Spatial-App Proposal / Outline

Description

The full-stack spatial web app (referred to as the Spatial Web App) will be the next iteration of the current Excel based spatial separation table used internally by GHL tech staff. The purpose of this app will be to provide an improved user experience compared to the existing spatial separation table, as well as provide a platform for future expansion into a deployable software as a service (SaaS) product.

The current Excel based spatial table does not have create, read, update and delete (CRUD) functionality, and is not a user intuitive design. Users currently need to have a high level of technical knowledge to use the spatial table. The table is not easy to use for those without sound knowledge of Excel, in addition to extensive training on how to use the spreadsheet. Inserting a new compartment calculation is not possible without unhiding rows and shifting down cells, making organizing the table difficult. Deleting a row removes a 'link' to the interpolations table, which is not ideal. Excel in general is decent for basic math calculations, but is not an ideal platform for dynamic add/delete operations that require CRUD operations.

Features

The Spatial Web App will have full CRUD functionality providing users with the ability to add, remove, and update compartment calculations with ease as a web application platform.

The app will be browser based and built using the MERN stack (MongoDB, Express, React, Node.js). The app will be built with a web first approach, and will incorporate responsive web design to accomodate different screen sizes including mobile.

For the internal-use version of the Spatial Web App, the app can be hosted on a private server and accessed by GHL staff only via a web browser.

Users will import and save compartment calculations to the app by opening a .csv file, and will be able to view, edit, and delete calculations. A print feature will output the table of compartment calculations as a PDF, similar to how Excel exports a spreadsheet in PDF format. The app will have a straightforward UI that will be intuitive to use, and will be designed for those with little to no technical knowledge.

Design

Frontend

The frontend will be built using React and TypeScript, and will be a single page application (SPA). The frontend will be built with a modular design, allowing for future expansion into a SaaS product. The frontend will be built with a web first approach, and will incorporate responsive web design to accomodate different screen sizes including mobile.

Algorithm

The method used to calculate spatial separation will be the same as the current Excel based spatial table by looking up the applicable Building Code tables and bi-linearly interpolating values. The JavaScript code developed two summers ago and shared with JDM will serve as a base. This code will be rewritten in TypeScript for best design practices, and modified to work with the MERN stack.

Database and Storage

It is understood that GHL follows OQM file management standards, and the current spatial table is stored in file explorer on the projects drive on a per project basis. To continue meeting these requirements, the database for the internal version of the app will remain as the current file explorer structure with a .csv file for each project. The .csv file will be read into the app and be stored in memory as a JSON object. The JSON object will be used to populate the data within the app, and will be written back to the .csv file when the user saves the table.

For the future SaaS version of the app, the database will be migrated to a cloud based database service such as MongoDB Atlas.

Testing

Testing is an important part of the development process, especially to minimize maintenance after the summer. Unit testing will be performed using Jest or a similar unit testing framework. The testing framework will be used to test the algorithm by comparing to the results of the current Excel based spatial table.

It is anticipated that the frontend UI will be simple enough such that manual testing will suffice. However, the testing framework can be used to test the frontend components with different inputs and ensuring the correct outputs are produced.

Integration testing will be performed to ensure the frontend and backend are working together as expected. End to end testing will be performed to ensure the app is working as expected from a user perspective.

Technologies Used

The tech stack will primarily consist of the following technologies:

- React
- mongoDB
- Node.js
- Express

Note that not all of the technologies used above will be used in the initial stages of designing the internal-use version of the app. The technologies used will be expanded upon as the app is developed.

Timeline

I have high confidence that a working internal-use version of the app can be developed and deployed by the end of the summer, and that the architecture for an externally deployed app can be prepared for future progress and development. The following is a rough timeline of the work that will be completed:

Weeks	Description of Work
1 - 4	Setup development environment, create React app, UI Figma sketches, create basic frontend with filereader to read .csv file into app, review and refine existing JavaScript code, begin writing new code in TypeScript
4 - 8	Create basic CRUD functionality, build out basic UI from Figma, integrate algorithm to calculate spatial separation into React App, build basic print functionality,begin building backend services
8 - 12	Create import and export to .csv functionality, create basic print functionality, set up cloud database to store calculations, write API endpoints, connect frontend and backend with Axios
12 - 16	Setup user authentication for SaaS version, advanced testing, write documentation, internal deployment, end to end testing, write documentation on next steps for SaaS version

Progress updates to the principal(s) will be provided as frequently as requested.

Time Spent

An anticipated allowance of 30 hours per week will be needed on this project over the course of the summer. This includes time spent on design, development, testing, documentation, and progress updates as described above.