
line.setAttribute("y2", y);
line.setAttribute("stroke", "#555");
svg.appendChild(line);
}

// Draw node circle
const circle = document.createElementNS("http://www.w3.org/2000/svg", "circle");
circle.setAttribute("cx", x);
circle.setAttribute("cy", y);
circle.setAttribute("r", nodeRadius);
circle.setAttribute("fill", "#ffccbc");
circle.style.cursor = "pointer";
circle.addEventListener("click", () => {
    needs to exist where we report back to the supervisory Robotics application
    });
svg.appendChild(circle);
}

// Draw node label
const label = document.createElementNS("http://www.w3.org/2000/svg", "text");
label.setAttribute("x", x);
label.setAttribute("y", y + 4);
label.setAttribute("text-anchor", "middle");
label.setAttribute("fill", "ffff");
label.textContent = node.name;
svg.appendChild(label);

// Draw children
if (node.children) {
let startX = currentX;
node.children.forEach(child => {
currentX += xSpacing;
drawNode(child, depth + 1, x, y);
});
currentX = (startX + currentX - xSpacing) / 2;
}

drawNode(treeData, 0, null, null);

//***** the supervisory Robotics application is able to
trigger any java script function with a single string as the parameter
//***** can likely do more parameters with ease as
well as we pass as a full encoded string
function MABScript(messageTxt) {
alert('you clicked onnnn: ${messageTxt}');
}

function getGreeting(name) {
we see this data
return `back at ya, ${name}`; //***we can and need return functions for some,
}

function message(messageTxt) {
alert(`script called with parameter: ${messageTxt}`);
}

function setBackground(color) { // will likely want some visual adjustment style functions to be available
document.body.style.backgroundColor = color;
alert(`script called with parameter: ${color}`);
}

// -----
// SEND message to C#
// -----
//THIS IS THE IMPORTANT FUNCTION THAT ALL WEB PAGE ACTIVITIES NEED THAT TRIGGER EXTERNAL RESPONSIVE
ACTOIVITY form the Supervisory Robotics application
function webPageActivity(txt) {
    window.chrome.webview.postMessage("MABScript was called with: " + txt);
}

/*C:\_DEVEL\_WebView2Embedder\WebView2-VS Embed-vB\WinFormsApp.sln*/
/*C:\comprehensiveROBOTICsolution\HTML\treeviewGraphicalTest.html*/
/*C:\_DEVEL\_WebView2Embedder\WebView2Browser.vi*/

</script>
</body>
</html>
```

Any robot + Our products = **INSTANT AUTOMATION**



We didn't make the robot...  
**We made the robot EASY.**

# Interaction with Supervisory Application

## From supervisory Application

- Initialize display
  - Sends info for list build
  - Can be multiple calls, or whatever is needed to send full menu structure info
- Update display
  - Supervisory application will have live data and live status info for each item in the menu structure which will need to be refreshed (button enables based on status, change in subtext info, etc.)

## To Supervisory Application

- Button presses to trigger actions
- Active sub menu (to know which nodes need to be updated for live data)

Any robot + Our products = **INSTANT AUTOMATION**



We didn't make the robot...  
**We made the robot EASY.**

# Related Code Snippets

- `webPageActivity(string);`
  - Sends a string to the supervisory application when something is clicked
  - This needs to be incorporated in your menu java code

```
circle.style.cursor = "pointer";
circle.addEventListener("click", () => {
    webPageActivity(node.name); //*****
});
```

- We can call (trigger) from our supervisory application any java script function. Easiest if they receive a single string as the parameter, but can send multiple parameters too as needed.
- These will be for initialization and updates to the live display data

```
function getGreeting(name) {
  return `back at ya, ${name}!`; //***we can and need return functions for some, we see this data
}

function message(messageTxt) {
  alert(`script called with parameter: ${messageTxt}`);
}

function setBackground(color) { // will likely want some visual adjustment style functions to be available
  document.body.style.backgroundColor = color;
  alert(`script called with parameter: ${color}`);
```