CIOBrain Design Document

ABSTRACT

This document walks through the design of the CIOBrain web application. It covers the reasons behind certain design decisions, the breakdown of the technical components and interactions between the different classes, and presents the graphical user interface of the application. The document also covers the rationale behind the design decisions the team made, and connects those design decisions to the initial requirements of the application to confirm that the right product is being built.

VERSION HISTORY

Version #	Implemented By	Revision Date	Reviewed By	Approved By	Approval Date	Rationale for Change
1						Initial Design Document

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

This document outlines the design decisions for the CIOBrain web application along with the API. It is divided into three main parts: the graphical user interface design, the class diagram of the API, and the sequence diagram. The CIOBrain application is a web application designed to generate a graph of all connected assets a CIO manages so that they are able to better visualize the scope and interconnectedness of their projects. The basic flow of the site is: the CIO imports their data in the form of spreadsheets, which the application then reads and parses. The CIO can then choose an asset type from their data: application, database, or infrastructure. After they choose the type, a list of all the assets of that type will appear below the asset type and the CIO can then choose the specific asset they want to view. The graph will then be generated. The root node of the graph will be the asset they chose, and it's connections will be displayed in a hierarchical format. Each node is the color of the asset type it corresponds with. The nodes also have text that indicate the name of the asset and what type it is, and also contain icons for more visual ease when it comes to identifying the node type. All these design decisions were made with ease of use in mind, so that the CIO is able to quickly identify the assets and how they're connected.

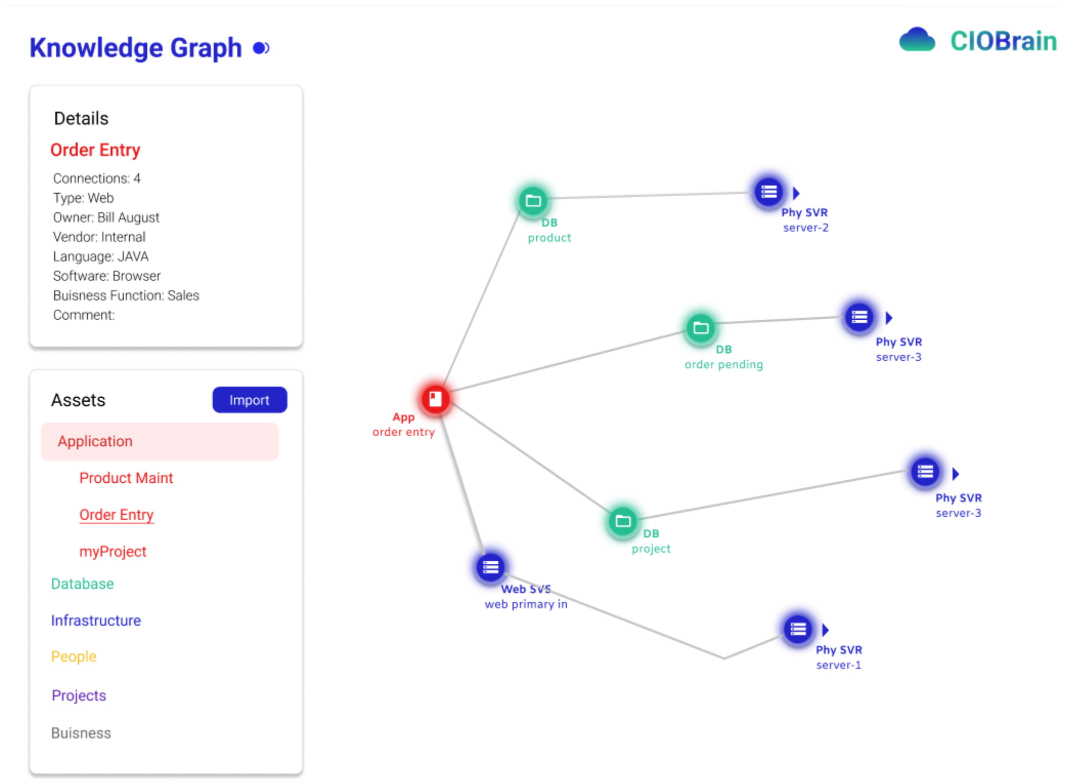
GUI (Graphical User Interface) Design

The Graphical User Interface design was primarily driven by the main use case for the application: it displays a graph where each node represents an asset, and all of the assets are connected to visualize their connections with each other. This is to help the CIO better understand how all of the different projects he/she is overseeing are related to help them do their job more efficiently. The CIO must first import the spreadsheet files of the data. The CIO can then select the specific asset they want to view the graph for by first clicking on the asset type in the bottom left menu, which then displays a list of that type of asset. They can then choose which specific asset they want to view. After they do so, the graph for that particular asset is displayed with the asset they chose as the root node, and the other assets' nodes appearing in their respective asset type colors (each asset type is the same color of the corresponding excel sheet), and arranged in a hierarchical view. The user can expand the graph and look at more connections by clicking on the leaf node they wish to view the connections of. The details of the chosen asset can then be seen at the top left of the page, with details such as the number of connections, the owner of the asset, etc.

The hierarchical design was initially proposed by the sponsor of the project, and research was provided as to how to implement the graph. When other types of graphs were researched, it did not appear as though they would provide as much clarity as the hierarchical format when displaying all the connections. For example, a radial graph would display the root node in the center, and then all the connections would be displayed in a spiral around it. This type of graph would appear too crowded and wouldn't be easy to read or interpret for the CIO.

The design of CIOBrain is clean and simple, and very straightforward for the user to interact with. The center of the design is the graph, with clear nodes and specific details provided for the asset they chose. The application is focused.

Below is the Figma design that was reviewed and approved by the sponsor.



Connections: 4
Type: Web
Owner: Bill August
Vendor: Internal
Language: JAVA
Software: Browser
Buisness Function: Sales
Comment:

Figure 1: Figma UI Design

STATIC MODEL

CLASS DIAGRAMS

The class diagram in Figure 1 details the components of the CIOBrain API. The current implementation of the API does include a backend database for storing the information that will be displayed on the graphs, but rather parses data from excel sheets provided by the sponsor. The information for the three types of assets (application, data, and infrastructure) is extracted from three separate excel files. There will be a corresponding model class for each type of asset that depends on the data from their respective excel file. Once a backend database is implemented, the model classes can be modified to obtain data from data access objects, rather than an excel file. There are also three controller classes that handle incoming requests from the web application and return the appropriate data for that type of asset. Each controller class has a dependency relationship with all of the model classes, so they have access to multiple data models. This allows the controllers to retrieve the connections to an asset, so that the application can create hierarchical data for a node and display a corresponding graph.

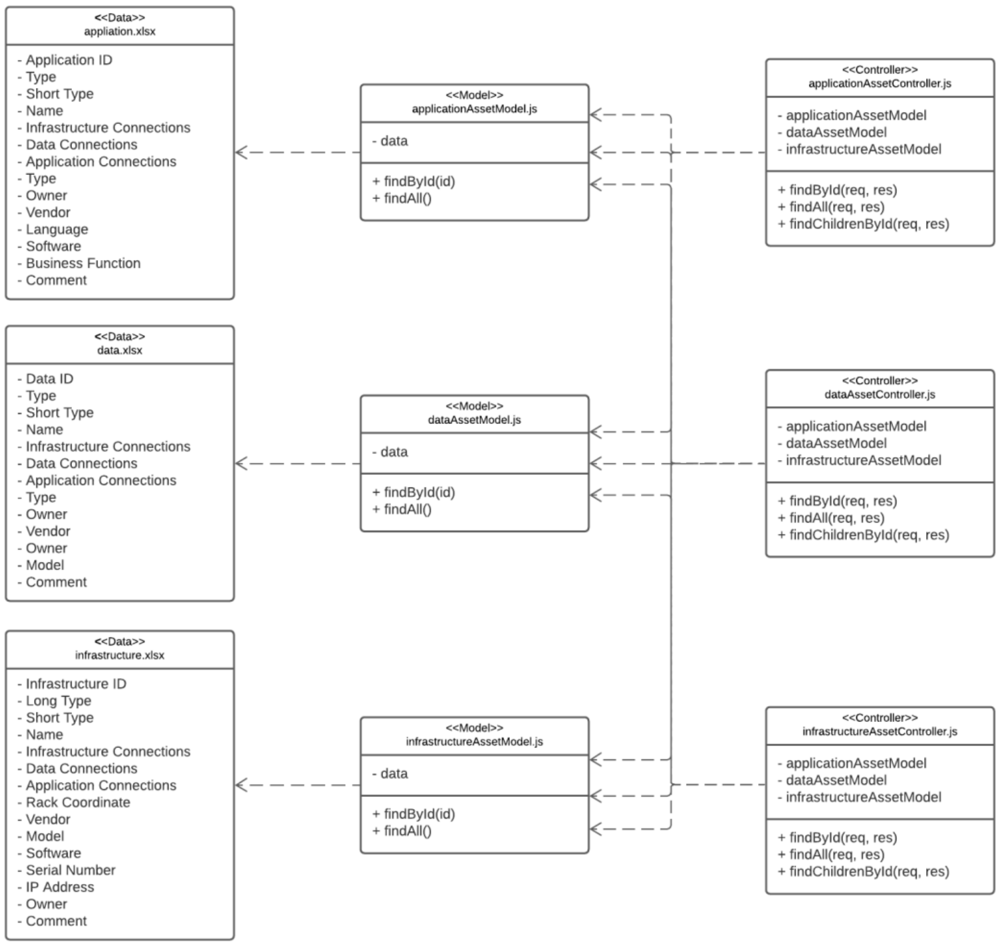


Figure 2: Class diagram of API

DYNAMIC DIAGRAMS

SEQUENCE DIAGRAMS

The sequence diagram depicts the interactions between the CIO and the system to populate the CIO asset graph. The CIO will navigate the website and be presented with an asset menu to choose an asset category: Application, Data, Infrastructure. The website will communicate back to the CIOBrain API to retrieve the current list of all assets under that category and display it to the user in a dropdown in the asset menu. The CIO can then select an asset. In order to generate the correlating asset graph, CIOBrain application will send an HTTP request to CIOBrain API for children assets linked to the asset selected. When the API returns a list of the children, CIOBrain will build a hierarchical dendrogram graph and display it to the users. CIOBrain will also send an HTTP request to the API for asset details of the selected asset and populate the retrieved asset details to the user.

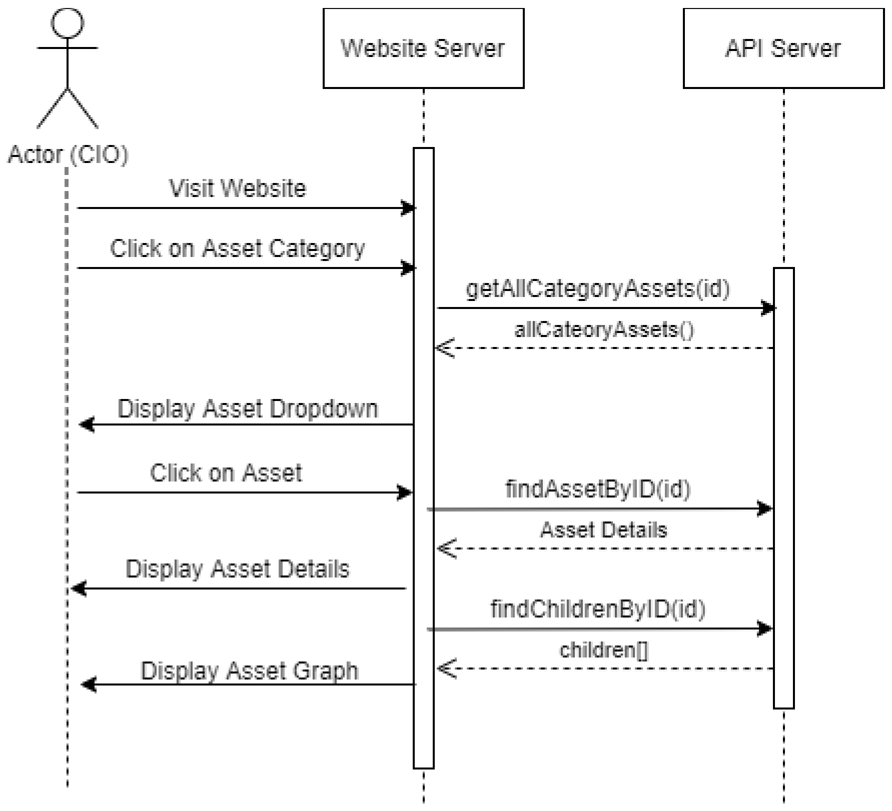


Figure 3: Sequence Diagram

RATIONALE FOR YOUR DETAILED DESIGN MODEL

The rationale for the design model was based on the requirements of the client. The goal of the model was to ensure the CIO client is able to select categories of assets in their organization in a hierarchical manner. For the GUI color scheme, the design is utilizing a multi color scheme approach in order to distinguish different assets. The multi color scheme is extended to the nodes, therefore the client is able to quickly recognize which nodes belong to which asset. The current design model for data flow is that the application retrieves data through excel sheets, but in the next phases it will retrieve data from a database. The design model ensures the next phase of development will be able to continue as a database design has been incorporated into the model through an API. Once data is present in the application, the sequence diagram displays the natural flow of events a user will take in order to retrieve knowledge from the category assets. The sequence diagram showcases the design of how the application will communicate with the API in order to achieve data retrieval.

TRACEABILITY FROM REQUIREMENTS TO DETAILED DESIGN MODEL

The major requirement for this system is:

> The system shall display asset graph connections.

Therefore, in the following discussion, we will map out how this requirement is achieved and represented in the front end wireframe, class diagram and sequence diagram.

The front end interface design provides a wireframe of the menu to gather user input necessary for the application to process a graph for the user. Additionally, the graph prototype selected is an expandable dendrogram to provide a simplistic view and interface for viewing graph connections. By being able to minimize and expand specific connections, CIOs will also have control over what asset relationships they want to view. The selected graph type also addresses the following constraint noted by the sponsor:

> The system should provide a user experience for displaying graph connections that is better than the other enterprise architecture tools in the market.

A major competitor that we are designing our application against is Product A who offers Relation Explorer maps and Interface Circle maps. The former is a dendrogram but requires multiple clicks to identify relationships in an asset graph and the latter is a radial dendrogram, displaying all relationships at the start but is hard to understand. The designed interface and prototyped dendrogram should eliminate these issues by providing simplicity and usability. User interactions to display the asset connections of a depth of 1 require only 2 clicks and color coding and icons are used to allow the user to make meaning of the information more quickly.

The application is able to display graph connections by building hierarchical data from the xlsx sheets. This sequence described by the diagram describes the dynamic interaction between the CIO and the system to achieve this by taking in user input, displaying the asset options, asset details and the respective asset's connection graph by calling API endpoints for asset information including:

- a list of assets under an asset category
- the asset entity
- the children of an asset entity

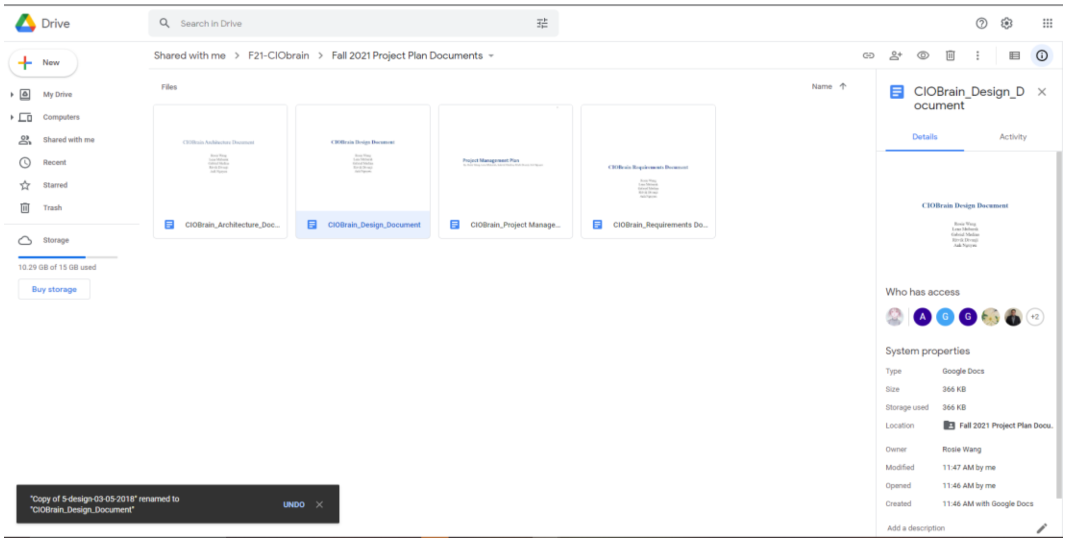
The separation of functionality between the front end and back end between the web server (CIOBrain application) and the api server (CIOBrain API) also provides low cohesion between the data and user display, allowing for future scalability.

The class diagram shows the dependencies between the xlsx files and the api controllers to implement the service logic for findChildrenById() which returns hierarchical data for the front-end application to build and display graph connections. The class diagram also describes how the api server is organized further to provide cohesion within assets being handled in the application and low cohesion between the api controllers and the data models to support the following constraint:

> The system shall be able to implement a graph database in the future to store assets and their connections.

EVIDENCE THE DESIGN MODEL HAS BEEN PLACED UNDER CONFIGURATION MANAGEMENT

The detailed design document has been placed under configuration management in a shared google drive for the team.



Shared with me > F21-CIObrain > Fall 2021 Project Plan Documents

CIOBrain_Architecture_Doc...
CIOBrain_Design_Document
CIOBrain_Project Manage...
CIOBrain_Requirements Do...

CIOBrain_Design_Document

Who has access

System properties

Type	Google Docs
Size	366 KB
Storage used	366 KB
Location	Fall 2021 Project Plan Docu...
Owner	Rosie Wang
Modified	11:47 AM by me
Opened	11:46 AM by me
Created	11:46 AM with Google Docs

Add a description

"Copy of 5-design-03-05-2018" renamed to "CIOBrain_Design_Document"