# Employee Tree Product Requirement Document (PRD)

## Overview

- **Purpose**: To develop a web application that effectively manages and visualizes the hierarchical structure of employees within an organization.
- **Problem Statement**: Organizations need a tool to understand and navigate complex employee hierarchies for better management and reporting.

## Goals

- To provide a user-friendly interface for searching employees.
- To display each employee's managerial lineage up to the organization's root.
- To show the total count of direct and indirect reports of employees.

## Success Metrics

- Accuracy in displaying hierarchical data.
- Efficient handling of edge cases.

## Assumptions & Dependencies

- Assumption that user input will be accurate and in the correct format.
- Dependency on modern web technologies like Next.js, React-D3-Tree, TypeScript, and TailwindCSS.

## Requirements

- Functional Requirements:
  - Ability to search for employees by name.
  - Display the names of all managers up to the root for a selected employee.
  - Show the total count of direct and indirect reports for an employee.

    - Handle multiple data sets where users can choose which set to use during a search.

- Edge Cases:

    1. Employees Lacking Hierarchical Connections
        - **Description**: This happens when certain employees, like "Keane", are not integrated into any hierarchical structure, missing both superiors and subordinates.
        - **Example**: Consider an employee `{"id": 4, "name": "Keane", "managerId": null}` who lacks a reporting manager and has no direct reports.
        - **Action**: The system should recognize this situation and alert with a message like, “Unable to process employee hierarchy. Keane not having hierarchy.”
    2. Employees Reporting to Multiple Superiors
        - **Description**: Occurs when an employee is assigned to more than one superior, disrupting the usual single-manager structure.
        - **Example**: An employee `{"id": 4, "name": "Linton", "managerId": 2}` and `{ "id": 4, "name": "Linton", "managerId": 3 }` with records showing two different managers.
        - **Action**: The system should flag this anomaly and prompt, “Hierarchy error: Linton reports to multiple superiors: Fletcher, Tabitha.”
    3. Self-Reporting Employees
        - **Description**: Arises when an employee is erroneously listed as their own manager.
        - **Example**: An employee entry like `{"id": 5, "name": "Alice", "managerId": 5}`.
        - **Action**: The application should point out this error with a message such as, “Employee Alice reports to self. Please correct the hierarchy.”
    4. Circular Management Hierarchy
        - **Description**: Identified when there's a loop in reporting, leading to an unresolved management chain.
        - **Example**: If "Linton" reports to "Fletcher", who in turn reports to "Linton".

- **Action**: The system should indicate the loop with a warning like, “Circular reference detected for employee Linton.”

5. Employee Reports to a non-existing manager
    - **Description**: This happens when certain employees, like “Peg”, are assigned to non-existing manager.
    - **Example**: An employee `{"id": 7, "name": "Peg", "managerId": 11}` **with records showing that no employee with id 11.**
    - **Action**: The system should recognize this situation and alert with a message, “Manager with id 11 does not exist for employee Peg. Please correct the hierarchy.”

## Technical Specifications

- The application will be developed using Next.js, TypeScript, and React-D3-Tree for tree hierarchy visualization.
- UI design will be implemented using TailwindCSS.
- The application will store employee data in memory without a database.
- Implementation will focus on OOP principles, readability, modularity, and maintainability.
- Unit tests will be written using Jest and React Testing Library.
- Code coverage integration with GitHub and Codecov for quality assurance.
- Docker containerization for easy deployment and execution.

## Delivery & Deployment

- The application will be hosted on GitHub with documentation on building, running, and testing.
- A Dockerfile will be provided for containerized deployment.
- The application will be deployed on Vercel via Github Action.

## Optional Specifications

- Integration of GitHub Actions for CI/CD to automate testing and deployment.

- An employee table showing the employee data with searching and filtering feature