System Design for MTOBridge

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

January 18, 2023

1 Revision History

Date	Developer(s)	Change
15/01/2023	Farzad	Initial Draft
17/01/2023	Victor	Added Timeline and Protocol Design

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
MTOBridge	Explanation of program name
[... —SS]	[... —SS]

Contents

List of Tables

List of Figures

3 Introduction

The following is a high level system design document for the MTOBridge software, a program for load rating bridges experiencing strain caused by self driving truck platoons. For more information on this software, you can consult the following documents as needed:
Link to SRS
Link to HA
Link to V&V
Link to MG
Link to MIS

4 Purpose

This document is intended to provide a high-level view of the design decisions made with regards to the user interface component of the MTOBridge software, along with a planned timeline of implementation.

5 Scope

Note that this document, as well as all others written by this team, will focus purely on the visualization and user interface component of MTOBridge, ignoring the backend MATLAB solvers as this component will not be designed by this team, nor does this team have any hand in that design. A simple diagram outlining the system boundaries can be found below.

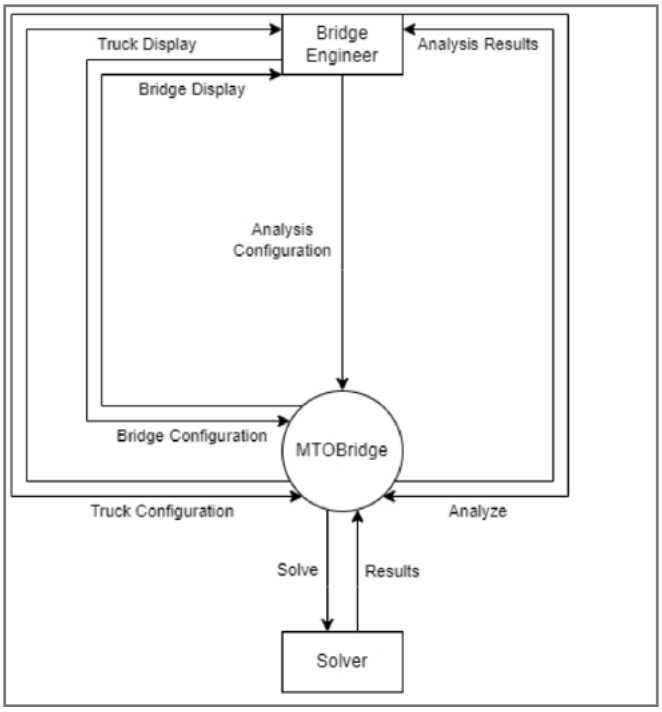

Figure 1: System Boundaries Diagram

6 Project Overview

6.1 Normal Behaviour

Assuming everything goes smoothly, the most general user scenario for MTOBridge is as follows:

1. A user launches the program
2. They set up/load a truck platoon configuration

   2a. They may choose to save their configuration for later
3. They set up/load a bridge configuration

   3a. They may choose to save their configuration for later
4. They set up/load a solver configuration

   4a. They may choose to save their configuration for later
5. They run the calculations

6. A visualization of the results is presented to them
7. The output report is created and shown to them
8. The out report is saved to the file system
    8a. They may choose to return to step 1
9. They exit the program

6.2 Undesired Event Handling

As this is a user interface, robustness and reliability are of the utmost importance for a good user experience. Crashes, slowdowns, or hangs as a result of unexpected events will be highly detrimental to the quality of the program. With this in mind, the plan is to attempt to foresee and catch as many different types of errors as possible, and otherwise allow the program to fail in a palatable way. For example, if during step 4 the user inputs an invalid bridge configuration (i.e choosing "swaws" as the length of their bridge), the program will be set up to prompt them to enter another length, as their current one is invalid and it needs to be a number between x and y, say. Another consideration is to design the program in such a manner so as to limit the possibility of undesired events occurring, by removing user freedom. For example, only allowing users to drag a slider from x to y meters to determine the length of their bridge rather than having them type it in, or some other form of input where the range of inputs can be controlled. This can make the program feel constrained, but also create a safer runtime where the user is not given enough rope to hang themselves. Which of these undesired event handling techniques is applied will depend on the context, something like user input is simple and its errors are able to be caught well enough that the first method seems feasible. However, the amount of interaction with the visualization/animation of the calculation results available to the user is a much less discrete design space, and therefore might have to be limited using the second method. The choice of method and its application will be refined through testing, both for robustness and quality of life.

6.3 Component Diagram

N/A as there are no hardware or electrical components to this project

6.4 Connection Between Requirements and Design

Table 1: Connection Between Requirements and Design

Decisions	Requirement
Concurrency	NFR.9: UI elements react promptly NFR.10 UI will not be unreasonably slow.
MVC architecture	NFR.13: The product shall be easily maintainable. NFR.14: The program should be able to be easily translated into other languages. NFR.3: The product will appear correctly on different display resolutions.
Local File system	SR-8: The system must automatically save to a file. SR-3: The system must produce a log of function calls made to MATLAB. The log will include timestamps along with software environment information such as the current input.

Following concurrency best practices and its design implications ensures that the UI thread will never get stuck. Therefore, it can remain responsive and reacts promptly.

Incorporating the MVC architecture enables the separation of presentation components from the core logic. Firstly, this allows for a flexible presentation since different view modes can be "plugged" and "unplugged". Secondly, taking advantage of such a feature makes altering the front-end source code more convenient because of it's low coupling with other components.

Since the number of automatic saves is high, it's wise to have a local data storage. Moreover, in order to prevent the automatic saving from taking a lot of space thus decreasing it's value, a simple file system (vs a database) can be created where automatic saves are stored with the same filename overwriting the previous auto-saves. However, custom saves with a unique name given by the user will remain intact.

7 System Variables

7.1 Monitored Variables

N/A

7.2 Controlled Variables

N/A

7.3 Constants Variables

N/A

8 User Interfaces

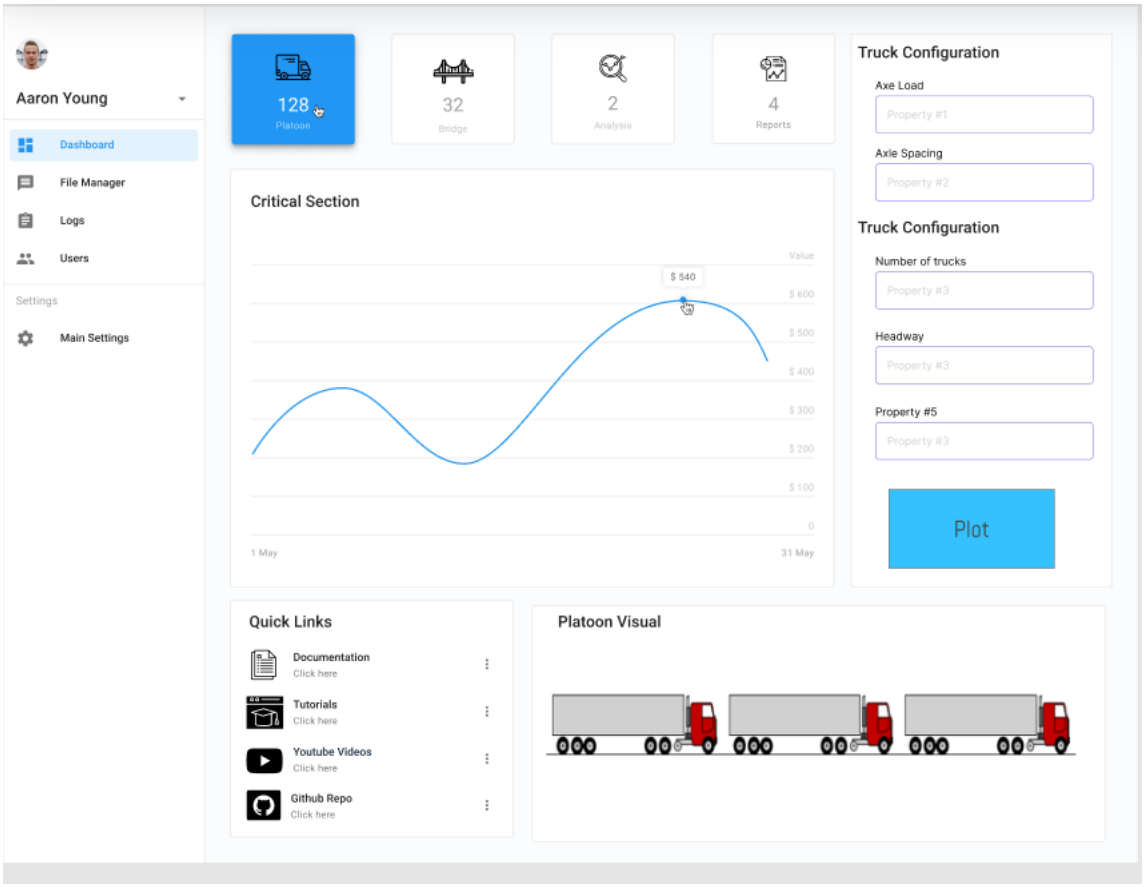

Figure 2: User interface Platoon Diagram

The screen where the user inputs information related to the truck configuration including the axle load, axle spacing, number of trucks and headway. A visual feedback of the inputted information is given at the bottom of the page to the user for confirmation. The effects of the specified platoon configuration to the critical/concerned section is displayed real-time as a graph in the middle of the screen.

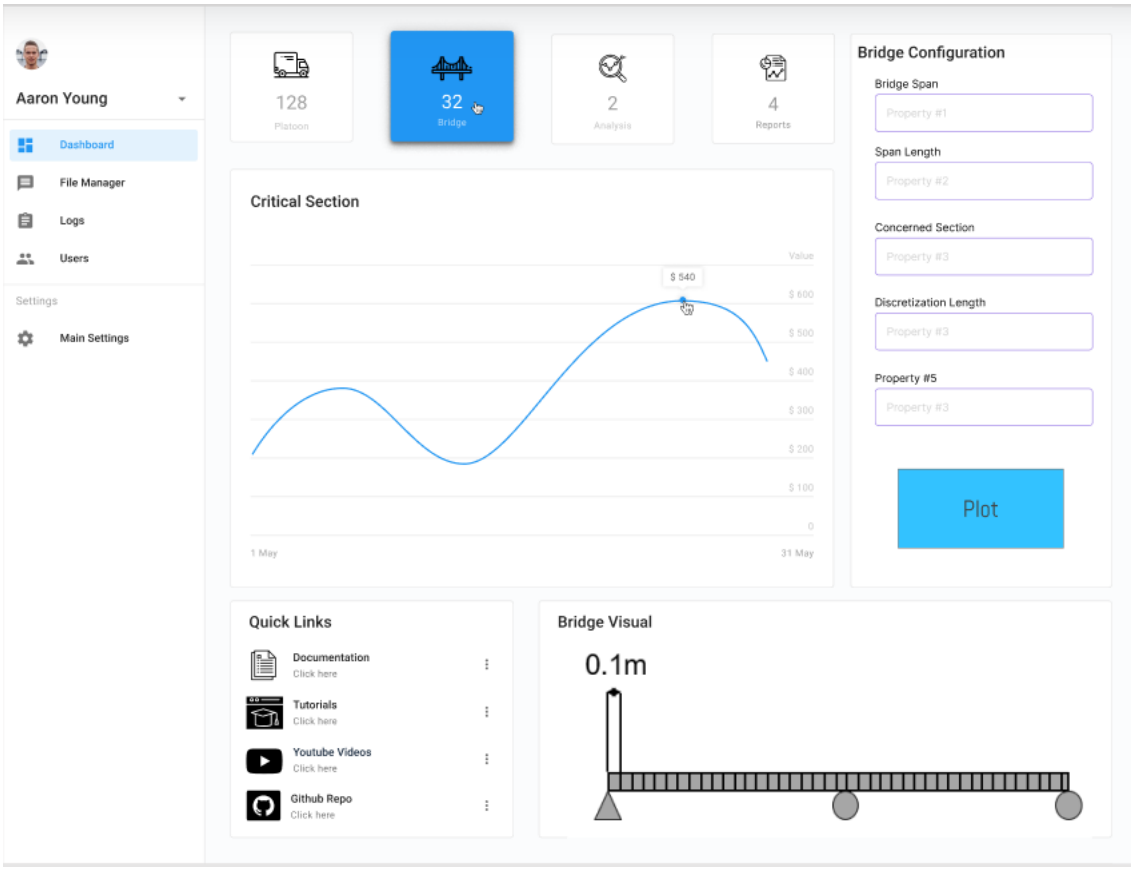

Figure 3: User interface Bridge Diagram

The screen where the user inputs information related to the Bridge configuration including the bridge span, span length, section of concern and the discretization length. A visual feedback of the inputted information is given at the bottom of the page to the user for confirmation. The effects of the specified bridge configuration to the critical/concerned section is displayed real-time as a graph in the middle of the screen.

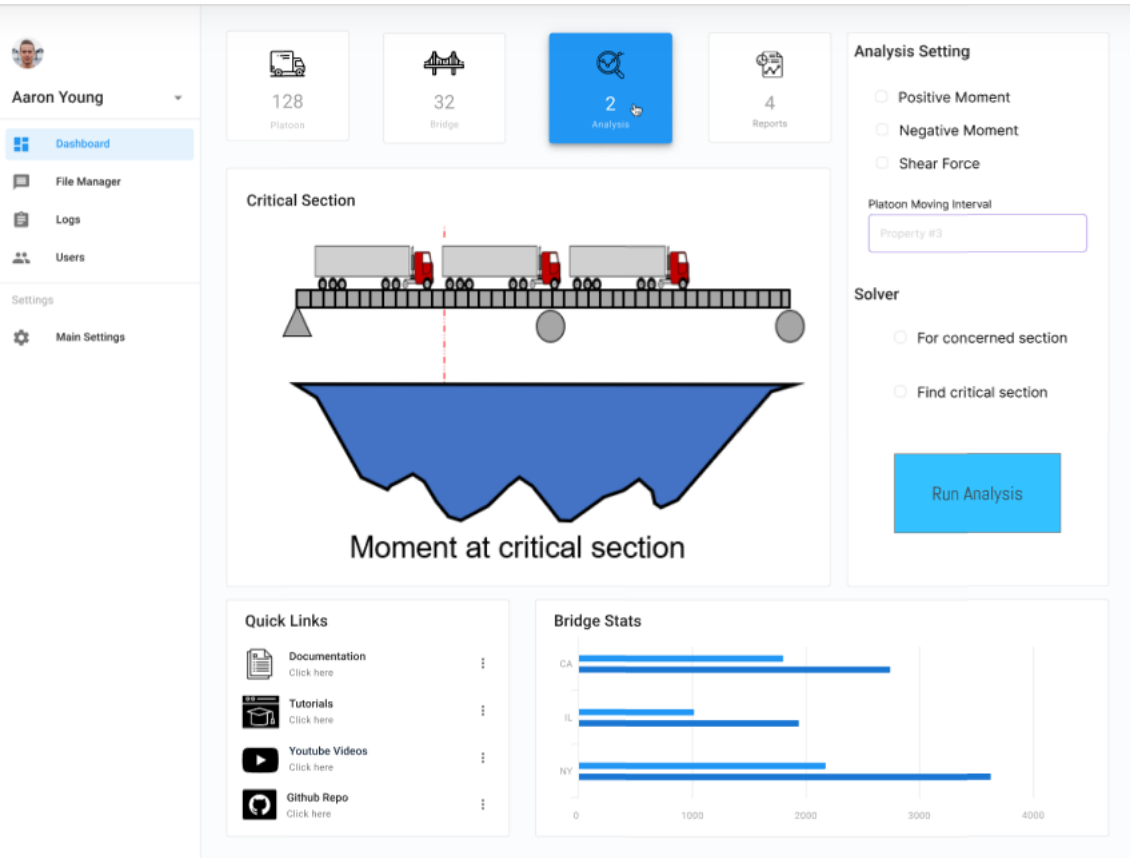

Figure 4: User interface Analysis Diagram

The screen where the user determines the Analysis setting as well as the the section to be analyzed.

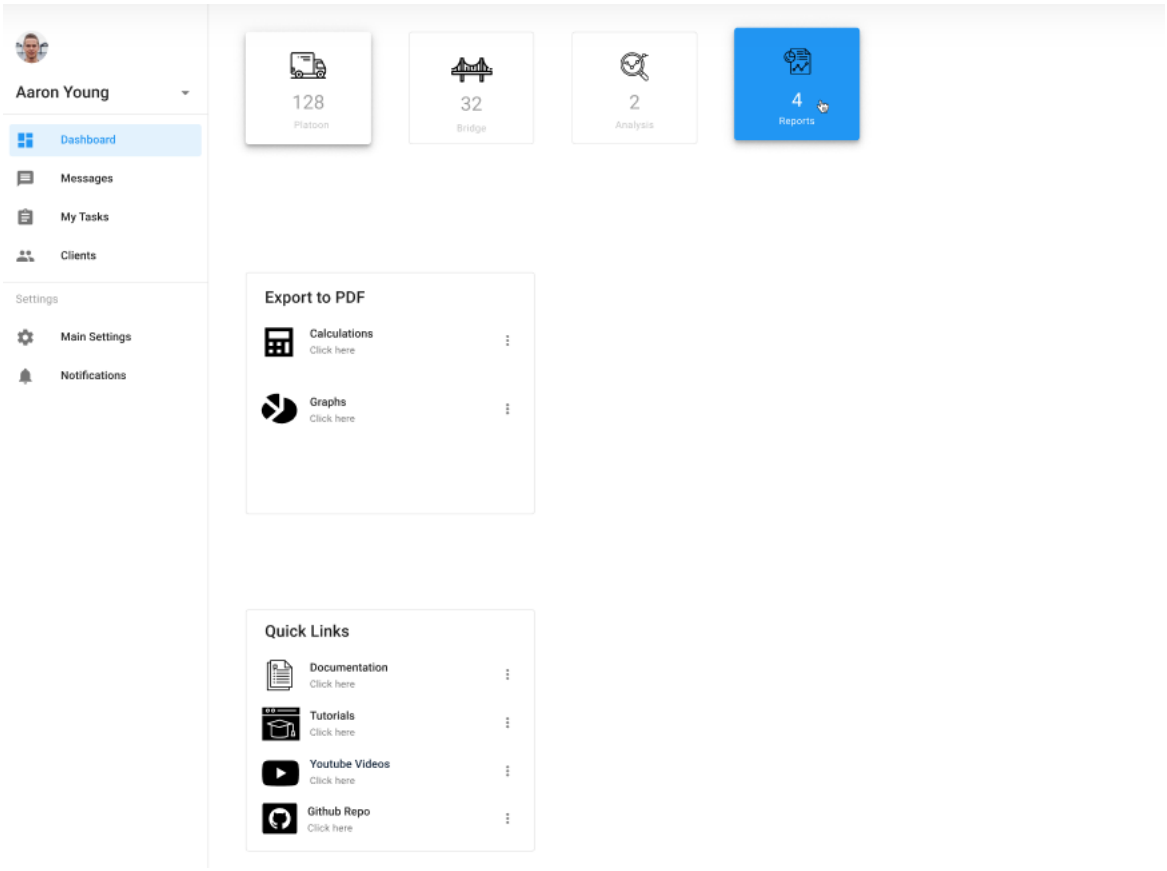

Figure 5: User interface Report Diagram

The screen where the calculations done by the solver and the outputted graphs can be exported to PDF.

9 Design of Hardware

N/A

10 Design of Electrical Components

N/A

11 Design of Communication Protocols

The only communication protocol used by the program is when it communicates with the MATLAB script. This actually uses two protocols: the method names and parameters defined in the MATLAB code, and the functionality implemented by the OS that allows our

program to call the MATLAB script. Neither of these communication protocols has been designed by the project team.

12 Timeline

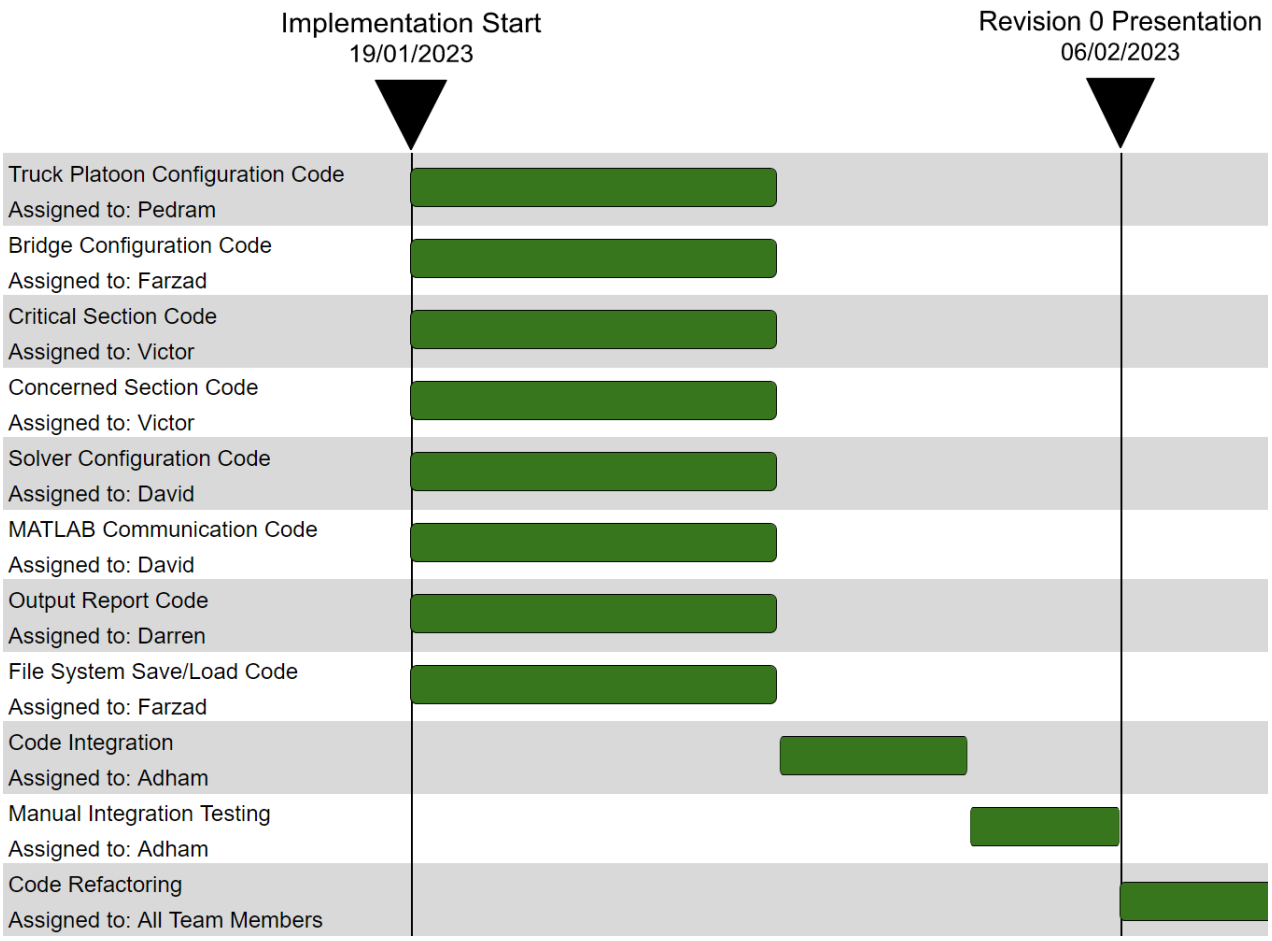

Figure 6: Project Timeline Section 1

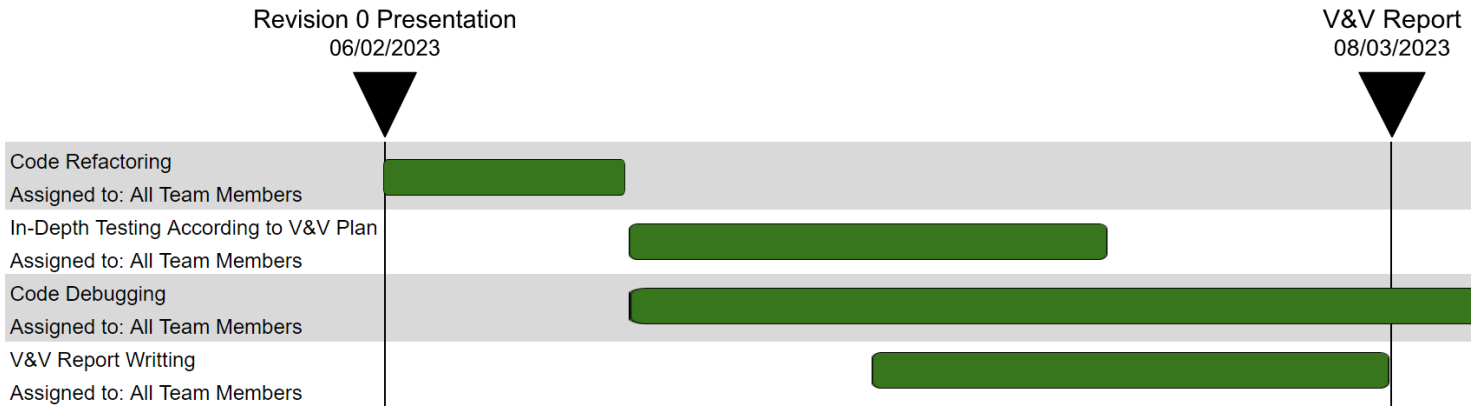

Figure 7: Project Timeline Section 2

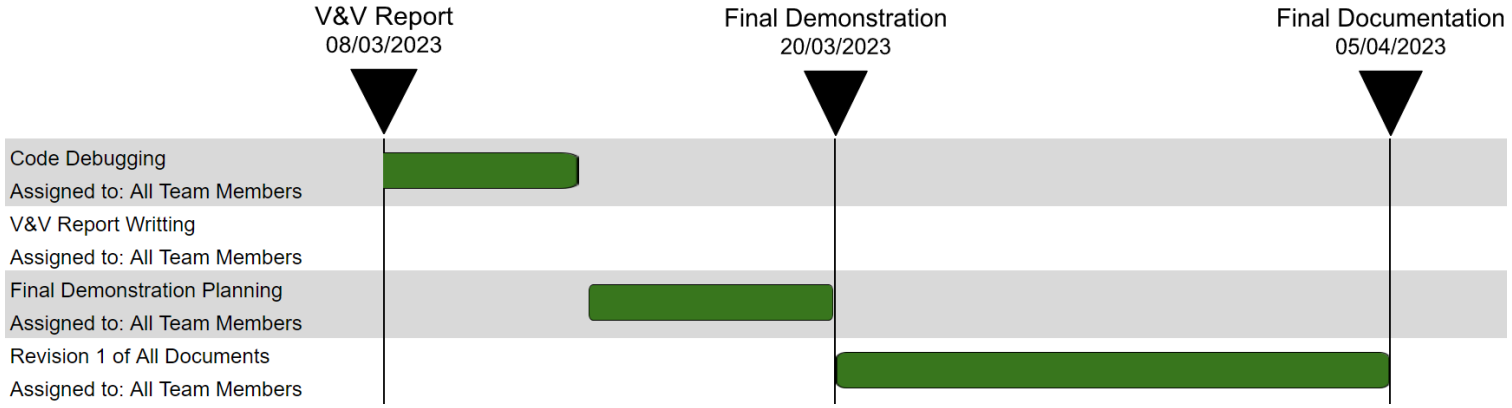

Figure 8: Project Timeline Section 3

A Interface

[Include additional information related to the appearance of, and interaction with, the user interface —SS]

B Mechanical Hardware

N/A

C Electrical Components

N/A

D Communication Protocols

N/A

E Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design. Please answer the following questions:

1. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

2. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select documented design? (LO_Explores)

Adham
I think the main aspect of the design that might be limited by time/resources is the presentation of the calculation results. In the worst case it may end up as just a 2D representation or graph, when ideally we would want a 3D fully animated visual of the platoon as it travels across the bridge and the effects it has over the course of its trip.

I think the main point of discussion as far as how to design the system was the level of modularity involved. As it stands, the project currently has 26 anticipated changes corresponding 1 to 1 to 26 modules. over the course of the design process this ballooned as high as 30+ and some suggested designs shrank it down to closer to 10. We decided to err more on the side of highly atomic modules in the end, for clarity and easy of understanding in the documentation, but still combined some modules that were extremely trivial to avoid

committing too dogmatically to a design principle even when it could be hurtful in practice.

Darren
Given time, could look at further configurability of platoons, e.g. accounting for trucks of different lengths, models, or wheel counts. Also for the animation to be a more faithful representation to better reflect the forces at play, such as alter the colors of the bridge display to reflect the stress on any given point.

Considered documenting further detail on subjects such as file I/O and graphics rendering. A generic visualizer module defining the exact nature of displaying a graphic to the screen was discussed to rigorously detail the design. We determined it was more important to focus efforts on the design of the system itself and the interaction of its parts rather than these modules to avoid reinventing the wheel.

Victor
I think one of the biggest limitations of our current design is the bridge input, as it allows for very little customization. That is, with more time we could implement a system that would allow the user to run calculations on more types and variations of bridges than the current design allows for.

We had considered designing our system so that it strictly followed the Model-View-Controller architectural pattern. While MVC would ease development and improve maintainability, we decided that strictly following it would limit our design freedom too much. We decided to use our current design as it follows some of the same patterns, allowing us to take advantage of many of the pros of MVC, while also giving us more flexibility.

David
I think, as Darren mentioned, the configurability of the truck platoon and the bridge could be enhanced greatly. For the bridge configurability, allowing for a larger number of bridge sections, defining bridge materials and section geometry could also be implemented. We could also enhance the way reports can be used, since they are essentially just a text file, where it could be made into a format that accepts pictures (or maybe it creates the report in pdf form).

The calculation call module was one that we considered different solutions for. A design solution to this was to have many external access programs, all of which would be a single call to the matlab engine. This would make the module simpler, because it would avoid all the conditional logic. However, we decided against this because it would've just shifted the responsibility to the users of this module. We instead decided to only have a single external

access program, with a unified input and output, and it's up to the other modules to decide how to use that output.

Pedram
With dozens of different Civil Engineering applications in the market, there is the opportunity to work on connecting our application with other platforms. This can be as simple as Import and Export functionalities or even adding our application as a plugin to their frameworks. The application, in theory, should also be capable of a cloud deployment where people can access through a webapp.

Creating an intuitive yet detailed UI with lots of options proved to be a main design challenge. The team debated on various layouts featuring design elements from animations, forms, tabs, and dynamic graphing. Members favored creating the UI from scratch as opposed to using partial templates to achieve specific functionalities enabled through a unique set of interactions between the graphs and the animations across all tabs.