2 Project Plan

2.1 Project Management/Tracking Procedures

2.1.1 Management Style and Justification

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

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

2.1.2 Progress Tracking and Management Tools

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

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

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

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

2.2 Task Decomposition

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

1. Project Planning & Defining
    - 1.1. Team dynamic planning
        - 1.1.1. Begin team communication
        - 1.1.2. Setup primary communication channels between team members and advisors (TA Jacob Conn, and Professor Mathew Wymore)
        - 1.1.3. Setup primary communication channels with all other connected parties (Alliant Energy, ETS, ETG, EPRC)
          [Dependent on 1.1.2]
        - 1.1.4. Assign leadership roles among team members
          [Dependent on 1.1.2]
    - 1.2. Develop Requirements
        - 1.2.1. Meet with professor Mat Wymore
        - 1.2.2. Meet with client: Alliant Energy
          [Dependent on 1.2.1]
        - 1.2.3. Develop requirements documentation
          [Dependent on 1.2.1, 1.2.2]
        - 1.2.4. Determine engineering standards to follow in software development for this project
          [Dependent on 1.2.3]
    - 1.3. Project Plan Instantiation
        - 1.3.1. Documentation setup
        - 1.3.2. Document breakdown team meeting
          [Dependent on 1.3.1]
        - 1.3.3. Finalize original Project Plan document - living document for when the need for improvements, or alterations
          [Dependent on 1.3.1, 1.3.2]
    - 1.4. Software stack planning
        - 1.4.1. Determine specific software stack
        - 1.4.2. Get familiar with tech stack - Typescript React and Go languages
          [Dependent on 1.4.1]
            - 1.4.2.1. Perform individual practice to familiarize members with unique syntax
        - 1.4.3. Create basic proof of concept for general functionality plans
            - 1.4.3.1. Sending information from back to front for image generation
    - 1.5. Project Merge and Pull Request (PR) Protocol
        - 1.5.1. Team initialization of the preliminary process of getting new code accepted
          [Dependent on 1.3.3]
        - 1.5.2. Limits on number of team members approval for a single PR
          [Dependent on 1.5.1]

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

2. DevOps & Tech Setup
   [Dependent on 1.4]
   2.1. Initialize project website
   2.2. Set up CI/CD (Continuous Integration/Continuous Deployment) pipeline
       2.2.1. Choosing the technologies that best integrate with our software.
       2.2.2. Implementing the chosen technologies and verifying they will continue to work for the 10 rated years of project lifetime.
       2.2.3. Testing the chosen technologies to ensure they deliver correct results.
   2.3. Set up deployment environments
   [Dependent on 2.2]
       2.3.1. Testing the chosen technologies on the deployment servers to ensure that deployments go smoothly
       2.3.2. Testing the technologies to ensure they will continue to operate in the deployment environments even after host and software updates.
   2.4. Set up individual team member work environment
   [Dependent on 1.4]
3. Software Design & Functional Design Verification
   [Dependent on 1.2]
   3.1. Create User Interface (UI) Mock-Ups
   [Dependent on 1.2]
   3.2. 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

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.]
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
8. Integration Testing and Unit Testing
   - 8.1. Unit testing being a part of team PR protocol will occur with continuous software development
     [Dependent on 1.5]
   - 8.2. Integration testing of software produced at program finalization stages
     [Dependent on 4.0, 6.0, 7.0]
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]

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 Integration Testing and Unit Testing, 9.0 User Acceptance Testing and Deployment.

Task Decomposition version number: 0.1.0

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestones

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

Evaluation

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

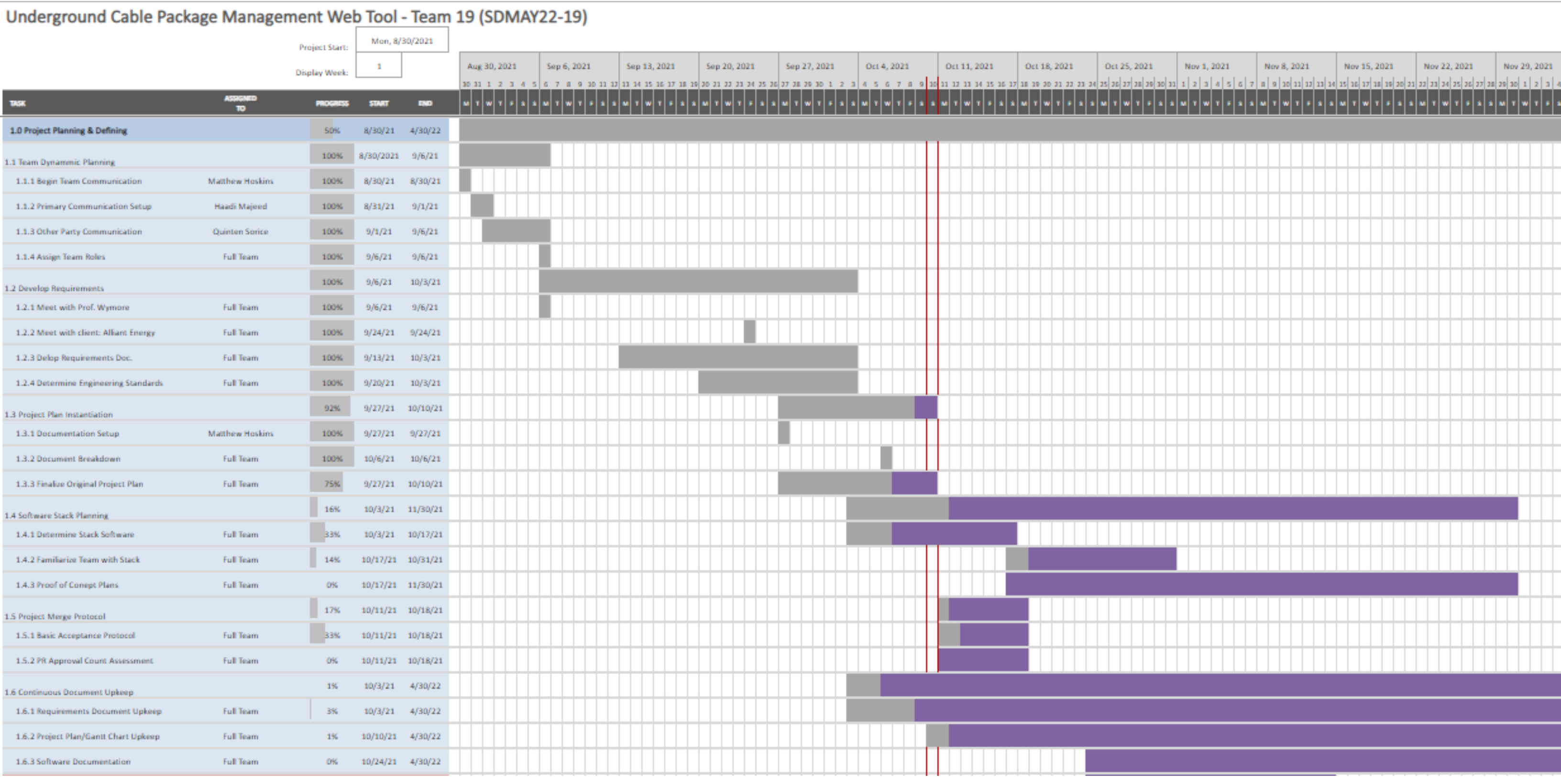
2.4 Project Timeline/Schedule

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

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

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

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



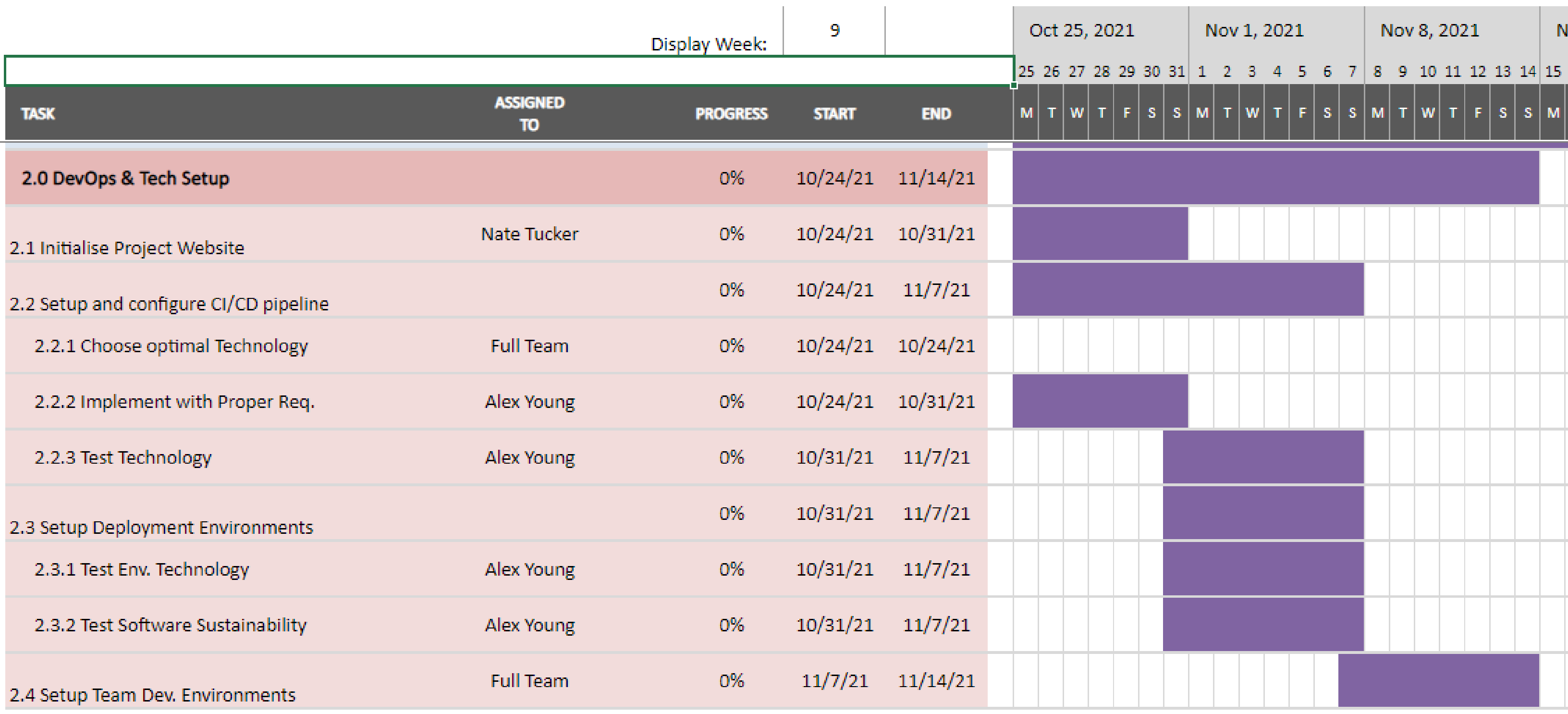
Underground Cable Package Management Web Tool - Team 19 (SDMAY22-19)

Project Start: Mon, 8/30/2021

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
1.0 Project Planning & Defining		50%	8/30/21	4/30/22
1.1 Team Dynammic Planning		100%	8/30/2021	9/6/21
1.1.1 Begin Team Communication	Matthew Hoskins	100%	8/30/21	8/30/21
1.1.2 Primary Communication Setup	Haadi Majeed	100%	8/31/21	9/1/21
1.1.3 Other Party Communication	Quinten Sorice	100%	9/1/21	9/6/21
1.1.4 Assign Team Roles	Full Team	100%	9/6/21	9/6/21
1.2 Develop Requirements		100%	9/6/21	10/3/21
1.2.1 Meet with Prof. Wymore	Full Team	100%	9/6/21	9/6/21
1.2.2 Meet with client: Alliant Energy	Full Team	100%	9/24/21	9/24/21
1.2.3 Delop Requirements Doc.	Full Team	100%	9/13/21	10/3/21
1.2.4 Determine Engineering Standards	Full Team	100%	9/20/21	10/3/21
1.3 Project Plan Instantiation		92%	9/27/21	10/10/21
1.3.1 Documentation Setup	Matthew Hoskins	100%	9/27/21	9/27/21
1.3.2 Document Breakdown	Full Team	100%	10/6/21	10/6/21
1.3.3 Finalize Original Project Plan	Full Team	75%	9/27/21	10/10/21
1.4 Software Stack Planning		16%	10/3/21	11/30/21
1.4.1 Determine Stack Software	Full Team	33%	10/3/21	10/17/21
1.4.2 Familiarize Team with Stack	Full Team	14%	10/17/21	10/31/21
1.4.3 Proof of Conept Plans	Full Team	0%	10/17/21	11/30/21
1.5 Project Merge Protocol		17%	10/11/21	10/18/21
1.5.1 Basic Acceptance Protocol	Full Team	33%	10/11/21	10/18/21
1.5.2 PR Approval Count Assessment	Full Team	0%	10/11/21	10/18/21
1.6 Continuous Document Upkeep		1%	10/3/21	4/30/22
1.6.1 Requirements Document Upkeep	Full Team	3%	10/3/21	4/30/22
1.6.2 Project Plan/Gantt Chart Upkeep	Full Team	1%	10/10/21	4/30/22
1.6.3 Software Documentation	Full Team	0%	10/24/21	4/30/22

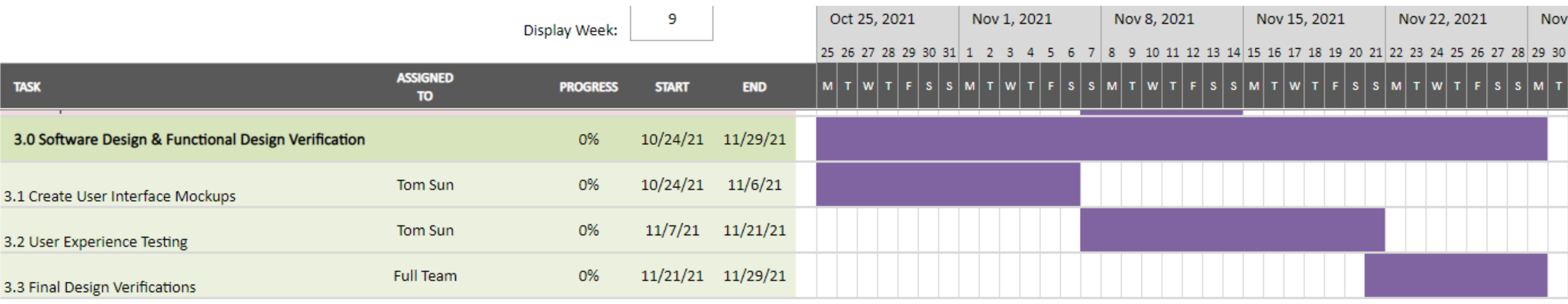
This zoomed out section of the project gantt chart is task 1.0 Project Planning & Defining (shown above). This section continues to span the weekly schedule as it includes the upkeep of various documents relevant to the project. It began on week one of the planner and continues from there with a variety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



Display Week: 9

TASK	ASSIGNED TO	PROGRESS	START	END
2.0 DevOps & Tech Setup		0%	10/24/21	11/14/21
2.1 Initialise Project Website	Nate Tucker	0%	10/24/21	10/31/21
2.2 Setup and configure CI/CD pipeline		0%	10/24/21	11/7/21
2.2.1 Choose optimal Technology	Full Team	0%	10/24/21	10/24/21
2.2.2 Implement with Proper Req.	Alex Young	0%	10/24/21	10/31/21
2.2.3 Test Technology	Alex Young	0%	10/31/21	11/7/21
2.3 Setup Deployment Environments		0%	10/31/21	11/7/21
2.3.1 Test Env. Technology	Alex Young	0%	10/31/21	11/7/21
2.3.2 Test Software Sustainability	Alex Young	0%	10/31/21	11/7/21
2.4 Setup Team Dev. Environments	Full Team	0%	11/7/21	11/14/21

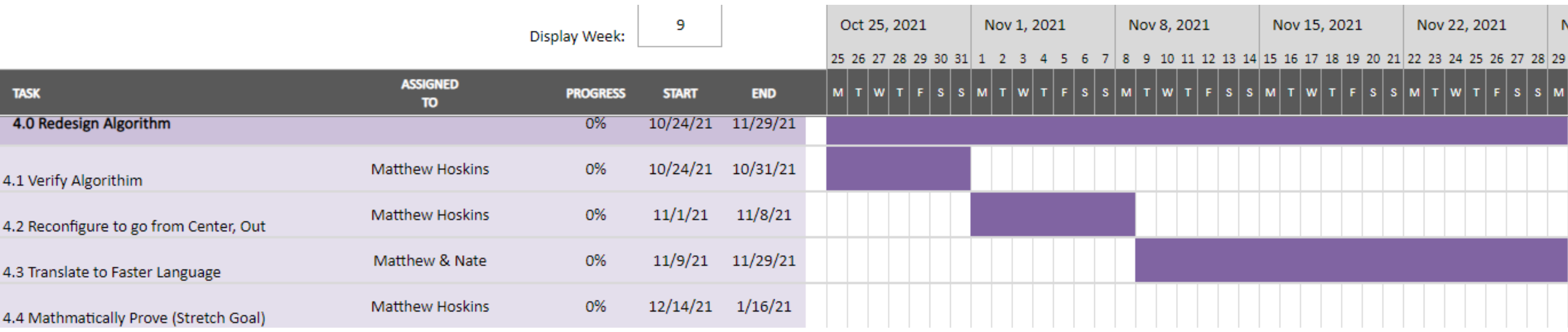
Task 2.0 DevOps & Tech. Setup (shown above), is the early planning, staging, and setup of technology and CI/CD pipelining.



Display Week: 9

TASK	ASSIGNED TO	PROGRESS	START	END
3.0 Software Design & Functional Design Verification		0%	10/24/21	11/29/21
3.1 Create User Interface Mockups	Tom Sun	0%	10/24/21	11/6/21
3.2 User Experience Testing	Tom Sun	0%	11/7/21	11/21/21
3.3 Final Design Verifications	Full Team	0%	11/21/21	11/29/21

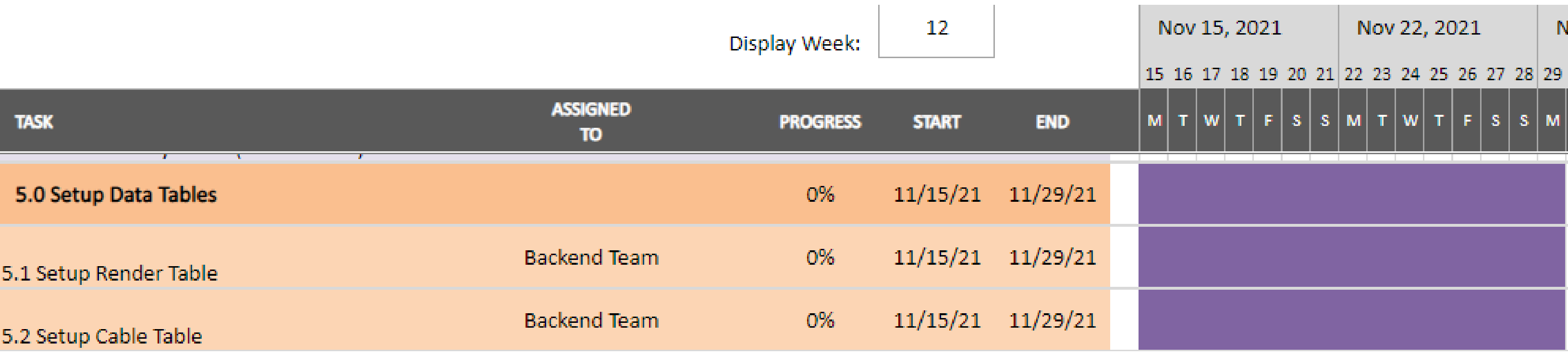
Task 3.0 Software Design & Functional Design Verification (shown above), will involve a few tasks dependent on each other in some form that will allow for the planning for the eventual UI of the final software project.



Display Week: 9

TASK	ASSIGNED TO	PROGRESS	START	END
4.0 Redesign Algorithm		0%	10/24/21	11/29/21
4.1 Verify Algorithim	Matthew Hoskins	0%	10/24/21	10/31/21
4.2 Reconfigure to go from Center, Out	Matthew Hoskins	0%	11/1/21	11/8/21
4.3 Translate to Faster Language	Matthew & Nate	0%	11/9/21	11/29/21
4.4 Mathmatically Prove (Stretch Goal)	Matthew Hoskins	0%	12/14/21	1/16/21

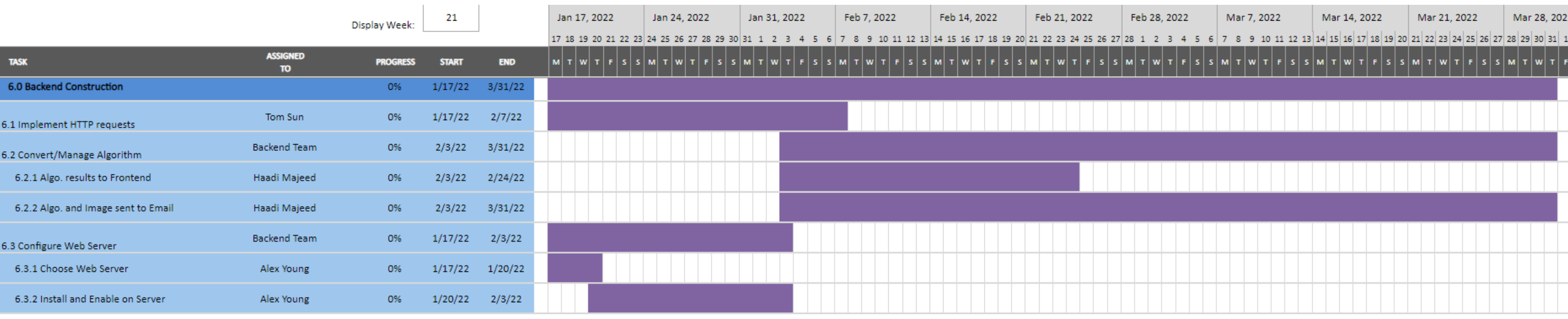
For task 4.0 Redesign Algorithm, is a math focused major task that will need to occur early on in order to properly ensure the current Python code can be verified, and converted to fit the projects' needs.



Display Week: 12

TASK	ASSIGNED TO	PROGRESS	START	END
5.0 Setup Data Tables		0%	11/15/21	11/29/21
5.1 Setup Render Table	Backend Team	0%	11/15/21	11/29/21
5.2 Setup Cable Table	Backend Team	0%	11/15/21	11/29/21

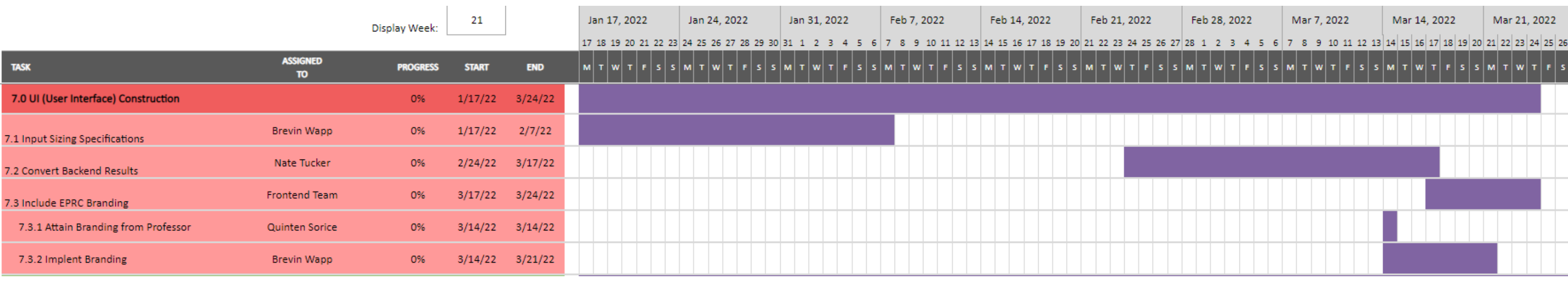
Task 5.0 Data Tables Setup (shown above), is a short major task that will be done in order to ensure all possible data points for entry are stored and available to the user input.



Display Week: 21

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

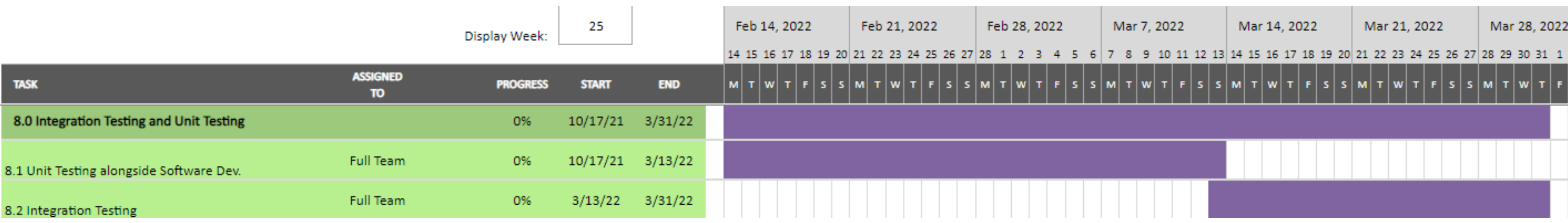
Task 6.0 Backend Construction (shown above), involves the long development of all the backend components to this software project. It starts after final design approval, and proof of concept programming.



Display Week: 21

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

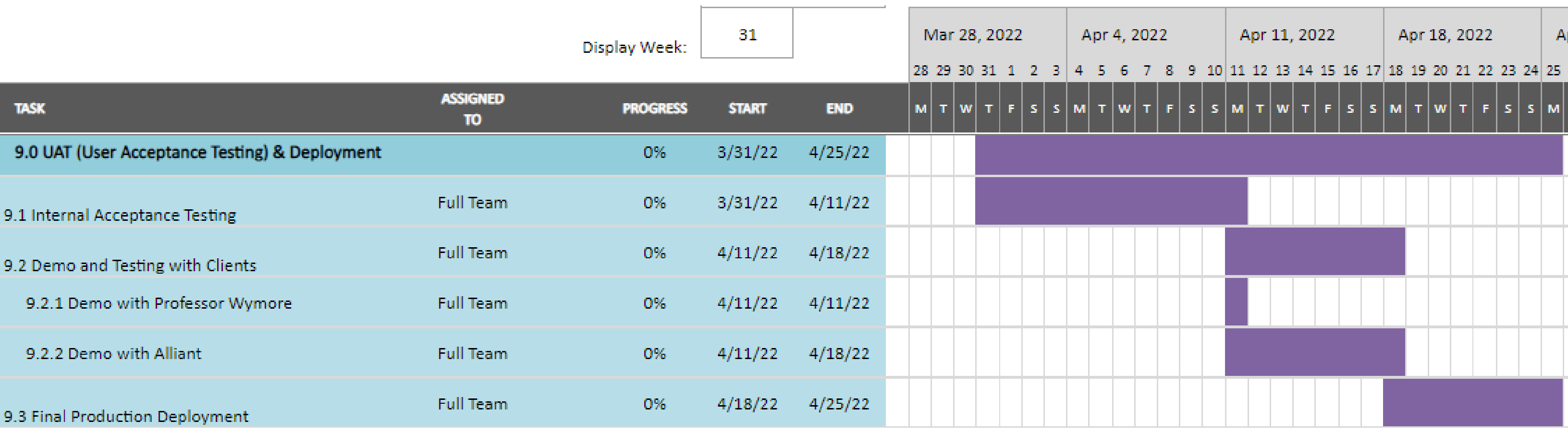
Task 7.0 UI Construction (shown above), will begin at a later point after proof of concept software has been constructed, as well as, general design given approval.



Display Week: 25

TASK	ASSIGNED TO	PROGRESS	START	END
8.0 Integration Testing and Unit Testing		0%	10/17/21	3/31/22
8.1 Unit Testing alongside Software Dev.	Full Team	0%	10/17/21	3/13/22
8.2 Integration Testing	Full Team	0%	3/13/22	3/31/22

As for task 8.0 Integration Testing and Unit Testing (shown above), what is shown here is the last section of it with the ending of the unit testing at the end of new code production followed by the start and completion of integration testing.



Display Week: 31

Mar 28, 2022 · Apr 4, 2022 · Apr 11, 2022 · Apr 18, 2022

TASK	ASSIGNED TO	PROGRESS	START	END
9.0 UAT (User Acceptance Testing) & Deployment		0%	3/31/22	4/25/22
9.1 Internal Acceptance Testing	Full Team	0%	3/31/22	4/11/22
9.2 Demo and Testing with Clients	Full Team	0%	4/11/22	4/18/22
9.2.1 Demo with Professor Wymore	Full Team	0%	4/11/22	4/11/22
9.2.2 Demo with Alliant	Full Team	0%	4/11/22	4/18/22
9.3 Final Production Deployment	Full Team	0%	4/18/22	4/25/22

The last major task, 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.

More detailed documentation of the gantt chart is in the gantt chart excel file.

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. Integration Testing and Unit Testing
    - Lack of comprehensive tests
        - Unlikely, Major: With incomplete testing, there is a chance that edge cases in how our app is used could go unnoticed which would be frustrating for users that encounter them. Or edge case calculations could turn out wrong, and missing them would mean the possibility of incorrect charging of clients for bore sizing.
        - We will have to ensure that our testing methodology is thorough and we know the results we are looking for.

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

9. UAT & Deployment

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

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

2.6 Personnel Effort Requirements

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

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

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

UI Mock-Up Verification	14	2 hour long meeting with clients
Verify Current Algorithm	15	Time for getting familiar with the tool and extensive testing
Redesign Algorithm	40	Includes time for development and testing
Convert Programming Languages	15	Includes time for development and testing in new language
Mathematical Proof of Algorithm	40	Some research and information seeking may be required
Set Up DataBase	15	Infrastructure should be already set up
Create Data Table - Render	4	Includes time to test created table
Create Data Table - Cable	4	Includes time to test created table
Backend Construction - HTTP	100	Includes time to develop and test all functions
Backend Construction - Transferable format	50	Some Prototyping may be required
UI Construction - Inputs	50	Some Prototyping may be required
UI Construction - Visualize results	100	Prototyping and some research into visualization tools required
UI Construction - EPRC Branding	40	Adding styles/icons to the constructed software shouldn’t take too long
Unit Tests	200	Should be done alongside development, estimated 1 hours per week per person
Integration Testing	100	Includes time for extensive test and making any fixes/adjustments
Internal Acceptance Testing (IAT)	14	2 hour meeting to review all aspects of the application

Demo and User Acceptance Testing (UAT)	14	2 hour meeting to review all aspects of the application
Final Production Development	60	Includes time to deploy and fix any last minute issues. Includes some time for monitoring after deployment

---

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.