PIQUE Expansion:

Adding a GUI and Database

By: Maria Gallivan, Dawson Kanehl, Zoe Norden
Connor Snow

CSCI 483R
Dr. Clemente Izurieta
April 25th, 2022

Introduction

Today's society runs on code. Code is built into every service and product. This presents many opportunities for cyber criminals to exploit systems preying on both individuals and corporations. According to the University of North Georgia, 43% percent of cyber attacks target small business, 64% of companies have experienced web-based attacks, 62% experienced phishing & social engineering attacks, 59% of companies experienced malicious code and botnets, and 51% experienced denial of service attacks [1]. Additionally, the estimate for the average cost of data breaches in 2020 was $150 million USD [1]. These staggering numbers continue to grow yearly in favor of the criminals.

Most programmers and companies are unaware of how vulnerable their systems are to cyber attacks. This is an enticing area for crime with a wide victim set and easy targets. A need to perform quality assessments on code persists in the industry as well as in personal projects. Through the comparison of their code to modern practices, we can enable companies to raise their standard of security. This, in turn, would hinder cyber crime and drop the number of cases. By strengthening the code’s defense, we disincentivize criminals to attack as it is more difficult to infiltrate the system and make money.

The PIQUE (Platform for Investigative software Quality Understanding and Evaluation) software was developed to help analyze code against current standards to evaluate its quality. The first step is creating a hierarchical tree of importance to score their code in key areas such as security, readability, usability, etc. Unfortunately, this tree must be modeled by hand and manually transferred into a JSON file, before being sent through the PIQUE application.

Our prototype would allow the users to model this tree in the application and auto-populate the necessary files. For ease, these files would be exported into a database that our group is designing. This database will store the finished tree, the nodes used, weights of the nodes, the links between nodes, the total scores, and different versions of the code and tree.

Qualifications

Please see resumes towards the end of the document.

Background

Our team is expanding the PIQUE software [2]. This software is written in JavaScript and bootstraps with Create React App [3]. We are creating a GUI and database to allow the users to model a priority tree in the app and auto-populate the necessary files. The PIQUE software, itself, aims to return a quantitative result of how secure code is based on the user created model.

PIQUE was designed as a solution to assess software through quality modeling, with regards to concepts such as functional stability, performance efficiency, compatibility, reliability, security, maintainability, and portability. It does not simply analyze given code and assign a score to each metric. Instead, the software utilizes a quality model provided by the user as a framework to score and aggregate quantitative results. Each user will have unique metrics for assessment as software quality can be subjective. Two users with the same code but different priority trees can get vastly different scores due to the nature of the PIQUE app's process. The users weigh each metric to ensure quality by their own evaluation.

The PIQUE software takes the user's priority model as a JSON file containing configuration information, various categories of qualitative metrics, and the relevant parent-child relationships between those metrics as defined by the model. This information goes first to the Calibration phase, where the app ensures the code is scored based on modern practices and the priority tree. It then enters the Scoring Phase, where an actual number is assigned to the code. Currently to access these phases, the developers must translate a hand-written priority tree into its corresponding JSON file manually.

In this JSON file, certain tools for evaluation are specified. A majority of the tools exist on a suite called Roslynator, which is an open-source collection of over 500+ analyzers and refactorings. Analyzers, for example, run and evaluate the source code looking for adherence to general rules and code styles that should be followed. These are tools that are specified at the bottom level of the hierarchy and there's many options available depending on how the model needs to be configured for the client's usage. Many of these analyzers pertain to code formatting, generally accepted practices, and any sorts of security concerns such as unused variables, parameters or maintaining proper access rights for variables. All these factors are taken into consideration when developing the quality model.

The quality model allows us to assess, predict, and improve our software/systems through a modeling framework that scores individual characteristics based on importance. The user determines which quality characteristics will or will not be taken into account when evaluating the properties of a piece of software. The quality/score of the software is the degree to which the system satisfies the stated and implied needs of its various stakeholders. Those stakeholders' needs (functionality, performance, security, maintainability, etc.) are precisely what is represented in the quality model, which categorizes the quality model into a unique tree for that user with its own set of characteristics and sub-characteristics [6]. A score will be given to each of these sub categories as well as an overall score.

We can look to PIQUE for a practical example of this being done and implemented. In its current state, PIQUE is an operational platform that would allow users to conduct quality modeling on their own code. This is being done manually where the team will hand write all the JSON code to describe the user's unique hierarchical tree of importance. This is both time consuming and not realistic long term or on a large scale. Our team's goal is to streamline this process and make it accessible for users, this is outlined below.

Work Schedule

This project is divided into two semesters. The first semester is devoted to developing a clear understanding of how the PIQUE software works and planning for the project. During winter break, we will begin to code our GUI and implement the database. This development will continue into the following Spring semester. We intend to use a Kanban framework for the Waterfall Methodology to manage our workflow. Although we plan to follow the Waterfall Methodology, we will still have weekly meetings where we outline weekly goals and progress made. Following our weekly meetings, we will meet with our customer, where we will share updates of our progress.

We have chosen a Kanban framework for the Waterfall Methodology as our development method because of the straightforward path the Waterfall approach takes. It guides the project through an analyze, design, implement, then test workflow. This cultivates success for our group due to conflicting schedules which impede consistent changes to the design and progress check ins. We trust that our requirements and design will lead us to a positive implementation and each

member of the team will do their job. This work environment naturally falls into the Waterfall Software Development Methodology. To implement a Kanban framework, we will also be using Trello as a tool to keep track of our software development progress.

Actual Implementation of Work Schedule

Our team used Trello as a tool to keep track of key tasks and dates as they approached. In addition to the features in Trello to assign tasks, we also held discussion while in our weekly meeting over Discord. In combination with these two apps, we are able to effectively delegate tasks for the week and assign them from “to do”, “in progress” and eventually “done”. Due to Trello’s ability to visually see our names next to tasks as well as a color coding system for the two groups, we were able to see how much work each team has done as well as what is left to get done.

In addition, we used the aforementioned Kanban and Waterfall strategies as intended throughout the duration of the project.

Milestones

Proposed Milestones

GUI Milestones

1) Implement a drag and drop feature for nodes
2) Establish connections between nodes via lines
3) Tie the drag and drop feature to a work space
4) Implement metadata on nodes: weights, color, name
5) Export a finished tree into a JSON file
6) Establish a connection to the database to save finished trees

Database Milestones

1) Design a database that will store:
    a) Quality model tree,
    b) Nodes with associated metadata

    c) Links between node
    d) Versions of the code being compared
    e) Scores for each version
2) Decide what kind of graph database (Neo4j)
3) Begin designing database through ERD diagrams
4) Generate and implement the database
5) Ensure it interfaces with the GUI as intended

Actual Milestones

GUI Milestones

1) Create workspace and menu
2) Add nodes with a button functionality
3) Implement ability to name and edit nodes
4) Implement drag and drop feature to work space
5) Establish connections between nodes via links
6) Save current node structure to JSON formatted file
7) UI/Visual improvements

Database Milestones

1) Setup Google Firebase
2) Implement link between PIQUE App and Google Firebase
3) Load saved JSON formatted file to node structure
4) Database structure improvements

Work Schedules

Proposed Work Schedules

GUI

Week 1-4

- Start implementing drag and drop / draw.io API (Maria)
- Able to drag and drop nodes and connections (Maria)
- Start aesthetic (Zoe)

Week 5-8

- Wrap up drag and drop for nodes and connections (Zoe)
- Implementing the metadata (Maria)

Week 9-11

- JSON file population (Maria/ Zoe)
- Aesthetics (Maria)
- Start presentation slides (Zoe)

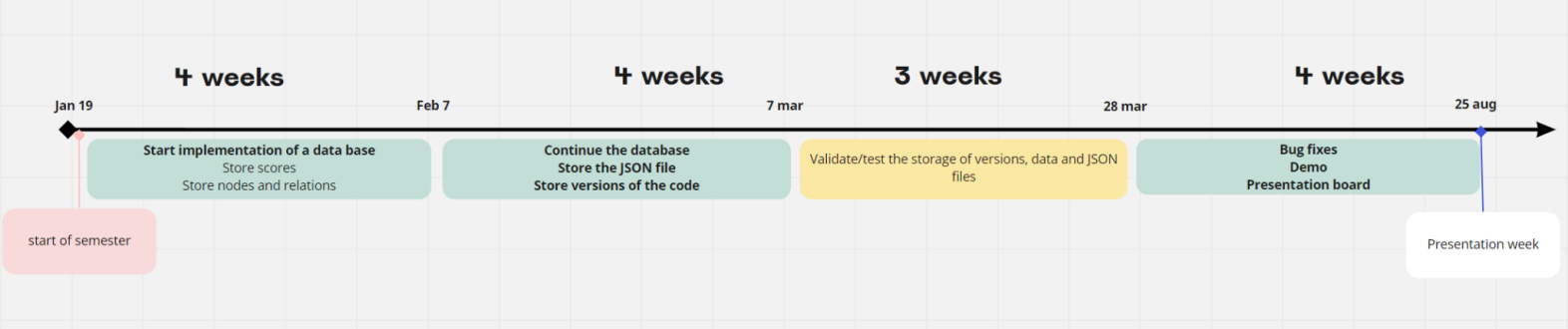
Week 12-15

- Bug fixes, demo (Maria/ Zoe)
- Presentation slides/project board (Maria/ Zoe)

Week 16

- Presentation and demos (Maria/ Zoe)

Database



Week 1-4

- Start the implementation of the database (Connor/ Dawson)
- Store the scores of nodes (Connor)
- Store the nodes and the relations (Dawson)

Week 5-8

- Continue the work on the database (Connor/ Dawson)
- Store the JSON file (Connor)
- Store the versions of the code (Dawson)

Week 9-11

- Validate/test the storage of key data, JSON files, and versions (Connor/ Dawson)

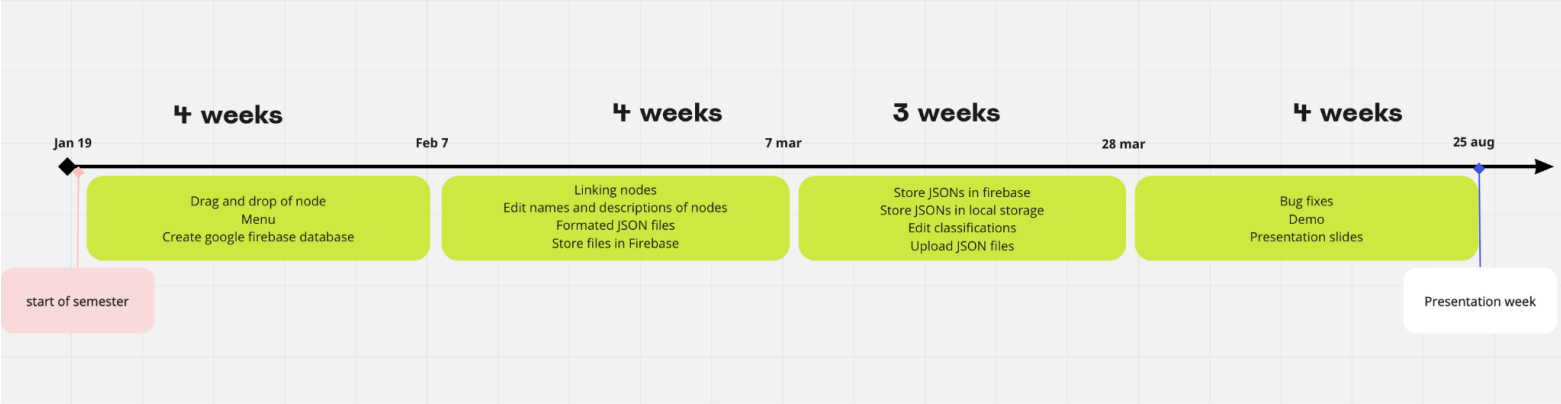
Week 12-15

- Bug fixes, demo (Connor/ Dawson)
- Presentation slides/project board (Connor/ Dawson)

Week 16

- Presentation and demos (Connor/ Dawson)

Actual Work Schedules



Week 1-4

- Creation of web workspace (Maria)
- Start implementing creation of nodes (Maria)
- Start implementing drag and drop nodes (Connor/ Maria)
- Creation of menu (Dawson/ Zoe)
- Creation of Google Firebase (Dawson/ Zoe)
- Link the Google Firebase to PIQUE app (Dawson/ Zoe)

Week 5-8

- Implementing linking of nodes (Connor)
- Wrap up drag and drop for nodes and connections (Connor/ Maria)
- Implementation of load/save as menu choices (Dawson/ Zoe)
- Edit names and descriptions of nodes (Connor)
- Add functionality to the creation of nodes (Maria)
- Continue the work on the database (Dawson/ Zoe)
- Store JSON files to firebase (Dawson/ Zoe)

Week 9- 14

- Load JSON Files (Connor/ Maria)
- Export JSON Files (Connor/ Maria)
- Edit and delete node fixes (Connor)

- Updating Proposal and posterboard (Dawson/ Zoe)
- Creation of abstract (Dawson/ Maria/ Zoe)
- Update Trello (Dawson/ Zoe)
- Read Me (Dawson)

Responsibilities

As mentioned previously, our project is split into two sections. The first section is a database that will save and store the quality model tree, nodes, weights of the nodes, the links to each node, versions of the code being compared, and the different scores for each version. The second section is a GUI that allows the user to drag and drop software quality nodes and export the finished tree to a JSON file. The two groups are connected by the finished tree which will be stored in the database.

We decided to split the group evenly and assign two people to each section. Although all parties intend to work on both systems, we did have primary focuses as described below:

Proposed

Maria Gallivan: *Start implementing drag and drop / draw.io API, implement metadata, JSON file population, finish up aesthetic, presentation slides/ bug fixes, demos*

Dawson Kanehl: *Database implementation, Store the nodes and relations, store versions of the code, bug fixes, demo, project board*

Zoe Norden: *Start aesthetic, finish drag and drop, JSON file population, presentation slides, bug fixes, demos*

Connor Snow: *Database implementation, Store the scores of nodes, store JSON files, Bug fixes, Demo, project board*

Actual

Maria Gallivan: *Deployed the React development environment, implemented a node class and a node "playspace" for building the quality model tree. Added functionality to store nodes as JSON objects within an array and save that array to a JSON formatted file. Designed functions for reading nodes and populating them on screen with their relationship lines.*

Dawson Kanehl: *GUI menu, Google Firebase realtime database implementation, link database to PIQUE web app, store the nodes and relations in the database, save trees to JSON files in the database, Read me, Proposal document, Database comparison, poster for presentation*

Zoe Norden: *GUI menu, database implementation, link database to PIQUE web app, store the nodes and relations in the database, save trees to JSON files in the database/locally, Poster for presentation, slides, misc. Administrative tasks, database comparison, assisted with loading JSON files to produce PIQUE trees*

Connor Snow: *Linking of nodes, implementing a drop down menu with options for node manipulation: showing node information, deleting nodes, changing node names, adding/removing connections to nodes. Created a draggable node "template" that spawns a new instance of a node and prompts the user to fill out name, description, and classification. Added functionality for parsing JSON files.*

Proposal Statement

The PIQUE Software needs a graphical user interface to allow users to model a priority tree more easily as well as a database to store the completed tree. Currently, when a tree is crafted, the developer must hardcode large JSON files since the tree was drawn by hand. This JSON file gives the software the ability to read the tree and move into the calibration phase.

The PIQUE developers are frustrated by the time wasted transferring hand-written trees into JSON files, so our team will build a tool to expedite this process. The tool will be implemented into the PIQUE software and allow designers to directly model their risk priorities in the app and store the results in a database, which is also designed by our team. This interactive GUI will save time and effort on both the users' and developers' side.

Requirements

1) Creation of nodes, classifications, children, and description must save to a JSON file
2) A JSON file must be saved locally to the PC and on the database
3) When loaded, a JSON file must display the tree correctly
4) Must be able to edit node properties
5) Must be able to delete links/nodes
6) Must be responsive, fast, and scalable
7) Meets ISO/IEC 9126 and 25010 standards

Architectural Design Documents

Tools Used

Initially, the team looked at creating the PIQUE expansion using Javascript React libraries and the graph database, neo4j. We chose Javascript React because the portion of PIQUE we are expanding was already written in Javascript with React bootstrapped in. In addition, we figured that this would allow the original development team to make edits and changes to the code without the extra effort of learning a new language.

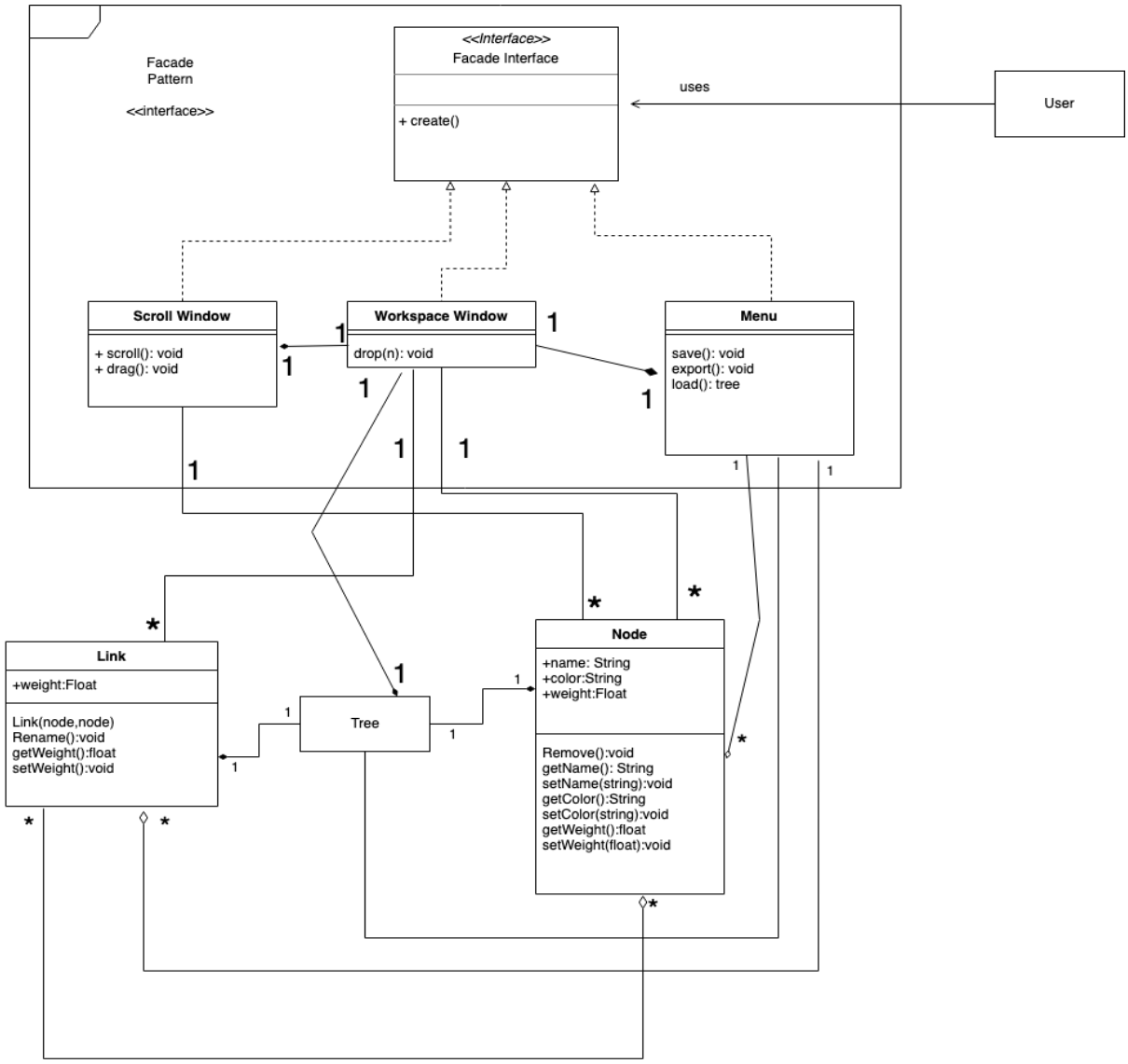
However, we decided not to use neo4j as our database component. Instead, because we're building a database with the intent to store large JSON files, we decided to go with Google Firebase. Google Firebase natively stores information in the JSON file structure, so it made sense to choose a database with that in-built characteristic.

Diagrams

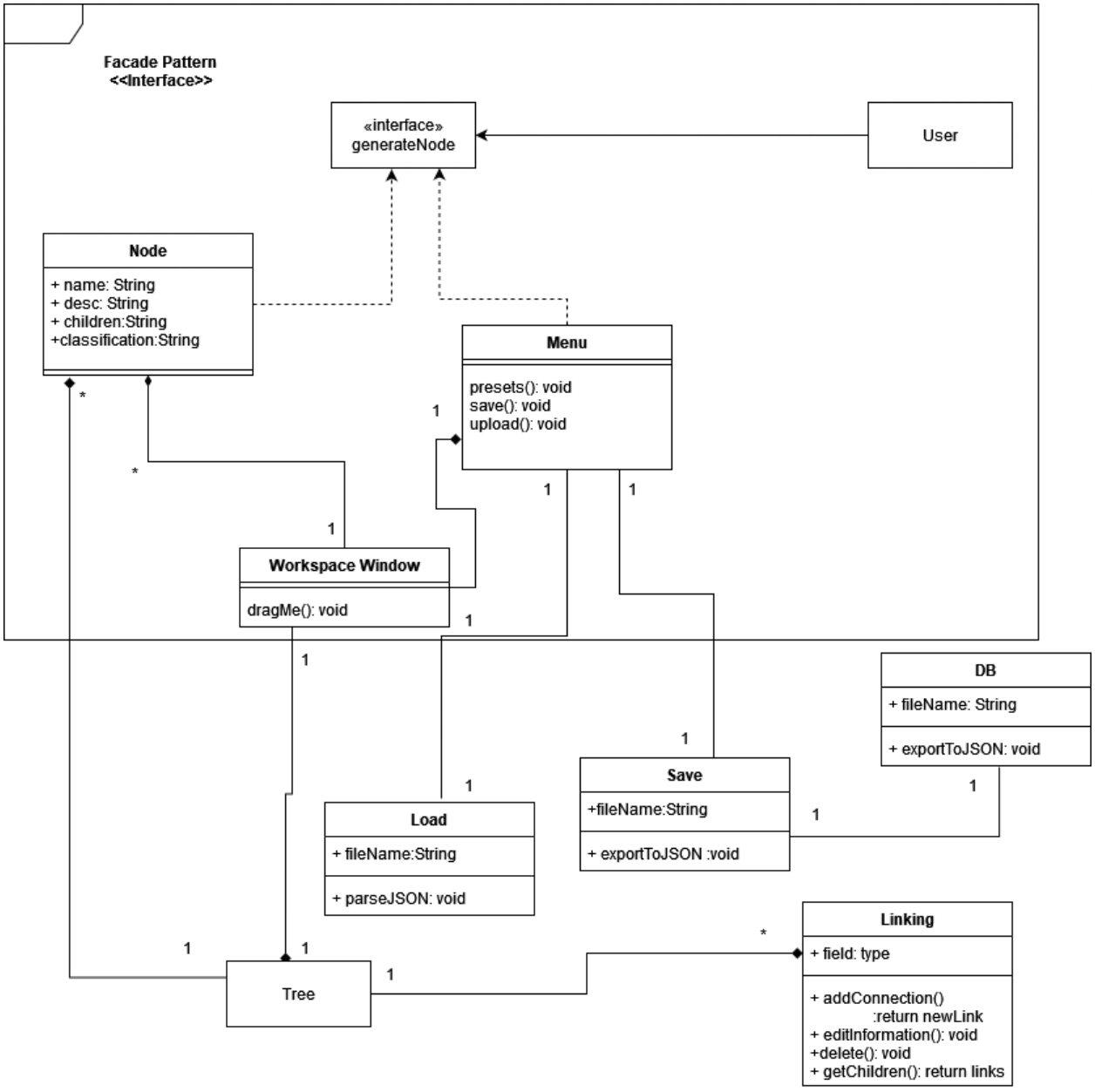
UML Diagram: GUI

The GUI UML diagram is shown below. It utilizes the Facade Pattern, which analogous to a facade in architecture, creates an object that serves as a front-facing interface masking a more complex underlying code [4]. We used this design pattern because it improves the readability and usability of our code: the facade object can be created and all the pieces needed come along with it. Additionally, it removes dependencies on each subsystem allowing us to avoid bugs. Overall, this pattern leaves us with a simplified interface with which to interact.

Proposed

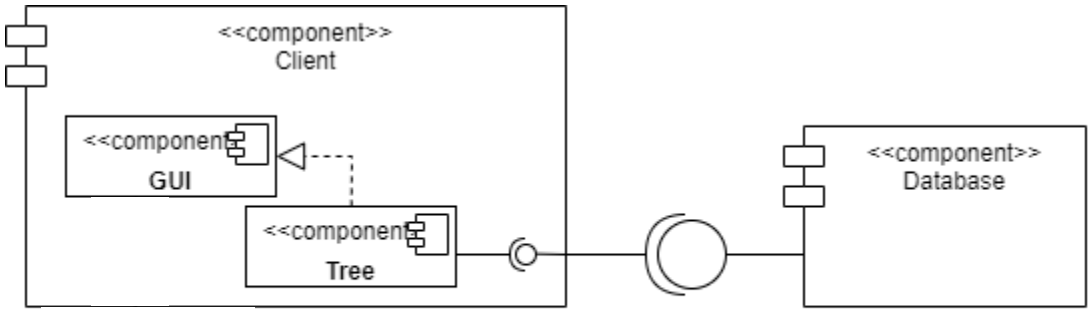

Facade
Pattern
<<interface>>
<<Interface>>
Facade Interface
+ create()
uses
User
Scroll Window
+ scroll(): void
+ drag(): void
Workspace Window
drop(n): void
Menu
save(): void
export(): void
load(): tree
Link
+weight:Float
Link(node,node)
Rename():void
getWeight():float
setWeight():void
Tree
Node
+name: String
+color:String
+weight:Float
Remove():void
getName(): String
setName(string):void
getColor():String
setColor(string):void
getWeight():float
setWeight(float):void

Final


Facade Pattern
<<Interface>>
«interface»
generateNode
User
Node
+ name: String
+ desc: String
+ children:String
+classification:String
Menu
presets(): void
save(): void
upload(): void
Workspace Window
dragMe(): void
DB
+ fileName: String
+ exportToJSON: void
Load
+ fileName:String
+ parseJSON: void
Save
+fileName:String
+ exportToJSON :void
Linking
+ field: type
+ addConnection()
:return newLink
+ editInformation(): void
+delete(): void
+ getChildren(): return links
Tree
1
*

Component Diagram

This diagram demonstrates how all the different components used in this project will interact. Due to the nature of our project, we will not be interacting with the PIQUE software. Instead, we're adding an additional tool that can later be incorporated with the software.

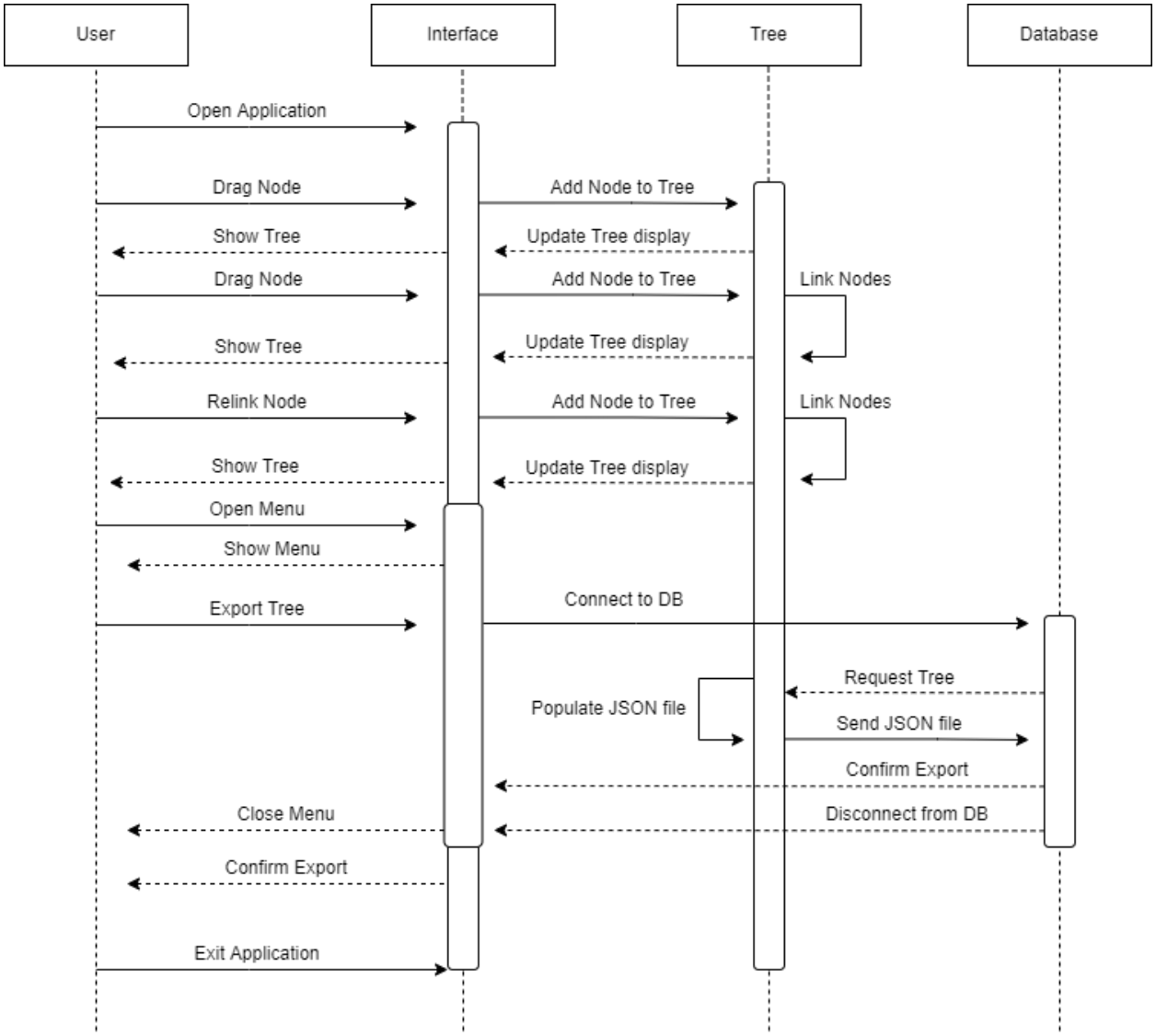


The GUI Use Case Diagram

This shares potential scenarios of interaction with the user. By listing different steps the user could take, we defined necessary components, classes, attributes, and relationships.

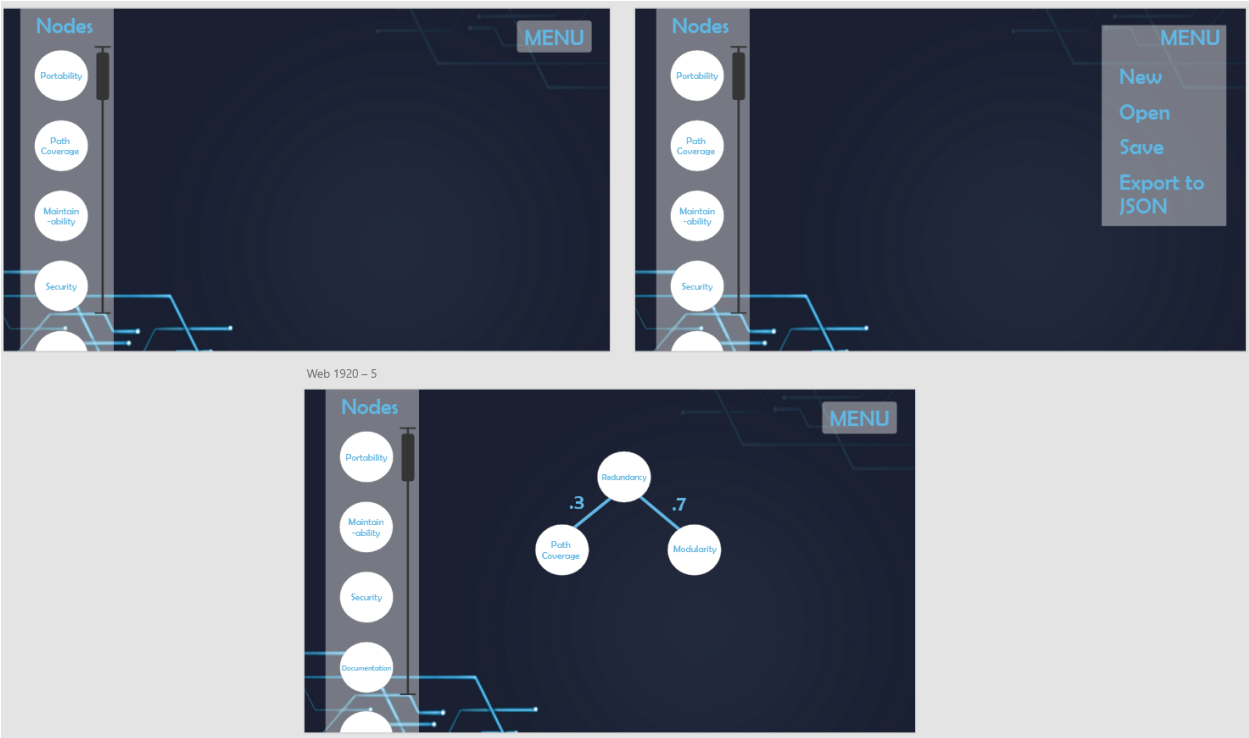
Sequence Diagram

This diagram represents an overview of the sequence of events that will occur when the user interacts with our program. This particular sequence is for a two node tree that is exported to the database.



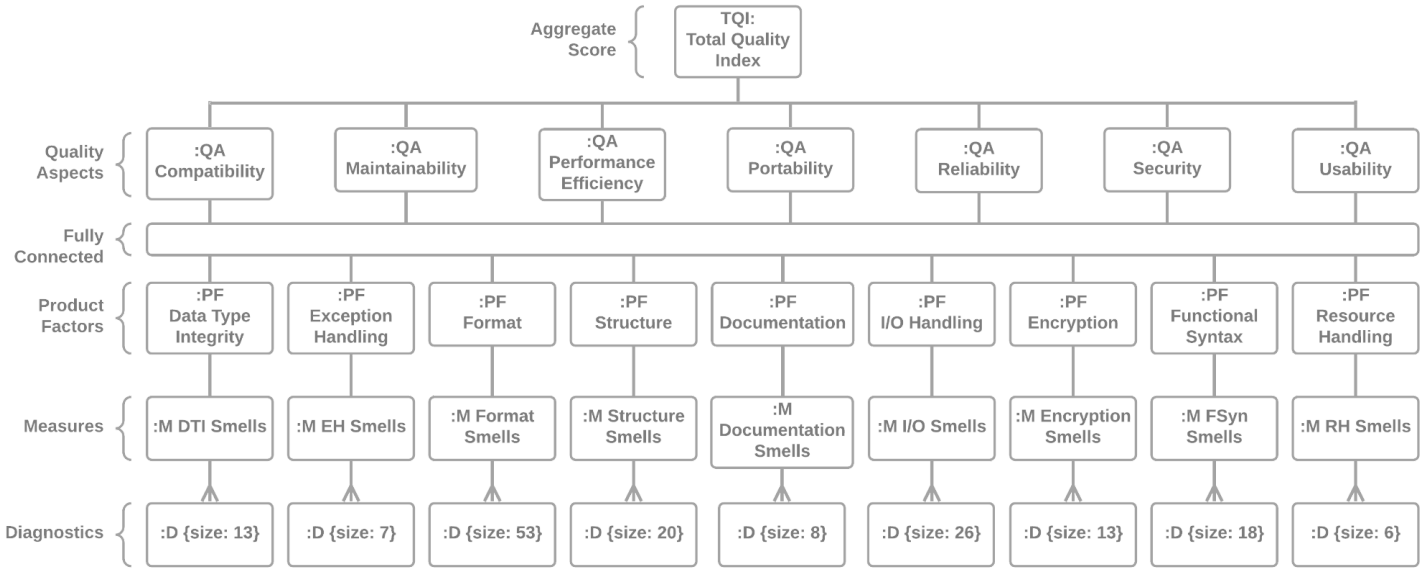
High Fidelity Model of the GUI

This is a snapshot of the model that the GUI team intends to create. The idea is that you drag and drop nodes into the workspace from the scrollbar on the right. The links between two nodes can be created by clicking on a node and dragging your mouse to another node. This design is not set in stone and could change as we dive further into the project.



Generalized PIQUE Assessment Graph Model

The graph model below displays the relationships in the database and the possible schema used. Neo4j, a graph database, will be used for this project, and it supports a paradigm called “whiteboard modeling”, meaning that as the model is developed “on the whiteboard” is how the model is stored internally. The idea behind PIQUE is that a user can pick and choose their quality metrics that will be used for their assessment. This coincides with how the database will store the data - that is, the way the model is created is exactly how the respective data will be stored. At the bottom level of the hierarchy model are the diagnostic nodes, which consist of a myriad of various tools depending on the use case. The approach here uses a bottom-up approach by using the assessment of the various product factors the tool diagnostics could represent to drive the design of the measure layer nodes and the bottom-level product factors[2]. Note: this graph model is intended to be a *generalized* model, meaning that not all nodes and relations present in this diagram necessarily represent an actual user model.



Database Comparison

Initially, the team looked into the graph databases vs relational databases. Graph databases excel as displaying highly connected, retrieving information, and are more flexible with constantly changing nodes. Conversely, relational databases are more static in nature and better at storing information. Relational databases are more transactional.

After more digging we started to look into graph database frameworks like Neo4j and MongoDB. Neo4j, like all graph databases, is flexible, quick, and easy to query. Unfortunately, it's not very scalable and has its own query language that may not be easy to learn. After deliberation with the team and the client we discuss another route. In the past summer the PIQUE Lite team used an online database called Google Firebase. This database excels at JSON storage and that is exactly what we needed.

Our entire project revolves around creating a way to generate, load, edit, and save JSON files. The creation of these JSON files, when loaded, would create a PIQUE tree. This matched Google Firebase perfectly. Google Firebase natively stores all information in JSON files and allows easy access to retrieve and store new files. The downside to this database is that it does require an online connection to save to the database as well as an active google account that the database host can authorize to allow you to view and download these JSON files. Additionally there is a cost if the files become too large. For general use the free version will be fine but there may be cases where there is not enough storage for the files.

We recommend going forward with a Google Firebase implementation, even if it's only to a backup to local storage, with this method any verified user can login from any computer with an internet connection and download their JSON files to continue their work.

Expected Results

By the end of this project we expect to have a working user interactable tree that allows users to create a hierarchical tree for quality modeling. We also expect to have a database that is capable of loading and storing previously created PIQUE trees and all relevant metadata attached to them.

Appendix

App.jsx

```
/**********************
 * PIQUE (Platform for Investigative software Quality Understanding and Evaluation)
 * Quality Model Creation GUI
 * CSCI 483R: Interdisciplinary Capstone Project
 * Dr. Clemente Izurieta
 * Spring 2022
 **********************
 * Developed by:
 *  Maria Gallivan
 *  Dawson Kanehl
 *  Zoe Norden
 *  Connor Snow
 **********************/

/* Imports */
import React, {useState} from "react";
import "./App.css";
import Node from "./components/Node";
import TopBar from "./components/TopBar";
import Xarrow from "./components/Xarrow";
import {Xwrapper} from "react-xarrows";
import {Menu, MenuButton, MenuItem, SubMenu} from '@szhsin/react-menu';
import {getDatabase} from "firebase/database";
import {getStorage, ref, uploadBytes} from "firebase/storage";
import {initializeApp} from "firebase/app";
import {getAnalytics} from "firebase/analytics";
import {Button} from 'react-floating-action-button'
import Select from 'react-select';
import '@szhsin/react-menu/dist/index.css';

/* Developed with code forked from:
 * https://github.com/Eliav2/react-xarrows/tree/master/examples
 */

/* References:
 * https://www.w3schools.com/jsref/jsref_foreach.asp
 * https://www.delftstack.com/howto/javascript/arraylist-in-javascript/
 * https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage
 * https://www.youtube.com/watch?v=Px4Lm8NixtE
 * https://react-select.com/home
 */

// Import the functions you need from the SDKs you need
```

```
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
    apiKey: "AIzaSyALd8fTT_YORi0wwJ7bC_7O347ssGlItvg",
    authDomain: "capstone-pique.firebaseapp.com",
    projectId: "capstone-pique",
    storageBucket: "capstone-pique.appspot.com",
    messagingSenderId: "981342029277",
    appId: "1:981342029277:web:e5b3373ffd6425e60c2234",
    measurementId: "G-S0HHKBR67N"
};

// Initialize Firebase
const
    app = initializeApp(firebaseConfig),
    analytics = getAnalytics(app),
    database = getDatabase(),
    dbstorage = getStorage();

// Options for Node Types
const options = [
    {value:"tqi", label: 'TQI'},
    {value:"quality_aspects", label: 'Quality Aspects'},
    {value:"product_factors", label: 'Product Factors'},
    {value:"measures", label: 'Measures'},
    {value:"diagnostics", label: 'Diagnostics'},
    {value:"name", label:'Model Name'},
    {value:"additionalData", label:'Additional Data'},
    {value:"global_config", label:'Global Config Information'}];

const bool_options = [
    {value: false, label: "False"},
    {value: true, label: "True"}];

const config_options = [
    {value: "benchmark_strategy", label: "Benchmark Strategy"},
    {value: "weights_strategy", label: "Weights Strategy"},
    {value: "normalizer", label: "Normalizer"}];

const
    // Arrays for storing nodes and lines
    storage = [],
    childlines = [],
    // For formatting shapes
    TYPE = ["node"],
```

```
    // Framework for creating/exporting model
    model_object = {
        "name": "Default",
        "additionalData":{},
        "global_config":{},
        "factors": {
            "tqi": {},
            "quality_aspects": {},
            "product_factors": {}
        },
        "measures": {},
        "diagnostics": {}
    };

// For copying current info to compare with new info
let currentNodeInfo = {
    nodeName : null,
    nodeDesc : null,
    nodeType : "other",
    toolName : null,
    is_positive : null,
    config_key: null,
}

// Current node gets set when the edit node function is called on the selected node
let currentNode = {};

const App = () => {
    /* References:
     * https://www.delftstack.com/howto/javascript/arraylist-in-javascript/
     * https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage
     */

    // Default node values
    let
        nodeName = null,
        nodeDesc = null,
        nodeType = "other",
        toolName = null,
        is_positive = null,
        config_key = null;

    const
        [interfaces, setInterfaces] = useState([]),
        [nodes, setNodes] = useState([]),
        [lines, setLines] = useState([]),
        [selected, setSelected] = useState(null),
        [actionState, setActionState] = useState("Normal");
```

```
// Handles Selected Nodes
const handleSelect = (e) => {
    if (e === null) {
        setSelected(null);
        setActionState("Normal");
    } else setSelected({id: e.target.id, type: "node"});
};

// Checks existence of nodes
const checkExistence = (id) => {
    return [...nodes, ...interfaces].map((b) => b.id).includes(id);
};

// Set functions for Node properties
function setName(prop){ nodeName = prop.target.value; }
function setDesc(prop){ nodeDesc = prop.target.value; }
function setType(prop){ nodeType = prop.value; }
function setTool(prop){ toolName = prop.target.value; }
function setBool(prop){ is_positive = prop.value; }
function setConfigKey(prop){ config_key = prop.value; }
// function setConfigValue(prop){ config_value= prop.target.value; }

/*
 * For all these change (set) functions, they check the initial value of the selected node
 * and compare it to the current value coming in, if they are different, then the user provided
 * changes are written to the standard node fields. If they are not different, then the standard
 * node fields get set to the initial values of teh selected node.
 */
function changeName(prop){
    if (prop.target.value !== currentNodeInfo.nodeName)
        nodeName = prop.target.value;
    else
        nodeName = currentNodeInfo.nodeName;
}
function changeDesc(prop){
    if (prop.target.value !== currentNodeInfo.nodeDesc)
        nodeDesc = prop.target.value;
    else
        nodeDesc = currentNodeInfo.nodeDesc;
}
function changeType(prop){
    if (prop.value !== currentNodeInfo.nodeType)
        nodeType = prop.value;
    else
        nodeType = currentNodeInfo.nodeType;
}
```

```
    function changeTool(prop){
        if (prop.target.value !== currentNodeInfo.toolName)
            toolName = prop.target.value;
        else
            toolName = currentNodeInfo.toolName;
    }
    function changeBool(prop){
        if (prop.value !== currentNodeInfo.is_positive)
            is_positive = prop.value;
        else
            is_positive = currentNodeInfo.is_positive;
    }
    function changeConfigKey(prop){
        if (prop.value !== currentNodeInfo.config_key)
            config_key = prop.value;
        else
            config_key = currentNodeInfo.config_key;
    }

    /**
     * Handles node drag-drop functionality. Pushes node information into an entry on the local "storage"
     * array. In addition, when a node is added a properly formatted JSON object is pushed onto the
"model_object"
     * that is used in creating the model that is exported to the user.
     * @function
     */
    function handleDropDynamic() {
        closeForm();
        let l = nodes.length;
        /*TODO: Set function to look for event, get the bounding client,
         * and set x, y coordinates to the bounding rectangle client.
         */
        // let { x, y } = e.target.getBoundingClientRect();
        let object = TYPE[0];
        while (checkExistence("node" + l)) l++;
        if(nodeName === null || nodeName === ""){
            nodeName = "node" + l;
        }
        // This adds to the local storage array containing all nodes
        let newNode = {
            id: nodeName,
            desc: nodeDesc,
            type: nodeType,
            children: {},
            positive: is_positive,
            t_name: toolName,
            c_key: config_key,
            x: 500,
```

```
        y: 500,
        /*TODO: The x and y coordinates should be mapped to the
         * place they are dropped on the screen
         */
        // x: e.clientX - x,
        // y: e.clientY - y,
        shape: object};
    setNodes([...nodes, newNode]);
    storage.push(newNode);
    console.log(storage);
}

/**
 * Grabs the selected node information and manipulates the HTML to display the current node info
 * @function
 */
function showInfo(selected) {
    const index = nodes.findIndex(item => {
        return item.id === selected;
    });
    const name = document.getElementById("info-name");
    if (nodes[index].id != null)
        name.innerHTML = nodes[index].id.toString();
    const desc = document.getElementById("info-desc");
    if (nodes[index].desc != null)
        desc.innerHTML = nodes[index].desc.toString();
    const type = document.getElementById("info-type");
    if (nodes[index].type != null)
        type.innerHTML = nodes[index].type.toString();
    const pos = document.getElementById("info-pos");
    if (nodes[index].positive != null)
        pos.innerHTML = nodes[index].positive.toString();
    const tool = document.getElementById("info-tool");
    if (nodes[index].t_name != null)
        tool.innerHTML = nodes[index].t_name.toString();
    const config = document.getElementById("info-config");
    if (nodes[index].c_key != null)
        config.innerHTML = nodes[index].c_key.toString();
    openInfo();
}

function setEdit(selected) {
    let index = nodes.findIndex(element => element.id === selected);
    currentNode = nodes[index];
    currentNodeInfo.nodeName = currentNode.id;
    currentNodeInfo.nodeDesc = currentNode.desc;
    currentNodeInfo.nodeType = currentNode.type;
    currentNodeInfo.toolName = currentNode.t_name;
```

```
    currentNodeInfo.is_positive = currentNode.positive;
    currentNodeInfo.config_key = currentNode.config_key;
    currentNodeInfo.config_value = currentNode.config_value;
    openEdit();
}

/**
 * @function
 * Maps all accepted changes to the currently selected node object. Called in the onClick event of
 * the HTML element for the edit menu
 * @param props provided by props const array
 * @param selected when edit is called (within TopBar.jsx) a copy of the selected item in the
 * storage array is assigned to a global currentNode object.
 */
function acceptEdit(props, selected) {
    closeEdit();
    props.setNodes((storage) => {
        if ([...storage, ...props.interfaces].map((a) => a.id).includes(nodeName)) {
            alert('Name already in use, please choose another!');
        }
        else if (nodeName === null) {
            return;
        }
        return storage.map((node) => (node.id === selected.id ? {
            ...node,
            id: nodeName,
            desc: nodeDesc,
            type: nodeType,
            t_name : toolName,
            is_pos : is_positive,
            c_key: config_key,
        } : node));
    });
}

/**
 * Prompts user for filename and creates a JSON file that is downloaded the user's
 * downloads folder.
 * @function
 */
function nameFile(){
    populate_model();
    let d = new Date();
    let t = d.getMonth() + "_" + d.getDay() + "_" + d.getHours() + ":" + d.getMinutes();
    let fileName = window.prompt("Enter the filename: ", t);
    export_to_JSON(fileName);
}
```

```
/**
 * As nodes are added to the screen, they are added into the existing global JSON "model object",
 * however, the children cannot be added to the model object until after the nodes are defined.
 * This function parses the model object with the parent key name in hand and adds the child name
 * to the parent - children structure.
 * @function
 */
function addChildren(l){
    let parent = l.props.start,
        child = l.props.end;
    const obj = JSON.parse(JSON.stringify(model_object)),
        factors = obj.factors,
        measures = obj.measures;
    for (let factor in factors){
        switch(factor) {
            case "tqi":
                let tqi_type = factors.tqi;
                for (const name in tqi_type) {
                    if (name === parent) {
                        model_object.factors.tqi[parent].children[child] = {};
                    }
                }
                break;
            case "quality_aspects":
                let qa_type = factors.quality_aspects;
                for (const name in qa_type) {
                    if (name === parent) {
                        model_object.factors.quality_aspects[parent].children[child] = {};
                    }
                }
                break;
            case "product_factors":
                let pf_type = factors.product_factors;
                for (const name in pf_type) {
                    if (name === parent) {
                        model_object.factors.product_factors[parent].children[child] = {};
                    }
                }
                break;
            default:
                console.log("Cannot add node to model.");
        }
    }
    for (let name in measures) {
        if (name === parent) {
            model_object.measures[parent].children[child] = {};
        }
    }
```

```
    // Diagnostic type nodes don't have children, so we don't need to worry about those.
}

function export_to_JSON(prop) {
    lines.forEach(addChildren);
    nodes.sort((a, b) => (a.type > b.type) ? 1 : -1);
    const data = new Blob([JSON.stringify(model_object)], {type: 'application/json'});
    const a = document.createElement('a');
    a.download = (prop + ".json");
    a.href = URL.createObjectURL(data);
    a.addEventListener('click', (e) => {
        setTimeout(() => URL.revokeObjectURL(a.href), 30 * 1000);
    });
    a.click();
    //saves the json to the DB under the same name that the user entered
    let storageRef = ref(dbstorage, 'uploaded/' + prop);
    uploadBytes(storageRef, data).then((snapshot) => {
        console.log('Uploaded a blob or file!');
    });
    window.alert("JSON data is save to " + prop + ".json");
}

/**
 * The incoming JSON file is stored as a local object and passed into this function, it
 * parses though it and adds entries to our local storage object. Later that object storage
 * is iterated over and the nodes are populated onto the screen.
 * @function
 */
function parse_JSON(incoming_json){
    // makes JSON object parse-able
    const obj = JSON.parse(JSON.stringify(incoming_json));
    const
        name = obj.name,
        config = obj.global_config,
        factors = obj.factors,
        measures = obj.measures,
        diagnostics = obj.diagnostics;
    // handles model name
    store_node_from_JSON(
        name,
        "",
        "name",
        null,
        null,
        null,
        null);
    // handles config nodes
    for (let config_type in config) {
```

```
        switch(config_type) {
            case "benchmark_strategy":
                store_node_from_JSON(
                    config[config_type],
                    null,
                    "global_config",
                    null,
                    null,
                    null,
                    config_type
                )
                break;
            case "normalizer":
                store_node_from_JSON(
                    config[config_type],
                    null,
                    "global_config",
                    null,
                    null,
                    null,
                    config_type
                )
                break;
            case "weights_strategy":
                store_node_from_JSON(
                    config[config_type],
                    null,
                    "global_config",
                    null,
                    null,
                    null,
                    config_type
                )
                break;
        }
    }
    for (let factor in factors){
        switch(factor) {
            case "tqi":
                let tqi_type = factors.tqi;
                for (const data in tqi_type) {
                    store_node_from_JSON(
                        data,
                        tqi_type[data].description,
                        factor,
                        tqi_type[data].children,
                        null,
                        null
```

```
                )
            }
            break;
        case "quality_aspects":
            let qa_type = factors.quality_aspects;
            for (const data in qa_type) {
                store_node_from_JSON(
                    data,
                    qa_type[data].description,
                    factor,
                    qa_type[data].children,
                    null,
                    null
                )
            }
            break;
        case "product_factors":
            let pf_type = factors.product_factors;
            for (const data in pf_type) {
                store_node_from_JSON(
                    data,
                    pf_type[data].description,
                    factor,
                    pf_type[data].children,
                    null,
                    null
                )
            }
            break;
        default:
            console.log("Key:Value pair not in model.");
    }
}
// measures
for (let data in measures) {
    store_node_from_JSON(
        data,
        measures[data].description,
        "measures",
        measures[data].children,
        measures[data].positive,
        null
    )
}
// diagnostics
for (let data in diagnostics) {
    store_node_from_JSON(
        data,
```

```
            diagnostics[data].description,
            "diagnostics",
            diagnostics[data].children,
            null,
            diagnostics[data].toolName
        )
    }
    // Copies loaded JSON into model object for exporting
    populate_model();
}

/**
 * Populates the model object when a file is uploaded. The store_node_from_JSON function only populates the
 * local storage array that holds the node info necessary for populating the screen. Without this function,
 * a blank file will be exported.
 * @function
 */
function populate_model() {
    for (let i = 0; i < nodes.length; i++) {
        let nodeType = nodes[i].type,
            config_key = nodes[i].c_key,
            nodeName = nodes[i].id,
            nodeDesc = nodes[i].desc,
            children = nodes[i].children,
            is_positive = nodes[i].positive,
            toolName = nodes[i].t_name;
        switch (nodeType) {
            case "name":
                model_object.name = nodeName;
                break;
            case "global_config":
                model_object.global_config[config_key] = nodeName;
                break;
            case "tqi":
                model_object.factors.tqi[nodeName] = {};
                model_object.factors.tqi[nodeName].description = nodeDesc;
                model_object.factors.tqi[nodeName].children = children;
                break;
            case "quality_aspects":
                model_object.factors.quality_aspects[nodeName] = {};
                model_object.factors.quality_aspects[nodeName].description = nodeDesc;
                model_object.factors.quality_aspects[nodeName].children = children;
                break;
            case "product_factors":
                model_object.factors.product_factors[nodeName] = {};
                model_object.factors.product_factors[nodeName].description = nodeDesc;
                model_object.factors.product_factors[nodeName].children = children;
                break;
```

```
            case "measures":
                model_object.measures[nodeName] = {};
                model_object.measures[nodeName].description = nodeDesc;
                model_object.measures[nodeName].positive = is_positive;
                model_object.measures[nodeName].children = children;
                break;
            case "diagnostics":
                model_object.diagnostics[nodeName] = {};
                model_object.diagnostics[nodeName].description = nodeDesc;
                model_object.diagnostics[nodeName].toolName = toolName;
                break;
            default:
                console.log("Key:Value pair not in model.");
        }
    }
}

// variables to remember number of each type of node passed to store_node_from_JSON
let t, c = 0;
// Formats incoming JSON into the proper format for viewing on screen
function store_node_from_JSON(nodeName, nodeDesc, nodeType, nodeChildren, isPositive, toolName,
configType){
    let object = TYPE[0],
        nodewidth = nodeName.toString().length,
        xpos, ypos;

    // sets height of node by Type
    switch (nodeType){
        case "name":
            ypos = 90;
            break;
        case "global_config":
            ypos = 180;
            break;
        case "tqi":
            ypos = 270;
            break;
        case "quality_aspects":
            ypos = 360;
            break;
        case "product_factors":
            ypos = 450;
            break;
        case "measures":
            ypos = 540;
            break;
        case "diagnostics":
            ypos = 630;
```

```
            break;
        default:
            console.log("Node has no place in model.");
    }
    if(t !== nodeType){
        c = 0;
        t = nodeType;
    } else{
        c++;
    }
    //sets x placement of node by number type already places
    //xpos = 850 + c*150*(Math.pow(-1, storage.length % 2)); //more centered
    xpos = 250 - nodewidth*3.2 + c*200; //left justified

    //creates the node from load
    let newNode = {
        id: nodeName,
        desc: nodeDesc,
        type: nodeType,
        children: nodeChildren,
        positive: isPositive,
        t_name: toolName,
        c_key: configType,
        x: xpos,
        y: ypos,
        shape: object};
    setNodes([...nodes, newNode]);
    nodes.push(newNode);
    for (let k in nodeChildren) {
        let p = {props: {start: nodeName, end: k}};
        setLines([...lines, p]);
        childlines.push(p);
    }
}

// Displays info popup
function openInfo() {
    document.getElementById("info").style.display = "block";
}
// Hides info popup
function closeInfo() {
    document.getElementById("info").style.display = "none";
    const name = document.getElementById("info-name");
    name.innerHTML = "No Name";
    const desc = document.getElementById("info-desc");
    desc.innerHTML = "No Description";
    const type = document.getElementById("info-type");
    type.innerHTML = "No Type";
```

```
    const pos = document.getElementById("info-pos");
    pos.innerHTML = "Invalid";
    const tool = document.getElementById("info-tool");
    tool.innerHTML = "No Tool Name";
    const config = document.getElementById("info-config");
    config.innerHTML = "No Config Type";
}
// Display edit node information popup
function openEdit() {
    document.getElementById("edit").style.display = "block";
}
function closeEdit() {
    document.getElementById("edit").style.display = "none";
    document.getElementById("inputName").value = "";
    document.getElementById("inputDesc").value = "";
    document.getElementById("toolName").value = "";
}
// Displays entry form popup
function openForm() {
    document.getElementById("popup").style.display = "block";
}
// Hides and resets entry form popup
function closeForm() {
    document.getElementById("popup").style.display = "none";
    document.getElementById("inputName").value = "";
    document.getElementById("inputDesc").value = "";
    document.getElementById("toolName").value = "";
    //I just don't know why it won't reset like inputName and Desc
    //document.getElementsByTagName("Select").defaultValue = {value: "other", label: 'Other'};
}

/**
 * This function is called on the click event for the preset options. It takes a JSON preset name,
 * defines the local file path to that JSON, loads the JSON into local storage, and passes the
 * JSON object to the parse_JSON function to be read in and loaded to screen.
 * @function
 */
function load_preset(name) {
    let filename = name + ".json",
        JSON_preset = require(`./presets/${filename}`);
    parse_JSON(JSON_preset);
}

/**
 * Prompts user for file name and loads a user provided file that must be located within the
 * user_uploads folder.
 */
function load_file() {
```

```
    let name = window.prompt("Enter file name (without the file extension)"),
        filename = name + ".json",
        loaded_JSON = require(`./user_uploads/${filename}`);
    parse_JSON(loaded_JSON);
}


// Properties
const props = {
    interfaces,
    setInterfaces,
    nodes: nodes,
    setNodes: setNodes,
    selected,
    showInfo,
    setEdit,
    handleSelect,
    actionState,
    setActionState,
    lines,
    setLines
};

// Node properties
const nodeProps = {
    nodes: nodes,
    setNodes: setNodes,
    selected,
    showInfo,
    setEdit,
    handleSelect,
    actionState,
    setLines,
    lines
};

///download from firebase here
```

```
// HTML
return (
    <div>
        {/* Workable area needs to be wrapped in Xwrapper so Xarrows dynamically re-render */}
        <Xwrapper>
            {/* Root Canvas */}
            <div
                className="canvasStyle"
                id="canvas"
```

```
onClick={() => handleSelect(null)} >
{/* Drag and Drop Tool Bar*/}
<div className="toolboxMenu">
    {TYPE.map((shapeName) => (
        <div
            key={shapeName}
            className={shapeName}
            onDragStart={(e) =>
                e.dataTransfer.setData("shape", shapeName)
            }
            draggable
        >
            {"Drag Me!"}
        </div>
    ))}
</div>
{/* Nodes Play Space */}
<div
    id="nodesContainer"
    className="nodesContainer"
    onDragOver={(e) => e.preventDefault()}
    onDrop={openForm}
>
    {/* Dropdown Node Options */}
    <TopBar {...props} />
    {/* New Node Mapping */}
    {nodes.map((node, i) => ( <Node
            {...nodeProps}
            key={i} // this seems to be the way to make sure every child has a unique id in a list
            node={node}
            position="absolute"
            sidePos="middle"
        />
    ))}
    {/* Add Node Popup Menu */}
    <div className="form-popup" id="popup">
        <div className="form-container" id="form">
            <h2>Input Node Information</h2>
            <b>Node Name</b>
            <input type="text"
                placeholder="Name"
                id="inputName"
                onChange={setName}/>
            <b>Description</b>
            <input type="text"
                placeholder="Description"
                id="inputDesc"
                onChange={setDesc}/>
```

```
                <br/>
                <b>Classification</b>
                {/*tqi, quality_aspects, product_factors, measures, diagnostics*/}
                {/* drop-down messed up for showing values or resetting value*/}
                <Select id="inputType"
                        options={options}
                        value={"Other"}
                        onChange={setType} />
                <br/>
                <b>Positive? (for Measures)</b>
                <Select id="positiveType"
                        options={bool_options}
                        value={"Bool"}
                        onChange={setBool} />
                <br/>
                <b>Tool Name (for Diagnostics)</b>
                <input type="text"
                       placeholder="Name"
                       id="toolName"
                       onChange={setTool}/>
                <br/>
                <b>Global Config Info</b>
                <Select type="configType"
                        options={config_options}
                        value={"Config"}
                        onChange={setConfigKey} />
                {/*<input type="text"*/}
                {/*       placeholder="Config Value"*/}
                {/*       id="configValue"*/}
                {/*       onChange={setConfigValue}/>*/}
                {/* Submission Button */}
                <Button
                  id="submit-btn"
                  tooltip="Submit"
                  styles={{backgroundColor: "#04AA6D", color: "#FFFFFF"}}
                  onClick={handleDropDynamic}
                />
              </div>
            </div>
            {/* Edit node popup menu*/}
            <div className="edit-popup" id="edit">
              <div className="edit-container" id="display-edit">
                <h2>Edit Node Information</h2>
                <b>Change Node Name</b>
                <input type="text"
                       placeholder="Name"
                       id="name-change"
                       onChange={changeName}/>
```

```
            <b>Change Description</b>
            <input type="text"
                placeholder="Description"
                id="desc-change"
                onChange={changeDesc}/>
            <br/>
            <b>Change Classification</b>
            <Select id="type-change"
                options={options}
                value={"Other"}
                onChange={changeType} />
            <br/>
            <b>Change Positive? (for Measures)</b>
            <Select id="pos-change"
                options={bool_options}
                value={"Bool"}
                onChange={changeBool} />
            <br/>
            <b>Change Tool Name (for Diagnostics)</b>
            <input type="text"
                placeholder="Name"
                id="tool-change"
                onChange={changeTool}/>
            {/* Submission Button */}
            <Button
              id="submit-btn"
              tooltip="Submit"
              styles={{backgroundColor: "#f65503", color: "#FFFFFF"}}
              onClick={function(){acceptEdit(props, currentNode)}}
            />
          </div>
        </div>
        {/* Node Information Popup */}
        <div className="info-popup" id="info">
          <div className="info-container" id="display-info">
            <h2>Current Node Info</h2>
            <b>Name</b>
            <p className="tab" id="info-name">No Name</p>
            <b>Description</b>
            <p className="tab" id="info-desc">No Description</p>
            <b>Classification</b>
            <p className="tab" id="info-type">No Type</p>
            <b>Positive</b>
            <p className="tab" id="info-pos">Invalid</p>
            <b>Tool Name</b>
            <p className="tab" id="info-tool">No Tool Name</p>
            <b>Config Type</b>
            <p className="tab" id="info-config">No Config Type</p>
```

```
                    <Button
                      tooltip="Exit"
                      styles={{backgroundColor: "red", color: "#FFFFFF"}} onClick={closeInfo}
                    />
                  </div>
                </div>
                {/* Menu Interface */}
                <div className="Menu">
                  <Menu menuButton={<MenuButton className="btn-primary">Menu</MenuButton>}>
                    <MenuItem onClick={load_file}>Upload</MenuItem>
                    <MenuItem onClick={nameFile}>Save</MenuItem>
                    {<><SubMenu label="Preset">
                      <MenuItem id="csharp"
                            value="test"
                            onClick={function(){load_preset('pique-csharp-sec-model')}}
                      >Csharp Model
                      </MenuItem>
                      <MenuItem id="bin"
                            value="test"
                            onClick={function(){load_preset('pique-bin-model')}}
                      >Bin Model
                      </MenuItem>
                      {/*TODO: Add more presets here, if necessary*/}
                    </SubMenu></>}
                  </Menu>
                </div>
              </div>
              {/* Xarrow Connections for Building New Models */}
              {lines.map((line, i) => (
                <Xarrow
                  key={line.props.root + "-" + line.props.end + i}
                  line={line}
                  selected={selected}
                  setSelected={setSelected}
                />
              ))}
              {/* Xarrow Connections for Loading Preset and Existing Models */}
              {childlines.map((line, i) => (
                <Xarrow
                  key={line.props.start + "-" + line.props.end + i}
                  line={line}
                  start={line.props.start}
                  end={line.props.end}
                />
              ))}
            </div>
          </Xwrapper>
        </div>
```

```
    );
};

export default App;
```

./components/Node.jsx

```
// Forked from: https://github.com/Eliav2/react-xarrows/tree/master/examples
import React from "react";
import "./Node.css";
import Draggable from "react-draggable";
import { useXarrow } from "react-xarrows";

const Node = (props) => {
    const updateXarrow = useXarrow();
    const handleClick = (e) => {
        //so only the click event on the node will fire on not on the container itself
        e.stopPropagation();
        if (props.actionState === "Normal") {
            props.handleSelect(e);
        } else if (
            props.actionState === "Add Connections" &&
            props.selected.id !== props.node.id
        ) {
            props.setLines((lines) => [
                ...lines,
                {
                    props: { start: props.selected.id, end: props.node.id },
                }
            ]);
        } else if (props.actionState === "Remove Connections") {
            props.setLines((lines) =>
                lines.filter(
                    (line) =>
                        !(line.root === props.selected.id && line.end === props.node.id)
                )
            );
        }
    };

    let background = null;
    if (props.selected && props.selected.id === props.node.id) {
        background = "#24bd57";
    } else if (
        (props.actionState === "Add Connections" &&
```

```
        props.lines.filter(
          (line) => line.root === props.selected.id && line.end === props.node.id
        ).length === 0) ||
      (props.actionState === "Remove Connections" &&
        props.lines.filter(
          (line) => line.root === props.selected.id && line.end === props.node.id
        ).length > 0)
    ) {
      background = "#ffeb33";
      // Keep the model name and the config nodes the normal color to imply that you don't draw connections to it
      if ((props.node.type === "name") || (props.node.type === "global_config")) {
        background = "#00b1e1"
      }
    }
```

```
    return (
      <React.Fragment>
        <Draggable
          onStart={() => props.position !== "static"}
          bounds="parent"
          onDrag={updateXarrow}
        >
          <div
            ref={props.node.reference}
            className={`${props.node.shape} ${props.position} hoverMarker`}
            style={{
              left: props.node.x,
              top: props.node.y,
              background
            }}
            onClick={handleClick}
            id={props.node.id}
          >
            {props.node.name ? props.node.name : props.node.id}
          </div>
        </Draggable>
      </React.Fragment>
    );
};

export default Node;
```

./components/Xarrow.jsx

```
import React, { useState } from 'react';
import Xarrow from 'react-xarrows';

export default ({ setSelected, selected, line: { props } }) => {
  const [state, setState] = useState({ color: 'coral' });
  const defProps = {
    passProps: {
      className: 'xarrow',
      onMouseEnter: () => setState({ color: 'IndianRed' }),
      onMouseLeave: () => setState({ color: 'coral' }),
      onClick: (e) => {
        //so only the click event on the node will fire on not on the container itself
        e.stopPropagation();
        setSelected({
          id: { start: props.root, end: props.end },
          type: 'arrow',
        });
      },
      cursor: 'pointer',
    },
  };
  let color = state.color;
  if (selected && selected.type === 'arrow' && selected.id.root === props.root && selected.id.end ===
props.end)
    // selected.stopPropagation();
    color = 'red';
```

```
  return <Xarrow {...{
    ...defProps,
    ...props,
    ...state,
    color,
    // TODO: You can change this field to true to show the heads
    showHead: false,
    path: "straight"}} />;
};
```

./components/TopBar.jsx

```
// Forked from: https://github.com/Eliav2/react-xarrows/tree/master/examples
import React from 'react';
import './TopBar.css';

const actions = {
```

```
    node: [
      'Edit Node Information',
      'Show Information',
      'Add Connections',
      'Delete Node'],
    arrow: ['Edit Properties', 'Remove Connection'],
};

const TopBar = (props) => {
    const handleEditAction = (action) => {
      switch (action) {
        case 'Edit Node Information':
          props.setEdit(props.selected.id);
          break;
        case 'Add Connections':
          props.setActionState(action);
          break;
        case 'Remove Connections':
          props.setActionState(action);
          break;
        case 'Show Information':
          props.showInfo(props.selected.id);
          break;
        case 'Remove Connection':
          props.setLines((lines) =>
            lines.filter(
              (line) => !(line.props.root === props.selected.id.root && line.props.end === props.selected.id.end)
            )
          );
          break;
        case 'Edit Properties':
          props.setLines((lines) =>
            lines.map((line) =>
              line.props.root === props.selected.id.root && line.props.end === props.selected.id.end
                ? {
                  ...line,
                  menuWindowOpened: true,
                }
                : line
            )
          );
          break;
        case 'Delete Node':
          if (window.confirm(`Are you sure you want to delete ${props.selected.id}?`)) {
            // first remove any lines connected to the node.
            props.setLines((lines) => {
              return lines.filter(
                (line) => !(line.props.root === props.selected.id || line.props.end === props.selected.id)
```

```
                );
            });
            // if it is a node remove from nodes
            if (props.nodes.map((node) => node.id).includes(props.selected.id)) {
                props.setNodes((nodes) => nodes.filter((node) => !(node.id === props.selected.id)));
            }
            props.handleSelect(null);
        }
        break;
    default:
    }
};

const returnTopBarAppearance = () => {
    let allowedActions = [];
    if (props.selected) allowedActions = actions[props.selected.type];
    switch (props.actionState) {
        case 'Normal':
            return (
                <div className="actionBubbles">
                    {allowedActions.map((action, i) => (
                        <div className="actionBubble" key={i} onClick={() => handleEditAction(action)}>
                            {action}
                        </div>
                    ))}
                </div>
            );
        case 'Add Connections':
            return (
                <div className="actionBubbles">
                    <p>Which node do you want to connect to?</p>
                    <div className="actionBubble" onClick={() => props.setActionState('Normal')}>
                        Finish
                    </div>
                </div>
            );

        case 'Remove Connections':
            return (
                <div className="actionBubbles">
                    <p>Which connection do you want to remove?</p>
                </div>
            );
        default:
    }
};
```

```
    return (
        <div
            className="topBarStyle"
            style={{ height: props.selected === null ? '0' : '60px' }}
            onClick={(e) => e.stopPropagation()}>
            <div className="topBarLabel" onClick={() => props.handleSelect(null)}> </div>
            {returnTopBarAppearance()}
        </div>
    );
};

export default TopBar;
```

App.css

```
:root {
    --interfacesBarWidthPx: 100px;
    --canvasWidthVw: 100%;
    --canvasHeightVh: 100%;
    background: #1b2034;
}

.tab {
    margin-left: 20px;
}

.fixedBoxStyle {
    border: 1px #999 solid;
    border-radius: 10px;
    text-align: center;
    position: static;
    width: 100%;
    height: 50px;
    display: flex;
    align-items: center;
    justify-content: center;
    background: white;
}
.fixedBoxStyle:hover {
    background: rgb(230, 230, 230);
}

.interfacesBarStyle {
    /* position: relative; */
    display: flex;
    flex-direction: column;
```

```
    justify-content: space-evenly;
    align-items: center;
    height: 100%;
    width: var(--interfacesBarWidthPx);
    border: solid black 1px;
}

.interfaceTitleStyle {
    font-size: 15px;
    color: black;
    position: absolute;
    margin-top: 5px;
    top: 0;
}

.nodesContainer {
    position: relative;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
    width: 100%;
}

div.Menu{
    text-align: right;
    font-family: Arial, Helvetica, sans-serif;
    /*color:white;*/
    padding: 8px;
    position:-webkit-sticky;
    position: sticky;
    top: 40px;
}
.btn-primary{
    text-align: center;
    font-family: Arial, Helvetica, sans-serif;
    font-size: large;
    color:white;
    background-color: #00B1E1;
    padding: 8px;
    margin-right: 100px;
    border-radius: 20%;
}

/*code adapted from https://www.w3schools.com/howto/howto_js_popup_form.asp*/
.info-popup {
    display: none;
    border: 3px solid #f1f1f1;
    border-radius: 5%;
```

```
    z-index: 9;
    position:absolute;
    top:50%;
    left:50%;
    margin-left:-130px;
    margin-top:-360px;
}

.info-container {
    max-width: 300px;
    border-radius: 4%;
    padding: 10px;
    background-color: white;
    color: black;
}

.form-popup {
    display: none;
    border: 3px solid #f1f1f1;
    border-radius: 5%;
    z-index: 9;
    position:absolute;
    top:50%;
    left:50%;
    margin-left:-175px;
    margin-top:-360px;
}

.form-container {
    max-width: 300px;
    padding: 10px;
    background-color: white;
    color: black;
    border-radius: 4%;
}

.form-container input[type=text] {
    width: 75%;
    padding: 15px;
    margin: 5px 0 22px 0;
    border: none;
    background: #f1f1f1;
}

.edit-popup {
    display: none;
    border: 3px solid #f1f1f1;
    border-radius: 5%;
```

```
    z-index: 9;
    position:absolute;
    top:50%;
    left:50%;
    margin-left:-175px;
    margin-top:-360px;
}

.edit-container {
    max-width: 300px;
    padding: 10px;
    background-color: white;
    color: black;
    border-radius: 4%;
}

.edit-container input[type=text] {
    width: 75%;
    padding: 15px;
    margin: 5px 0 22px 0;
    border: none;
    background: #f1f1f1;
}

.canvasStyle {
    display: flex;
    position: relative;
    width: var(--canvasWidthVw);
    height: var(--canvasHeightVh);
    color: black;
    background-image: url("img/background.jpg");
    background-size: contain;
    background-repeat: repeat-y;
    background-color: #1b2034;
}

.toolboxMenu {
    width: 90px;
    height: 90px;
    background: transparent;
    text-align: center;
    position: fixed;
    right: 25px;
    bottom: 25px;
    padding: 5px;
    z-index: 8;
}
```

```
.node {
    border: 1px black solid;
    text-align: center;
    border-radius: 50%;
    background: #00B1E1;
    display: flex;
    height: 100%;
    justify-content: center;
    align-items: center;
    padding: 10px;
    margin: 10px;
}

.node {
    height: 50px;
}

.switchIcon {
    top: 0;
    width: 50px;
    height: 50px;
}

.iconContainer {
    background: white;
    border-color: #999;
    border-style: solid;
    border-width: 1px;
    display: flex;
    align-items: center;
    flex-direction: column;
    border-radius: 10px;
}
```

./components/Node.css

```
/* XArrows Code forked from: https://github.com/Eliav2/react-xarrows/tree/master/examples */
.absolute {
  position: absolute;
}
.hoverMarker:hover {
  background: #00e1e1;
}
.switchIcon {
  top: 0;
  width: 50px;
```

```
  height: 50px;
}

.iconContainer {
  border-color: #999;
  border-style: solid;
  border-width: 1px;
  display: flex;
  align-items: center;
  flex-direction: column;
  border-radius: 10px;
}
```

./components/TopBar.css

```
/* Forked from: https://github.com/Eliav2/react-xarrows/tree/master/examples */

.topBarStyle {
  display: flex;
  overflow: hidden;
  position: absolute;
  width: 100%;
  background: transparent;
  padding: 0 20px;
  transition: all 0.2s ease-out;
  height: 90px;
  align-items: center;
  /* border: solid 1px black; */
  /* z-index: 1; */
  /* overflow-x: auto; */
}

.topBarLabel {
  display: flex;
  align-items: center;
  border-radius: 30px;
  background: white;
  margin: 5px;
}
.topBarLabel:hover {
  background: azure;
}
.topBarLabel:active {
  background: deepskyblue;
}

.actionBubbles {
```

```
  color: white;
  display: flex;
  align-items: center;
  width: calc(100% - 200px);
  justify-content: space-evenly;
}

.actionBubble {
  background: #999;
  border-radius: 20px;
  height: 80%;
  display: flex;
  align-items: center;
  padding: 10px;
  margin: 5px;
}
.actionBubble:hover {
  background: #00e1e1;
}
.actionBubble:active {
  background: DeepSkyBlue;
}
```

Resumes

Dawson Kanehl

(727)-409-3369 kanehld@yahoo.com

2336 Trail Crest Drive, Bozeman, MT 59718

EXPERIENCE

Cybersecurity Education and Research, Bozeman Mt. — *Internship*

Current

DOD funded internship that empowers and teaches AFROTC students interested in Cyber security through internships and hands-on teaching. Complete online training and a cybersecurity focused capstone.

ACE Cyber Security Boot Camp, Rome Ny. — *Internship*

June 2020 - Aug. 2020

Created websites for DOD and completed various security cyber tasks. Lead at the tactical level as a commander for mach drone warfare. Managed a small team to accomplish tasks efficiently.

The Chocolate Moose, Bozeman Mt. — *Server and Creator*

June 2018 - Feb 2020

Created new chocolate desserts, milkshakes, coffees, and window displays for the business. Consistently provides premier customer service.

OPS Air Force PDT, Patrick AFB FL. — *Class Leader*

June 2019

Was selected as class leader and was responsible for maintaining accountability, being the link between cadets and cadre, as well as handling any concern or issue that other cadets had while on the trip.

Palm Harbor United Methodist Church, Palm Harbor FL. — *Tech Lead*

Oct. 2017 - May 2018

One of the leading technology and AV staff for the Sunday services and set up all other events put on by the organization. Learned many skills such as Media Shout, projector cameras, a media switch and video editing.

Lockheed Martin, Dunedin FL. — *Internship*

Aug. 2016 - May 2017

Responsibilities included: completing time studies, updating databases, observing managers and assemblers, report observations to team lead, complete saving packages, and assist the team in any way possible.

EDUCATION

Montana State University, Bozeman MT.

Currently Enrolled

Pursuing a Computer Science B.S. degree, as well as Air Force ROTC.

RELATED COURSES

Computer Security

Web Design

Social and Ethical issues in computing

Human Computer Interaction

CHARACTERISTICS

Driven

Eye for detail

Adaptable

Dependable

Team Player

CERTIFICATION/AWARDS

Adobe InDesign, Photoshop, Illustrator - Creative Cloud

Microsoft Office Suite, Certified Specialist - (Word, PowerPoint, and Excel)

Secret Security Clearance

Cadet of the Semester Fall 2019 - Professional Officer Course

HS Diploma Summa Cum Laude - Engineering and Business Academy 2017

SKILLS

Leadership

Communication

Photo and video editing/capture

HTML, CSS, Python, Java, and C, GoLang, Lisp

MARIA GALLIVAN

A student with a passion for Cyber Security

CONTACT

PHONE:
(206) 653-4665

EMAIL:
maria.gallivan@gmail.com

PROJECTS

Udemy CompTIA PenTest+ Seminar
04/2021 – present
Studying from Michael Solomon on ethical hacking and high-level penetration testing.
Working towards taking the PenTest+ certification by 4/2022.

National Cyber League Competition
02/2021 – 04/2021
Competed in the NCL competition both individually and on a team solving security problems in cryptography, log analysis, network scanning, open-source intelligence, and web app exploitation.
Related Courses: Web Design
Databases
Cyber Security
Related Tools: Wireshark
Bash Commands
VMware

Networked Battleship Game
09/2019 – 05/2020
Created a network-based game focused on understanding a client/server architecture.
Related courses: Networks
Computer Systems
Languages: Python, C++, Assembly

Hour of Code Program
09/2018 – 03/2019
12/2019
Helping 20-30 students through a Code.org Hour of Code program.
Coordinating and recruiting volunteers.

Multi-Discipline Engineering Project
09/2019 – 12/2019
Built an app controller for an RC car from scratch while collaborating with a team of Mechanical and Electrical Engineers.
Related courses: Technical Writing
Software Engineering
Language: Dart

EDUCATION

Montana State University - Bozeman, MT | 2018 - 2022
Bachelor of Science: Computer Science | 3.92
Bachelor of Science: Applied Mathematics | 3.56

Awards and Honors: President's List (MSU 1x), Dean's List (MSU 3x), Washington State Honors Award (2018), President's Award for Outstanding Educational Excellence – Gold Award (2018)
Unsung Hero (MSU Athletics Department – 2021)

WORK EXPERIENCE

IT Support Technician | 05/2021 – 08/2021
Montana State University Facility Services – Bozeman, MT
Managed department DNS and database for hardware records.
Configured and onboarded desktops, laptops, and iPads in a department of 100+ workers.

Cyber Systems Engineer Intern for Aerospace | 5/2020 – 7/2020
Northrop Grumman – Azusa, CA
Ensured cyber requirements are met by each hardware and software group on the current Northrop Space project.
Performed code reviews (C++), static code analysis, system modelling, trade studies and education trainings.
Presented my findings to managers and team members while participating in an Agile work environment.

Undergraduate Course Assistant | 8/2019 – 5/2020
Montana State University – Bozeman, MT
Supported 25+ undergraduates learning to code in Java, C, and Python through weekly labs and study sessions.

LEADERSHIP EXPERIENCE AT MONTANA STATE

Hacker Cats Club – President | 04/2021 – present
Plan, implement, and lead a regular schedule of speakers, meetings, and presentations to invite discussion and deeper learning of cyber security. Facilitated collaboration on NCL and Over the Wire activities.

Association of Women in Computing - President | 12/2019 – present
Plan meetings, and events to promoting a safe, supportive space that advocates for diversity.
Build a strong community and manage a team of students to make ideas and opportunities come to life.
Through the COVID semesters, continued running the club virtually.

College of Engineering Ambassador | 08/2019 – present
Inspire prospective students to explore engineering majors through tours and presentations.
Build relationships with members of all disciplines to better relay their information, activities, and goals.

Spirit Squad Cheerleader | 09/2018 – present
Work 20+ hours a week as a student athlete at practice, games, and community appearance while successfully balancing school.

ZOE NORDEN

CONTACT

970 - 692 - 3299

znorden17@outlook.com

https://github.com/znorden17

EDUCATION

Computer Science, BS
Montana State University
Bozeman, MT | May 2022

China Studies, Minor
Montana State University
Bozeman, MT | May 2022

LANGUAGES

C/C++, Python, Java, HTML/CSS, Javascript

SKILLS

Adobe Creative Cloud Suite, Linux systems, conversational Mandarin

PROJECTS

Seed Labs
Jan. 2021 - May 2021
Successfully demonstrated how vulnerabilities were exploited to gain root access to a system.

Home Security System
Oct. 2021 - PRESENT
Developed a mobile application that receives alerts from a server everytime a door is opened/closed.

PROFILE

My background involves living and traveling the world for most of my high school career and being able to experience several different unique cultures. These experiences have helped me to develop interpersonal and problem-solving skills, a high level of independence, good time management, flexibility, and the ability to work with people of different backgrounds.

EXPERIENCE

IT Intern
Lumibird | Bozeman, MT | Nov. 2021 - PRESENT
- Assist in getting this branch of Lumibird NIST SP 800-171 certified
- Assist 100+ workers with day-to-day technology problems

Teaching Assistant at Montana State University
Montana State University | Bozeman, MT | Jan. 2021 - PRESENT
- Communicated C to individuals with some computer science experience
- Communicated Python to individuals with no computer science experience.

Cyber Security Intern
Zoot INC. | Bozeman, MT | Dec. 2020
- Researched phishing educational and tracking programs

Computer Science Help Center Volunteer
Montana State University | Bozeman, MT | Jan. 2021 - PRESENT
- Communicated many different programming languages and concepts to many individuals of a variety of different backgrounds.

AWARDS

Montana State University Dean's List
Montana State University | Bozeman, MT | Fall 2020, Spring 2021

Montana State University Grace Hopper Sponsorship Award
Montana State University | Bozeman, MT | 2019, 2021

Connor Lowe

Bozeman, MT 59714 | 406.370.4348

con.lowe406@gmail.com | github.com/conlowe009

EDUCATION

Bachelor of Science, Computer Science (Interdisciplinary Option) + Minor in Japan Studies

Montana State University, Bozeman, MT

Specific Skills: Java, Python, C, C++, Go, Dart, Kotlin, Delphi, Test-Driven Development, SQLite, HTML/CSS, Linux, Bash/Shell Programming, Systems Administration, Git & Subversion Version Control

Relevant Curriculum: Data Structures and Algorithms, Software Engineering, Multi-Disciplinary Engineering Design, Computer Systems, Concepts/Programming Languages, Discrete Structures, Multivariable Calculus, Data Mining, Operating Systems, Database Systems

ACADEMIC ACTIVITIES

- Montana State University Global Ambassador, 2017
- International Student, Nanzan University, Japan 2016
- Notetaker for Students with Disabilities, 2015-2016
- Montana State University Student Senate, 2015
- FIRST Robotic Competition, 2012-2014

ACHIEVEMENTS AND AWARDS

- Montana State University Honor Roll, 2014-2017
- Benjamin A. Gilman Scholarship Alumni, 2016
- American Association of Teachers of Japanese Grant Recipient, 2016

WORK EXPERIENCE

Delivery Driver and Manager

Domino's Pizza, Bozeman, MT — May 2016 - December 2020

- Pizza delivery in a fast-paced, dynamic environment. Lead the team of drivers during night shifts.

Software Engineering Intern

Teledyne Photon Machines, Bozeman, MT — July 2021 - Present

- Software Engineering Internship focusing on control software developed in-house for Laser Ablation Machines used in Inductively Coupled Plasma Mass Spectrometry Analysis.
- Assist in design, build, and validation of new product software and technologies.
- Generate documentation for new and existing product designs and technologies.
- Create software GUIs for the customer facing part of the software.
- Help other engineers with bug fixes and feature development.
- Work with cross functional teams including sales, marketing, and production.

References

[1] “Cybersecurity: A Global Priority and Career Opportunity.” *University of North Georgia*, 2021, https://ung.edu/continuing-education/news-and-media/cybersecurity.php

[2] Rice, David Mark. “An Extensible, Hierarchical Architecture for Analysis of Software Quality Assurance.” *Montana State University*, Dr. Clem Izurieta, 2020, pp. 1–156. https://www.cs.montana.edu/izurieta/thesis/Rice.pdf

[3] Github for PIQUE-Lite app, https://github.com/MSUSEL/PIQUE-Lite

[4] “Facade Pattern.” *Wikipedia*, Wikimedia Foundation, 26 Dec. 2020, https://en.wikipedia.org/wiki/Facade_pattern.

[5] Github for Draw.io Integration, https://github.com/jgraph/drawio-integration

[6] “ISO 25000 Portal.” *iso25000.Com*, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010.

[7] Trello: https://trello.com/invite/b/VwP5jW1o/42edf5121778a5e430416d3a2d890da9/guidatabase

[8] Github Repository: https://github.com/MGallivan00/Capstone_22

Code References

Google. “Add Firebase to Your JavaScript Project | Firebase Documentation.” *Google*, Google, https://firebase.google.com/docs/web/setup#available-libraries.

Song, Zheng. “A Customisable and Optimised React Menu Library with Accessibility.” *React.js Examples*, React.js Examples, 12 Oct. 2020, https://reactjsexample.com/a-customisable-and-optimised-react-menu-library-with-accessibility.

Eliav2. “React-Xarrows/Examples at Master · Eliav2/React-Xarrows.” GitHub, https://github.com/Eliav2/react-xarrows/tree/master/examples.

Foundation, Mozilla. “JavaScript Documentation Reference.” *DevDocs*, MDN Contributors, https://devdocs.io/javascript/.