Technical Documentation:

Architecture + Tech Stack :

Frontend:	React.js with Material-UI for tables and charts.
Backend:	Node.js with Express.js for the REST API.
Database:	PostgreSQL for relational data.
Graph Visualization:	D3.js for a customizable hierarchical tree.
Profile Picture/Avatar:	Gravatar
Deployment Frontend:	Vercel
Deployment Backend:	Render
Deployment Database:	Render
Version Control:	GitHub

Frontend:	Employee List View Employee Graph View Add + View + Edit Employee Other components such as “Are you sure?” messages.

Backend:	**Get** : /employees + /employees/search/{query} + /employees/{id}
	Post : /employees
	Put : /employees/{id}
	Delete : /employees/{id}

Database:	Employee table

Design Patterns :

The Agile Design Pattern was used as this provided the most flexible working environment for constant updates with easy version control through GitHub.

There was a “main” branch which had the most recent fully working deployed modal on it. Then a “develop” branch which combined any changes from feature branched. Finally, there were “feature/{feature}” branches which were used for adding and updating new and existing features.

Technology + Justifications :

The notable technology used was D3.js. This was used as it allowed for easy and flexible graph generation directly through the JavaScript frontend using data received from the backend.

The entire program was built using JavaScript for coherent legibility and ease when writing code.

Render was used for the backend as it has free tiers which has good availability and easy to integrate using CI/CD on GitHub.
Furthermore, Render was used for the Database hosting since it integrates the best and easiest with a Render hosted backend.

Finally, Vercel was used for the hosting of the frontend because of its seamless CI/CD integration with GitHub and its versatile ability to host JavaScript applications.