Underground Cable Packing Web Tool

Final Report and Design Document

Iowa State University CPRE/CYBE/EE/SE 491
Fall 2021 - Spring 2022
Team Number 19

Professor Mathew Wymore - Faculty Advisor
Jacob Conn - Teaching Assistant
ISU's Electrical Power Research Center - Client
Alliant Energy - Client

Team Member - Project Leadership Role
Alexander Young - DevOps and System Engineer
Brevin Wapp - Scrum Master
Haadi Majeed - Quality Assurance Engineer
Matthew Hoskins - Team Lead
Nate Tucker - Tech. Lead
Tom Sun - User Experience and Requirements
Quinten Sorice - Client Point of Contact

Team Email: sdmay22-19@iastate.edu
Team Website: http://sdmay22-19.sd.ece.iastate.edu/

Revised 12/05/2021
Version: 1.0.0

Executive Summary

Development Standards and Practices

During this project, many standards and practices were considered to best suit the development of the project, and some of the most important and planned upon standards to follow are included here. For more information on the development standards and practices that this team's project group will be following can be found in this document's *1.3 Engineering Standards* section.

For broad standards of professionalism and ethics, that were considered during the design phase of this project, include: the National Society of Professional Engineers (NSPE) ethics, the Software Engineering (SE) code of ethics, the IInstitute of Electrical and Electronics Engineers (IEEE) code of ethics, and the Association for Computing Machinery (ACM) code of ethics. The decision to primarily follow the SE code of ethics was made from the consideration that this project will be a software development project. For more information on these standards and ethics see section *5. Professionalism* of this document.

The specific standards that this project looked into and will be following during design processes are listed below with a brief description and expanded upon in section *1.3 Engineering Standards*.

- IEEE 1016: create and maintain software design documentations
- IEEE 830: create a Software Requirements Specification (SRS)
- ISO/IEC/IEEE 29119: software tests defined, operated, and documented properly

For each of these standards the process that this project group will follow can be seen in various sections of this document. IEEE 1016 focuses on design documentation which would be the whole of this document. IEEE 830 would be the section regarding software requirements, *1.2 Requirements and Constraints*. ISO/IEC/IEEE 29119 process can be found in this document's testing section *4. Testing*.

Summary of Requirements

Provided here is a brief summary of this project's requirements, for more information on these requirements see section *1.2 Requirements and Constraints*. The requirements listed here are the broad perspective and most significant of the requirements identified by the team and advisors.

- Final product must be a web application
    - Run on mobile and common browsers
- Must run on ISU server
- Output is a graphical visualization in a portable format
- User input of cables/ducts
- UI must contain ERPC branding

Applicable Courses from University Curriculum

Iowa State University courses that contain content applicable to this project would include:

- CPRE 185: Introduction to Problem Solving I
- SPCM 212: Fundamentals of Public Speaking
- COMS 227: Object-Oriented Design
- COMS 228: Data Structures
- CPRE 230: Cyber Security Fundamentals
- CPRE 231: Cyber Security Concepts and Tools
- COMS 252: Linux Operating System Essentials
- COMS 309: Software Development Practices
- CPRE 310: Theoretical Foundations of Computer Engineering
- COMS 311: Introduction to Algorithm Design and Efficiency
- ENGL 314: Reporting, Documenting, and Technical Communication
- SE 317: Introduction to Software Testing
- SE 319: Construction of User Interfaces
- SE 329: Software Project Management
- SE 339: Software Architecture Design
- SE 409: Software Requirements Engineering
- SE 417: Software Testing
- CPRE 421: Software Analysis and Verification for Safety and Security

Knowledge Acquired

The curriculum up to this project has covered a large amount of what this project has and will require considering the focuses of this project include: a frontend connecting to a server backend with an algorithm as the primary component (COMS 309, COMS 311, SE 319, etc.), proper team project documentation (COMS 309, ENGL 314, SE 329, etc.), project planning (SE 329), and team member, advisor, client interaction, and others interaction on a professional level (SPCM 212, etc.). All of which has been covered by the curriculum to an extent.
Ultimately resulting in the knowledge being acquired by the team members on this project being more specifics. For example: programming in languages and frameworks that were not specifically covered by a class (Golang), high level code sharing and production management through GitLab, or proper conversion of existing algorithms to web application versions with a good level of efficiency.

Table of Contents

List of Figures, Tables, Symbols, and Definitions

List of Tables

List of Figures

0 Team Section

0.1 List of Members and Roles

Team Member	Leadership Role
Alexander Young	DevOps and System Engineer
Brevin Wapp	Scrum Master
Haadi Majeed	Quality Assurance Engineer
Matthew Hoskins	Team Lead
Nate Tucker	Tech. Lead
Tom Sun	User Experience and Requirements
Quinten Sorice	Client Point of Contact

Table 1: Team Member and Role List

0.2 Required Skill Sets for Project

This section outlines a basic set of skills that would be required by the team in order to complete the set requirements for this project.

1. **Formal project documentation** creation and upkeep
2. **Project communication** between fellow team members, advisors, clients, and other roles
3. **Project planning** for timing, work division, and resources
4. **GitLab code sharing and protocol** allowing smooth development of the project
5. **Functional project design** process enabling quality software development
6. **Algorithm analysis and testing** ensuring quick and valid results
7. **UI/UX development** that will create appealing and easy-to-use web applications
8. **Software testing** encompassing unit, interface, security, integration, system, regression and user acceptance testing
9. **React frontend** communication to Golang backend on server
10. **Golang Backend** on server communication to React frontend
11. **Visual output creation** from mathematical results
12. **Secure development** for web application; preventing unauthorized access of data
13. **Database creation and management** of potential user input options
14. **Software documentation** creation that effectively communicates the software functionality
15. **Server-side setup and upkeep** for hosting application during development

0.3 Skill Set Covered by Team

Included, separated by major, is a summary of the skills that each team member has that is applicable to the development of this project (skills are only listed in the event that it matches a required skill set for the project listed in section *0.2 Required Skill Sets for Project*).

0.3.1 Computer Engineering Majors

Alexander Young: 1. Formal Project Documentation, 2. Project Communication, 3. Project Planning, 4. GitLab Code Sharing and Protocol, 5. Functional Project Design, 7. UI/UX Development, 8. Software Testing, 9. React Frontend, 10. Golang Backend, 11. Visual Output Creation, 12. Secure Development, 13. Database Creation and Management, 14. Software Documentation, 15. Server-Side Setup and Upkeep

Haadi Majeed: 1. Formal Project Documentation, 2. Project Communication, 3. Project Planning, 4. GitLab Code Sharing and Protocol, 5. Functional Project Design, 6. Algorithm Analysis and Testing, 7. UI/UX Development, 8. Software Testing, 9. React Frontend, 12. Secure Development, 13. Database Creation and Management, 14. Software Documentation, 15. Server-Side Setup and Upkeep

Tom Sun: 1. Formal Project Documentation, 2. Project Communication, 3. Project Planning, 4. GitLab Code Sharing and Protocol, 5. Functional Project Design, 7. UI/UX Development, 8. Software Testing, 12. Secure Development, 13. Database Creation and Management, 14. Software Documentation, 15. Server-Side Setup and Upkeep

0.3.2 Software Engineering Majors

Brevin Wapp: 1. Formal Project Documentation, 2. Project Communication, 3. Project Planning, 4. GitLab Code Sharing and Protocol, 5. Functional Project Design, 7. UI/UX Development, 8. Software Testing, 9. React Frontend, 11. Visual Output Creation, 14. Software Documentation

Matthew Hoskins: 1. Formal Project Documentation, 2. Project Communication, 3. Project Planning, 4. GitLab Code Sharing and Protocol, 5. Functional Project Design, 6. Algorithm Analysis and Testing, 7. UI/UX Development, 8. Software Testing, 14. Software Documentation

Nate Tucker: 4. GitLab Code Sharing and Protocol, 5. Functional Project Design, 6. Algorithm Analysis and Testing, 7. UI/UX Development, 8. Software Testing, 9. React Frontend, 10. Golang Backend, 11. Visual Output Creation, 12. Secure Development, 13. Database Creation and Management, 14. Software Documentation, 15. Server-Side Setup and Upkeep

Quinten Sorice: 1. Formal Project Documentation, 2. Project Communication, 3. Project Planning, 4. GitLab Code Sharing and Protocol, 6. Algorithm Analysis and Testing, 8. Software Testing, 10. Golang Backend, 14. Software Documentation, 15. Server-Side Setup and Upkeep

1 Requirements Section

1.1 Problem Statement

As many companies with the need to distribute cabling, on a large scale, are shifting to more underground cabling, in order to better withstand environmental disasters as well as other potentially damaging occurrences, the need for software to assist in these actions have become even more necessary.

To that end, Iowa State University's Electrical Power Research Center (EPRC) and Professor Mathew Wymore have requested an expanded web tool version of an existing executable program with the addition of new features and improved primary functionality such as: enhanced algorithm, mobile support, and ease of use. This web tool will also allow for more readily available functionality being an application available on EPRC's website for immediate use.

The tool's primary use comes from streamlining the billing process for underground cabling companies and contractors with a known user being Alliant Energy. Other expected uses, from user interaction, include the assessment of proper bore sizing, project based material calculation, ease of optimal calculation, and ease of project specific utility file creation.

1.2 Requirements & Constraints

The following is a list of project requirements and constraints that have been identified by the team, the faculty advisor, and an industry client representative. These requirements are broken down into various categories with clear notation stating constraints. All of these requirements are taken into consideration for the design and implementation of this project.

Functional Requirements:

- Processing time targets, assuming a test case of one dozen cables/ducts or less
    - < 20 seconds if user must wait for results on page **(constraint)**
        - Stretch goal: < 5 seconds **(constraint)**
    - < 10 minutes if user can be emailed results asynchronously **(constraint)**
- Front end must run on common browsers, including desktop Chrome, Firefox, Safari and Edge (as time allows)
- Must be a web application
- Back end must run on target infrastructure (ISU’s ETG servers)
- Capable of sharing recent results via URL
- Output a graphical visualization of the final packing, and of the attempted packing in the outer diameter one size smaller, in a portable format (PDF, PNG, JPG, etc.)
- Correct results must be achieved, with correct defined as:

    - All given cables fit in resulting outer diameter
    - All given cables will NOT fit in outer diameter one size smaller
    - Stretch goal: formally prove algorithm is correct
- Configurable:
    - Set of predefined cables/ducts
    - Set of outer diameters to check
    - Unit of measurement (in or cm)
- Stretch goal: export results into downloadable spreadsheet that could have more information added to it
- Stretch goal: ability to plan out multiple sections of cable run at the same time
- Stretch goal: Setup admin accounts for companies and specific groups for the purpose of setting a group profile of tables and settings

Qualitative Aesthetics Requirements:

- The (User Interface) UI needs to use Iowa State University (ISU) Electric Power Research Center (EPRC) branding
- For the output - Cables should be packed to the center of the circle in this visualization

Resource Requirement:

- The use of Iowa State owned servers for hosting our application

UI Requirements:

- Clean and intuitive UI
- UI is functional and usable on mobile Chrome and Safari
- Stretch goal: improvements on visualization on invalid duct sizing

Software Requirements:

- Web stack must be well-known and documented technologies expected to be maintained for at least ten years **(constraint)**

1.3 Engineering Standards

The following are the engineering standards that the team has researched and implemented for the creation of this software development project. As there is not any hardware testing or maintenance to be implemented by the team, there are not any standards with the focus on hardware. The standards below are concerned with the software development, testing, and documenting of this project.

Create and maintain a software design description or design document as indicated by Institute of Electrical and Electronics Engineers (IEEE) standard IEEE 1016. This document will be used for recording design information, addressing various design concerns, and communicating that information to the design's stakeholders in the form of data design, architectural design, interface design, and procedural design.

Create a software requirements specification (SRS) set forth by IEEE 830. This will entail a document that lays out functional and nonfunctional requirements. The requirements document will be updated as needed as the project progresses, and the requirements evolve.

When making software based tests of any component of this project, following the standard set by Internal Organization for Standardization (ISO), International Electrotechnical Commision (IEC) in ISO/IEC/IEEE 29119 any tests will be properly defined, operated, and documented. This will entail creating sound tests that will be recorded in documentation.

1.4 Intended Users and Uses

Our primary users would be the users of the original cable packing application, our secondary users would be those involved in the process of packing and installing underground cable packages that have a use and access to this software, and anyone else that would be involved in or interested in the management of underground cable packing would be our tertiary users.

Alliant Energy and Iowa State University's Electric Power Research Center (EPRC) being the clients of the project; act as the primary users of the resulting software. Along with the secondary users of underground cable contractors, and anyone with access to the web tool that would want to use the product acting as the tertiary users of the resulting project.

Our application is designed to act as a standard for the cable packing industry to remove guesswork and create consistency. Our primary users would be able to reference our tool to calculate a consistent price based upon the cables involved that shows both sides how the calculation was made. Additionally, a by-product of showing the cable packing is that contractors would be able to determine the best fit for arranging the cables before they lay them down.

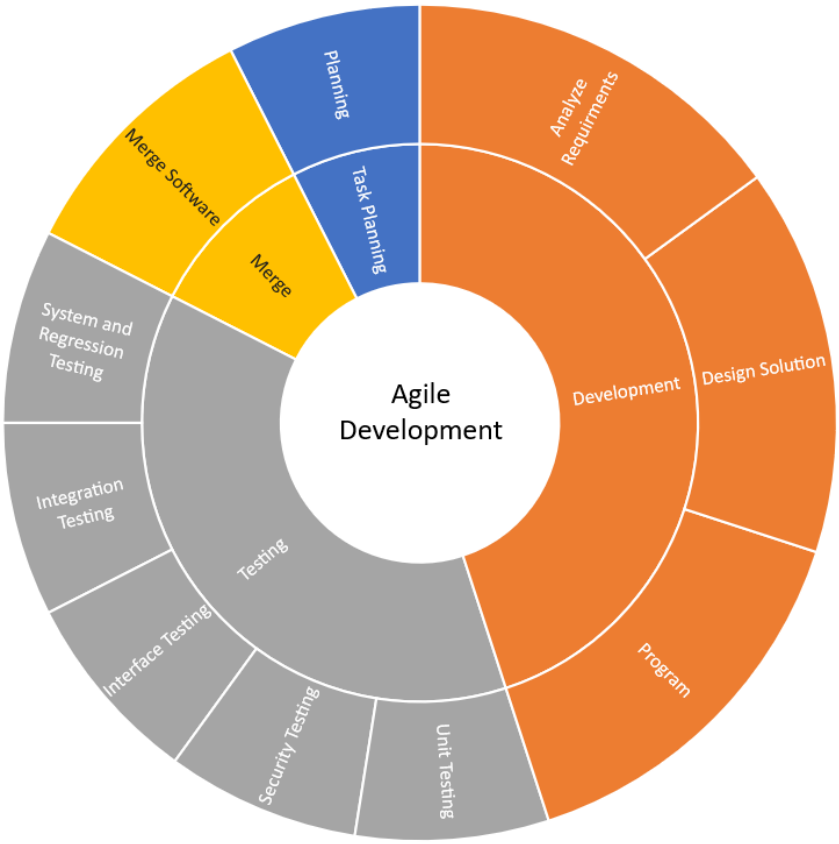
2 Project Plan

2.1 Project Management/Tracking Procedures

2.1.1 Management Style and Justification

Being completely a software development endeavour, this project's goal is to create a functioning and stable web tool. It is best suited to the agile management style because it will allow for stable and tested development that can have requirements fulfilled and improved upon without the risk of excessive planning or scope creep.

The project management style that the team has adopted is an agile project management flow style. This is due to a variety of reasons, including: our group's ability and knowledge to effectively and efficiently create software with this type of workflow, members ability to take lead in time management tasks toward development, and the general approach towards software development that involves iterable improvement.



See above (*Figure 1: Basic Agile Development Visualization*) is a basic visual representation of the development process in a clockwise order starting from "Planning" and ending at "Merge".

2.1.2 Progress Tracking and Management Tools

For this project, our team will make use of a number of tools and are all documented as follows.

Tools regarding produced elements: first, GitLab will be used for team collaboration in producing stable software, as well as document progress and tasks related to the software development of the project. Second, Google Drive will be used for other produced materials - primarily along the lines of documents and presentations.

Time scheduling and management devices: first, a Google Calendar will be used to plan out all meetings and due dates not related to code development. Second, a project Gantt chart will be created and utilized to properly manage all project deliverables and software based milestones.

As for communication methods: first, a structured Discord server will be used for immediate communication between team members, teaching assistant (TA) Jacob Conn, and Professor Mathew Wymore. Second, project group email (through school email) will be used for communication to other parties such as: Alliant Energy representatives, Iowa State University's (ISU's) Engineering Technology Support (ETS), ISU's Electronics and Technology Group (ETG), and ISU's Electric Power Research Center (EPRC). Lastly, a Webex meeting room will be established for any meetings that the project team will be hosting.

2.2 Task Decomposition

Basic Task Decomposition with numerous subtasks per broad realized task listed below with expected dependencies and clear numbering system for this project.

1. Project Planning & Defining
    - 1.1. Team dynamic planning
        - 1.1.1. Begin team communication
        - 1.1.2. Setup primary communication channels between team members and advisors (TA Jacob Conn, and Professor Mathew Wymore)
        - 1.1.3. Setup primary communication channels with all other connected parties (Alliant Energy, ETS, ETG, EPRC)
          [Dependent on 1.1.2]
        - 1.1.4. Assign leadership roles among team members
          [Dependent on 1.1.2]
    - 1.2. Develop Requirements
        - 1.2.1. Meet with professor Mat Wymore
        - 1.2.2. Meet with client: Alliant Energy
          [Dependent on 1.2.1]
        - 1.2.3. Develop requirements documentation
          [Dependent on 1.2.1, 1.2.2]

1.2.4. Determine engineering standards to follow in software development for this project
[Dependent on 1.2.3]

1.3. Project Plan Instantiation
1.3.1. Documentation setup
1.3.2. Document breakdown team meeting
[Dependent on 1.3.1]
1.3.3. Finalize original Project Plan document - living document for when the need for improvements, or alterations
[Dependent on 1.3.1, 1.3.2]

1.4. Software stack planning
1.4.1. Determine specific software stack
1.4.2. Get familiar with tech stack - Typescript React and Go languages
[Dependent on 1.4.1]
1.4.2.1. Perform individual practice to familiarize members with unique syntax
1.4.3. Create basic proof of concept for general functionality plans
1.4.3.1. Sending information from back to front for image generation

1.5. Project Merge and Pull Request (PR) Protocol
1.5.1. Team initialization of the preliminary process of getting new code accepted
[Dependent on 1.3.3]
1.5.2. Limits on number of team members approval for a single PR
[Dependent on 1.5.1]

1.6. Continuous Documentation Upkeep / Technical Writing
1.6.1. Requirements Document changes when goals change
[Dependent on 1.2]
1.6.2. Project Plan and Gantt Chart updating
[Dependent on 1.3]
1.6.3. Software documentation as newly accepted PR's occur
[Dependent on 1.5]

2. DevOps & Tech Setup
[Dependent on 1.4]

2.1. Initialize project website

2.2. Set up CI/CD (Continuous Integration/Continuous Deployment) pipeline
2.2.1. Choosing the technologies that best integrate with our software.
2.2.2. Implementing the chosen technologies and verifying they will continue to work for the 10 rated years of project lifetime.
2.2.3. Testing the chosen technologies to ensure they deliver correct results.

2.3. Set up deployment environments
[Dependent on 2.2]
2.3.1. Testing the chosen technologies on the deployment servers to ensure that deployments go smoothly
2.3.2. Testing the technologies to ensure they will continue to operate in the deployment environments even after host and software updates.

2.4. Set up individual team member work environment
[Dependent on 1.4]

3. Software Design & Functional Design Verification
[Dependent on 1.2]
   3.1. Create User Interface (UI) Mock-Ups
   [Dependent on 1.2]
      3.1.1. Update and Improve UI Mock-Ups
   3.2. User Experience (UX) testing
   [Dependent on 3.1]
   3.3. Final design verification with clients and managing professor
   [Dependent on 3.2]
4. Redesign Algorithm
   4.1. Go through mathematical processes to verify effectiveness of current algorithm
   4.2. Redesign to work from the inside out of the duct
   [Dependent on 4.1]
   4.3. Convert to a compilable programming language for speed purposes
   [Dependent on 4.2]
   4.4. Prove mathematical algorithm - stretch goal
   [Dependent on 4.3]
5. Setup Data Tables
   5.1. Render
      5.1.1. Id
      5.1.2. List of Cables
      5.1.3. List of Cable positions
      5.1.4. Creation Date
   5.2. Cable
      5.2.1. Id
      5.2.2. Label
      5.2.3. Color? Arbitrary for display purposes
      5.2.4. Diameter
   5.3. Company Admin Account (Stretch Goal)
      5.3.1. Single account per company or specific group of users
      5.3.2. Allow updating of associated selection profile
      5.3.3. Profiles will be open to admin and non-admin users
      5.3.4. Creation of account design
6. Backend Construction
[Dependent on 2.]
   6.1. Implement HTTP requests
      6.1.1. Insert, delete, and read - no need for update
   6.2. Convert/manage algorithmic results to transferable format
   [Dependent on 4.3]
      6.2.1. Send format to frontend to be drawn
      6.2.2. Send results to specified email (if requested)
   6.3. Configure web server to load balance/distribute requests to different microservices

6.3.1. Choose web server, the choice will motivate a lot of API design choices
6.3.2. Install and enable on the server provided by ISU
[Dependent on 2.]
6.4. Admin accounts handling (Stretch Goal)
[Dependent on 5.3]
6.4.1. Security of every account
6.4.2. Update corresponding profile tables and settings
6.5. Calculate multiple sections of cable run at a given time (Stretch Goal)
6.6. Export results into a semi-formatted spreadsheet (Stretch Goal)

7. UI Construction
[Dependent on 3.]
7.1. Input desired duct and cable specifications that will be run through the algorithm
[Dependent on 6.2]
7.2. Receive backend results and convert to graph drawing/expected output
[Dependent on 6.2]
7.3. Include EPRC required branding
7.3.1. Communicate with Professor Mathew Wymore and EPRC about attaining necessary branding for the website
7.4. Selection of company or specific group profile settings and tables (Stretch Goal)
[Dependent on 6.4]
7.5. Input for calculating multiple sections of cable run at a given time (Stretch Goal)
[Dependent on 6.5]
7.6. UI improvement to visualizing invalid duct size (Stretch Goal)
7.6.1. Show “invalid” wires overlaid the “invalid” duct size
7.6.2. Visually alter color of “invalid” overlay
[Dependent on 7.6.1]

8. Development Testing Suite
8.1. Unit testing being a part of team PR protocol will occur with continuous software development
[Dependent on 1.5]
8.2. Interface testing of software for any UI in production
[Dependent on 3.]
8.3. Security Testing for applicable software after initial development
8.4. Integration testing of software produced at sprint finalization stages
[Dependent on 4.0, 6.0, 7.0]
8.5. System & Regression Testing with integrated software for each merge to main code branch
[Dependent on 8.4]

For more detailed information on testing tasks see document section 4.0 Testing

9. UAT & Deployment
9.1. Internal acceptance testing
[Dependent on 8.]
9.2. Demo and testing with clients
[Dependent on 9.1]

9.2.1. Demo with Professor Wymore
9.2.2. Demo with Alliant
9.3. Final production deployment
[Dependent on 9.3]

Table 2: Task Decomposition primary task enumeration and summary: 1.0 Project Planning and Defining, 2.0 DevOps and Technology Setup, 3.0 Software Design and Functional Design Verification, 4.0 Redesign Algorithm, 5.0 Data Tables Setup, 6.0 Backend Construction, 7.0 User Interface Construction, 8.0 Development Testing Suite, 9.0 User Acceptance Testing and Deployment.

Task Decomposition version number: 0.4.0

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestones

1. Protocols, Technologies, and Requirements have a team consensus.
2. Git is configured with CI/CD and individual work environments are set up.
3. Mockups are verified by client, professor, and TA.
4. Algorithm must produce the correct result within 20 seconds.
5. Frontend and backend can successfully communicate.
6. Application must pass all unit tests and produce expected results.
7. Application must be deployed on the Iowa State server.

Evaluation

For each task that will be represented as an issue in Git, they will be assigned an effort value of the expected amount of time required to complete each issue. In our Git Kanban style board, we can visually see how many issues for each milestone have been completed, and how many are left. This allows us to not only see how close we are to a milestone, but also to track individual progress.

2.4 Project Timeline/Schedule

The following is the teams' original Gantt chart. It includes: tasks, substasks, who each task is assigned to, the current tasks' progress, the start date, and the end date. The start date is the current recommended day in which the team or those assigned to the specific tasks should begin based on dependencies and due dates. Some tasks are given ample time to demonstrate their complexity, and expected time requirements.

There are a few tasks that are ongoing throughout most of the project. These tasks are continuous documentation (technical documents maintenance, and software documentation), and the continuous unit testing of code as software for both proof of concept and final project are created.

Each section on the Gantt chart is the major task derived from *2.2 Task Decomposition* with the various subtasks making up each different colored section. All known deliverables are therefore included in the chart in the form of tasks. Tasks that are associated are shown to be so in either being in the same major task section or through the excel gantt chart calculation of not having start days before ending dates of dependent tasks.

All date information is shown at the top of the excel sheet, which has all tasks on a single gantt chart, in relation to the project with a start date of August 30, 2021 (Week 1).

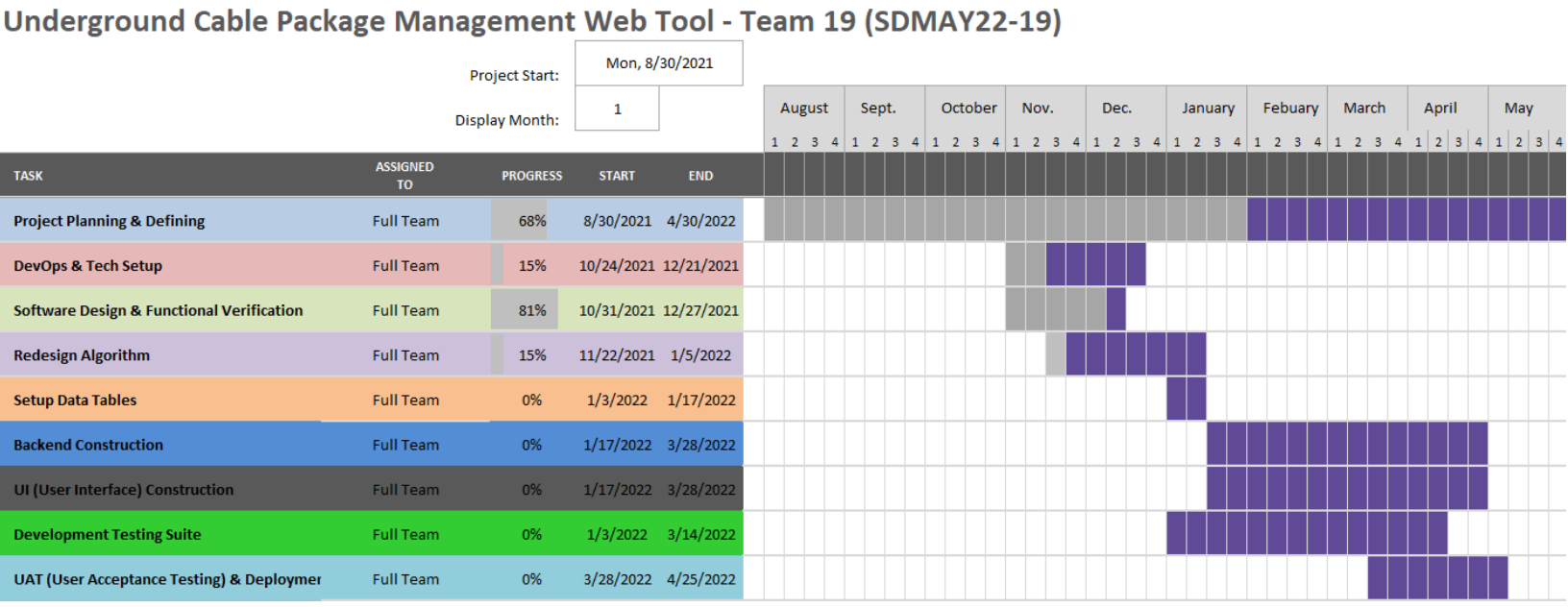
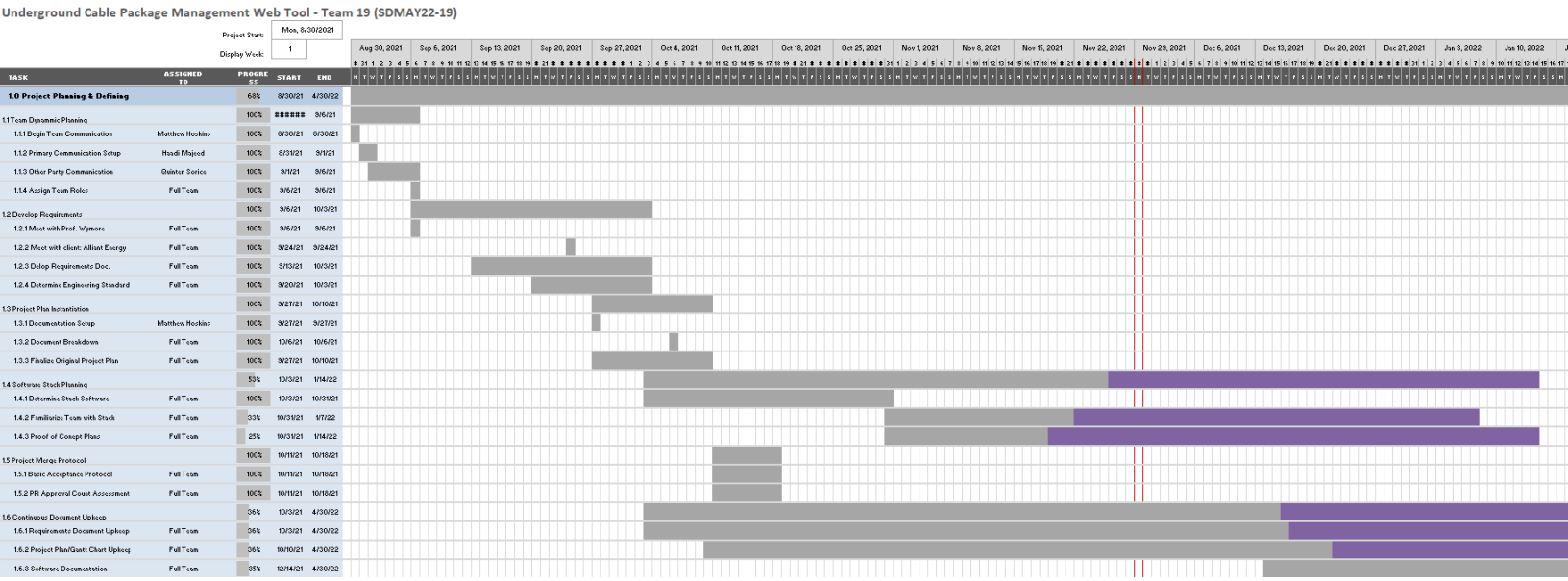
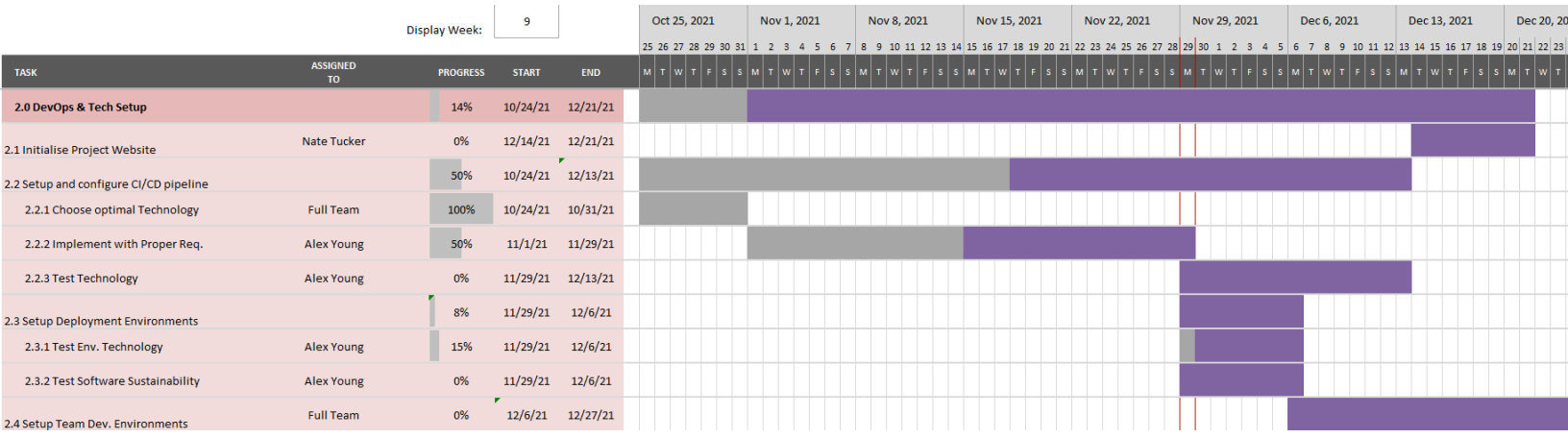


Figure 2: Gantt Chart by Major Task View

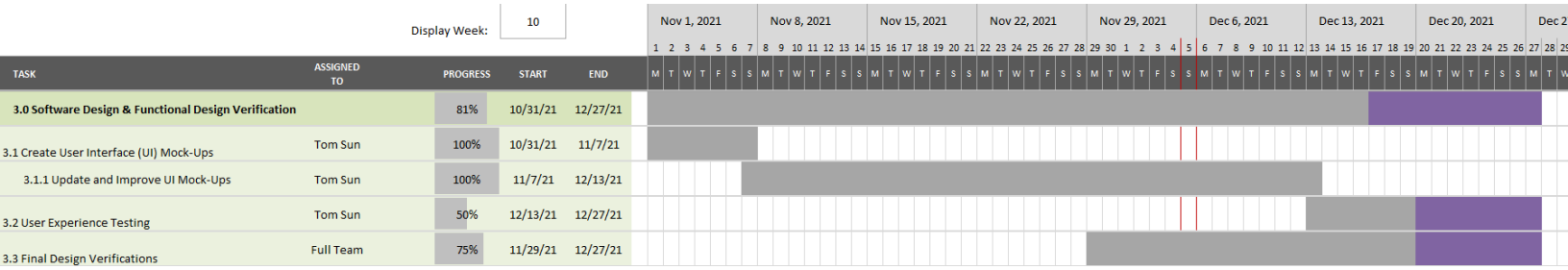
Seen above (Figure 2), is a generalized version of the gantt chart to the nine major tasks. More information can be found below in each individual major task gantt chart view.



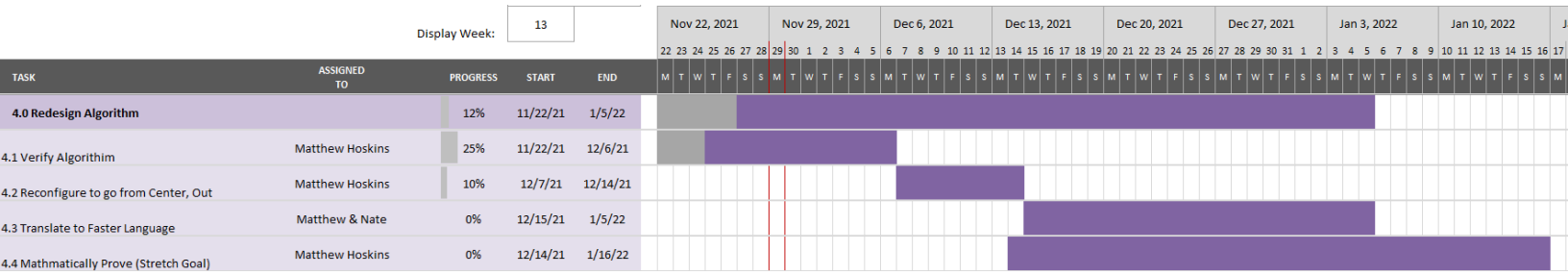
This zoomed out section of the project gantt chart is task *Figure 3: 1.0 Project Planning & Defining* (shown above). This section continues to span the weekly schedule as it includes the upkeep of various documents relevant to the project. It began on week one of the planner and continues from there with a variety of tasks that can be seen more closely in [*2.2 Task Decomposition*](#).



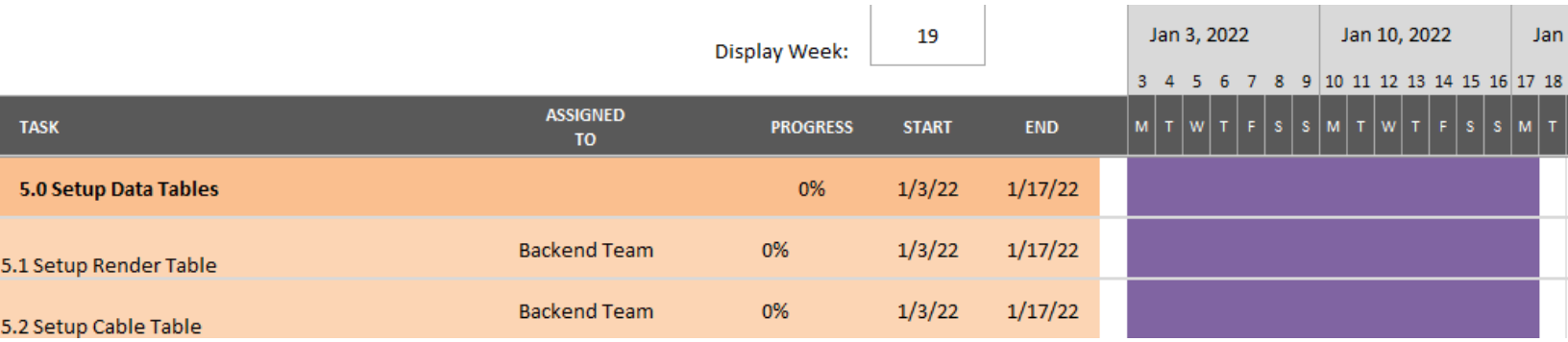
Task *Figure 4: 2.0 DevOps & Tech. Setup* (shown above), is the early planning, staging, and setup of technology and CI/CD pipelining.



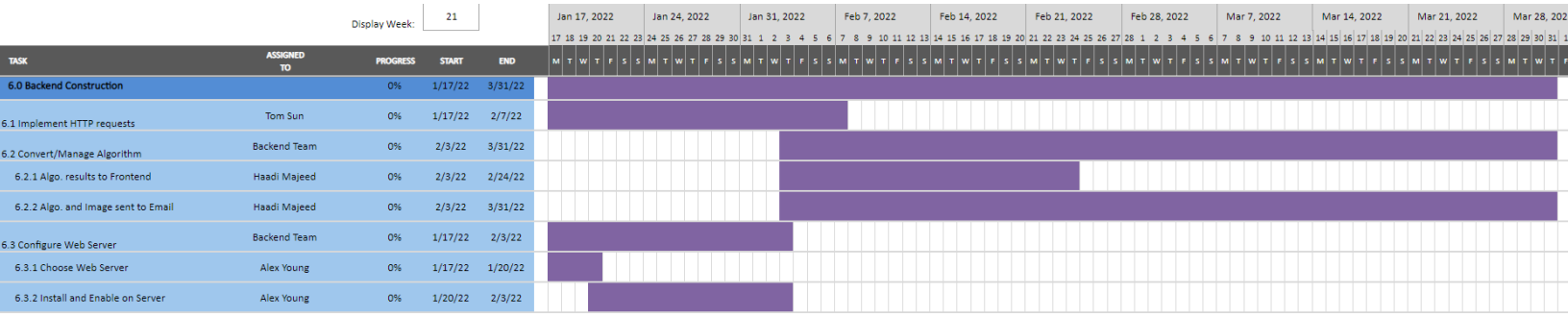
Task *Figure 5: 3.0 Software Design & Functional Design Verification* (shown above), will involve a few tasks dependent on each other in some form that will allow for the planning for the eventual UI of the final software project.



For task *Figure 6: 4.0 Redesign Algorithm*, is a math focused major task that will need to occur early on in order to properly ensure the current Python code can be verified, and converted to fit the projects' needs.

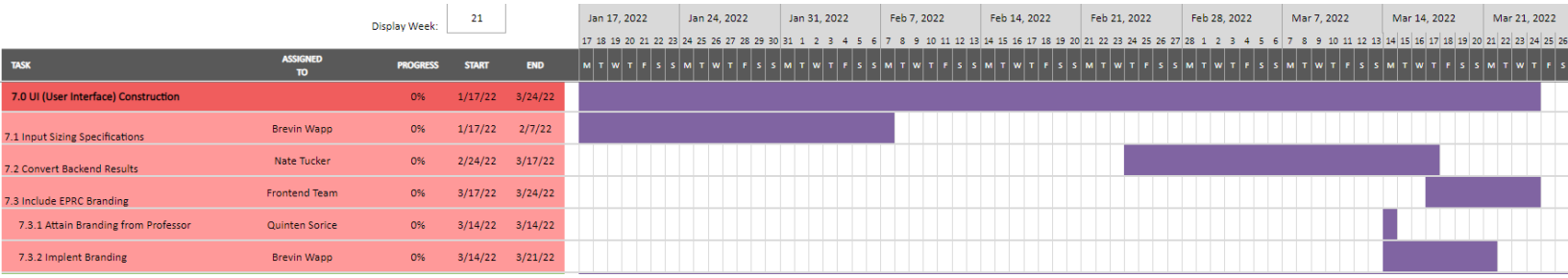


Task *Figure 7: 5.0 Data Tables Setup* (shown above), is a short major task that will be done in order to ensure all possible data points for entry are stored and available to the user input.



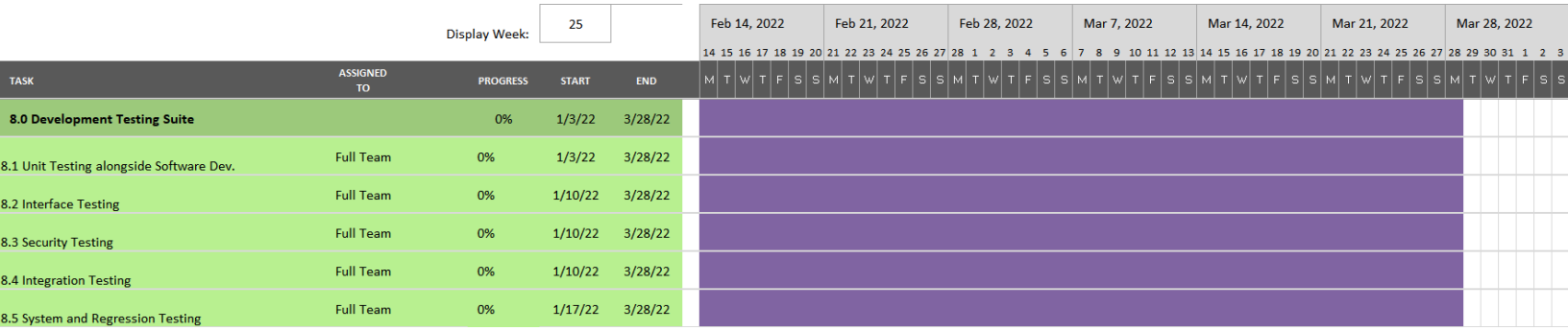
TASK	ASSIGNED TO	PROGRESS	START	END
6.0 Backend Construction		0%	1/17/22	3/31/22
6.1 Implement HTTP requests	Tom Sun	0%	1/17/22	2/7/22
6.2 Convert/Manage Algorithm	Backend Team	0%	2/3/22	3/31/22
6.2.1 Algo. results to Frontend	Haadi Majeed	0%	2/3/22	2/24/22
6.2.2 Algo. and Image sent to Email	Haadi Majeed	0%	2/3/22	3/31/22
6.3 Configure Web Server	Backend Team	0%	1/17/22	2/3/22
6.3.1 Choose Web Server	Alex Young	0%	1/17/22	1/20/22
6.3.2 Install and Enable on Server	Alex Young	0%	1/20/22	2/3/22

Task *Figure 8: 6.0 Backend Construction* (shown above), involves the long development of all the backend components to this software project. It starts after final design approval, and proof of concept programming.



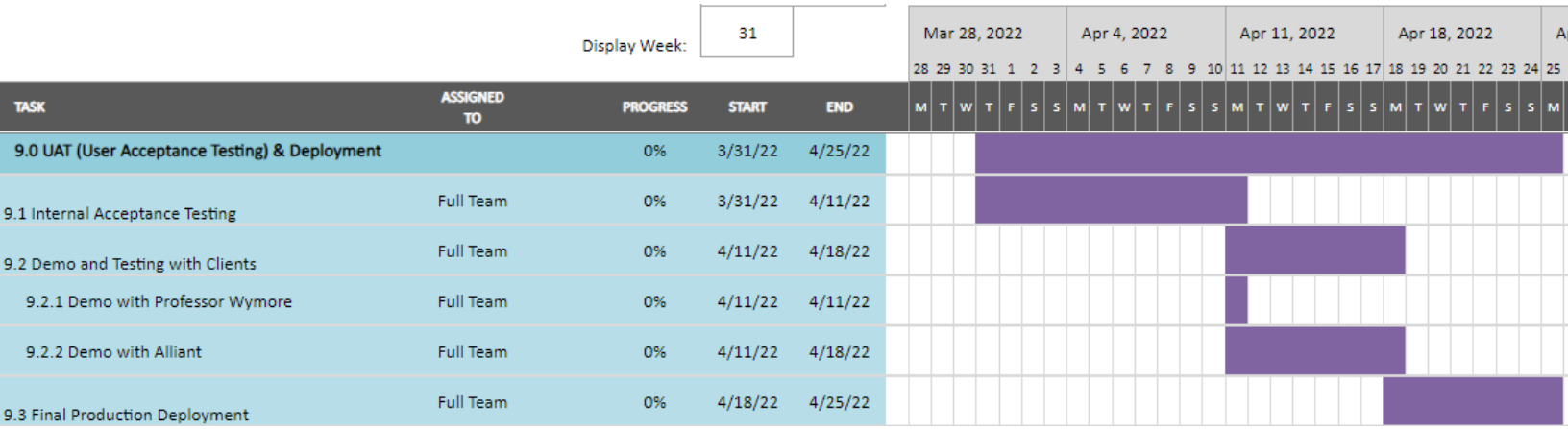
TASK	ASSIGNED TO	PROGRESS	START	END
7.0 UI (User Interface) Construction		0%	1/17/22	3/24/22
7.1 Input Sizing Specifications	Brevin Wapp	0%	1/17/22	2/7/22
7.2 Convert Backend Results	Nate Tucker	0%	2/24/22	3/17/22
7.3 Include EPRC Branding	Frontend Team	0%	3/17/22	3/24/22
7.3.1 Attain Branding from Professor	Quinten Sorice	0%	3/14/22	3/14/22
7.3.2 Implent Branding	Brevin Wapp	0%	3/14/22	3/21/22

Task *Figure 9: 7.0 UI Construction* (shown above), will begin at a later point after proof of concept software has been constructed, as well as, general design given approval.



TASK	ASSIGNED TO	PROGRESS	START	END
8.0 Development Testing Suite		0%	1/3/22	3/28/22
8.1 Unit Testing alongside Software Dev.	Full Team	0%	1/3/22	3/28/22
8.2 Interface Testing	Full Team	0%	1/10/22	3/28/22
8.3 Security Testing	Full Team	0%	1/10/22	3/28/22
8.4 Integration Testing	Full Team	0%	1/10/22	3/28/22
8.5 System and Regression Testing	Full Team	0%	1/17/22	3/28/22

As for task *Figure 10: 8.0 Development Testing Suite* (shown above), what is shown here is the ending of the testing suite that starts with the software development and ends with the development completion, and includes Unit, Interface, Security, Integration, System, and Regression testing.



The last major task, *Figure 11: 9.0 UAT & Deployment*, occurs at the end of the project with final testing and software fixes to be completed prior to final deployment. As testing will be occurring throughout the project ideally this will prove very efficient.

2.5 Risks And Risk Management/Mitigation

Each major task that was identified in *2.2 Task Decomposition* section is broken down individually for what risks could potentially occur along with an evaluation on the likelihood, and what the plan to mitigate these potential risks during the project development process. A table reference is listed below for each individual evaluation type. As this is an Agile project, risks and risk mitigation will be associated with each sprint.

1. Project Planning & Defining
   - Misunderstanding requirements
     - Unlikely, Catastrophic: Not properly understanding what our client is asking could mean building an application that is not useful or does not fit their needs. We will need to meet (and have been meeting) with our client to fully understand what they are looking for and how we can deliver an app that fits their needs.
2. DevOps & Tech set up
   - Mismanagement of setup
     - Possible, Negligible: If something in our virtual machine setup ends up being wrong, there is little hassle in getting the error fixed or configuration changed to resolve our problem.
3. Software Design & Functional Design Verification
   - Architectural problems
     - Rare, moderate/major: A major flaw with our architecture could result in problems throughout our project, so it will be paramount to select an architecture that will fit our needs before starting development
   - Wireframe issues

    - Unlikely, negligible: If our wireframes for design verification are not to the spec our client specifies, we will simply need to change them to fit requirements before implementing their design in the full application.

4. Redesign Algorithm
   - Mathematical Error
     - Rare, Major/catastrophic: An error in the calculations regarding the cable-fitting algorithm would result in delivery of incorrect results and the plethora of problems that delivering incorrect calculations to a client would entail.
     - Mitigation: Checking our algorithm results against the original application and against mathematically sound equivalent theorems.
   - Optimization
     - Likely, negligible: A low-consequence risk with redesigning an algorithm is that it is not as efficient or optimized as it possibly could be, so there could be a chance to reduce latency with a highly-optimized algorithm.
5. Setup Data Tables
   - Incorrect Table configuration
     - Unlikely, Minor: If a table for storing results or information is configured incorrectly, then a mitigation would be making a change in the database to accurately reflect our data, though going unchecked this could result in the mishandling of data storage.
6. Backend Construction
   - Improper data treatment and storage
     - Unlikely, Moderate: The worst outcome that can happen with a poorly built backend is returning incorrect data, which can mean inaccurate results and possibly a mischarge to a client. Ensuring that our backend returns the correct information and in a timely manner will be important as we build the application.
7. UI Construction
   - Confusing UI
     - Rare, Minor/moderate: If users cannot understand how to use the app, they will not be able to get the information they want out of it. It will be important for us to perform user acceptance testing so we can gauge how intuitive and easy to understand our application front end is.
   - Dysfunctional UI
     - Rare, Minor: If the UI is so poorly built that it either does not work or cannot give results, that would be frustrating as the user. A dysfunctional UI is hard to miss when using proper testing techniques, so this should be a very rare risk to occur.
8. Development Testing Suite
   - Lack of comprehensive tests
     - Unlikely, Major: With incomplete testing, there is a chance that edge cases in how our app is used could go unnoticed which would be frustrating for users that encounter them. Or edge case calculations could turn out wrong, and missing them would mean the possibility of incorrect charging of clients for bore sizing.
     - We will have to ensure that our testing methodology is thorough and we know the results we are looking for.

- Incorrect testing validation
    - Unlikely, Major: If tests run on the application are configured incorrectly or give false positives/negatives, there is a chance that an error would go unnoticed.
    - We will have to ensure that our testing methodology is thorough and we know the results we are looking for.

9. UAT & Deployment

- Inaccurate Results
    - Rare, Moderate: Should our user testing come back inconclusive or yield inaccurate results, then we would have a harder time improving the usability of the application should there be something substantially wrong with the design.

Consequence -> Likelihood \/	Negligible: 1	Minor: 2	Moderate: 3	Major: 4	Catastrophic: 5
Almost Certain: 5	5, Moderate	10, High	15, Extreme	20, Extreme	25, Extreme
Likely: 4	4, Moderate	8, High	12, High	16, Extreme	20, Extreme
Possible: 3	3, Low	6, Moderate	9, High	12, High	15, Extreme
Unlikely: 2	2, Low	4, Moderate	6, Moderate	8, High	10, High
Rare: 1	1, Low	2, Low	3, Low	4, Moderate	5, Moderate

Table 3: Risk Management and Mitigation Numeric Definition

2.6 Personnel Effort Requirements

We have a seven-person team. As a result, the tasks that require individual effort of every team (such as meetings and validations) will be scaled up accordingly to reflect total personnel effort. These evaluations are listed in table form with the task name, current estimated hours required, and a brief explanation of resultant estimation.

Task Name	**Est. hrs**	**Explanation**
Begin Team Communication	3.5	Half-hour each to set up communication channels
Set up communication with advisors	7	Half-hour long meeting each to meet with Jacob Conn and Mathew Wymore
Set up communication with external stakeholders	17	2-hour long meetings (total) to meet with external stakeholders. 3 hours for email communications

Assign leadership roles	7	1 hour long team meeting
Requirements - Wymore meeting	7	2 x half hour long meetings
Requirements - Alliant Energy	14	2 x hour long meetings
Requirements Document	15	1 hour team meeting + 1 hour individual work time + time to proof-read and submit assignment
Engineering Standards	14	Half hour team meeting + half hour individual work
Project Plan Set Up	14	2 hour individual work time
Project Plan Task Breakdown	7	1 hour long meeting
Finalize Project Plan	10.5	1 hour individual work and half hour meeting to finalize the document
Determine Software Stack	21	1 hour individual work and 2 hour team meeting
Get Familiar with Tech Stack	60	8 hour for each person, with some additional time
Tech Stack PoC	40	Basic stack set up, should be fairly simple
Project PR Standards Meeting	14	2 hour long meeting
Continuous Documentation and Technical Writing Up-keep	175	1 hour per-person for 25 weeks
Initialize Project Website	10	Infrastructure is already set up, so the team just need to construct the html pages
CI/CD Pipeline Set Up	15	Creating initial pipelines and integrate with Gitlab, some learning may be required
Set Up Deployment Environment	20	May involve meetings with IT services, set up VM/server
Set Up Individual Work Environment	28	4 hours per-person, since some learning/experimenting may be required
Create UI Mock-Ups	80	Includes time to learn mock-up tools and creating

		iterations of mock-ups
UX Testing	20	Include time to construct tests, meeting times with stakeholders and compiling data
UI Mock-Up Verification	14	2 hour long meeting with clients
Verify Current Algorithm	15	Time for getting familiar with the tool and extensive testing
Redesign Algorithm	40	Includes time for development and testing
Convert Programming Languages	15	Includes time for development and testing in new language
Mathematical Proof of Algorithm	40	Some research and information seeking may be required
Set Up DataBase	15	Infrastructure should be already set up
Create Data Table - Render	4	Includes time to test created table
Create Data Table - Cable	4	Includes time to test created table
Backend Construction - HTTP	100	Includes time to develop and test all functions
Backend Construction - Transferable format	50	Some Prototyping may be required
UI Construction - Inputs	50	Some Prototyping may be required
UI Construction - Visualize results	100	Prototyping and some research into visualization tools required
UI Construction - EPRC Branding	40	Adding styles/icons to the constructed software shouldn't take too long
Unit Tests	200	Should be done alongside development, estimated 1 hours per week per person

Integration Testing	100	Includes time for extensive test and making any fixes/adjustments
Internal Acceptance Testing (IAT)	14	2 hour meeting to review all aspects of the application
Demo and User Acceptance Testing (UAT)	14	2 hour meeting to review all aspects of the application
Final Production Development	60	Includes time to deploy and fix any last minute issues. Includes some time for monitoring after deployment

Table 4: Personal Effort Breakdown

For this current estimation, this would evaluate to 1,474 hours that would be split evenly between the team of seven people. This would make it about 210.5 hours per team member over the course of two regular length school semesters. *These numbers are subject to revaluation as the project progresses.*

2.7 Other Resource Requirements

As stated previously, this project is completely software based, and was not provided a budget making the total amount of resources small to begin with. Work hours from the team and its partners will be required for the project's completion, but can be found in section *2.6 Personnel Effort Requirements* for the team directly.

Aside from these resources, the only other resource requirement for this project is an Iowa State University server that will be hosting the web tool which will be negotiated and set up in conjunction with ETS.

3 Design

3.1 Design Context

3.1.1 Broader Context

The broader context of this design problem is situated around the concept of converting an executable program that exists for the purpose of making the transition between aboveground wiring (specifically electrical in nature) into an underground cabling setup.

The communities that this project is being designed for can be broken down into two entities being those in the electrical community that would be involved in boring of underground cabling, and the other community would be representatives of the professor client and EPRC. The communities that would be affected by this design would be those that will utilize the resultant software for determining overall bore sizes of the underground cabling. These communities will be affected purely on their usage of the web tool for determining proper bore size without any knowledge other than that of what wiring will need to be buried.

The societal need that this project addresses, is the need for easily accessible software for determining overall bore size of given wiring input that is an accurate representation of what the correct bore results should be. The expected use will be for the purpose of pricing and acquisition of appropriate amounts of wiring, meaning that the societal needs will be focused in that area specifically.

List of relevant considerations related to the project in each of the following areas:

Area	**Description**	**Examples**
Public health, safety, and welfare	A welfare connection between the project would be the incentive of easing the overall planning process for moving wiring underground to prevent powerlines from being affected by weather and causing damage or injury.	Reduces the possibility of having improper ducting/boring sizes making the move to underground wiring more stable of a wiring option.

Global, cultural, and social	The values and practices of the project and the process for the project would not be any realm that would cause concern regarding any affected communities (Iowa State University EPRC, Potential users).	This project is a web tool for specific calculations and visualizations. The process of creation and subsequent usage will follow programming standards for ease of use and user acceptability.
Environmental	An indirect impact created by this project would be, since it enables an ease of underground cable construction, the amount of tampering with top level earth would change as well as the amount of existing above ground cabling.	This would imply an increase in undergrounding boring for laying underground cabling, and maintaining said cable. The other effect would be less vertical above ground cabling structures that would either obstruct natural elements or become potential debris.
Economic	As the final project web tool is to be used in a planning and financing sense by potential clients, this project would have the ability to save labor and reduce human error. The only expense of the project would be by ISU for maintaining the code and server hosting the program.	This would mean that there would be a small cost to the hosting user (ISU), and a free tool for calculations for all other users that can ensure accurate planning of wire/duct sizes and corresponding amounts.

Table 5: Project Area Considerations

3.1.2 User Needs

List of Users:

The primary users of this project: ISU's EPRC acting as hosts and eventual software controllers, primary client groups consisting of companies with underground cabling needs (specifically, Alliant Energy)

The secondary users of this project: various contractors that would be working with either a primary user or another outside party, with the ultimate goal of setting up underground cable packages.

The tertiary users of this project would be every other user of the final product as it will be an openly available web tool with association to ISU's EPRC.

Individual User Needs:

The primary user, in regards to ISU's EPRC, requires a web based software that will be able to complete some basic wiring and bore size functions because an existing executable program was requested, by the other subcategory of primary users, to be implemented as an openly sharable program.

The other primary users need a program that can be openly available while easily and efficiently calculating optimal wire package size that can be shown as a visual representation with proof of optimal measuring. This is for the purpose of being able to correctly predict the sizing and amount of wires and ducts for purchasing, and to appropriately assess each individual jobs pricing for contractors.

The secondary users group need a way to have easy access to the algorithm without having to obtain the existing executable software from EPRC to prevent property violations, so that this group can confirm pricing with the primary users as well as gain the ability to create their own calculations.

The tertiary users group needs a way to gain access to quality software that can clearly show the results of circular objects embedded inside other circular objects because this could be a complicated mathematical problem that would be made simple from this software that is already being created for the use of the primary and secondary groups.

3.1.3 Prior Work/Solutions

A previous example of a solution that fulfills the same need as our project has been made at Iowa State, namely the Python-based desktop application created by our professor contact Mathew Wymore, however the Python app has some shortcomings. The Python application works well enough in that it retains a simple interface and can give accurate results quickly but lacks portability (as a static desktop application) and has no online connectivity. The web version of the application we intend to build this year will open the door for additional features and would be more user-friendly to a wide range of clients that may not want to set up a desktop application to get an accurate cost calculation for laying underground cables. A web version would also allow for the exporting of results in a more streamlined fashion compared to a screenshot of the Python application's input and output.

3.1.4 Technical Complexity

The design of this project will not simply entail the porting of the existing Python application into a webpage; there will be an analysis and redesign of the algorithm that determines cable best-fit in a given diameter pipe which will have to be verified against existing mathematical models, a new interface that can interact with our algorithm though a webpage, and the addition of new features like sharing results through links, storing results for future reference, and in the future offer mobile browser support, something the Python application cannot currently offer at all.

To implement such updates and improvements, we will have to build an updated model of the existing cable packing tool's algorithm to ensure optimal performance and accuracy. The algorithm will then need to have an interactive front-end, back-end, and database to allow users to interact with the app, retrieve results, and store/lookup previous calculations respectively. While existing technologies will aid us in building the structure for this application, such as JavaScript libraries to give us more front-end functionality or backend frameworks to allow faster querying of the algorithm, there are no plug-and-play solutions on the market that would fulfill the same goals as us building this tool ourselves.

3.2 Design Exploration

3.2.1 Design Decisions

Below is a list of some key design decisions that the team has made in relation to the solution that we have devised. The project is purely a software development project making a clear limit to software based decisions. It should be noted that the software will be hosted on an ISU server, and decisions related to said server have been made together with standard implementation and technology in mind.

1. We decided that in order to keep our project lightweight and easy to use, we will not be implementing a user account system. There was debate on being able to save the diagrams to a user account, but we have found alternatives that do not require as much involvement from the user. Instead, a potential stretch goal (2.2 Task Decomposition part 5.3) Company Admin Account, was created with the idea of having a single account per user group to add special data sets of cables and bores.
2. After the initial meeting with a representative from the expected user (and non-primary client), Alliant Energy, it became clear that a variety of features they had envisioned were too disparate from the originally planned project end-goal put forth by ERPC and Professor Wymore. The original response that the team went with was to hold off on configuring the design of the project to fit these proposed functionalities, and to continue with the original vision that was created after requirements talks with the primary client

representative of Professor Wymore. Moving forward in the design process we took into account Alliant Energy's ideas to build upon the planned features and requirements in order to accommodate their hopes for the project while making sure the focus was still on the original project vision. Many stretch goals for this project (featured in section 1.2 Requirements & Constraints) come from some of the additional functionality that Alliant Energy expressed interest in.

3. We were initially open to multiple technologies as there are not many external requirements when it came to technology. However, because of our familiarity with React and its broad applicability, we decided to use React for our project.

3.2.2 Ideation

The process of exploring potential design decisions was a multistep procedure that involved the identification of the project or tasks requirements, followed by identifying technologies, programming languages, and/or frameworks that could fulfill these requirements. After that step the team would come together and decide on a technology to move forward with (after weighing the pros and cons of each option. This would then lead to testing to ensure that the decided technology would be able to accomplish the requirements. At that point, one of two actions would be taken, if it would be suitable that technology would proceed and be used for implementation otherwise the team would go through an abridged version of the process to find a better technology for the job.

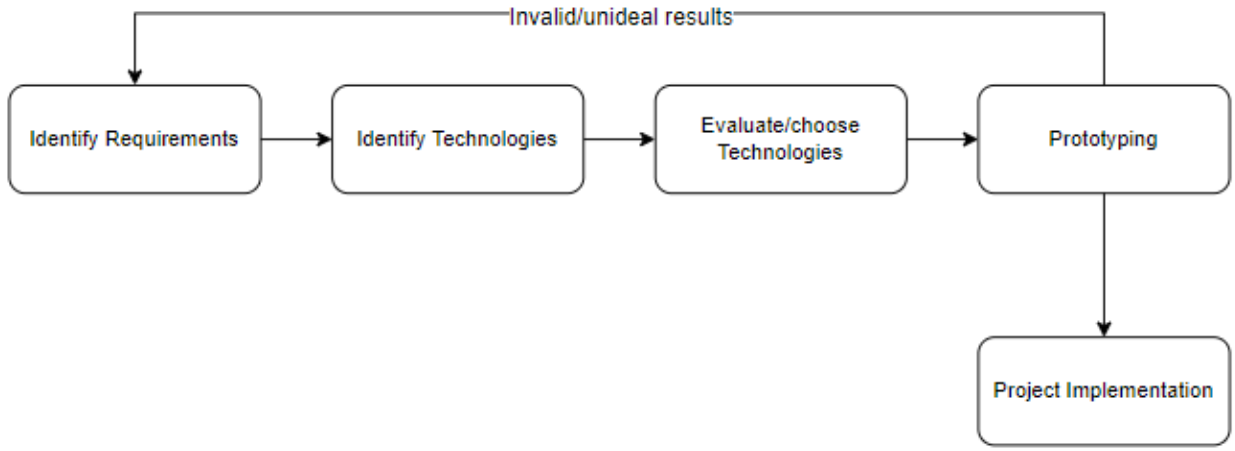


Figure 12: Design Approach

Shown above is a basic outline of the full process of deciding a specific technology at a broad perspective. Each of these steps included a more in depth process, for example: at the identifying technologies stage, the team generated their own ideas for technologies that could be used. These ideas were compiled based on prior knowledge of suitable technologies, programming languages, and frameworks that could apply as well as searching various potential solutions over the internet; that would fit with the requirements identified in the previous task. From that point on the design would then be based around the technology that was evaluated to be the best suited for the task.

One design decision that had an in-depth process towards which technology would be used and how it would then work into the software as a whole was that of the backend/algorithm section. The task that needed to be completed was deciding on the programming language for the backend and algorithm section of this project that would then be able to fill those roles while connecting to the already ideated frontend.

Some of the evaluating criteria that was then setup was based on the project requirements that would connect to this broad area of topic as well as the focus of deciding a language that would have other potential benefits. The process of languages that could fit these criteria was compiled by the team given prior knowledge/experience and research towards a language that would align with the needs laid out that the team was not as familiar with. The top five languages that were identified to have potential satisfaction were as follows: Python, C, Java, Golang, and Ruby.

All of these languages met the requirement that the software should be a well-known technology with documentation and the expectation to be maintained for at least the next ten years. These languages also had the added benefit that at least one member of the team already had working experience of the identified languages.

Python had the benefit of being the language used for the executable program that the web application was going to be based off of, but was ultimately removed from the list because of the requirement for processing times of the algorithm. Python as well as Java were identified as having potential of being too slow for the calculation process, and would then not be able to meet the processing time target.

After breaking it down to just C, Golang, and Ruby. C was a language that every team member had experience in, but was ultimately eliminated from the running along with Ruby because those that were familiar with Golang and research into the language determined it to be the more efficient language in most instances. Meeting the processing time requirement was what ultimately led to the decision of Golang with the thinking that it would enable the most efficient algorithm and backend.

At this point the remaining decisions for the backend/algorithm portion of the project were simply how it would be set up in a broad sense and how it would connect to the frontend. For more detail on the proposed design see section *3.3 Purposed Design*.

3.2.3 Decision-Making and Trade-Off

The largest thing that affected our tech stack decisions was speed. Either speed of development or application speed. This eliminated a few choices right away that, while extensible and more feature-dense in the end, would take far too much setup and time to get off the ground.

The other decision was application speed. For that reason we chose a split frontend-backend that would present as quick of UI as possible and do number crunching as quickly as possible.

For the frontend we landed on React JS because of its pretty and snappy feel and how easily it would be to display the data, and for the backend we ultimately landed on Go because of its builtin http server module and its speed. Compiled languages were always going to be the best choice, and Go had the best http implementation.

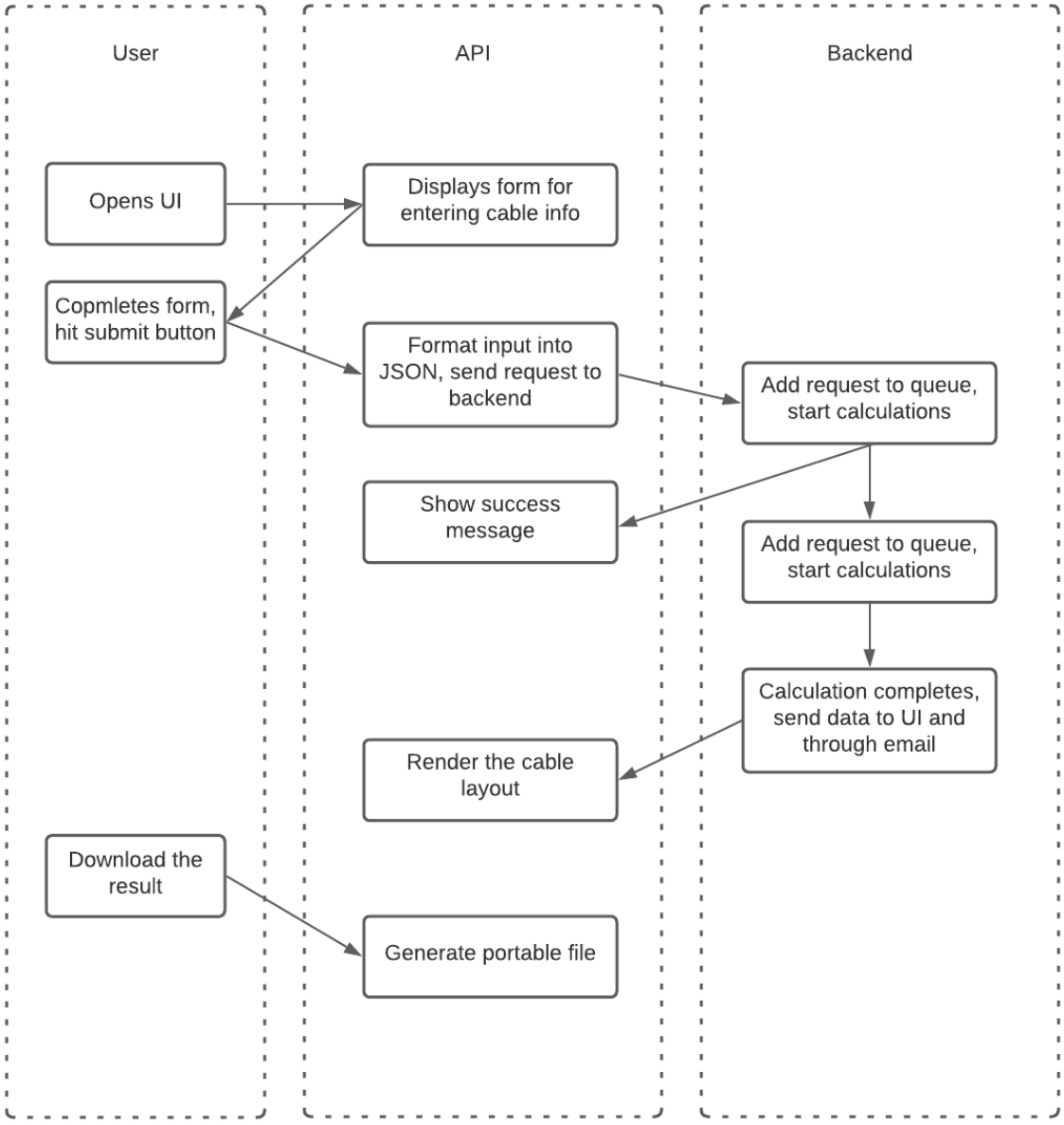
A final decision was our requirements for testing, internationalization, and accessibility. React has some very simple to use and extensible options for all of these requirements and presents the best options for a good user experience.

3.3 Proposed Design

The following includes designs that have been implemented, tested, attempted, and mocked-up. These designs were created with the requirements of the project in mind as well as other general usability standards and visually appealing frontend plans.

3.3.1 Design Visual and Description

API & User Interaction Diagram: The following diagram shows the interactions between the user, UI, and back-end APIs, with respect to time. (*FIgure 13: API & User Interaction Diagram)*



Software Architecture Diagram: The following diagram (Figure 14) shows the large point-of-view software architecture of the designed software with the major components being the browser, the server, and the database of cables and ducts.

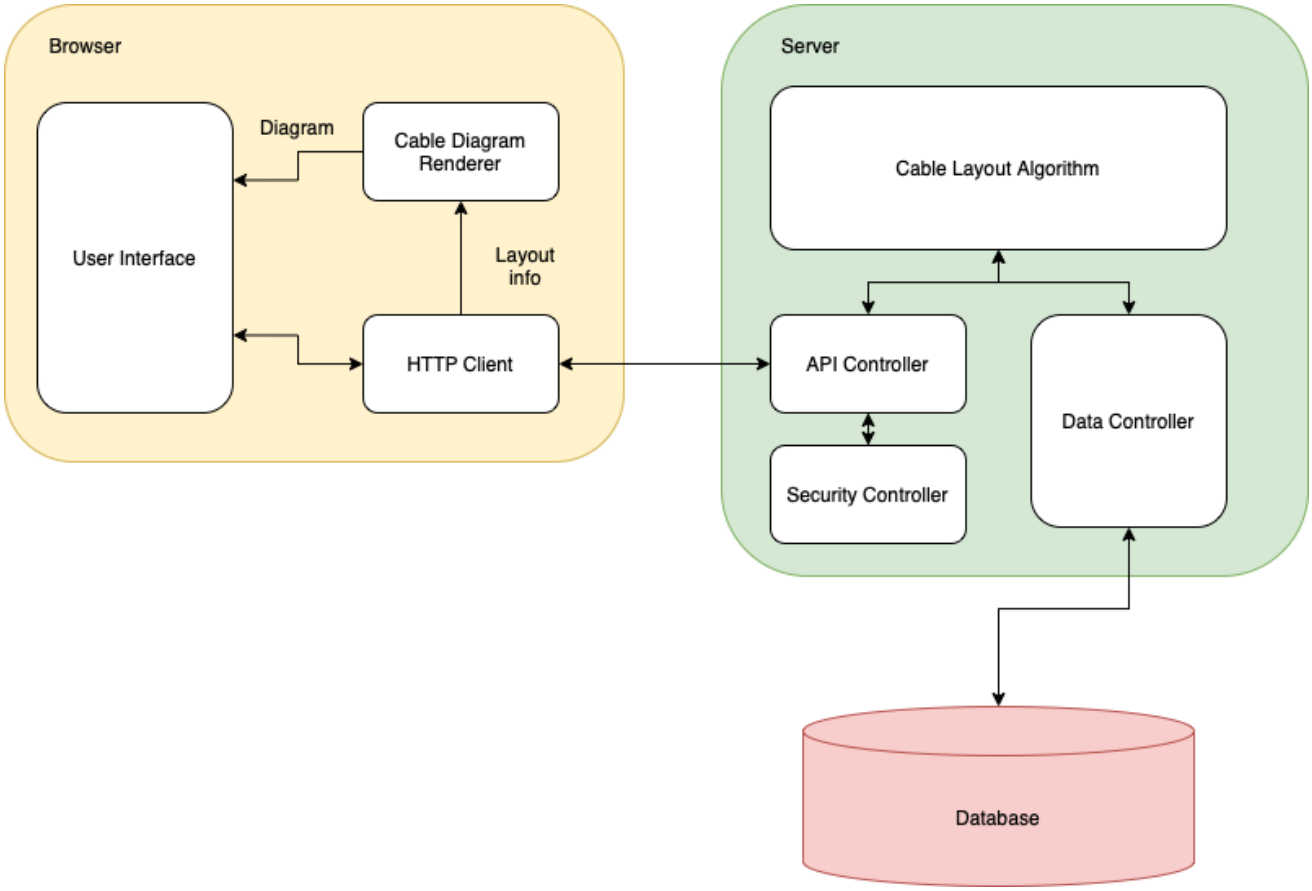


Figure 14: Software Architecture Design

The browser will be the interaction point for the user, with the UI being the visible components to the user. The “Cable Diagram Renderer” will take data sent from the algorithm backend and create the final images to be sent to the user. The “HTTP Client” will then function as the point of data interchange between the frontend and backend.

The server will be the ISU hosted server running the backend/algorithm functionality of the software. The “API Controller” will be the data interchange component for the backend taking the user input and sending results back to the frontend. The “Cable Layout Algorithm” is the algorithm component that will take the transmitted data and get the results to be sent back to the frontend. The “Security Controller” will perform any security checks needed, and the “Data Controller” will perform any access of information to the database.

The database will function as the database that holds any cable information to user presets.

UI Mock-Ups:

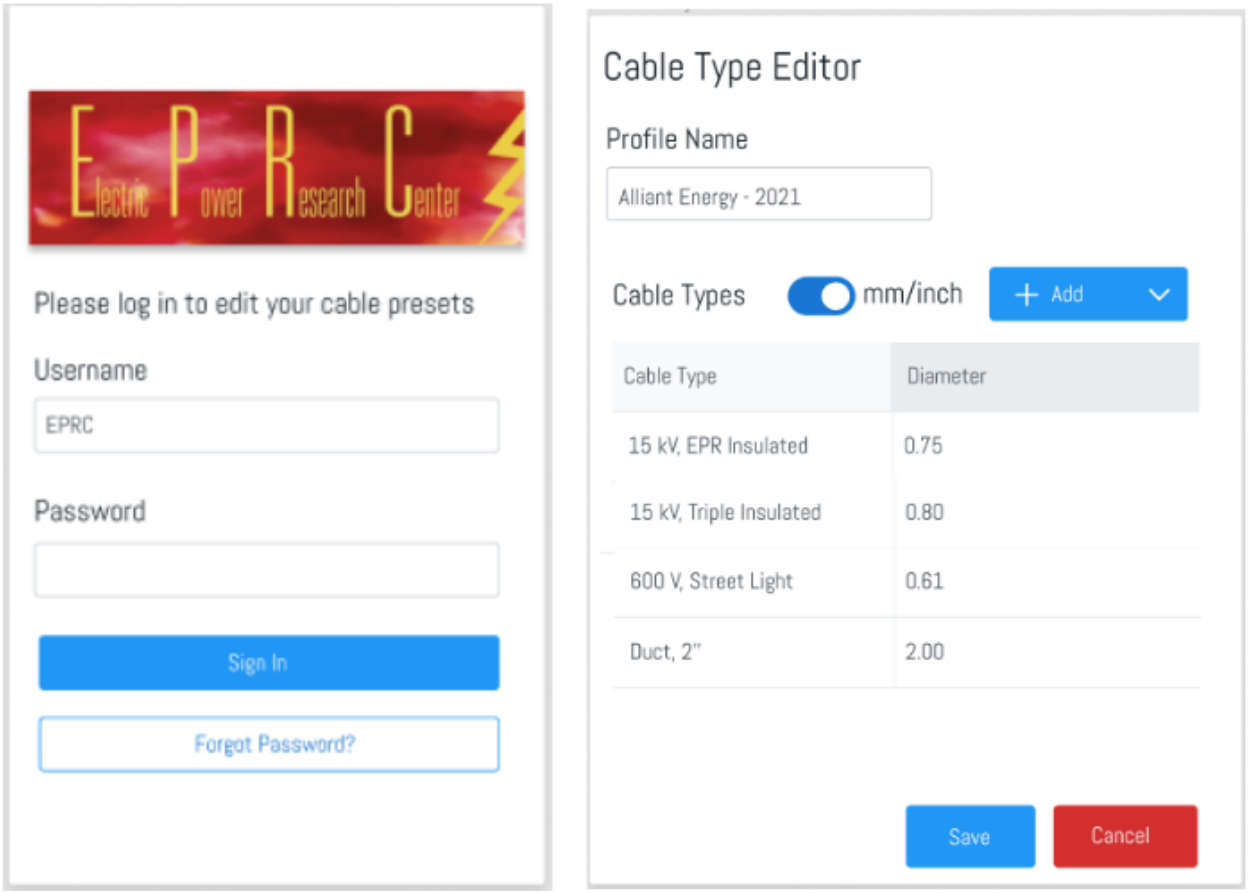


Figure 15: Overlay for Login screen and Cable Setup Mock-up

This (Figure 15) is the mock-up for what a user would see when setting up a profile for a company/group of users. First, users log in through an overlay panel shown on the left. It allows the user to add a cable type/description along with a diameter in a specified measurement all under a specific profile name. A save and cancel action are also available for the expected uses during the editing or creation of a cable package profile.

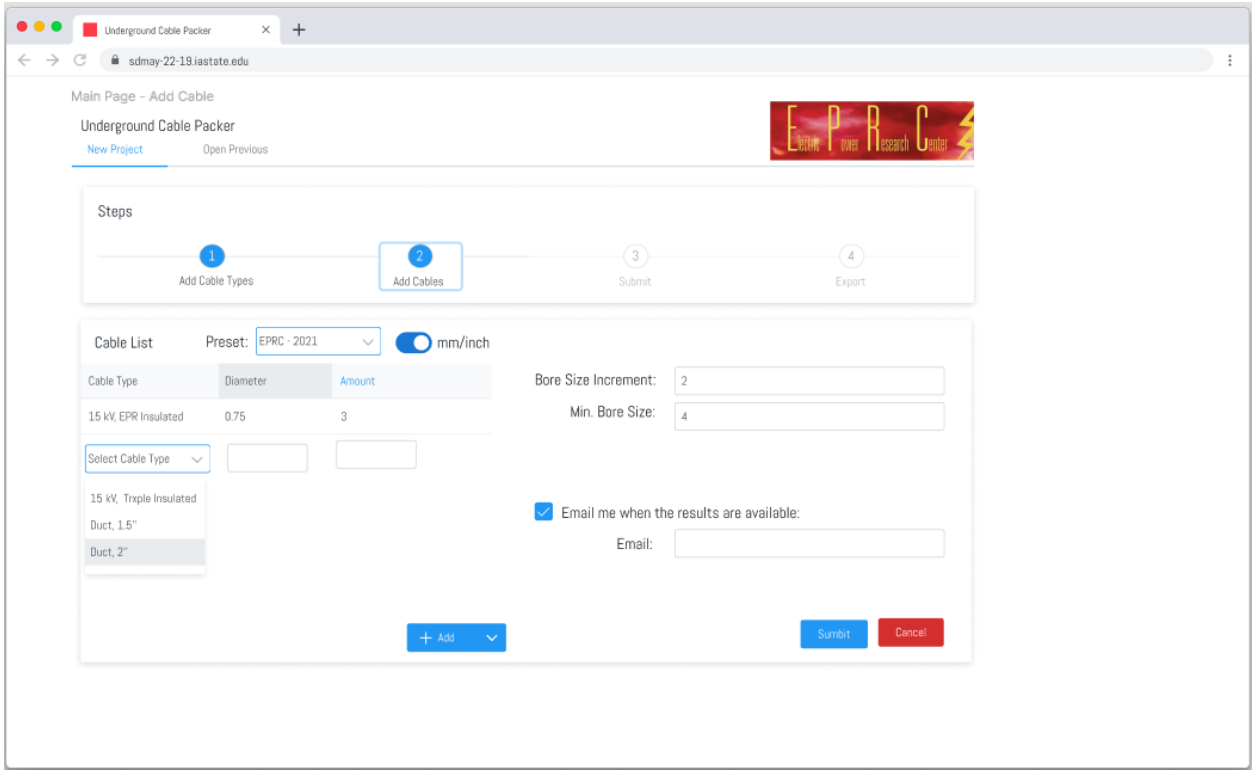


Figure 16: Cable Input Selection Mock-up

This (Figure 16) is a mock-up visualization of the web application during the process of inputting their specifications for a certain query. The preset dropdown allows for the user to choose which preset to use for available cable types.

As for the primary input section located on the lower left of the screen, the left column (“Cable Type”) of the input field includes a dropdown where a user can select a cable in the preset they are using, or to type in a placeholder if desired. The “Diameter” column will automatically be populated with the correct measurements if the cable type selected has a designated diameter otherwise the user will fill in the desired diameter for that row of cables. Lastly, the right column (“Amount”) specifies how many of a certain cable is desired for this iteration of calculations. Each row is for a different cable type, and rows can be added as the user requires them.

Other input locations that are optional are the “Bore Size Increment” input field, the “Min. Bore Size” input field, and the email checkbox and input field.

The “Bore Size Increment” field allows a user to specify the outermost bore increment value when checking for the smallest valid, and largest invalid bore sizes. For example, if the increment is set to two (inches) and the algorithm determines the smallest outer bore size to be six from an array of incrementing by two starting from zero (0, 2, 4, 6), then the smallest minimum is six and the largest invalid is four, and those would be the final images outer bore size rendered in the results.

The “Min. Bore Size” allows a user to provide a minimum bore size for their query meaning that should the user specify four (inches) then the algorithm will not consider any outer bore size smaller than four inches, and will go through the default or specified incrementation value from four.

The remaining optional input fields are concerned with if the results are sent to the user’s email function as expected. If a user marks the checkbox for “Email me when the results are available” it will then have the user fill in an email address in the input field for “Email”, and send an email to the specified account with the results.

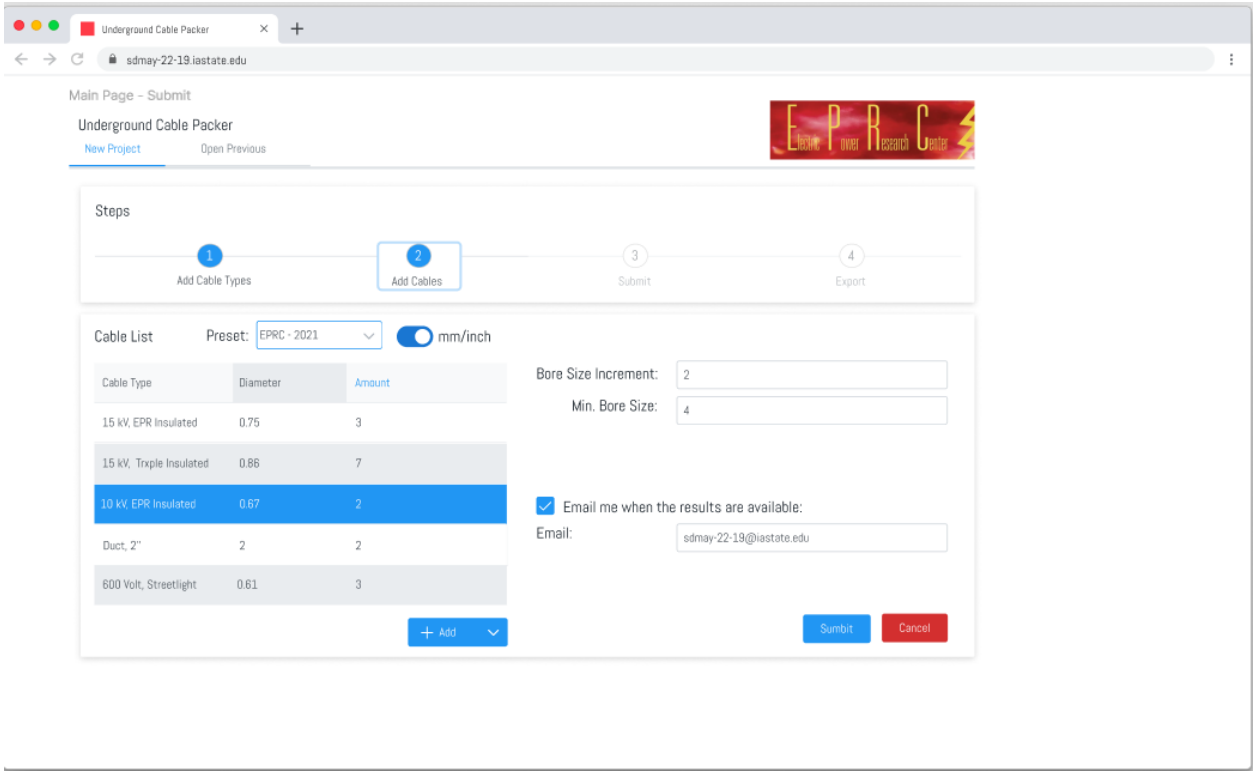


Figure 17: Cable Input Selection Mock-up (Filled out)

This (Figure 17) shows a completely filled out query still on the same page of the web application as Figure 16. After a user has completed the entry of all the desired information they then can either press the “Submit” button or the “Cancel” button. Doing both what would be expected. The “Cancel” button would not submit but cancel the query. The “Submit” button would send the input to the backend, located on the ISU internally hosted server, and run the inputs through the algorithm to then send the results back to the frontend when this has completed.

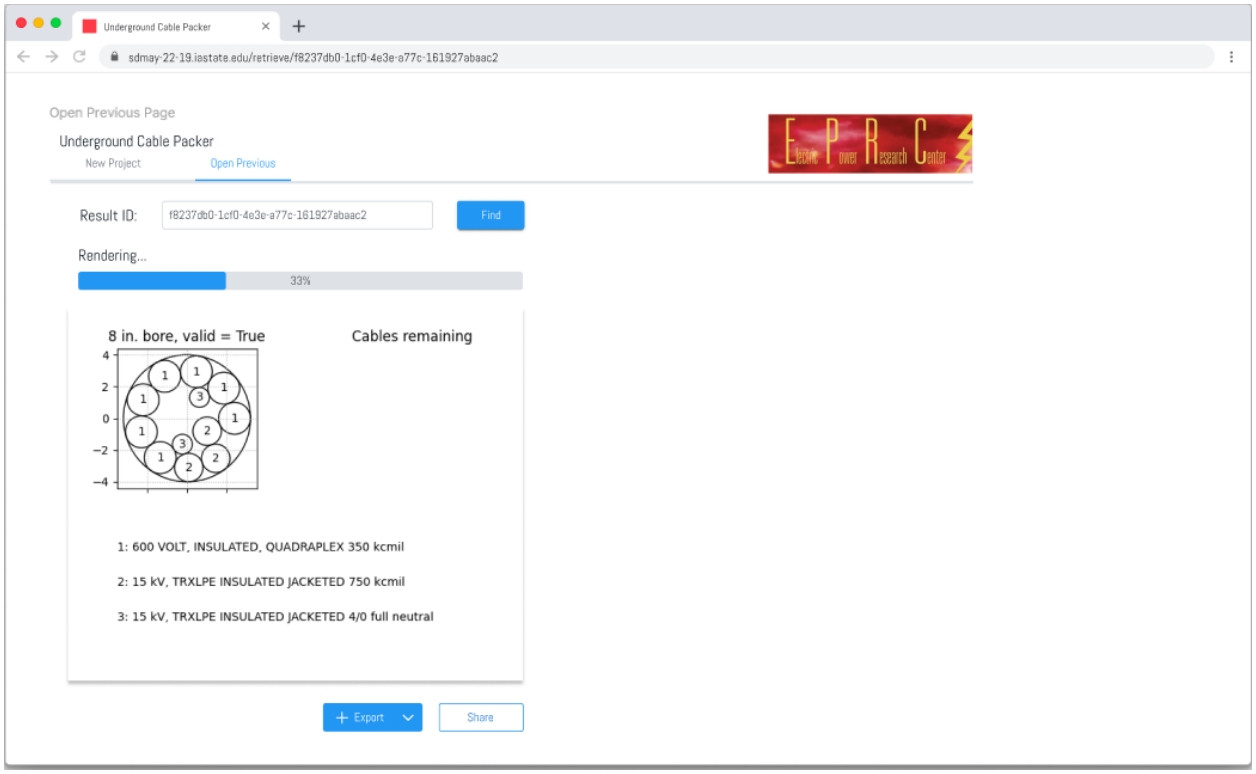


Figure 18: Results Page Mock-up

This (Figure 18) is a mock-up of the results page. The main aspect is the visualization of the calculated maximum bore size with cables packed into it for clear validation proof. The other visualization that would also show up on this screen is the next smallest bore size (based on input or default incrementation) showing how one size smaller than the produced valid is not valid with a list of what cables/ducts push it over into needing the next size larger.

On the top of the screen, there is a “New Project” tab and a “Open Previous” tab. The “New Project” tab will allow the user to begin a new query from the first step. The “Open Previous” will temporarily hold the users created run throughs of the process for either reference or going back to access again for whatever reason they may have.

The “Result ID” is a section that allows users to open a previously created result by putting in the “Result ID” that will be given to the user upon query completion. This allows the user to have the ability to go back and look-up results should they have not saved them.

At the bottom of the screen there is a “Share” button that allows users to share the created results through the supported methods of the web application. Also at the bottom of the screen, there is a “Export” button that allows users to save the results in the supported methods of the web application onto their computer.

3.3.2 Functionality

The design is intended to operate as an interactive website, which can be accessed by a modern browser. It will have fields in which a user can input their desired cables, ducts, and other underground utilities, and additionally, adjust the other parameters of the application. Once the user is satisfied with the selections they have made, the application will then perform the calculations and return a generated image and properties of the bore as well as how the cables will fit in the bore. Additionally, the application will return the next smallest bore size and display what cables would not fit should that bore size be used. The user may then opt to generate a link that can be shared to the results of the application.

We believe that the current design will satisfy the requirements laid out by the requirements documentation. While our design is still in the planning and research phase, and the UI is only in mockup, we believe that all requirements can be met.

3.3.3 Areas of Concern and Development

One concern being that a potential primary user that has input on the projects requirements (i.e. Alliant Energy) has shown desire to create a more specialized tool that would be specific to their system, but the main goal of this project is to create a generalized version of the predecessor that will be open and available to whoever wishes to use it. In light of this, we have developed a few possible solutions that we are looking into to resolve them. Such as for the generalization of Alliant Energy's requests and making it so the data can be exported to various different file types for usability.

A second concern we have identified is with software running the algorithm and concerns that it will not perform to the standards we have set for it, as in it may not run at speeds desired, slowing down the process. To avail this, we have considered rewriting the algorithm into a compiler language instead and even look into refactoring the code to further improve the speeds it can perform at.

Finally, we realised that scaling such a program to be correctly output and interacted with on a small screen such as a phone screen. This could have multiple ways to go about it, however we have not finalized which way this application will follow. Our current idea for it is to output a static picture at the top with details below, in contrast to having it on the side. Additionally using a front end structure that can determine screen size and adapt accordingly will be useful to assist in this.

3.3.4 Technology, Frameworks, and Libraries

This project will involve a variety of technologies, frameworks, and libraries given the need to develop a web application with a frontend, backend, and database. Along with the software development is the necessary project documentation, scheduling, and planning (as for communication see 7.5 Team Contract). A generalized list of these with the used version number and a brief explanation for their use is included in this section.

For the Web Application Project:

React, version 17.0.2: Frontend development of the web application to create the UI/UX that will run on a web browser, gather input from users, show results of user run queries, and connect to the backend.

Golang, version 1.17: Backend and Algorithm development of the application that will handle the calculation of maximum valid duct sizes, connecting to the database content, and sending results to the frontend.

PostgreSQL, version 14.1: Database management system to handle the various cables and ducts that a user can choose from during the input stage of using the application.

Prettier, version 9.0.0: Software standardization to ensure consistency of the code that is written between the team during the development process.

GitLab, version 14.4.2: Software development and sharing tool that will enable effective code merging and task management.

JSON, version 2020-12: Data-interchange formatting between the frontend and backend.

Google Drive, online version: Project documentation creation and editing that will enable full team collaboration and ease of use toward being actively updated. Will be able to store all created documents in a single access point with complete version control.

From the Existing Program:

Python, version 3.9.5: Primary programming language of the existing program that the web application will be based on. The conversion of this language into either React or Golang will be necessary to create the web application.

Matplotlib, version 3.4.2: Visualization library used to create an output of the existing algorithms results. This will occur on the frontend of the web application.

NumPy, version 1.20: Mathematics library used in association with the programming language Python to create results from the algorithm.

4 Testing

With this project being purely a software development project, testing will occur on exclusively the software side. That being said, it is recognized that testing will be extremely important for ensuring the validity of the code that each developer will produce, and subsequently ensuring that requirements are appropriately met. The tools that will be used will vary depending on the area of the development (frontend, backend, etc), but will be along the lines of software testing tools of the various languages and frameworks that are being utilized for the development of the project.

The testing strategy that will be utilized was agreed upon by the team, and uses modern practices of software testing in a development environment in accordance with existing software testing standards. These various testing components will be performed alongside the agile development methodology chosen for the development process of this project.

4.1 Unit Testing

We are going to utilize two testing frameworks for our separate codebases. First, for the client-side application, we will be using the built-in testing framework “react testing library” with “jest.” This gives us the ability to test individual React components, custom hooks, functions, and mock up items between them as necessary for as much granular testing as possible. It can simulate individual unit tests as well as simulate user input and firing of events to give us some UI/UX testing as well, although not as much as would be preferred by QA engineers, for example.

The server side application will use the builtin go testing framework. This will give us a lot of control over what is tested server side, so as not to go so far into testing where we are just testing the library. Individual functions, components, and modules will all need to be thoroughly tested and have those tests passed in order to be accepted into the main branch,

The unit tests will be a baked in component of the overall CI/CD pipeline as well. Should any tests fail or act not according to the testing criteria in the automated tests, it will not be accepted into the main branch. While developers are given liberties as to the specifics of their testing, one popular example the team is encouraged to follow is the ZOMBIE testing methodology.

4.2 Interface Testing

In this software system, we have two primary interfaces: the User Interface (UI), and the Application Programming Interface (API). The UI is the web-based interface that users can interact with to perform tasks as described in the requirements. It can take in commands and display the results to the users. The API communicates between the front-end and the back-end, sending users requests generated by the UI to the algorithm, and sending completed results back to the UI to be rendered and displayed.

To test the API, automated tests such as the Unit Tests described in section 4.1 Unit Testing and integration testing as described in section *4.3 Integration Testing* can be performed periodically, after each code push. This testing will ensure that the API is compliant with the requirements and prevent regression defects as development continues.

Similar to the API, the User Interface will also implement automated tests that are executed periodically. In addition, we will use manual testing to ensure that the user experience requirements from our clients are met as well. Each component's design will include a section that describes a UI test case, in the form of a step-by-step checklist. During testing, the validator will interact with the UI as described in the test case, and verify that the UI's behavior is exactly as expected. This combination of automated and manual testing will ensure that the interface meets the requirements.

4.3 Integration Testing

There are a few integration paths for our design. One is between the front end and back end, which we have designed to be via an API layer. There are also integration paths with how our software will interact with the hosting system, which we may also test, to ensure that the application is functioning correctly and is accessible to the users. For testing the integration between the front and back end, we have a few options on tools.

One such option is using a headless testing browser environment, such as Selenium. This will hook into a headless browser and run a full battery of tests against our entire application. This will ensure that the frontend and backend are producing expected results and are integrating together successfully and that they are producing the correct output to the user.

Another option we have is a test kit that hooks into the frontend, and triggers it to make requests to the backend, just as a real user would. This software may be partially custom written by us to best hook into our application. We will most likely be modifying a unit testing toolkit to make these requests.

All these tests will be run automatically within the confines of the CI/CD pipeline. We will probably be utilizing Docker or a system like it to spin up databases and other system dependencies so that we can test in an environment as close to production as we can. This will also allow us to create new databases and tables as we need, without disrupting the production environment.

4.4 System Testing

System testing is a level of software testing that validates the complete and fully integrated software, and as such will occur after the process of unit, interface, and integration testing. Considering the plan is for the unit tests to be extensive and required for eventual merging into

the main branch program this series of testing should occur with relative ease. Especially when considering that the integration testing will automatically ensure that these tests will properly run and integrate when the code base is sent to CI/CD pipeline on the project's GitLab. Due to this, the process of 4.5 Regression Testing will become a part of the system testing as it will pertain to the connecting of various developers code into the single system, and will thus require testing alongside general system testing for ensuring existing functionality in the system is not broken.

From all of this, the system testing will expect that all existing functionality will not be broken by new integration as well as testing the new functionality that the code being merged into the system remains from its individual testing.

As far as what will be required for this, will be the same as the previous sections (*4.1 Unit Testing*, *4.2 Interface Testing*, and *4.3 Integration Testing*). There should not be any need for additional tools for system testing as it will ensure the overall functionality of the system after merging of new functionality. This will mean that additional testing may be required to show full functionality of the system from end-point to end-point. Ideally, to ensure existing developer bias, this series of test creation and testing will be performed by someone other than the developer of what is being merged into the system.

In order to connect this to the requirements, any additional tests will be focused on the functionality of the requirement that the task being merged into the main system is supposed to create.

4.5 Regression Testing

We will be using gitlab's integrated CI/CD tools to verify that when a new push to a feature branch or master is made, the automated test suite to verify that functionality is intact will run, and should any errors occur, gitlab will either prevent the merge, if the request is made against master, or let the committer know the errors if it is a push to a feature branch. This way we will avoid cases where a new feature merging into the rest of the project potentially breaks the master or production branches, and feature branches will have assurance that all required features are working in the grand scheme of the project rather than just one developer's computer.

Ideally, there will be no issues with integrating new features into previous main builds of the project, however should a feature break one or more parts of the master branch, GitLab's CI/CD suite will be able to tell us what exactly is conflicting or breaking what so we don't need to guess at any point. The most basic regression test for us should see that the algorithm itself is not interfered with in any way, as it is the heart of what will make this project successful. Next, ensuring the backend calls to that algorithm stay functional will be most important as the app has no functionality without being able to get results out of the algorithm itself. Next most important is being able to interact with the backend through the user interface in the frontend, since without a human interface, no one will be able to use our application. These critical

features should always be tested for functionality with new builds and new features before those new features are merged into the rest of the project.

4.6 Acceptance Testing

For our acceptance testing, there are three primary steps. Should any step result in actions to be taken, the process will start again. The first step is the full implementation testing by the team. This is to verify that all the features we wanted to implement are completed and ready to move on to the next step. We will iterate through our requirements document and check that each requirement is being met. Step two would be a very similar process of going through the requirements documentation, but this time with our client Matt Wymore. In doing this, we can get their perspective on if the requirements are being met to the degree that they expect. The final iteration is then meeting with Alliant Energy, our industry reference. We repeat the process of going through the requirements, showing off the implementation of those requirements, and getting a consensus on if the requirements have been met. Additionally, should new requirements be requested, we can evaluate the feasibility of the new requirements, and either go back and implement them, or discuss alternatives.

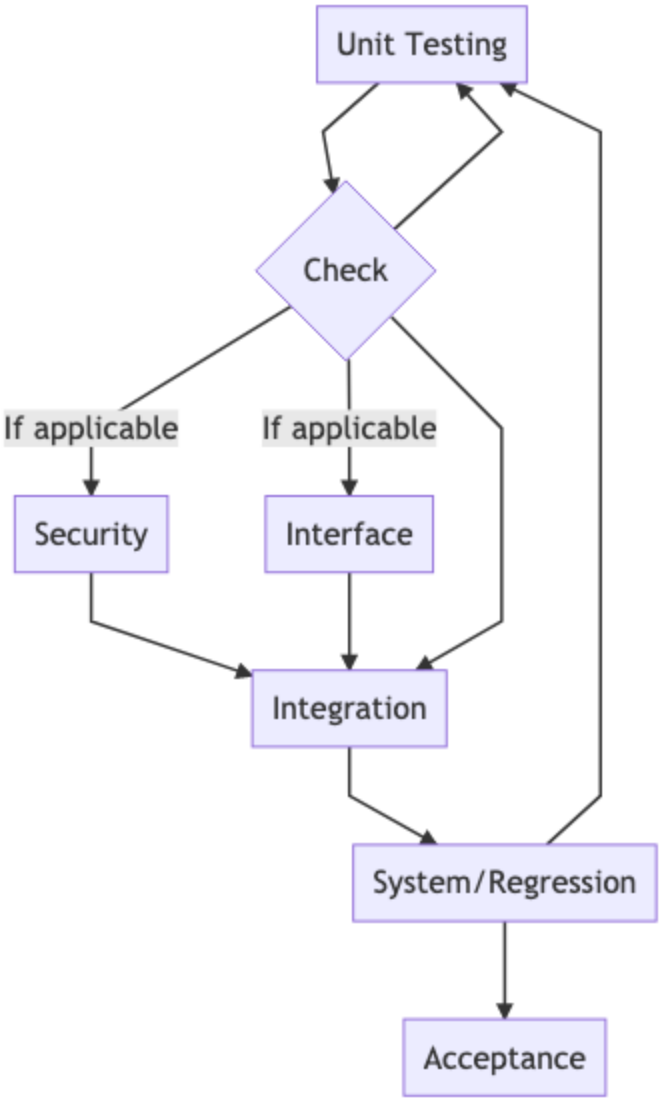
4.7 Security Testing

As we develop the application, along with the server side and client side components, it is important that we take into account the application of modern security procedures. As such, for components like the frontend web application, we will only send information that is explicitly needed. As we configure and set up the server side of the application, we will conduct a penetration test to ensure that access cannot be obtained beyond those who are authorised to have it. Additionally, prior to application publishing, removing any unnecessary accounts that may linger on the server and closing any ports that are unnecessarily open.

4.8 Results

At this time, the only results of existing testing is some basic smoke test results ensuring that the basic framework will correctly compile and launch.

The main method for ensuring compliance with the requirements of this project will be from the overall method of the project development - not just testing. This method is an agile methodology where a developer will take various tasks/issues per sprint that focus on a certain functionality that may be a full requirement, or a part of an overarching requirement. Once reaching the testing phase, compliance will be further ensured by having comprehensive testing that will be required to succeed for integration into the main branch of code. This testing will involve everything from checking validity of code to evaluating if the new code can correctly perform the specific task that it was intended to based on the requirement(s) that it was intended to address.



Shown above (*Figure 19: Testing Suite Procedure*), is a visual representation of the project's testing process. It begins with unit testing of the code that is being developed which will repeat as long as there is more functionality to be tested. Subsequently, once all unit tests have been completed and passed by the new code, the developer will check if either security or interface testing will be needed (as not all functionality will). After that has been completed and successfully passed, integration testing will occur. After successfully testing the integration of the new code it will be merged into the main branch where system and regression testing will occur in unison. At this point, if more functionality is to be added the cycle will repeat with the creation of the newer code for the remaining functionality. Otherwise the project will be complete and reach the acceptance testing phase. This will all occur with the agile development methodology meaning that the unit to system/regression testing phases will repeat many times, and at least once for each added functionality.

At this time, there is not a summary narrative concluding the usefulness of this testing design.

5 Professionalism

This section is with respect to the paper, “Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment”, *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012.

5.1 Areas of Responsibility

One of the codes of ethics, related to this project, (IEEE, ACM, SE) were chosen and then added onto the table provided in, “Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment”. A new column was added upon this table at the end with a brief description of how the selected code of ethics connected to each area of responsibility next to the National Society of Professional Engineers (NSPE) column.

The chosen code of ethics to base the evaluation off of was Software Engineering (SE) code of ethics, and the resulting table can be found below. The resulting description is based on the SE code of ethics for each of the seven professional responsibilities of the table.

The seven areas of professional responsibility in the assessment instruction with an additional column of the SE code of ethics outlined in “Computer Science and ACM Approve Software Engineering Code of Ethics”, *Computer Society Connection* pp.84-88, 1999.

Area of responsibility	**Definition**	**NSPE Canon**	**SE code of ethics**
Work of Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts.	Accept responsibility for one's own work while only approving software that is safe, meets requirements, passes appropriate tests without negative effects on quality of life.
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.	Ensure products, manufactured and modified, meet the highest of professional standards possible to ensure the utmost financial results.
Communication Honesty	Report work truthfully, without	Issue public statements only in an	One must accept responsibility for their

	deception, and are understandable to stakeholders.	objective and truthful manner; Avoid deceptive acts.	work while not knowingly working in an illegal or unethical manner.
Health, Safety, and Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	Approve software that won't diminish quality of life, harm the environment, and diminish quality of life while being fair and supportive to colleagues.
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	Keep confidential information of any client while ensuring proper documentation and evidence of nonproprietary or breach of property or ideas.
Sustainability	Protect the environment and natural resources locally and globally.		Do not purposefully accept software that will harm the environment.
Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.	End products should be of the highest professional standard with proper procedure in ethics in the development process with the hope of also participating in lifelong learning, and advance the integrity and reputation of the profession.

Table 6: Ethics Table Additions

The SE code of ethics differs from the NSPE code of ethics in each area slightly due to it being a code based around a more specific area of engineering than the broad area of NSPE. The following is grouped based on each of the seven professional responsibilities.

For the responsibility "Work of Competence", the difference between the two codes can be seen from the more broad perspective of NSPE as it is specifically concerned with engineers

performing tasks specific to their training while the SE code is specific to the work of software engineers.

For the responsibility “Financial Responsibility”, the difference comes from the SE code of ethics being not just financial responsibility toward employers or clients, but also general use of data and computer resources that may lead to undue financial burdens.

For the responsibility “Communication Honesty”, there is not a really substantial difference as the responsibility of communicating honestly to any party involved in any sort of project should be done without any deception or omission of facts for either code of ethics.

For the responsibility “Health, Safety, and Well-Being”, in a similar way to the communication honesty responsibility of not really being different in any real important means. It is simply a more specific statement for the SE code of ethics.

For the responsibility “Property Ownership”, is another responsibility that is not that different between the two codes as this responsibility is dealing with the respect towards one’s property and ideas which is consistent across any engineering project. The general conclusion is to properly act as a trustee of information for all those involved in the project.

For the responsibility “Sustainability”, the NSPE code of ethics was left blank in the original table of the seven responsibilities making this section a clear difference between the two codes. As for the SE code of ethics, the responsibility breaks down to any code that one develops should not actively have a negative impact on the environment.

For the responsibility “Social Responsibility”, is the final responsibility and is relatively similar between the NSPE code of ethics and the SE code of ethics as they both relate to the purpose of creating quality work at the highest level that will be done in a lawful manner while advancing the integrity of the profession.

5.2 Project Specific Professional Responsibility Areas

This section includes a brief explanation of the applicability and the degree to which the team has fulfilled each of the seven areas of professional responsibility that can be found defined above in the table of section *5.1 Areas of Responsibility*.

“Work of Competence”: As this means, “Perform work of high quality, integrity, timeliness, and professional competence”, this clearly applies to the team’s project in a professional context. This is because we want to be able to meet these attributes in the work that we put in. Our work should be of a high quality, while still getting completed in a timely manner with the professionalism that would be expected of us in a real work environment. The team, to this point, have been performing, in this professional responsibility, really well in getting work done to the level that would be expected while maintaining a competence towards the professionalism of how we do so. Giving a rating of either the high end of medium or just in the high degree of level.

“Financial Responsibility”: This responsibility applies to our project closely. The goal for the final product is to not only save the company time when working with underground cable bore holes, but also reduce unnecessary spending caused by disagreements between the company and its contractors. Our project will have this in mind, and maximize the time and financial benefits through deliberate design decisions. In addition we will also reduce the cost of operating and maintaining the tool that we create. Our team is performing highly in this category. We have created a design that fulfills the requirements described above and have considered the cost of long-term operation as well.

“Communication Honesty”: As we continue developing our project, we have strived to maintain full clarity on what we are doing and what our objectives are. Doing such keeps everybody involved well-informed on our decisions. This applies to this project because of the need to keep everyone informed between the team members, the advisors, clients, and teaching assistants. The team has successfully managed to maintain excellent communication throughout the entirety of the project planning and design process up to this point. This has been done with an instant chat communication method between team members, advisor, and teaching assistant, along with regular meetings. As well as communication with clients and other resources through school email with a designed format. As far as the honesty aspect, our team has been accurately portraying the project information truthfully to all stakeholders, team members, and everybody else involved .

“Health, Safety, and Well-Being”: Our application will be used to calculate the size of real-world underground bores and thus how much land will need to be moved to install the cables needed. By minimizing bore sizes, we reduce the impact on the environment albeit in a relatively small way. We must ensure that the calculations to determine bore size are correct and can be relayed to the user effectively so they can apply their estimates.

“Property Ownership”: While there is no physical property, information and ideas are highly relevant to the scope of our project. Alliant Energy has provided us with insights into their inner workings, and have trusted us with this. We believe that our team is performing at a high level to rate it. We have kept all information shared with us to ourselves, and have not distributed or shared it beyond our circle of which it is relevant. Additionally, we have met with representatives from their company in order to hear their ideas and tailor our project in order to accommodate, as their insight has been helpful in understanding the use cases. However, while we asked about NDA, they only recently got back to us and requested to have IP rights. We are still in conversation to determine if IP rights are what we want to agree to however.

“Sustainability”: We want to ensure that while our project will not produce any physical products, we can still be responsible for how many resources we use to host the application on various servers. While we do not have total control over how the servers themselves hosting this application will be run, we can ensure that we are not using high-power servers that would be better suited to heavy computing since our application should require minimal processing power. This does mean optimizing our code to only run calculations when necessary rather than all the time to reduce energy consumed by a server processor.

“Social Responsibility”: We want to ensure that our project is done to the best of our ability. Although we still have things to learn in regards to the technology and methodology we chose for our implementation, we will do it to the best degree possible. We will not be dealing with confidential or proprietary information, as this is a publicly and freely available tool, available to anyone who wishes to use it. However with that being said, we will be restricting who can update the publicly available profiles via an organization admin account to trusted individuals at said user companies. While this project is not yet complete, we hope the end result will be of high quality and provide a positive example of the software and computer engineering professions.

5.3 Most Applicable Professional Responsibility Area

For this section, one area of professional responsibility that is both important to this project and the team has demonstrated a high level of proficiency in the context of this project. This includes a description of how this responsibility is important to this project, the ways in which the team has demonstrated the responsibility in the project, and the impacts that it has led to for the project.

While there are multiple professional responsibilities areas that the team has performed really well thus far, one of the most applicable areas would be that of Communication Honesty. The team has a strong communication system setup for all of the types of communication that needs to be done. To add to the area of honesty, the information that has been communicated has been truthful without the intent to deceive, and if there were any uncertainty or miscommunication the team members would work to remedy the situation.

This has led to the transparency of the work that the team has been doing, and the effectiveness of the exchange of information has allowed the team to fully figure out project requirements, scheduling, and all of the necessary design decisions. There have been difficulties that have occurred during the project that have led to the need for more in depth communication or extra meetings between appropriate parties. The team’s communication system and communication honesty has helped to facilitate the alleviation of these difficulties.

6 Implementation

This section will be completed as the project implementation phase gets to a stage requiring documentation beyond software documentation.

7 Closing Material

7.1 Discussion

As of the date that this document was last updated [11/29/2021], there are not any lasting results of this project as most of what has been accomplished is planning and setup for implementation that will be occurring over the first few months of the year 2022.

7.2 Conclusion

With the content of the previous section (7.1 Discussion) in mind, what has been completed has been a project, and design plan (seen in this document). Various technological setup has also occurred, including: team based GitLab, an ISU server for hosting the application and development environments for the team, and communication channels between team members, advisors, and clients.

Most of the results of this project up to this point is reflected in this document, the team's webpage, the team's private google drive for documents, and the team's GitLab repository (with exceptions of software development and document creation that did not end up shared to one of these points of storage).

The overarching goal of this project is to create a web application that will be hosted by Iowa State University that will function as an improved implementation of the existing executable program, and what has been accomplished to this point has been largely set up for the eventual implementation of the project.

One goal that has already been achieved is working with ISU faculty to secure a server for hosting the project during and after development. Those of the appropriate role are in the process of setting it up in order to allow for initial development to begin which has been slightly constrained due to a slow moving back-and-forth process between the team and faculty. Plans to improve this is simply to maintain close communication and increase time allotment to tasks such as this.

7.3 References

"Computer Society and ACM Approve Software Engineering Code of Ethics", *Computer Society Connection* Vol. 32, Issue: 10, pp. 84-88, 1999.

"Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012.

IEEE. *IEEE-IEEE Code of Ethics*. IEEE.org. Retrieved November 7, 2021 from https://www.ieee.org/about/corporate/governance/p7-8.html

7.4 Appendices

This section includes any additional information that was deemed necessary or helpful to the evaluation of the design components of this document.

7.5 Team Contract

Team Name: sdmay22-19, Underground Cable Packing Web Tool

Team Members:

1) Alexander Young
2) Brevin Wapp
3) Haadi Majeed
4) Matthew Hoskins
5) Nate Tucker
6) Tom (Jidong) Sun
7) Quinten Sorice

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:

    - Monday 6:00 PM - Advisor meeting - Webex
    - Friday 5:00 PM - TA meeting - Virtual through Discord or Zoom (TA's choice)
    - Friday 5:30-6:30 PM - Team Meeting - In-Person (lLibrary or Senior Design Lab) or Virtual
    - Sunday 3:00-4:00 PM - Team Meeting - Virtual
    - Additional meetings can be scheduled as needed through proper communication channels

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

    - Discord server (Senior Design 19 (sdmay22-19)) for text updates
    - Google calendar for keeping track of due dates and meeting times/locations

3. Decision-making policy (e.g., consensus, majority vote):

    - Majority vote for decisions that impact the team or project

4. Procedures for record-keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

- The team will store meeting minutes on discord's #document channel and Google Drive

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

    - All members should attend scheduled meetings
        - Within 10-15 minutes
        - Communicate with the team if any emergencies
    - Give advance notice as early as possible for the team to accommodate.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

    - Do the thing you need to do before agreed deadlines
    - Team-based work should be completed with that team (or as much of that team as possible)

3. Expected level of communication with other team members:

    - Email messages should be responded to within: 24 hour
    - Discord text messages should be responded to within: Same-day
    - If unavailable for extended periods: provide advance notice

4. Expected level of commitment to team decisions and tasks:

    - Expectation to deliver A-level work and time commitment (school/academic terms)

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

    Alexander Young: DevOps and Systems Engineer

    Brevin Wapp: Scrum Master

    Haadi Majeed: QA Engineer

    Matthew Hoskins: Leader/Time Management

    Nate Tucker: Tech Lead

    Tom Sun: User Experience & Requirements

    Quinten Sorice: Client Point of Contact

2. Strategies for supporting and guiding the work of all team members:

    - Due dates for any assignment will be available for viewing on shared calendar
    - Due dates will be arranged with the contributors of the specific assignment to ensure it is effective and attainable.
    - Discussions revolving around any issues, or concerns can be brought during meetings or over an appropriate Discord channel
    - E-mail communications between advisor and TA for important or complex questions

3. Strategies for recognizing the contributions of all team members:

    - Weekly update meetings
    - Gitlab code impact scores / commit contributions

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

    - Matthew: Front-end web development with JavaScript and a little typescript, High level programming language experience in Python and C's (potentially related languages to project development), UI/UX project work, Algorithmic math, Time management capabilities, some MySQL and SQL experience, OOP programming.

    - Tom: Front-end web development with typescript, Back-end development with Spring. AWS (ECS, Lambda), Gitlab & Jenkins CD/CD. UI mockup with figma and UX testing. PostgresQL (and general SQL). Git

    - Quinten: Front-End development through Javascript and Adobe Experience Manager, backend through Spring/SpringBoot, OOP with Java, general C, Docker deployment

    - Haadi: Front End development web dev with JS and TS, back end with C# / .NET and Node JS, databasing via SQL, MSSQL, Postgres.

    - Nate: Front End development with React JS in JS in TS. Back end development with Node JS, Django, ASP.NET Core, Springboot. AWS Serverless deployments with Amplify, API Gateway, Lambda, and Dynamo DB. Unit testing with react testing library and junit, Fluent Assertions in C#.

    - Alex: Front End development with React JS Old in ES6 and TS. Backend development with ASP.NET Core, Go, Ruby, and with Azure Technology Fleet. Systems administration of Windows and Linux environments. Automation experience with PowerShell, Ruby,

Puppet Labs Puppet, Red Hat Ansible. SQL Experience with Microsoft SQL Server, and PostgreSQL.

- Brevin: Front End development with React JS in JS, TS. Backend development with .NET Entity Framework Core (C#), Spring Boot (Java), C, C++, Clojure, SQL via MYSQL, PostgreSQL, Bash.

2. Strategies for encouraging and support contributions and ideas from all team members:

- Allow each person to explain/talk uninterrupted and then give thoughts/feedback
- Brainstorming sessions for generating ideas
- “Help Hours” where team members can come together to ask questions

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

- As for identifying:
  - Repeated disagreements
  - Difficulty identifying a correct/best course of action, or decision
  - Failure to produce work without visible effort or seeking help
  - Lack of communication
- Resolving:
  - Take it to a team vote if needed
  - Defer to Project Manager
  - Have conflicting parties meet with TA to settle the dispute if unable to resolve internally

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

- Create a web application design and algorithm design
- Potential proof-of-concept mockup
- Create an implementation plan
- Work as a team to produce results
- Create requirements for clients and implementation
- Finish most dev-ops tasks (setting up project manager board, CI/CD, infrastructure access, etc.)

2. Strategies for planning and assigning individual and teamwork:

- GitLab built-in board for ticket/task management
- Whole-Team or stack-based team meetings for ticket assignment

3. Strategies for keeping on task:

    - Weekly update meetings, sharing progress and roadblocks
    - Adding time estimates to issues
    - Ask for help if needed, both internally and externally
    - Stay on-topic during team meetings, having meeting agendas

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

    - Acknowledge the problem, be it personally or apparent team problems
        - Address that problem using team management skills

2. What will your team do if the infractions continue?

    - Consult Senior Design administration

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*

b) *I understand that I am obligated to abide by these terms and conditions.*

c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Alexander Young DATE September 18, 2021

2) Brevin Wapp DATE September 18, 2021

3) Haadi Majeed DATE September 18, 2021

4) Matthew Hoskins DATE September 18, 2021

5) Nathan Tucker DATE September 18, 2021

6) Jidong Sun DATE September 18, 2021

7) Quinten Sorice DATE September 18, 2021