Table 1: Revision History

Date	**Developer(s)**	**Change**
September 21 2022	Darren	Team information sections
September 21 2022	David	Technology sections
September 21 2022	Adham	Project Scheduling Section
September 22 2022	Victor	Technology sections
...	...	...

Development Plan
MTOBridge

Team 15, Alpha Software Solutions
Badawy, Adham
Yazdinia, Pedram
Jandric, David
Vakili, Farzad
Vezina, Victor
Chiu, Darren

[Put your introductory blurb here. —SS]

1 Team Meeting Plan

The team will meet on Tuesdays on campus at 5:30PM weekly for 1 to 1.5 hours. Team members will meet in-person when possible but will join virtually when unable to be physically present. Meetings are expected to review progress made since the previous meeting, review the status of the project, and discuss and divide upcoming work among team members.

Adham has reserved a room in Gerald Hatch Centre and will chair meetings. Darren will open the call for team members to join virtually. Meeting minutes will be taken as needed when group decisions are reached on topics, responsibility for which will pass between team members as is appropriate.

2 Team Communication Plan

Git issues will be used to track ongoing tasks on the project. Aside from meetings and impromptu conversations between shared classes, the team is communicating primarily through a group chat in Discord.

The team will communicate with the client via email and Zoom calls. Team members will CC the rest of the team when emailing the client.

3 Team Member Roles

- Adham will act as team leader and chair meetings.

- Darren will facilitate virtual meetings and has experience in C++.
- David has experience in C++ and is most familiar with the UI implementation framework we currently expect to use.
- Farzad possesses the most domain knowledge in civil engineering and is the main point of contact with the client.
- Pedram will maintain the git repo.
- Victor has experience in C++.

4 Workflow Plan

- How will you be using git, including branches, pull request, etc.?
- How will you be managing issues, including template issues, issue classificaiton, etc.?

The git will be separated into a dev and prod (master) branch. Developers will fork from the dev branch to implement changes. After merging changes into the dev branch and confirming no issues with integration, the dev branch will be merged into prod. CI/CD will be set up between prod and dev as well as between dev and feature branches.

5 Proof of Concept Demonstration Plan

What is the main risk, or risks, for the success of your project? What will you demonstrate during your proof of concept demonstration to convince yourself that you will be able to overcome this risk?

6 Technology

6.1 Programming Language

C++20 will be used for the GUI, and will be connected to the backend made in MATLAB.

6.2 Linter

In order to maintain a consistent format, we will use clang-format for linting. Using this linter, we can also easily select a coding standard to follow, which will come from a file in our repo.

6.3 Unit Testing Framework

6.4 Code Coverage Measuring Tools

6.5 Continuous Integration

We will use GitHub's built-in CI/CD in multiple ways to help the development of this project. Firstly, we will automatically run clang-format when code is pushed to the repo, allowing us to ensure that any code on GitHub complies with the coding standard. We will also automatically run pdflatex on uploaded .tex files so that the PDF versions of all documents are guaranteed to be up to date. Lastly, any uploaded code will be built and tested against our current suite of tests to ensure that it has not introduced any errors. This testing will also include the use of Valgrind to check for any runtime bugs that may be introduced.

6.6 Performance Measuring Tools

Although we don't expect the performance of our code to have much effect on the performance of the project as a whole (the majority of computing resources will be taken up by the MATLAB back-end), we will still use Valgrind to measure the performance of our program to ensure that it is within reasonable bounds. We will also use Valgrind as a way to check for runtime errors in our code.

6.7 Libraries

A GUI framework called Qt will be used.

- This framework will allow for easy creation of a simple GUI, but also allows for extensibility which may be required.
- It contains basics for creating buttons, text fields etc., and more complex components like graphs (or even 3D views) with good performance.
- It also contains a testing framework for easily testing the GUI, as well as benchmarking performance.

MATLAB is being used for the backend, and in order to connect it with the GUI, libraries provided in MATLAB are needed in order to call MATLAB functions from C++.

6.8 Tools

We will use the following tools during our development:

- GitHub will be used as our version control system. It will also be used to track issues and implement continuous integration.
- MATLAB will be used to run the back-end code provided to us from the MTO.

- LaTeX will be used to create the documentation for this project.
- While each developer will use the IDE of their preference, a number of us will use Visual Studio.
- CMake will be used to facilitate building the C++ code.
- As stated in the linter section, we will use clang-format to enforce a standard on the code we write.
- As stated in performance measuring tools, Valgrind will be used both to test performance and check for runtime bugs in the code.

7 Coding Standard

We will be using the Google C++ Style Guide for our project as it is one of the most popular C++ style guides available. Using a standard will allow our code to have a consistent style no matter which of us wrote it. The Google Style Guide is also supported by clang-format, which will allow us to easily check the compliance of code.

8 Project Scheduling

Scheduling will be done in somewhat of a sprint format, week to week, to avoid what we see as relatively futile attempts to predict where along the project we will be 8 weeks from now in spite of all the stacking uncertainties between now and then. With that in mind, a significant part of the weekly meetings will be discussing upcoming milestones, and decomposing larger tasks, such as this development plan, into single person-sized chunks to be split amongst the team, as well as scheduling smaller submeetings as needed. We will be deciding who does what based on individual preferences and suitabilities to different portions of each step of the project.

An important part of scheduling work is to ensure the schedule is the correct mix of both efficent and also feasible to accomplish. Another advantage to this week-by-week sprint based scheduling is that a very granular look can be taken at the free time available to each group member each week, and adjusting accordingly, as schoolwork tends to come in waves, and we can therefore take advantage of lighter weeks to allow us to ease up on tougher weeks when possible, instead of assuming a static amount of free time per week. As well, maintaining a living schedule every week means the work ends up getting spread out across the project, and minimizes last second cramming.

The major milestones for this project will follow along with the deliverable schedule fairly well for the most part, if pushed up a few days. i.e. though the external deadline for the requirements document may be October 5th, an

internal deadline of Oct 3rd will be maintained, circumstances permitting, to ensure suitable time to review and double check deliverables before they are to be presented to stakeholders. This example would go for all other stated deliverables as well, however there may be more sub-milestones identified within the weekly meetings, especially where the deliverable schedule leaves very large gaps. Namely, as we get a better grasp on the finer points of the implementation, the sizeable chunk of time with no deadlines between Nov. 25th and Jan. 18th will materialize into many sub deliverables and internal milestones for the group, working around exams and holidays of course.