

Underground Cable Packing Web Tool

Final Report and Design Document

Iowa State University CPRE/CYBE/EE/SE 491+492
Fall 2021 - Spring 2022
Team Number 19

Professor Mathew Wymore - Faculty Advisor
ISU's Electrical Power Research Center - Client
Alliant Energy - Prospective User

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/>
Web Tool Link: <https://ucp-webtool.ece.iastate.edu/>

Revised 4/28/2022
Version: 1.4.2

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 Institute 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 EPRC 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
- COMS 363: Database Management
- SE 409: Software Requirements Engineering
- SE 417: Software Testing
- CPRE 421: Software Analysis and Verification for Safety and Security

Knowledge Acquired

The curriculum leading up to this class has covered a majority of the skills and technologies required to successfully develop this application: a frontend connecting to a server backend with an algorithm as the primary component (COMS 309, COMS 311, SE 319, et al.), proper team project documentation (COMS 309, ENGL 314, SE 329, etc.), project planning and management (SE 329), and team member, advisor, client interaction, and others interaction on a professional level (SPCM 212, etc.). The only areas of knowledge not covered by course curriculum leading up to this project and thus areas to gain the most experience in are specifics of the programming languages not explicitly covered in class (GoLang), high-level code sharing and development through GitLab, and in-depth algorithm research and development.

Table of Contents

Executive Summary	2
Development Standards and Practices	2
Summary of Requirements	2
Applicable Courses from University Curriculum	3
Knowledge Acquired	3
Table of Contents	4
List of Figures, Tables, Symbols, and Definitions	6
List of Tables	6
List of Figures	6
0 Team Section	7
0.1 List of Members and Roles	7
0.2 Required Skill Sets for Project	7
0.3 Skill Set Covered by Team	8
0.3.1 Computer Engineering Majors	8
0.3.2 Software Engineering Majors	9
1 Requirements Section	10
1.1 Problem Statement	10
1.2 Requirements & Constraints	10
1.3 Engineering Standards	12
1.4 Intended Users and Uses	12
1.5 Design Evolution Since CPRE/SE 491	13
1.5.1 Basic Functionality Change	13
1.5.2 Other Design Changes	13
2 Project Plan	14
2.1 Project Management/Tracking Procedures	14
2.1.1 Management Style and Justification	14
2.1.2 Progress Tracking and Management Tools	15
2.2 Task Decomposition	15
2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria	19
2.4 Project Timeline/Schedule	20
2.5 Risks And Risk Management/Mitigation	24
2.6 Personnel Effort Requirements	27
2.7 Other Resource Requirements	29
3 Design	31
3.1 Design Context	31
3.1.1 Broader Context	31

3.1.2 User Needs	32
3.1.3 Prior Work/Solutions	33
3.1.4 Technical Complexity	33
3.2 Design Exploration	34
3.2.1 Design Decisions	34
3.2.2 Ideation	35
3.2.3 Decision-Making and Trade-Off	36
3.3 Proposed Design	37
3.3.1 Design Visual and Description	37
3.3.2 Functionality	44
3.3.3 Areas of Concern and Development	44
3.3.4 Technology, Frameworks, and Libraries	44
4 Testing	46
4.1 Unit Testing	46
4.2 Interface Testing	46
4.3 Integration Testing	47
4.4 System Testing	47
4.5 Regression Testing	48
4.6 Acceptance Testing	49
4.7 Security Testing	49
4.8 Results	49
4.8.1 Unit Testing Results	50
4.8.2 Interface and Security Testing Results	51
4.8.3 Integration Testing Results	51
4.8.4 System and Regression Testing Results	52
4.8.5 Acceptance Testing Results	52
5 Professionalism	53
5.1 Areas of Responsibility	53
5.2 Project Specific Professional Responsibility Areas	55
5.3 Most Applicable Professional Responsibility Area	57
6 Implementation	58
6.1 Development Process Abstraction	58
6.2 Web Tool Results	58
7 Closing Material	63
7.1 Discussion	63
7.2 Conclusion	63
7.3 References	64
8 Appendices	65
8.1 Appendix I. [Operation Manual]	65

8.2 Appendix II. [Alternative Design Versions]	66
8.3 Appendix III. [Other Considerations]	66

List of Figures, Tables, Symbols, and Definitions

List of Tables

Table 1: Team Member and Role List	7
Table 2: Task Decomposition	15-19
Table 3: Risk Management and Mitigation Numeric Definition	26
Table 4: Personal Effort Breakdown	27-29
Table 5: Project Area Considerations	31-32
Table 6: Ethics Table Additions	53-54

List of Figures

Figure 1: Basic Agile Development Visualization	14
Figure 2: Gantt Chart by Major Task View	20
Figure 3: 1.0 Project Planning & Defining	21
Figure 4: 2.0 DevOps & Tech. Setup	21
Figure 5: 3.0 Software Design & Functional Design Verification	22
Figure 6: 4.0 Redesign Algorithm	22
Figure 7: 5.0 Data Tables Setup	22
Figure 8: 6.0 Backend Construction	23
Figure 9: 7.0 UI Construction	23
Figure 10: 8.0 Development Testing Suite	23
Figure 11: 9.0 UAT & Deployment	24
Figure 12: Design Approach	34
Figure 13: API & User Interaction Diagram	36
Figure 14: API Diagram	37
Figure 15: Software Architecture Design	38
Figure 16: Overlay for Cable Setup Mock-up	39
Figure 17: Cable Input Selection Mock-up	40
Figure 18: Cable Input Selection Mock-up (Filled Out)	41
Figure 19: Results Page Mock-up	42
Figure 20: Testing Suite Procedure	49
Figure 21: Landing Page	57
Figure 22: Presets Selection Demonstration	58
Figure 23: Input Process	58
Figure 24: Mobile UI	59
Figure 25: Results Imaging	60
Figure 26: Help Page Image	61
Figure 27: Account Page Image	61

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

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

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

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

Many companies with the need to install cabling on a commercial scale are shifting more towards the method of boring underground, so as to better mitigate environmental hazards and other risks associated with above-ground cable routing. Thus, the need for software to streamline these efforts is increasing ever more.

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 desktop 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 provide more readily available functionality through the application hosted on an EPRC website

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 a user interaction standpoint, include the assessment of proper bore sizing, project-based material calculation, streamlined calculation process, 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**)
- Must be a web application frontend must run on common browsers, including desktop Chrome, Firefox, Safari and Edge (as time allows)
- 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 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
 - Final outer bore size provided (Diameter or Radius)
 - Stretch goal: formally prove algorithm is correct
- Configurable:
 - Set of predefined cables/ducts
 - 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 browsers (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 International Organization for Standardization (ISO), International Electrotechnical Commission (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.

1.5 Design Evolution Since CPRE/SE 491

It is important to note that the core components of the project have not undergone a significant change, and that all of the items in the section are either small changes that did not affect the main function or had a small change in the output seen by the user. In fact, most of the overarching design has remained the same and, because of this, this document is largely the same as its predecessor, the final design document of CPRE/SE 491 for this team.

1.5.1 Basic Functionality Change

An important change decision, which was made in conjunction with Professor Wymore, is that the final output of the main function, the bore generation and calculation, would be a single visual representation with a numerical aid instead of a pair of graphs.

The software that this project is based on generated two visual outputs. The first, being a “valid” bore generation where it was the visual representation of the smallest bore size in a given range that could fit all of the user's inputs. The second, being an “invalid” bore generation where it was the visual representation of the next smallest bore size from the first output, and would only hold so many of the user's inputs inside of it. This was to show that the “valid” solution was the smallest bore in a given range that could hold all of the user's inputted values.

The change that this project has made from that, as well as the original output plan (being the same as the software this project is based on), is that the output will instead be the absolute smallest bore that the algorithm could generate without regard to an incrementation of the bore sizes. It will generate a single visual representation of the output with a bore perfectly fit to be the smallest given the input. The bore size is then also given as an output numerical value next to the graph for use by the user.

This change was made with the idea that it would simplify the inputs that the user has to provide (not having to provide bore increments), and it would simplify the user of the results as it would be a single absolute graph with a marking on the smallest the bore could be. This will then allow for user interpretation of the results on what bore would ultimately be chosen based on the more clear and easy to understand results.

1.5.2 Other Design Changes

A low-impact change that was made to the design was the removal of the option to email bore results once the calculations are completed. Users can still opt to have their results emailed to them after the algorithm has run and determined a bore, but we decided to remove the option to let the user wait for the algorithm to run in the background. The algorithm developed is much faster than originally anticipated and works with an expected runtime of under one second. This speed means that the usefulness of receiving results at the moment they finish, from a user perspective, is severely diminished since the wait time between start and finish of the calculation is negligible.

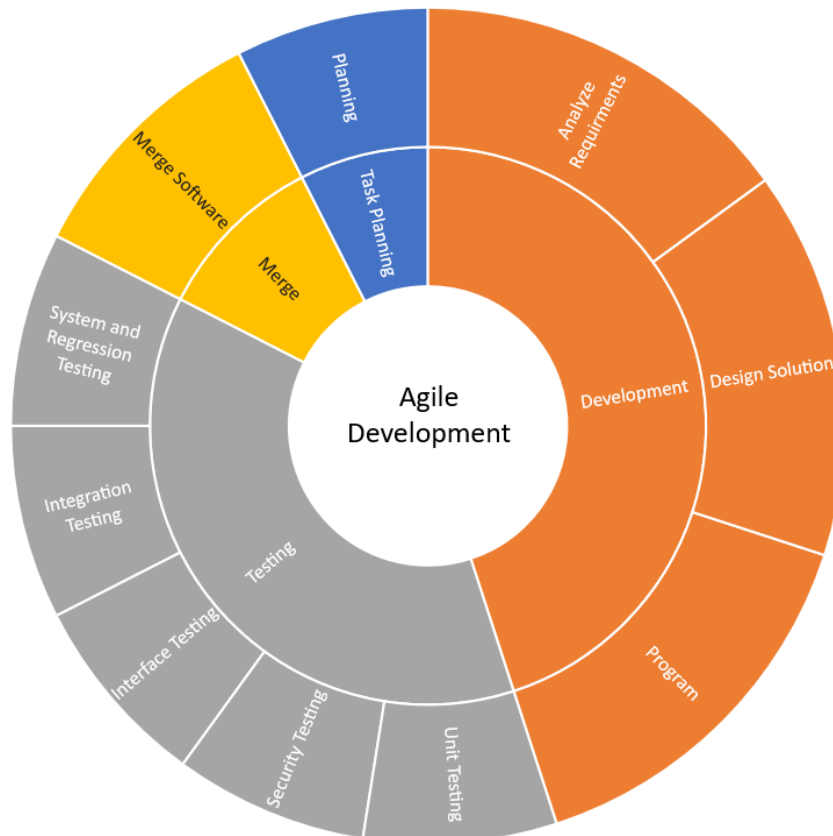
2 Project Plan

2.1 Project Management/Tracking Procedures

2.1.1 Management Style and Justification

Being completely a software development endeavor, 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 has been 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 has been used for other produced materials - primarily along the lines of documents and presentations.

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

As for communication methods: first, a structured Discord server has been used for immediate communication between team members, teaching assistant (TA) Jacob Conn, and Professor Mathew Wymore. Second, project group email (through school email) has been 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 has been 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: 1.0.0

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, subtasks, 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).

Underground Cable Package Management Web Tool - Team 19 (SDMAY22-19)

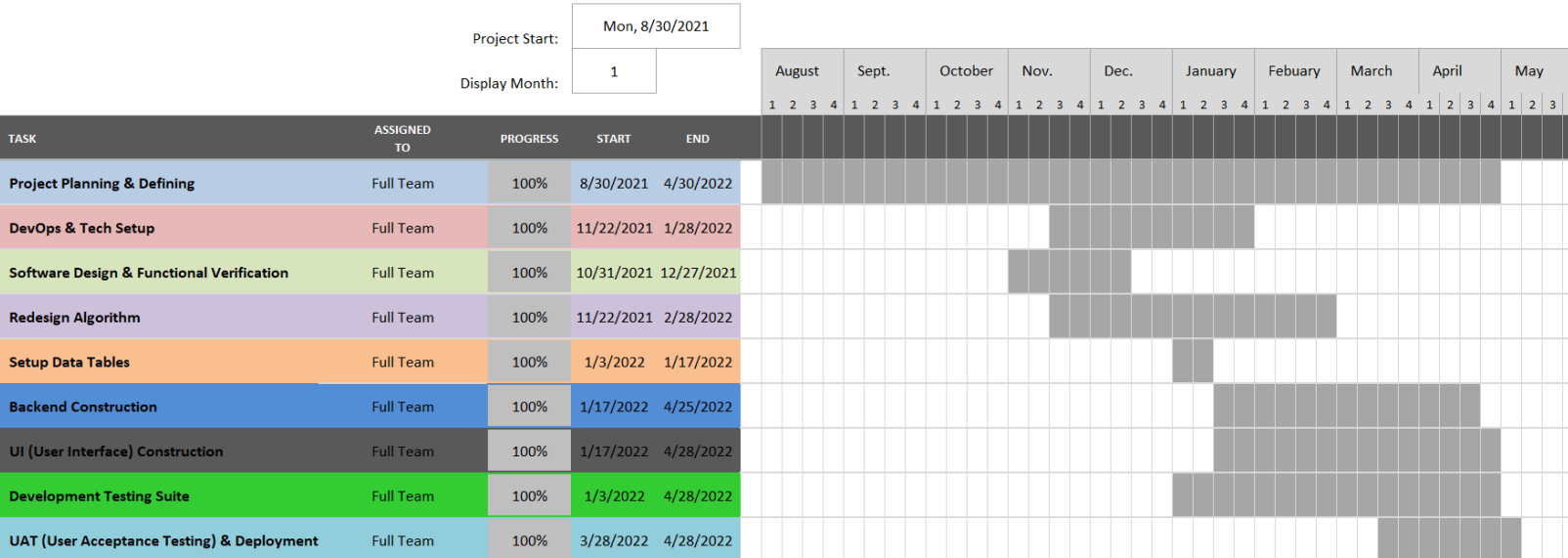
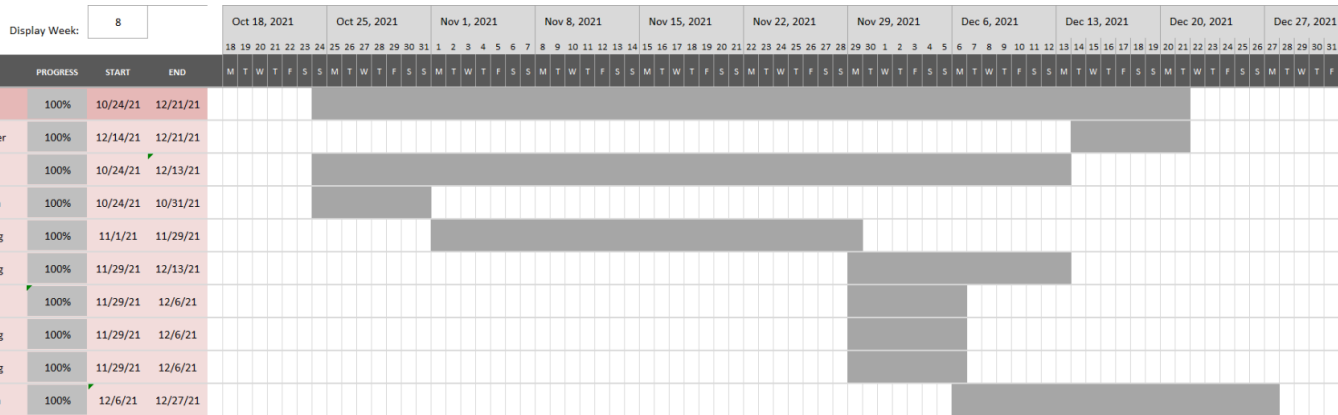


Figure 2: Gantt Chart by Major Task View

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.

					Apr 25, 2022							May 2, 2022							May 9, 2022						
					25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13		
TASK	ASSIGNED TO	PROGRESS	START	END	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F		
9.0 UAT (User Acceptance Testing) & Deployment		100%	4/28/22	4/28/22																					
9.1 Internal Acceptance Testing	Full Team	100%	4/28/22	5/9/22																					
9.2 Demo and Testing with Clients	Full Team	100%	5/9/22	4/28/22																					
9.2.1 Demo with Professor Wymore	Full Team	100%	4/28/22	4/28/22																					
9.2.2 Demo with Alliant	Full Team	100%	4/28/22	4/28/22																					
9.3 Final Production Deployment	Full Team	100%	4/28/22	4/28/22																					

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 V	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, hours required, and a brief explanation of the time requirements.

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	28	4 hours per-person, since some

Environment		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 a more easily-accessed software tool for determining the underground tunnel bore size given a set of cables that need to fit into it. 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 bore 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 through 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, thus limiting us to purely software-based decisions. It should be noted that the application is hosted on an ISU server, and decisions related to said server have been made together with standard implementation and technology in mind.

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 EPRC 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.

Further elaboration on potential and unimplemented design choices can be found in [Appendix ii](#).

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.

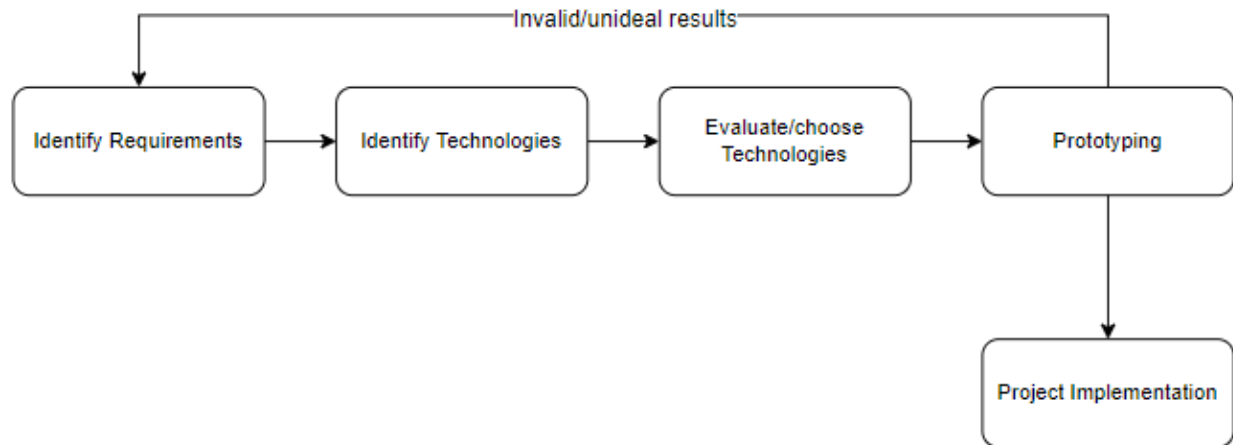


Figure 12: Design Approach

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.

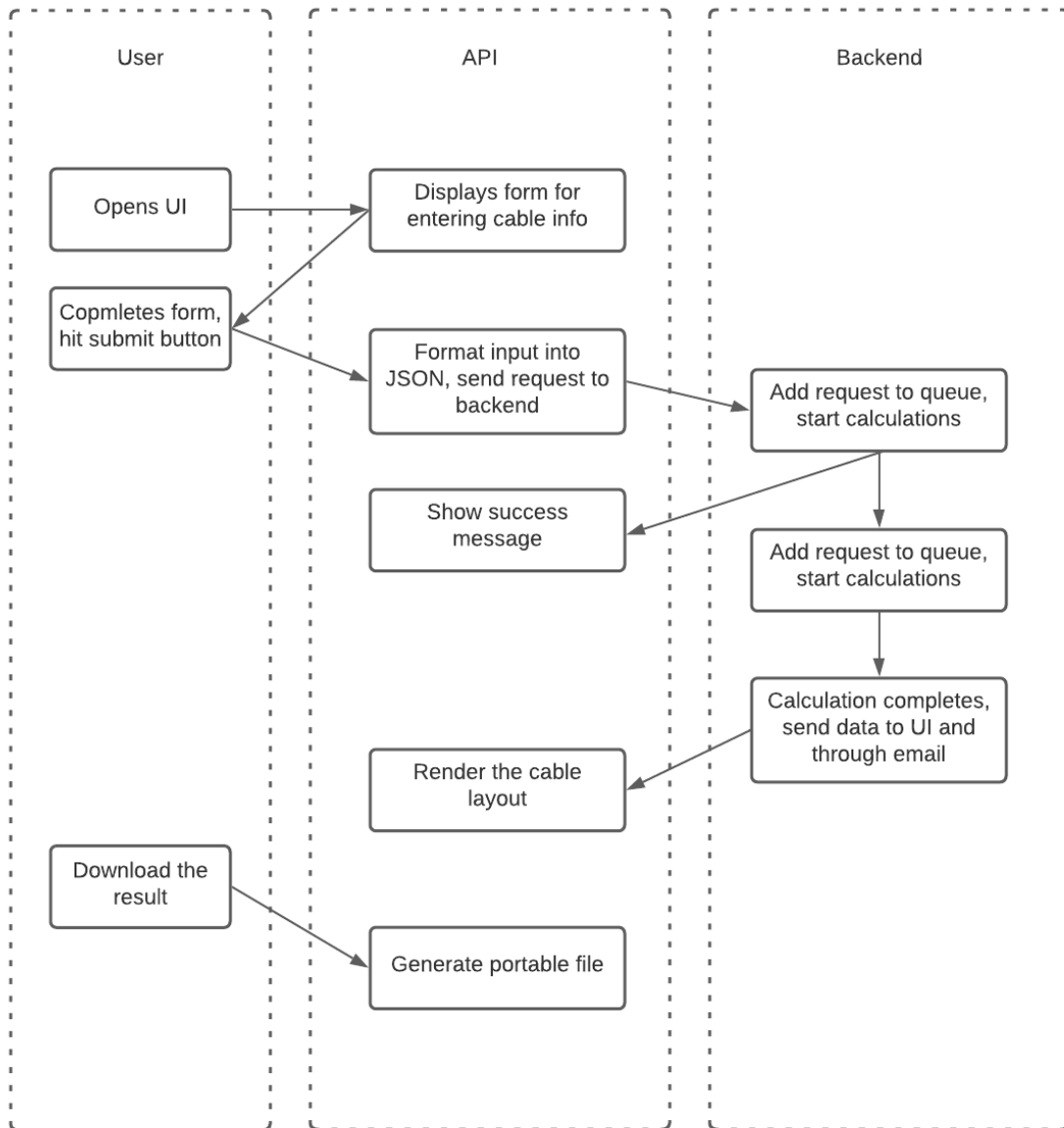
A final decision was our requirements for testing, internationalization, and accessibility. React has some very simple to use and extensible options for all of these requirements and presents the best options for a good user experience.

3.3 Proposed Design

The following includes designs for the primary implementation plan. These designs were created with the requirements of the project in mind as well as other general usability standards and visually appealing frontend plans. These designs were used as a guide for the implementation of the project (see [6. Implementation](#)), and as such there are slight alterations to the final design.

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*)



API Diagram: The following diagram shows the API and how it processes requests from the frontend. Another version of how the API fits in the backend see Software Architecture Diagram on the next page.

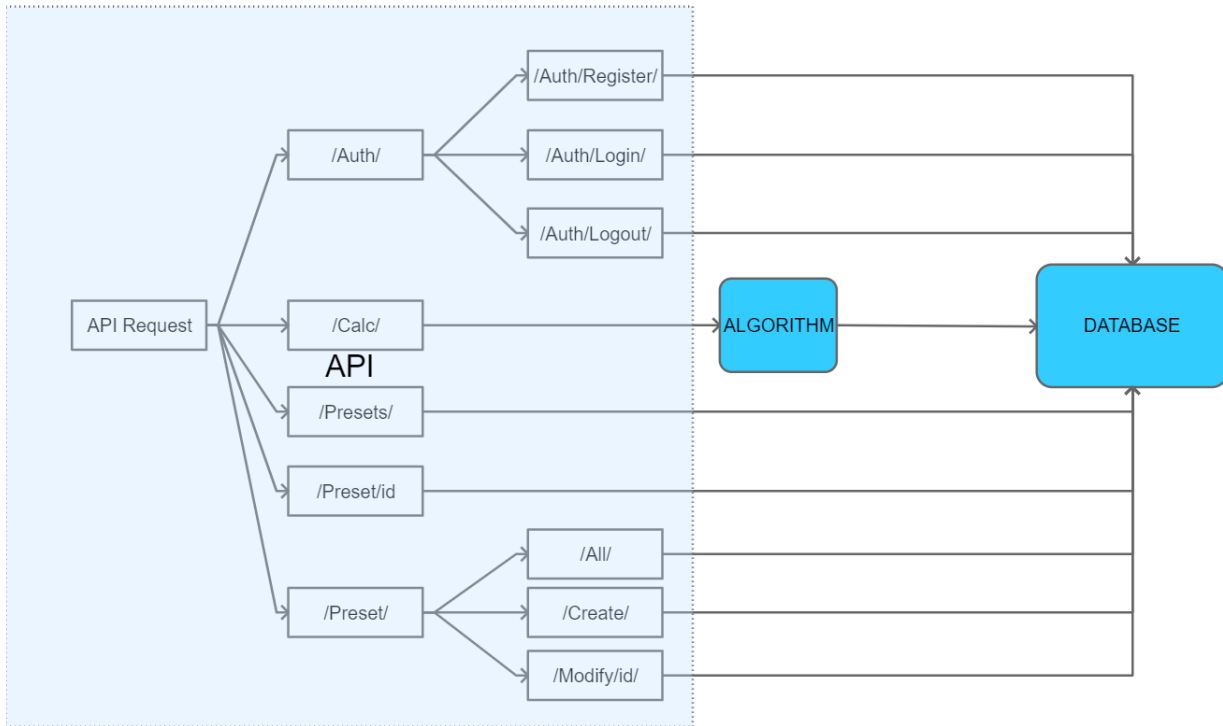


Figure 14: API Diagram

The Auth section of the requests pertain to the account that a user can make/have for the purpose of creating their own list of presets that would then be usable by other users as well as themselves. These requests are for registering an account, logging into an account, and signing out of an account. These requests will then connect to the database for the appropriate information to read or write.

The Calc section of the requests will call the algorithm of the code which in it has code that will access the database.

The Presets (with an s) is for accessing any made preset list of cables, so any user can access those lists. This gets the information off of the database with read permission, and nothing else.

The Preset/id section is for accessing any made preset list id, so that a drop down list can be viewed by a non-authenticated user by reading the information off of the database.

The Preset section is for users that can create/modify and use a preset list of inputs that will be written to the database and read off of it.

Software Architecture Diagram: The following diagram (Figure 15) 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.

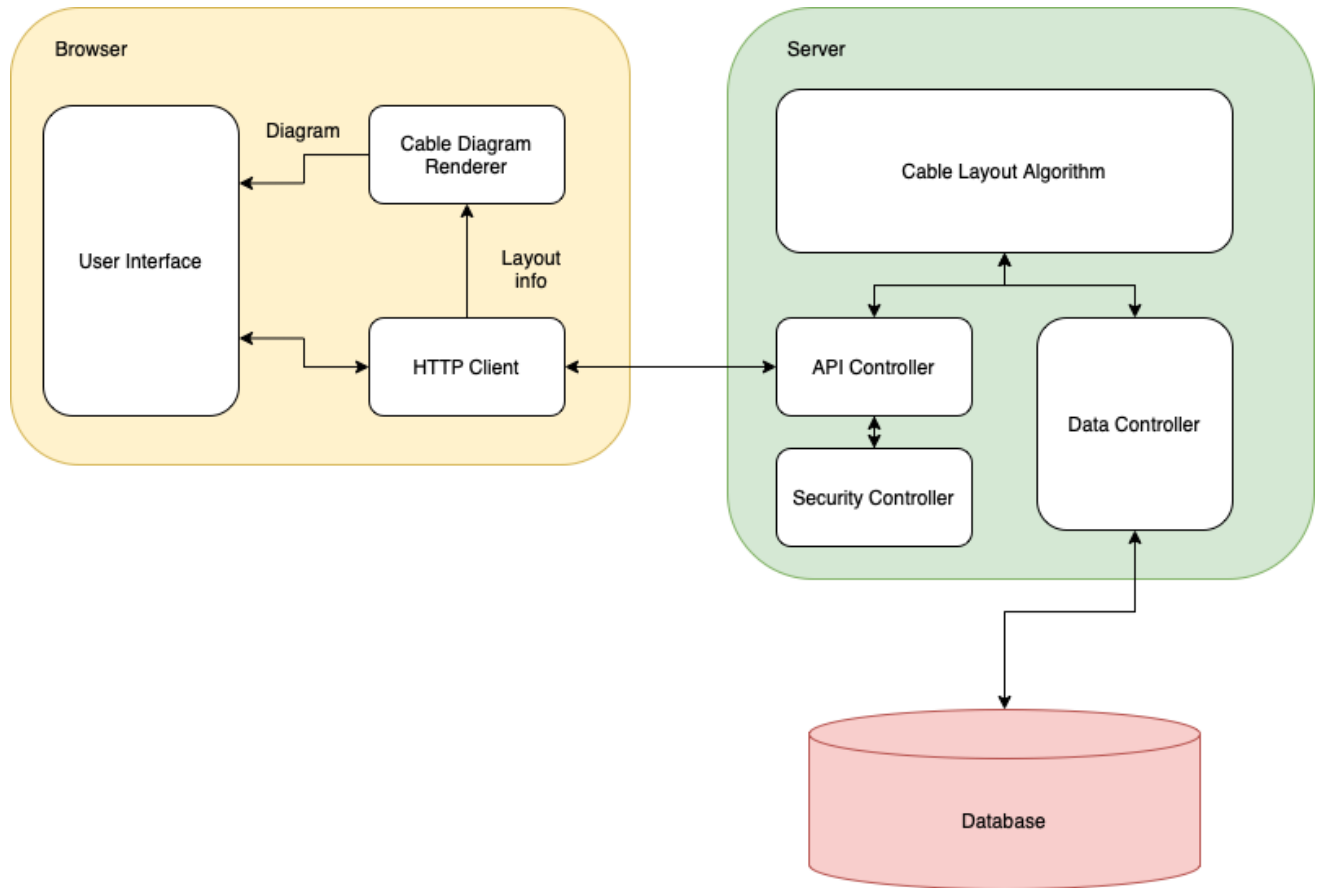


Figure 15: Software Architecture Design

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 for user presets.

UI Mock-Ups:

The image displays two UI mock-ups side-by-side. The left mock-up is a login screen for the Electric Power Research Center (EPRC). It features a red banner with the EPRC logo, a prompt to log in, and input fields for 'Username' (containing 'EPRC') and 'Password'. Below these are 'Sign In' and 'Forgot Password?' buttons. The right mock-up is the 'Cable Type Editor' overlay. It has a 'Profile Name' field with 'Alliant Energy - 2021'. A 'Cable Types' section includes a toggle for 'mm/inch' and an '+ Add' button. A table lists cable types and their diameters:

Cable Type	Diameter
15 kV, EPR Insulated	0.75
15 kV, Triple Insulated	0.80
600 V, Street Light	0.61
Duct, 2"	2.00

At the bottom of the editor are 'Save' and 'Cancel' buttons.

Figure 16: Overlay for Login screen and Cable Setup Mock-up

This (Figure 16) 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.

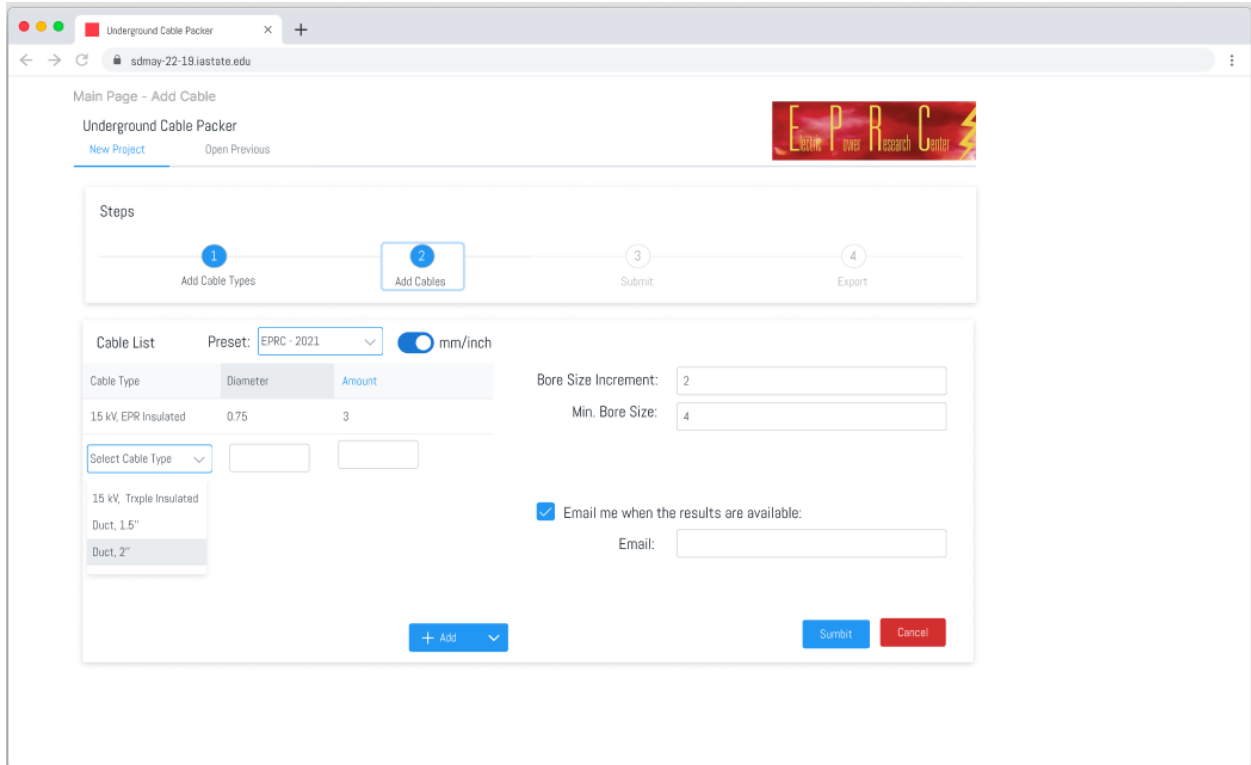


Figure 17: Cable Input Selection Mock-up

This (Figure 17) 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.

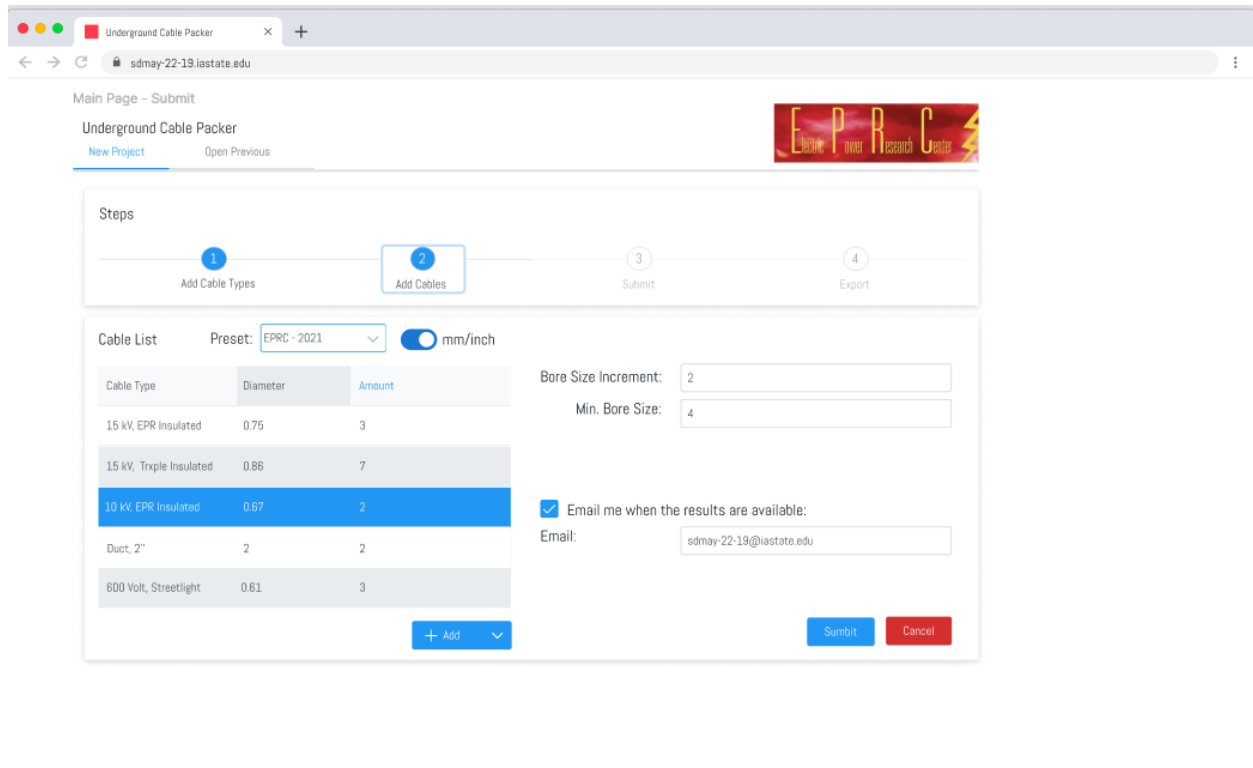


Figure 18: Cable Input Selection Mock-up (Filled out)

This (Figure 18) shows a completely filled out query still on the same page of the web application as Figure 17. 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.

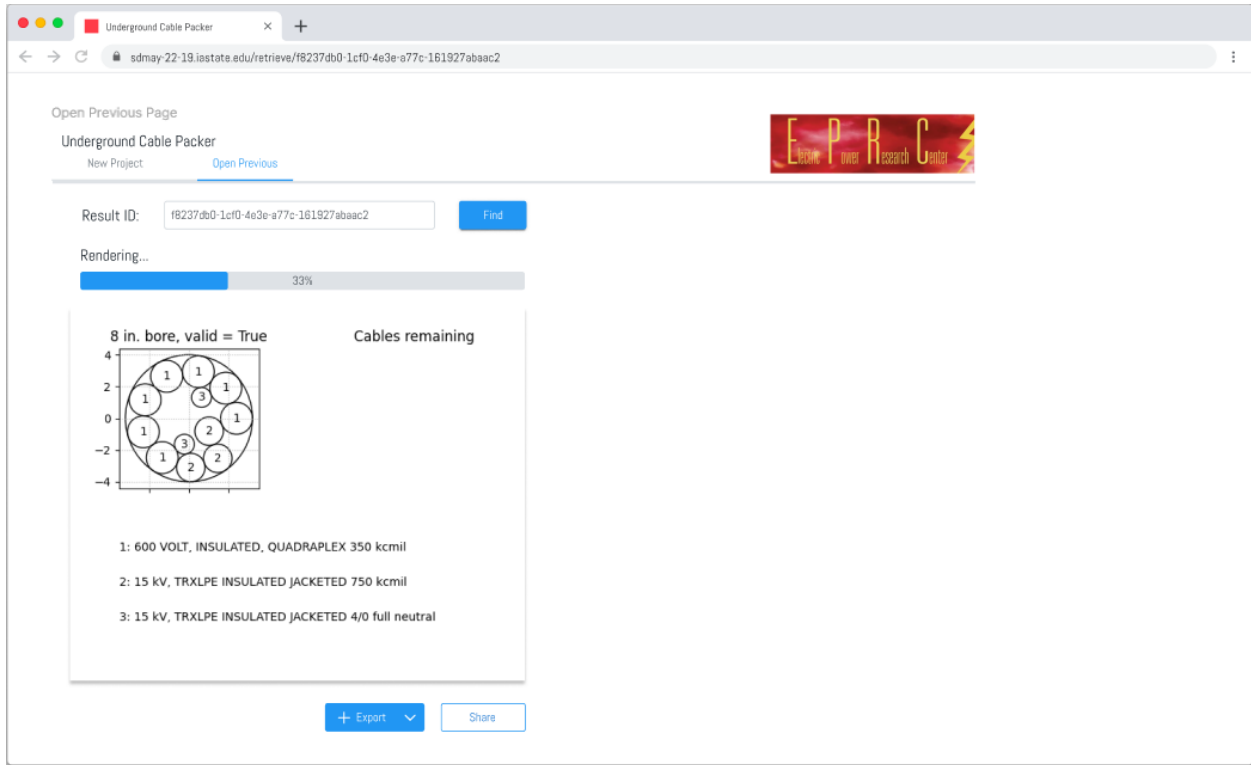


Figure 19: Results Page Mock-up

This (Figure 19) 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 realized 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 authorized 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

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.

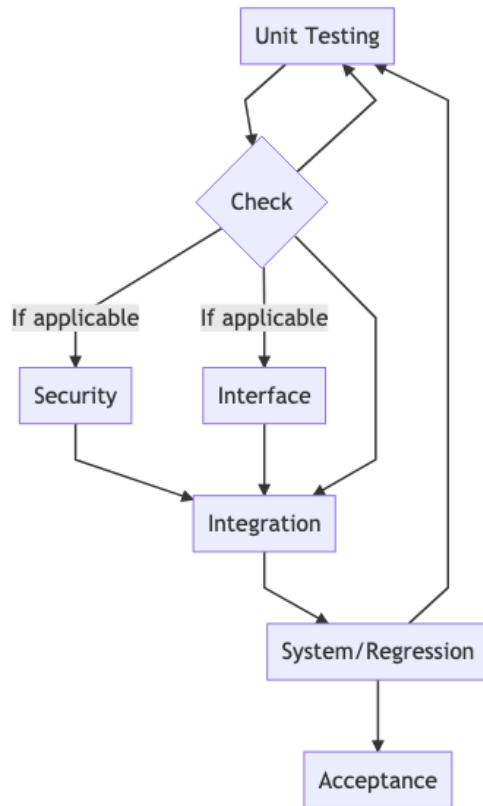


Figure 20: Testing Suite Procedure

Shown above (Figure 20: 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.

4.8.1 Unit Testing Results

The unit tests were incorporated into the CI/CD process, meaning that for any push of code it was standard to have all written tests to pass with any code changes that may have occurred. This created a realistic development environment, and led to all unit tests being passed tests.

Subsequently, the results of the unit tests aided in rooting out any small problems that may have existed in developed code, and was a large focus on having good tests with a good coverage of

functional code. The unit tests were key in rooting out code errors that may have been overlooked otherwise.

The backend algorithm code specifically had a multitude of unit tests, for testing logic and mathematical functionality, that reduced the chance of having an error or fault that would be difficult to detect when running through the algorithm process as a whole.

The total runed tests are many and varied to include specifics in this document, but they can all be found in the code base of the project.

For any additional information see ([4.1 Unit Testing](#)).

4.8.2 Interface and Security Testing Results

Interface testing as well as security testing are important parts of software testing, and from the develop strategy used in this project, are on the same stage of process. These tests both happen on the local side first prior to integrating new code into the project, and only under applicable circumstances.

Interface testing was done in a variety of methods as it pertains to API interfacing and UI. To test the API automated tests were written as well as some basic usability testing via using the frontend were used. The results aided in identifying differences in schema between the JSON data being interchanged between the frontend and the backend. The UI testing took on the form of some basic unit tests for frontend components that the user interacts with, and testing done by a developer simulating the various frontend processes. The results allowed the finer details to be corrected and improved upon which made the testing extremely helpful for improving the user experience.

As for security testing, when applicable being anytime that code was written that would send information back to the backend. Aside from keeping the software being used up-to-date, the means to test for security was to limit the interchange of data to only what was needed. When it was needed tests were made/taken in order to identify the potential weaknesses that would need to be addressed by the code in some manner. The results of not identifying major security risks was expected, but beneficial to have tests ensuring that the code was safe.

For any additional information see ([4.2 Interface Testing](#), [4.7 Security Testing](#)).

4.8.3 Integration Testing Results

For the integration between the frontend and the backend as well as the code and the hosting system, integration testing. This testing was set up completely to be done automatically via the CI/CD pipeline, in order to properly ensure that any code being pushed can integrate correctly with either the frontend or backend via the api or from the code base to the hosting server.

The results of the integration testing, as many automated pipelined tests, went relatively unnoticed most of the time. It was when there was a problem that a push would finish the

pipeline with an indicator of a failure that would contain information on what failed. This allowed for the team to easily identify and remedy any error found in the code.

For any additional information see ([4.3 Integration Testing](#)).

4.8.4 System and Regression Testing Results

System and regression testing results are combined because the way this project handled them was a sort of combined form where they occurred with the CI/CD process when a merge occurred. Meaning, that whenever a developer had gone through the proper process for getting their newly developed code merged into another branch or main, automatic testing setup in advance would occur.

The process for getting a merge completed is: developing new code for an issue, testing that code, pushing it to a specific branch, requesting a merge, getting approval for the merge, initiating that merge, and then the merge process with the CI/CD testing that would include the system and regression testing. This ensured that there would not be an issue with newly merged code causing a problem with the system, or negatively affecting the pre-existing code functionality.

The results of this testing were largely successful, without the need for correction - as the code was already extensively tested before reaching this stage in the development. That said, development was not without the occasional problem that showed code unexpectedly conflicted with the existing code, or the detection of code that could have had negative effects on the system.

For any additional information see ([4.4 System Testing](#), [4.5 Regression Testing](#)).

4.8.5 Acceptance Testing Results

The acceptance testing was done with multiple meetings with the client/advisor and prospective users as the development team started to reach its completed form.

The first meeting showed the basic features that were setup and expected of the project. The results of this testing session were received well and successful with some slight alterations noted.

The second series of meetings were for the purpose of demonstrating what had changed, and the final results of the project. The results of the final round of acceptance testing showed that all implemented features met expectations and functioned as intended both from design and user interaction standpoints.

For any additional information see ([4.6 Acceptance Testing](#)).

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

6.1 Development Process Abstraction

Following the process laid out in the project plan and the design section of this document as closely as possible, the development team began by setting up as much of the generalized framework and functions, setting up the server that will host the tool, followed by the communication between frontend and backend, and the finer details of the project. As the team made progress there was close contact between advisor/client, and the prospective user to properly adjust and redesign as progress was made. This led to some small changes from the original design, of course, which can be seen in section [1.5 Design Evolution](#).

Along with the development, as outlined in [4. Testing](#) section, testing occurred in order to ensure functionality of the code being written. This was extremely important for the algorithm functionality, all of the math involved, and the visualization of the results; these were some of the primary components of the core facet of the web tool, and as such needed to work as expected even under various potential edge cases.

6.2 Web Tool Results

Included in this section is the resulting visual/UI side of the project that can be seen and experienced further on the actual site (linked here: <https://ucp-webtool.ece.iastate.edu/>).

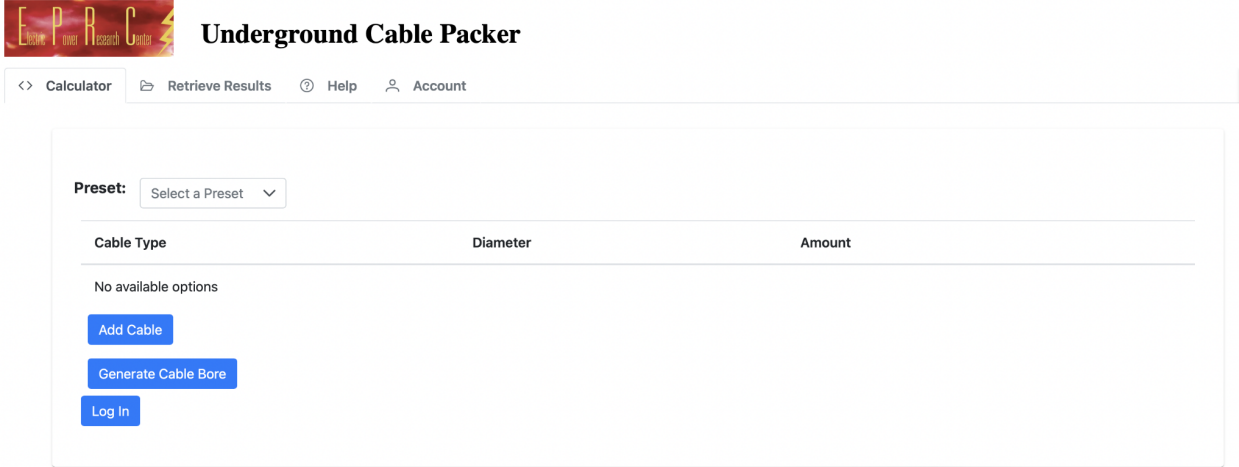


Figure 21: Landing Page

The landing page for the application. The calculator page’s main purpose is for adding cables to be considered for a bore calculation, and sending the request to see the generated bore.

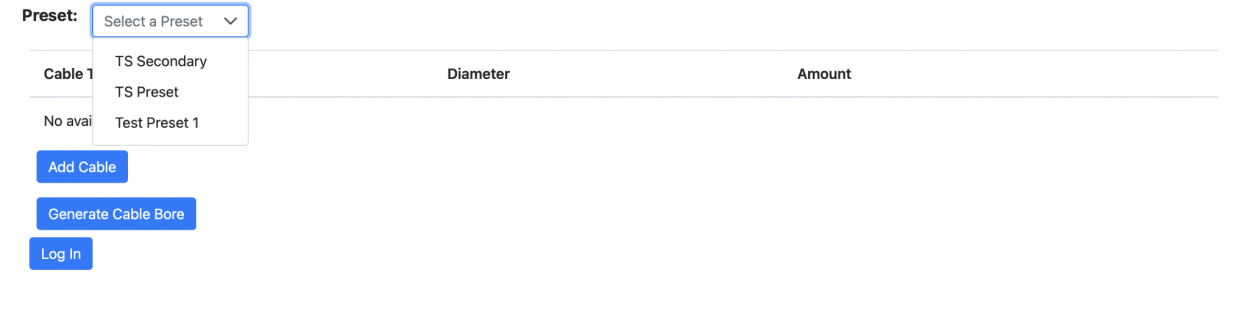


Figure 22: Presets Selection Demonstration

Presets allow individual companies to use their own cable specifications for bore generation. Presets are managed through an account page where each respective company’s cable names and diameters can be changed.

Preset: TS Secondary ▾

Cable Type	Diameter	Amount
<div style="border: 1px solid #ccc; padding: 2px;"> Select or add a Cable ▾ <ul style="list-style-type: none"> 30kV Gold 3kv triple insulated 2" duct 30kV Gold 3kv triple insulated 2" duct </div>	<input type="text"/>	<input type="text" value="1"/> <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✓"/> <input type="button" value="×"/>

Preset: Select a Preset ▾

Cable Type	Diameter	Amount
30 kV Triple Insulated	0.77	3 ✎
Duct, 2"	2	2 ✎

Figure 23: Input Process

Adding cables. Cables all have names and default diameters, though diameter can be changed if desired, and the amount of each type of cable is also specified here.

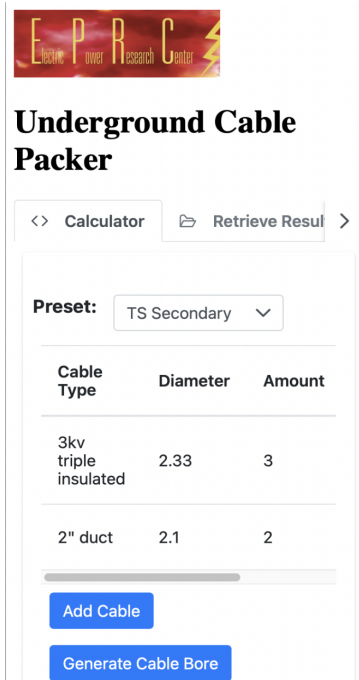


Figure 24: Mobile UI

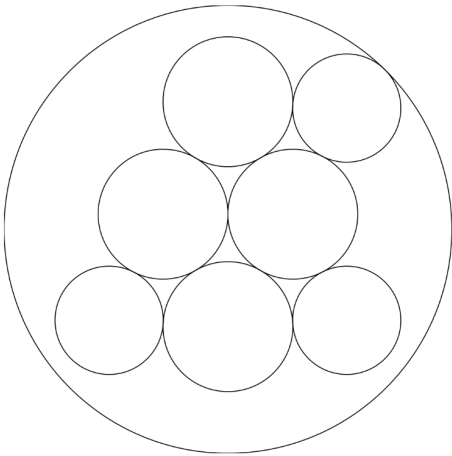
The mobile view of the application. One of the main reasons for rebuilding this application as a web-based tool was mobile compatibility. Completely scalable UI elements allow the app to be used on devices of any size.

Generate Cable Bore

Email to send Results To: [Send Email](#)

[Export to PDF](#) [Share](#)

[Log In](#)

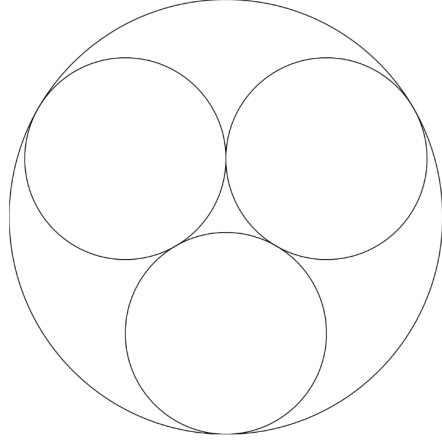


Result ID: 73d0d0d6-537f-4855-996b-35f3348ca5cb
 Rendered at: 2022-4-28 8:56:50
 Minimum Bore Size: 7.24

Type	Diameter
2" duct	2.10
2" duct	2.10
2" duct	2.10
2" duct	2.10
2" duct	2.10
30kV Gold	1.75
30kV Gold	1.75
30kV Gold	1.75
Bore	7.24

<> Calculator [Retrieve Results](#) [Help](#) [Account](#)

Result ID: [Find](#)



Result ID: a7e8fc78-fd82-4ddf-98f6-4df81822aa51
 Rendered at: 2022-4-28 8:58:26
 Minimum Bore Size: 3.77

Type	Diameter
30kV Gold	1.75
30kV Gold	1.75
30kV Gold	1.75
Bore	3.77

[Export](#)

Figure 25: Results Imaging

Generated cable bores. The cables selected above are run through our algorithm to determine what the minimum-radius circle (bore size) is that can enclose all other circles (cables) when packed together. The render has the list of cable diameters, minimum bore diameter, render generation time, and a unique result ID so users can retrieve this result later.

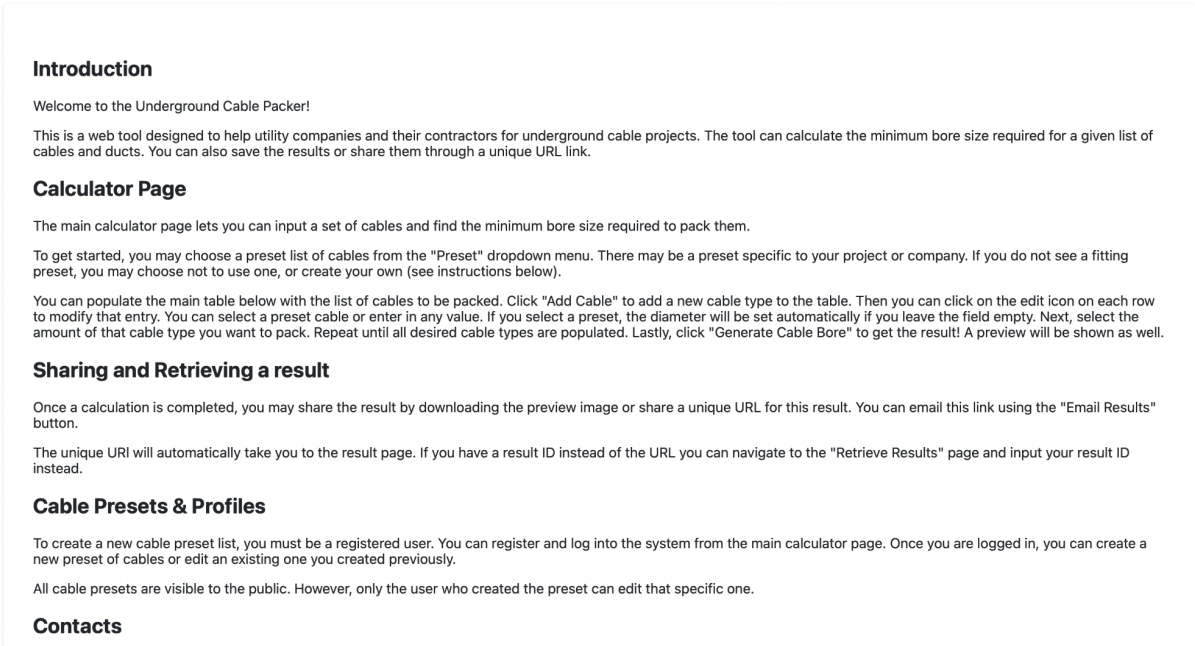


Figure 26: Help Page Image

The help page. Walks users through how to use the tool’s various features and explains what different sections of the site are for. Also provides contact information should users need or want to contact someone at EPRC for help.

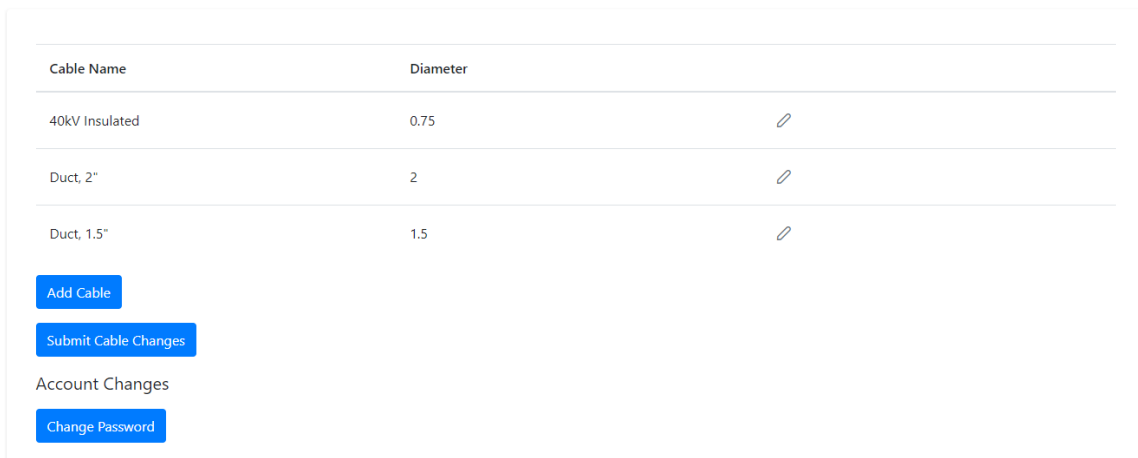


Figure 27: Account Page Image

The account page. Shows users their preset and the ability to create new or edit existing cables. The account changes portion has the functionality to change a password if a user has any need. Changing the password can be done on this page or when attempting to log in.

7 Closing Material

7.1 Discussion

This project, starting in August of 2021 and ending in May of 2022, has the results of a functional web tool that's primary component is that of a bore calculator given a set of inputted cables/bores/circle radii. As in any project, there were changes, delays, unexpected problems, and the need to be in close contact with all those involved, but this team believes it to have been a successful endeavor in, "preparing [the team] for entry to the workforce...using skills in technical writing, project planning, and design reporting", as well as, "the successful implementation and demonstration of designs".

The link to the created web tool can be found on the title page of this document, the GitLab readme, and here: <https://ucp-webtool.ece.iastate.edu/>.

7.2 Conclusion

With the content of the previous section ([7.1 Discussion](#)) in mind, along with all of the project plan, design plan, and completed work, we feel that the design and completion of this project over the last 8 months has given us a new insight into how applications are developed from end to end. We have also gained valuable knowledge not offered through traditional classes, such as interaction with an industry client and managing scope based on user feedback.

7.3 References

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

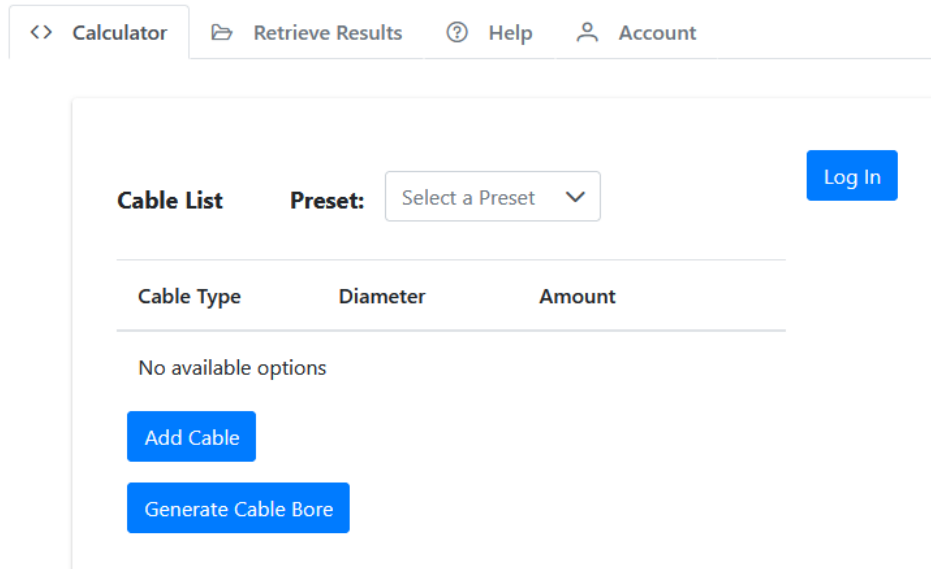
"Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416-424, 2012.

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

8 Appendices

8.1 Appendix I. [Operation Manual]

This operation manual is for the purpose of using the final product web tool application. This application can be found by [clicking here \(https://ucp-webtool.ece.iastate.edu/\)](https://ucp-webtool.ece.iastate.edu/) and simply requires a network connection. Upon reaching the landing page, the primary component of the application will already be open.



The user has the option to immediately begin a new bore calculation by using the “Add Cable” button and entering and modifying any number of cables. Upon desired configuration, the “Generate Cable Bore” should be clicked where it will be processed by the backend server to determine the configuration, at this time, the user would be allowed to enter an email address to which the results could be emailed to - if desired. Otherwise, the results would be shown below the UI (shown above) with a unique Result ID number, a time and date that it was rendered at, and finally the minimum sized bore needed to fit each cable. Additionally, once generated, the ability to download the result as a pdf and to get a sharable link presents itself for returning to the generation. There is also the ability to login to a profile, these profiles have preset cable listings under accounts with the same company or organization.

If the user arrives at the page with a Result ID from a previous bore generation, they can switch to the “Retrieve Results” tab on the top navigation bar. From there they would enter the number to view any previously generated results given the id is valid.

Additionally there are two more tabs in the top navigation bar, that being a Help page where the user can read a quick start guide and a contact link, and finally there is an Account tab, which when logged in, shows all preset cables defined by the account, the ability to create a new preset, create a new account with these presets, and finally change the account’s password in the event that such action is needed.

8.2 Appendix II. [Alternative Design Versions]

Though we iterated on our design from the beginning of the school year, fleshing out ideas for how users will use our application and what technologies we could use to accomplish our goals, we decided on a design that met our needs fairly early on in the process. This is not to say that we had other ideas, of course. This section serves as a continuum of [3.2 Design Exploration](#) for ideas that were briefly considered before more information was gathered to rule them out.

For example, 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.

Very early in the design process, before we knew many requirements of our project, we hoped we could use an AWS-centered approach to build a lightweight application that scaled quickly and could add features easily. This was ruled out as we found we needed to host our application on Iowa State servers and the team's overall lack of knowledge of AWS development meant we would have needed more time dedicated to learning the tech stack to develop effectively. Another plan was to port the old Python desktop application to the web and keep the original function of the app the same. This meant we could rely on the algorithm designed by Professor Wymore to keep development time for a new algorithm to a minimum, though we would be limited to the functionality of the original application, which fell short of our intended feature set.

Some more creative design ideas involved using an external tool to do the calculations for us and simply pass data to the external API through our frontend, though like the idea to use the Python application this would limit us on what features and data we could add to the algorithm should we want to change anything (and no such API fulfilled our needs). There was also a brief design that involved a random number generator to determine bore size, though through rigorous analysis this was found to be an ineffective solution.

8.3 Appendix III. [Other Considerations]

The most important information that would be classified as other considerations would be the research that went into developing a fast and as close to correct backend algorithm. An algorithm for packing unique circles within a circle is a popular computer science problem that has multiple aspects that need to be considered: ranging from how to orient the interior circles together, how to orient the interior circles with respect to the outermost circle, and all of the generalized arrangements. The project's algorithm attempts to address as much of the problem as possible while remaining fast and relatively simple.

The backend code for the algorithm does this by focusing on arranging the circles in a triangular orientation that becomes a hexagonal orientation overall which is the commonly held most efficient packing of circles regardless of the container. An example for this visually, while with uniform circles, would be that of a sheet of bubble wrap, and again different as those are oriented for an effective square or rectangular outer packing.

The algorithmic code uses mathematical functions to get the circles placed together correctly while utilizing a simplified algorithm for how to orient which circle where. For example, it gives priority to placing the largest circles toward the center and filling the holes with the smaller circles from the user input. It is far from the most optimal solution, but in a situation where the algorithm being written is that of NP hard, while remaining fast and effective, its efficiency is limited.

For more information on circle within a circle packing algorithms there are many scholarly articles and algorithms written by those whose sole focus of work was to make the most efficient algorithm possible - people who are smarter than us. Nevertheless, the resultant algorithm should be enough for this project's goal.