**1. Define the Project Scope and Requirements**

* **Objectives**: Clarify what the tool will do. In your case, it's about comparing GHG emissions from rail and road freight.
* **Features**: List the essential features, such as:
  + Input for container size, product volume, weight, origin, and destination.
  + Calculation of GHG emissions for both rail and road.
  + Comparison and output display.
* **Constraints**: Identify any constraints, such as specific routes, types of freight, or geographical regions.

**Objectives**

* **Primary Goal**: The main objective is to create a software tool that allows users to compare the greenhouse gas (GHG) emissions from transporting freight by rail versus road. This will help users make more environmentally informed decisions when choosing between these two modes of transport.
* **Secondary Goals**:
  + Educate users on the environmental impact of freight transportation.
  + Provide a user-friendly interface that makes it easy to input data and receive accurate comparisons.
  + Offer flexibility in the types of inputs, such as different container sizes, volumes, weights, and transport routes.

**Features**

* **Input Parameters**:
  + **Container Size**: Users should be able to select from different container sizes (e.g., 20ft, 40ft).
  + **Product Volume**: Allow input of the total volume of the products being transported. (cubic meter/ cubic feet/ liter)
  + **Weight of Goods**: Input for the weight of the freight in tons or kilograms.
  + **Origin and Destination**: Users should select or input the starting point and destination of the freight.
  + **Transport Mode**: Choose between rail and road for comparison.
* **Calculation Functionality**:
  + **Emission Calculation**: Based on the input parameters, the tool will calculate the GHG emissions for both rail and road.
  + **Comparison Output**: The tool will present a side-by-side comparison of emissions for both transport modes.
* **User Interface**:
  + **Form for Inputs**: A simple and intuitive form where users can enter their data.
  + **Results Display**: Clear and concise output showing the calculated emissions, possibly with graphs or charts for better visualization.
  + **Accessibility**: Ensure the tool is accessible and user-friendly across various devices and for all users, including those with disabilities.

**Constraints**

* **Geographical Limitations**: Depending on the available data, the tool might initially focus on a specific region, in this case Australia.
* **Data Availability**: The accuracy of the tool is dependent on the availability and reliability of GHG emissions data for both rail and road transport.
* **Technical Limitations**: The tool should be designed to handle a reasonable amount of data and user inputs, ensuring performance remains optimal.

**Success Criteria**

* **Accuracy**: The tool should provide accurate comparisons based on reliable emissions data.
* **Usability**: Users should find the tool intuitive and easy to navigate.
* **Performance**: The tool should perform calculations quickly and handle a variety of input scenarios without errors.
* **Impact**: The tool should help users make more environmentally friendly decisions regarding freight transport.

**2. Research and Data Collection**

* **Emission Factors**: Research the GHG emissions factors for rail and road transport. This includes:
  + Average emissions per kilometre for both rail and road.
  + Differences based on container size and freight type.
* **Data Sources**: Find reliable sources for emissions data. This could include government databases, transportation studies, or environmental organizations.

**3. Software Design and Architecture**

* **Choose a Platform**: Decide whether this will be a web-based tool, a desktop application, or a mobile app.
* **Design the User Interface**: Sketch out how users will interact with the tool. This includes the input forms, buttons, and results display.
* **Backend Design**: Outline how the calculations will be handled. For example:
  + A formula for calculating emissions based on user inputs.
  + Storage and retrieval of emission factors and other data.
* **Technology Stack**: Decide on the programming languages, frameworks, and tools you'll use (e.g., Python, JavaScript, Vue.js, Node.js).

**4. Develop a Prototype**

* **Build a Simple Version**: Start with a basic prototype that takes in user input and returns a simple calculation. This doesn’t need to be fully accurate or feature-rich but should demonstrate the core functionality.
* **Test and Iterate**: Test the prototype to see if it meets your initial requirements and make improvements based on feedback.

**5. Implementation and Development**

* **Develop the Full Application**: Expand the prototype into a full-fledged tool, incorporating all features and refining the calculations.
* **Integrate Data**: Incorporate real-world data for emissions factors and other relevant metrics.
* **Testing**: Perform thorough testing, including edge cases and varying input conditions.

**6. Deployment and Documentation**

* **Deploy the Tool**: Depending on your platform choice, this could be hosting a web app, distributing a desktop app, etc.
* **Documentation**: Create user guides, technical documentation, and any other necessary materials.

**7. Feedback and Iteration**

* **Collect User Feedback**: Once the tool is in use, gather feedback to understand how it’s performing.
* **Make Improvements**: Based on feedback, make necessary updates and enhancements.

**8. Final Report/Presentation**

* **Prepare Documentation**: Write up the project report, including your methodology, data sources, design choices, and results.
* **Present the Tool**: Be ready to demonstrate how the tool works, explaining its features and the benefits of using it.