GROUP 1 BOWTIE PROJECT

Lessons Learned from our Attempts to Use the React Flow Playground

When we first started building our bowtie diagram, our goal was to create an interactive visualization where we could clearly show (the Loss of control of the commercial vehicle) as the top event, along with the different causes and prevention controls. We decided to try using the React Flow Playground because it looked like a modern tool with draggable nodes, auto-connecting lines, and a flexible layout. However, working inside React Flow taught us several important lessons that eventually led us to pivot ourselves to another tool.

The first lesson We learned was that React Flow is not a simple drag-and-drop builder. Even though the Playground looks user-friendly at first, it still works like a developer tool. Every node needs an ID, position, and connection, and the tool does not automatically format or arrange anything for you. We had to manually move each box such as Loss of control of commercial vehicle, Driver Fatigue, Distracted Driving, Fatigue Monitoring, Telematics Speed Alerts, and more. Getting them into the right place took a lot of time, and one small movement affected the whole layout. The symmetry that a bowtie diagram normally has was very hard to recreate.

Another challenge was colors and visual styling. In the professional bowtie example from Open Risk, the nodes use color-coded layers (“Good,” “Fair,” “High,” “Medium,” etc.). In React Flow, we could not see where to add these colors or background sections. We discovered that React Flow requires custom code to style nodes, which are not available in the Playground. This made the diagram look very plain and not visually comparable to the professional examples.

We also struggled with layout and interactivity. We noticed that React Flow does not provide automatic hierarchical or bowtie layouts unless you install ElkJs, but ElkJS cannot be used inside the Playground. Because of this, we had to manually drag every node, one by one. As the diagram grew larger, it became harder to manage, and the connections became messy. Even after trying our best, the structure was uneven and did not match the clean style shown in the bowtie reference diagrams. Because of these problems, we realized that continuing in React Flow would consume too much time and still not produce a professional result. That is when we made the decision to pivot. We pivoted to OpenRisk/ Presight, which already has a built-in bowtie structure, automatic alignment, color coding, and labels for prevention and mitigation controls. With this tool, we were able to create a much cleaner, clearer, and more organized bowtie diagram suiting what we are expecting. Our attempt taught us that choosing the right tool is part of the visualization process. A tool should support clarity, not create barriers. React Flow helped us understand how bowtie diagrams are structured, but pivoting to Presight allowed us to deliver a more organized, readable, and effective final design