function handleSubmit() {

const link = document.getElementById('videoLink').value.trim();

const file = document.getElementById('fileUpload').files[0];

const label = document.getElementById('clipLabel').value.trim();

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

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

if (!link && !file) {

status.textContent = "Please paste a link or upload a file.";

return;

}

function startAnalysis(analyzeBtn, statusSpan, labelOrName, linkOrFile) {

analyzeBtn.disabled = true;

statusSpan.textContent = ' ⏳ Analyzing formations';

setTimeout(() => {

statusSpan.textContent = ' ⏳ Reading tendencies';

setTimeout(() => {

statusSpan.textContent = ' ⏳ Generating report';

setTimeout(() => {

const results = runThePlayMaster(linkOrFile, labelOrName);

fillAnalysisPage(results);

statusSpan.textContent = ' ✅ Analysis Ready by The PlayMaster';

analyzeBtn.style.display = 'none';

setTimeout(() => toggleView('analysis'), 1000);

}, 2000);

}, 2000);

}, 2000);

}

if (link) {

status.textContent = `Link received (${label || 'No label'}). Ready for analysis.`;

const newItem = document.createElement('li');

newItem.innerHTML = `

<span>🔗 ${label || 'Link'}: ${link}</span>

<button class="analyze-btn">Analyze</button>

<span class="analysis-status"></span>

`;

uploadItems.appendChild(newItem);

const analyzeBtn = newItem.querySelector('.analyze-btn');

const statusSpan = newItem.querySelector('.analysis-status');

analyzeBtn.addEventListener('click', () =>

startAnalysis(analyzeBtn, statusSpan, label || link, link)

);

return;

}

if (file) {

const fileName = file.name.toLowerCase();

const isZip = fileName.endsWith(".zip");

const newItem = document.createElement('li');

newItem.innerHTML = `

<span>${isZip ? '📦' : '🎥'} ${label || file.name}</span>

<button class="analyze-btn">Analyze</button>

<span class="analysis-status"></span>

`;

uploadItems.appendChild(newItem);

const analyzeBtn = newItem.querySelector('.analyze-btn');

const statusSpan = newItem.querySelector('.analysis-status');

analyzeBtn.addEventListener('click', () =>

startAnalysis(analyzeBtn, statusSpan, label || file.name, file.name)

);

if (isZip) {

status.textContent = `Zip file "${file.name}" uploaded (${label || 'No label'}). Auto-extraction and processing will be available soon.`;

} else {

status.textContent = `Clip "${file.name}" uploaded (${label || 'No label'}). Ready for analysis.`;

}

}

}

function toggleView(view) {

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

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

if (view === 'upload') {

uploadView.style.display = 'block';

analysisView.style.display = 'none';

} else {

uploadView.style.display = 'none';

analysisView.style.display = 'block';

}

}

function runThePlayMaster(linkOrFile, label) {

return {

text: [

`Opponent uses Cover 2 Man on most 3rd downs.`,

`Blitzes from nickel on 50% of plays.`,

`Run game struggles vs 3-4 fronts.`

],

visuals: {

successChart: "placeholder",

formations: ["Trips Right", "I-Form", "Gun Bunch"],

tags: [

{ play: "1st & 10", coverage: "Cover 2", blitz: "Nickel Edge" },

{ play: "3rd & 6", coverage: "Man", blitz: "None" }

]

}

};

}

function fillAnalysisPage(data) {

const textBox = document.querySelector('#analysisView .text-output');

textBox.innerHTML = `<h2>AI Breakdown</h2>`;

data.text.forEach(point => {

const p = document.createElement('p');

p.textContent = point;

textBox.appendChild(p);

});

const visualBox = document.querySelector('#analysisView .visual-output');

visualBox.innerHTML = `<h2>Visual Report</h2>`;

const chart = document.createElement('div');

chart.className = 'chart';

chart.innerHTML = `<h3>Play Success Rate</h3><p>[Chart Placeholder]</p>`;

visualBox.appendChild(chart);

const formations = document.createElement('div');

formations.className = 'formation-diagram';

const formationList = (data.visuals?.formations || []).map(f => `<li>${f}</li>`).join('');

formations.innerHTML = `<h3>Common Formations</h3><ul>${formationList}</ul>`;

visualBox.appendChild(formations);

const tagRows = (data.visuals?.tags || []).map(row => `

<tr><td>${row.play}</td><td>${row.coverage}</td><td>${row.blitz}</td></tr>

`).join('');

const table = document.createElement('div');

table.className = 'tags-table';

table.innerHTML = `<h3>Defensive Tags</h3><table>

<tr><th>Play</th><th>Coverage</th><th>Blitz</th></tr>

${tagRows}

</table>`;

visualBox.appendChild(table);

}