CSM-ODC
Project Management

June 15 2021

Jacob Tanchak, Michael Hessel, Eric Tyskiewicz, Petal Ladenson

Introduction

Our client was the Office of Construction and Design at Colorado School of Mines. The technical issue that we were tasked with solving was designing a software that could manage projects in a way that is convenient to the client. The current software that the client has is intended for large schools and businesses and therefore has many unnecessary details for their own use. The client typically stores all of their projects in excel spreadsheets, but the software that they use has no way of communicating with these spreadsheets so they must manually re-enter information that they had already entered before. Additionally, the software has no working method for the client to search through projects by certain fields such as budget or project manager. The software that we set out to create is intended to allow the client to upload the project spreadsheets, whereupon any important information will be extracted and stored in a database. This way, we could create options on a web server to allow the client to sort their projects by desired fields and pull reports. This would also mean that the client would no longer have to manually re-enter all of the project details multiple times for documentation purposes, as our software would hold important information and all of the details would still be within the spreadsheets on their computers, so the new software could be used to gain a broad overview and also point towards a spreadsheet that goes into further detail. Our software is a website that communicates with a Java backend, that in turn communicates with a database for object storage. The frontend was developed in Angular, and exists solely to fulfill the purpose of a user interface; any information the frontend receives is instantly passed along to the backend for processing. For example, a file uploaded to the frontend is passed along to the Java application in order to be parsed for data. The Java application then stores that information in the database, so that when the user requests information from the frontend, it can be retrieved and sent by the Java backend.

Requirements

The requirements are to build an application to search through key features of construction project spreadsheets and return summaries of all projects that meet given parameters. In addition, they are to build a framework for a more comprehensive project management system.

A list of the functional requirements-

- Contain and organize projects with several attributes
- Project information can be updated at any time
- Varying levels of access to project details
- Projects can be searched for and selected by certain fields
- Show that spending is approved and who approved it
- Update building list

A list of the non-functional requirements-

- Keep the groundwork open for future projects/functionality
- Simple interface
- Interface with Mines security
- Documentation and concise use instructions

System Architecture

The application consists of three key components: the frontend, the backend, and the database. These parts all interact as lined out by Figure 1 below:

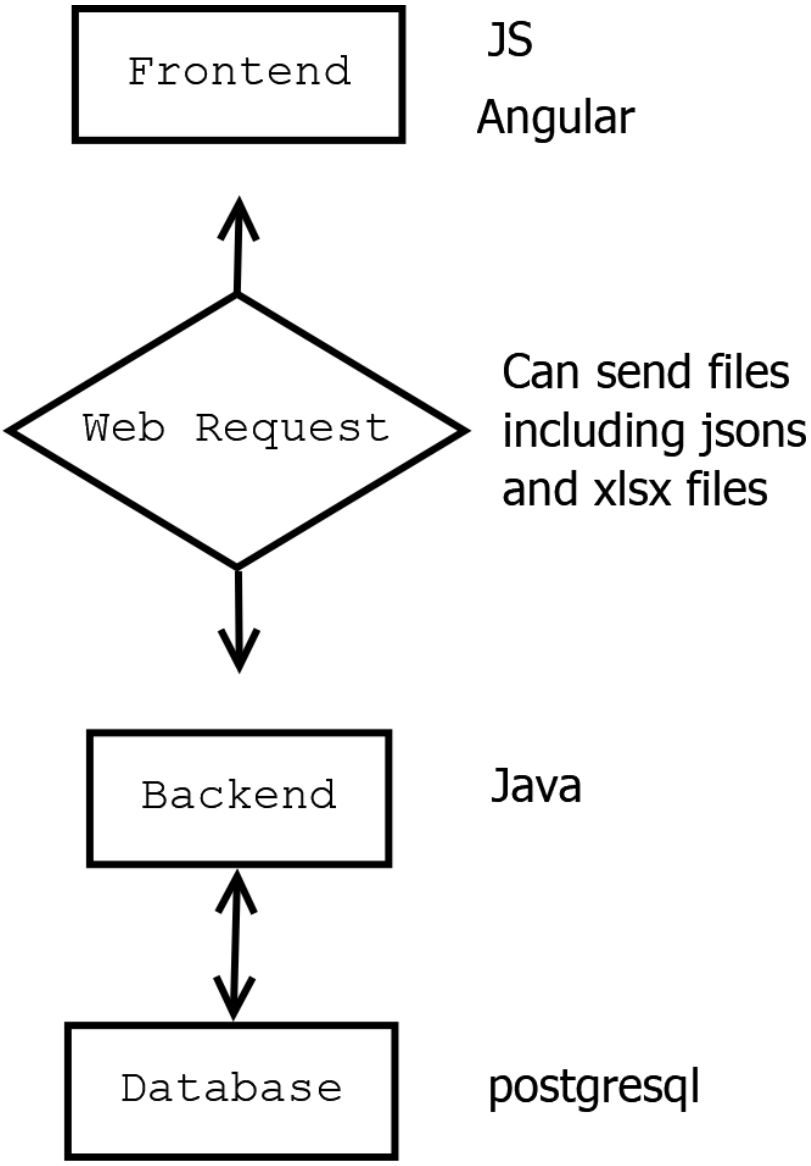

Figure 1

The frontend was developed in an Angular environment, which is just a more user friendly method of using HTML, TypeScript, and CSS. The purpose of the frontend is to create a user interface that is easy to use, and communicates with the backend for all major operations. The frontend has no complex operations that it should be doing, it only exists as a go between for the user and the backend. The frontend's simplicity is reflected in it's simple home page as seen in Figure 2 below. The fronted consists of four major components:

- An upload page, where excel spreadsheets can be uploaded and have their information stored
- A project searching page, where the user can select details to sort and search projects by and, upon receiving the results, select specific projects to see more details
- A report generator page that allows users to choose which projects to put in the report and which attributes of the projects should be shown; this component is only partially implemented and will need to be finished by future teams
- A resources page with information on the use of the website

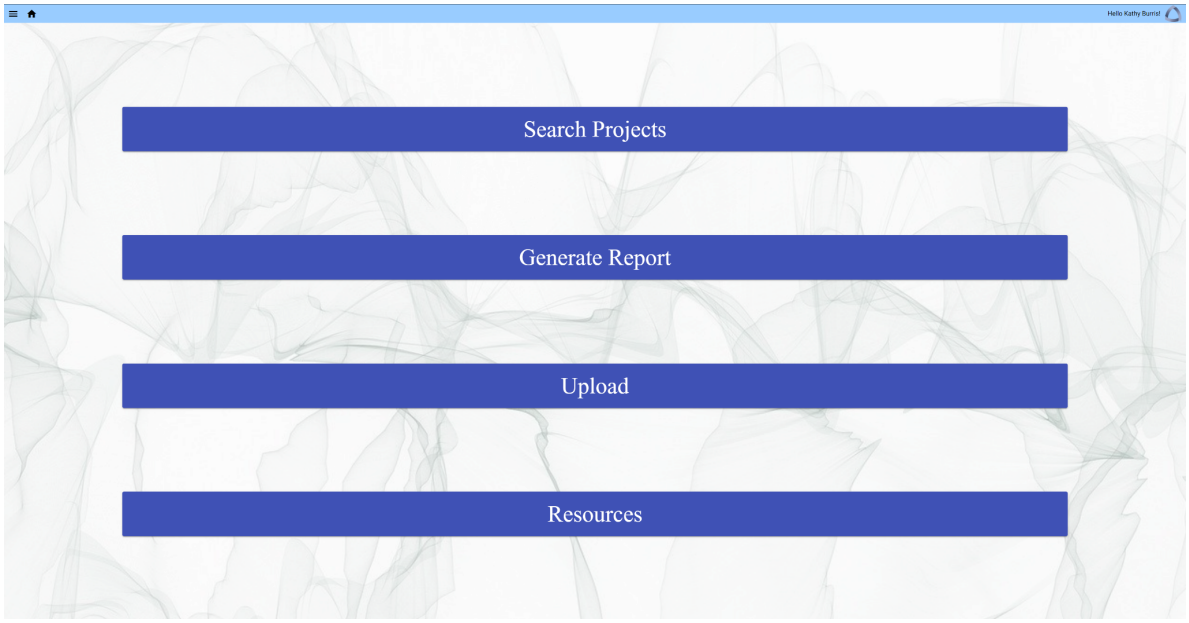

Figure 2

The backend was developed as a Java/Spring Boot application. The backend's purpose is to facilitate all interactions with the database that is needed in a way that the frontend can utilize. The backend's main way of interacting with the frontend is through endpoints on the Spring Boot side that are found through specific fetch requests. These endpoints take in and return a body (JSON decomposition of specific objects on each side) in order to accomplish their tasks. Examples of some of these endpoints are "/upload" (body in: multipart file, no body out) and "/getProjectBy" (body in: searchParams object, body out: list of Project objects). In order to interact with the database we have a list of functions that all query the database through postgresql calls. An example of one of these functions is the function getProject that takes in a project number and returns the project that corresponds to that number in the database by finding it through a SELECT statement.

The database is a postgresql database that contains eight tables. These tables are organized into two groups: one for permissions and the other for projects. The permissions group contains four tables: three specific tables and a cross reference table, while the project group contains four tables for each part of a project spreadsheet.

Technical Design

The database schema contains two groups of tables (Figures 3 and 4), each having their own ERD. The first one here is the project diagram.

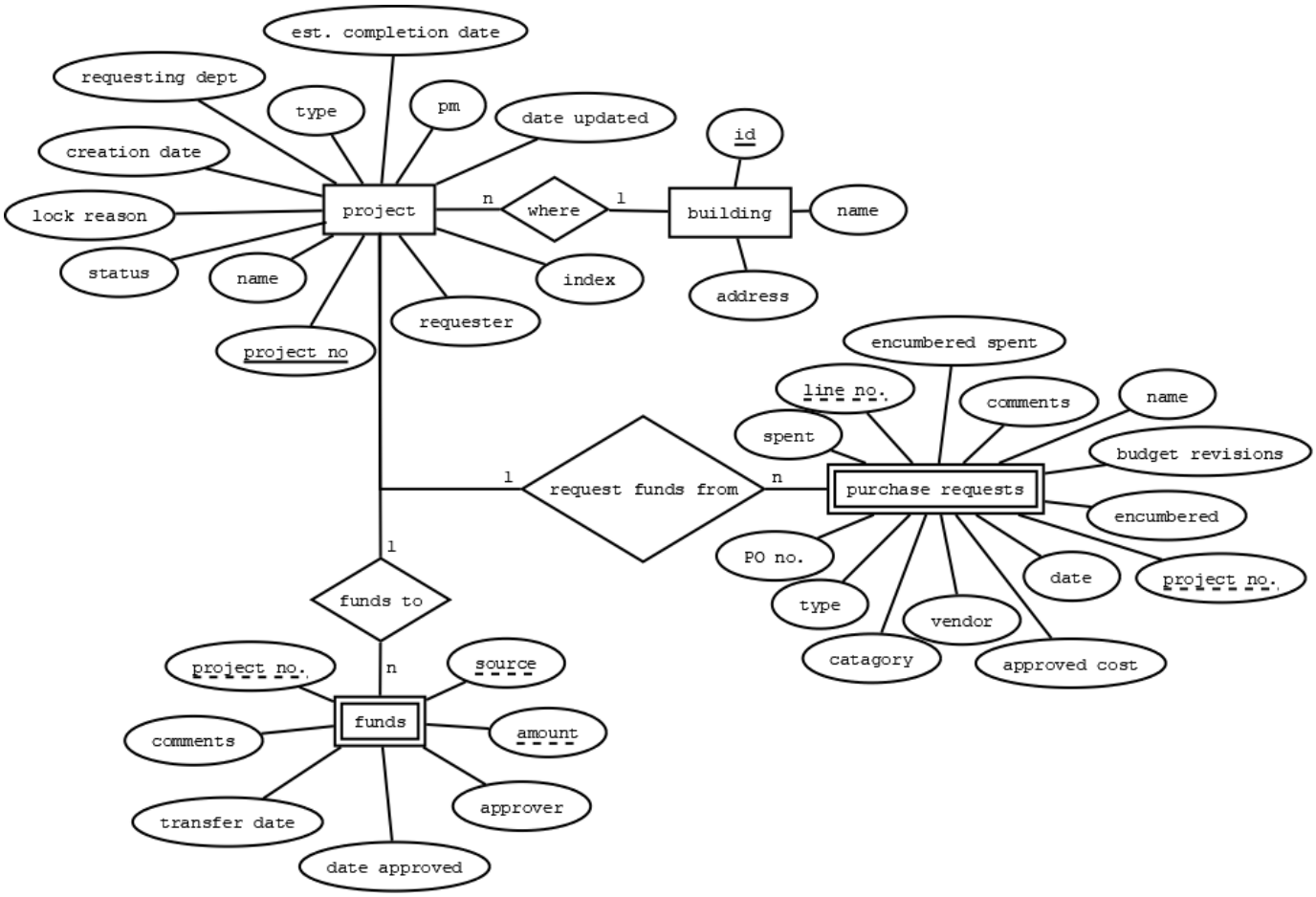

Figure 3

The project side, as seen in Figure 3, consists of four tables: purchase requests, funds, project, and building. The simplest table, building, is a list of all buildings that includes their id, name, and address. The next table, project, is the list of all individual projects keyed by the project number. This table includes all the information that is included once for each project such as name, creation date, project manager, etc. One thing in this list only included once is building id, the foriegn key to building, because each project only belongs to a specific building. The next two tables, funds and purchase requests, are each something that a project contains multiple of hence the many-to-one relationship. Each of these is a row in the spreadsheet so for every column of information these have, we store them in the database.

The permissions group, as seen in Figure 4, is the collection of tables in the database which contains the information of who is allowed to do what. To do this, we need three main tables: user, perm_group and perm_list. In this structure there are several perm groups that a user can belong to, each group then has a collection of perm lists connected to it by a cross reference table which contains the ids for both the group and list as a connection between the groups. And then perm_list is an identifier that a list of functions need to see in order to run without error. So whenever a user wants to run a function, before we run it we can check to see if the user's group contains the list that that function needs it to have in order for it to run and then either run the function or throw an error.

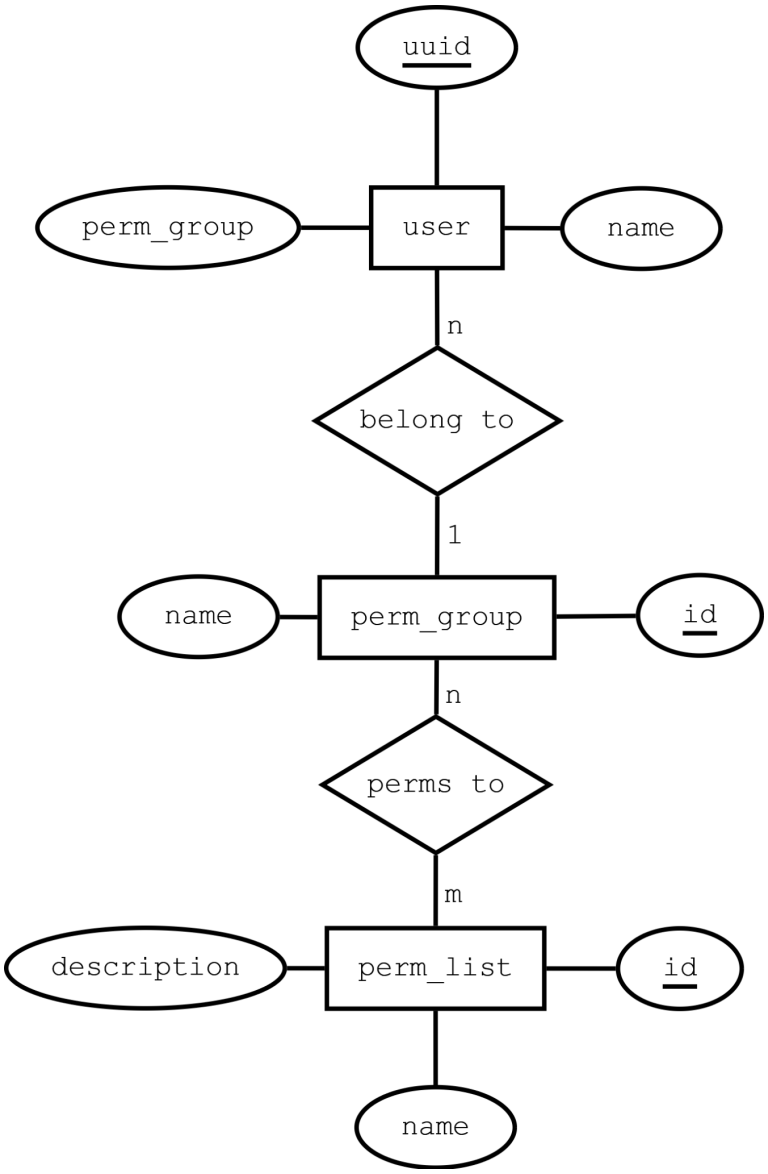

Figure 4

QA - UPDATE

Unit testing

- We wrote tests to make sure that each function that the frontend can call (and all the functions below those with non-trivial features) to test edge cases for them
- This makes sure that each part of the backend functions as expected

User interface testing (automated or otherwise)

- We manually tested every button of the frontend to make sure that everything is working as expected on the frontend
- This makes sure that the UI actually functions as expected

Integration testing

- We manually tested every feature of the frontend to make sure that everything is returning as expected
- This makes sure that the frontend and the backend are communicating
- We wrote tests to ensure that the backend communicates with the database in all the ways we want (this will most likely be an extension of the unit tests)
- This ensures that the backend is communicating with the database as expected

User acceptance testing

- Run through the frontend and its functions through our client
- This makes sure that our client has something that is they understand how to use and are satisfied with

Results

Our Software is divided into a frontend, backend, and database. All three of these have the minimum functionalities to be considered complete, but only the frontend is uploaded on our web server. We also did not have time to implement user authentication via multipass as well as create the report generator component. After testing by uploading an example spreadsheet to our project, the locally hosted project was able to extract all of the project details that we desired and store them in our database. In the future, we have advised our client to employ another group to add the authentication and report generation, as well as uploading the complete project to the server.

Lessons Learned:

- Deployment is half the battle
- “One month” is less time in practice than it appears to be when planning
- The amount of time you have to work is not always best used by constantly working, breaks are important
- Do not ignore any resources you are offered, you never know what might get you past a hurdle
- When learning to do new things, it will often take multiple attempts of several different methods
- Documentation is important and takes time
- It takes time to understand exactly what the client wants/The client may always be right but they may not always be informed
- Often when trying to solve a programming problem it is more important to know what to google than where to begin

Future Work

In the future, it is likely that any team working to improve our project will add the report generation functionality, as this is something that we did not have time to complete. The component for this functionality is already included in the Angular project, and the methods to communicate with the backend are there as well so this is most likely the easiest feature to add. Additionally, a future team might integrate the multipass functionality that we had wanted to include. We have already communicated with Mines ITS and they have already set up the functionality on their end, all that remains to do is have the future team actually integrate the project's interaction with it. The biggest improvement to make is actually uploading the backend and database to the web server so that the already uploaded frontend can interface with them on the website rather than just on a local machine.

Appendix

For future development of the product it is important to have Angular CLI, nodejs, a Java 8 IDE, maven, and a postgres server. In addition, the Angular material library is used so that will have to be added. Given all of these, getting the project to run should be as simple as cloning the github repository. Make sure that the Java project is imported as an existing maven project, an option that Eclipse has built in.

To view the web server of the project, make sure you are on the Mines network and navigate to https://odc-tracker.mines.edu.

The network is very difficult to navigate if you lack experience, for uploading the backend and database a lot of research may be required. For implementing the multipass functionality, it would be wise to consult with Christopher Painter Wakefield, as he is the most knowledgeable and available on that subject. Be sure to plan accordingly as this component may take quite a bit of time.