SYSC 4907 A

Progress Report: A Web-based Platform for the Study and Analysis of COVID-19 Spread

Supervisor: Professor Gabriel Wainer

By: Tyler Mak (101108389) Andy Ngo (101132278)

Date: December 9th, 2022

1. Introduction

This report will be about the progress of the project for creating a web based application that is able to perform use-case specific simulations using an existing spatial-spread simulation model. This project is being supervised by Professor Wainer and Bruno St-Aubin, conducted by Tyler Mak and Andy Ngo both in Computer Systems Engineering. The report will be covering the project description, progress summary, problems encountered, variation to proposal, and future work on the project. Each part will include statements of what has been changed from the proposal report.

2. Project Description

This project focuses on optimizing the transfer of simulation results between the frontend and backend. Currently the frontend needs to download the entire simulation results log file in order to render the simulation but with websockets this issue will be solved. To implement websockets we planned for the frontend to send an initial message to the backend using the REST architecture. This initial message will tell the backend to start a websocket connection with the frontend. After the websocket connection has been made, the simulation results can be sent through line by line and rendered by the frontend the moment it is received. These improvements aim to decrease the time it takes for the simulation to start after the results are calculated, reduce the memory utilization on the frontend and improve overall user experience. Additionally, depending if we have enough time we are planning on making the simulation interactive such that the user can pause and modify the simulation parameters at any moment of the simulation. This would also be implemented using websockets but would require the simulator to handle different cases for each simulation and two way communication between the frontend and backend.

3. Progress Summary

The progress for the past couple of weeks focused on getting familiar with how websockets worked in Java, analyzing and understanding the backend Java code provided to us by Bruno and understanding the communication methods of websockets. Alongside learning about websockets we learned about dependencies with Maven, and the Spring Boot framework. Each week we had recurring meetings with Bruno where we had the opportunity to ask any questions and report our progress so far. At the beginning, the meetings were mostly about understanding the general purpose of each file and method in the code Bruno provided us with. After researching and going through examples of different Java websocket code, it was easier to decide which parts from the examples would be applicable for our project. The goal for this was to create a websocket that will send back the simulation results log file line by line to the requested frontend to render the simulation. Since the team is using agile development, tickets were created to make it easier to understand what the next task at hand was.

So far we have accomplished the ability to read the simulation results log file and can send the results line by line. We have also tested many different websocket implementations and have a working demo chat application using websockets. Currently we are working on creating a demo frontend that will use the REST architecture to send the initial message and receive the simulation results line by line. We are also working on integrating what we have learned from the different testings we have done to create a websocket connection. Additionally, we are working on creating a demo web page with some basic elements and the ability to send the initial message to the backend using REST architecture.

4. Problems Encountered

4.1. Multiple different implementation methods of websockets in Java

While researching for websockets in Java, we realized quickly that there were multiple different implementations and each had their own unique set of dependencies and libraries. These dependencies and libraries also did not always work for every version of Java. This meant there was a large amount of time spent learning a certain implementation of websockets that may not even work with the dependencies. As well as the Java version used in the project raised the possibility that it would not align with the code. To figure out which implementations worked and which would not, testing was needed to be done for each different implementation method. When testing out some of these different methods, it resulted in not being able to open a proper websocket connection to using a different IDE, NetBeans, to open up a server. Having the websocket implementation on NetBeans would be a bit troublesome when applying it onto the given code.

4.2. Creating a test Maven project with Spring Boot framework

Another problem we ran into was when trying to create a fresh Maven project for testing different websocket implementations, the project would not be able to run. We were given strange errors that could have been due to multiple different factors. Even after asking for help from Bruno, we were unable to create a fresh Maven project with the Spring Boot framework. Unless the example had included their Maven project and all other necessary files, it was not possible to test it alone by creating a new Maven project. Testing implementations had to be done by trying to implement them in the backend Java code provided by Bruno. This essentially meant that testing could only be done through the use of pre-existing Maven projects.

5. Variations to Proposal

The timeline we made in the proposal has slightly changed, plans have been pushed due to being busy from midterms the first part of November and working on assignments and other projects in other courses. We have also learned after research, we would need to send an initial message from the frontend using the REST architecture to create an entry point. The purpose of this entry point is to ensure that the frontend will send a message to the backend to initialize the websocket connection.
The timeline made in the proposal can be seen below in the Appendix as Figure 1, followed by the updated timeline can be seen in Figure 2.

6. Future Work on Project

Our next steps for the project include implementing a REST entry point for the frontend of the system to notify the backend to initialize the websocket connection. Implementing websockets using the knowledge we have gained from various testing and the streaming of simulation results once we have set up the websocket connection. If there is enough time we also plan on using websockets to create two way communication between the frontend and backend for interactive simulation. At the present time we are working on two tasks: a demo frontend web page that will use the REST architecture to send an initial message for setting up the websocket connection, and the implementation of websockets on the front and backend for streaming of simulation results.

7. Appendix

Month	Deadlines / Tasks
October	- Proposal (Due October 21, 2022) - Get familiar with Cadmium and Simulation web services
November	- No specific deadlines - Begin optimizing the transfer of results - Test code
December	- Oral Presentation / Progress Report (Due December 9, 2022) - Debug code / implement Java code to C++
January	- Changes to Oral Presentation / Poster Fair - Present Oral Presentation - Finish implementing code on C++ - Begin implementing interactive simulation
February	- Deadline to request for software/hardware needs (Due February 14th, 2023) - Test implementations for interactive simulation
March	- Draft Final Report (Due March 14, 2023) - Debug code and perform more tests
April	- Final Report (Due April 12, 2023) - Finalize code and clean up repository

Table 1: Project Deadlines and Tasks

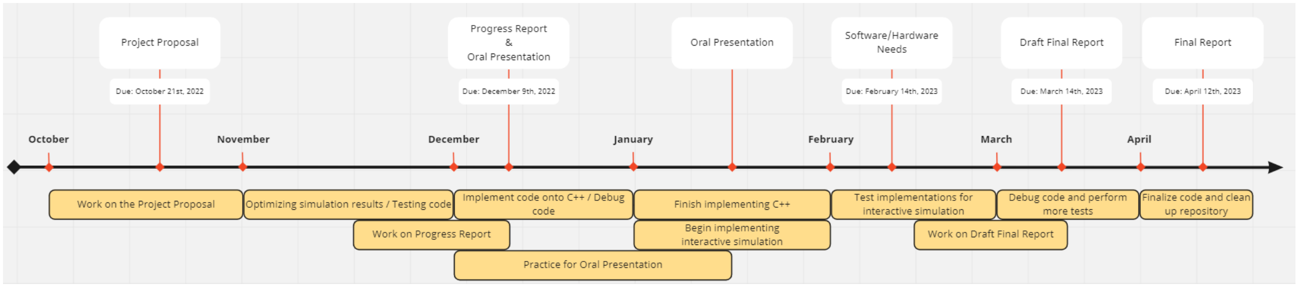
Project Proposal
Due: October 21st, 2022
Progress Report & Oral Presentation
Due: December 9th, 2022
Oral Presentation
Software/Hardware Needs
Due: February 14th, 2023
Draft Final Report
Due: March 14th, 2023
Final Report
Due: April 12th, 2023
October
November
December
January
February
March
April
Work on the Project Proposal
Optimizing simulation results / Testing code
Implement code onto C++ / Debug code
Finish implementing C++
Test implementations for interactive simulation
Debug code and perform more tests
Finalize code and clean up repository
Work on Progress Report
Begin implementing interactive simulation
Work on Draft Final Report
Practice for Oral Presentation

Figure 1: Project Timeline

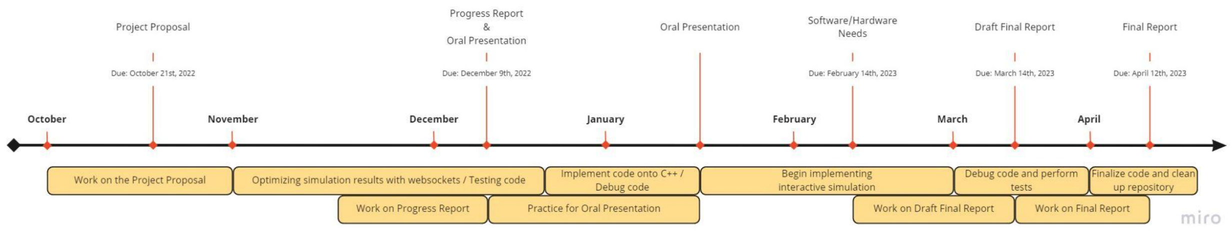
Project Proposal
Due: October 21st, 2022
Progress Report & Oral Presentation
Due: December 9th, 2022
Oral Presentation
Software/Hardware Needs
Due: February 14th, 2023
Draft Final Report
Due: March 14th, 2023
Final Report
Due: April 12th, 2023
October
November
December
January
February
March
April
Work on the Project Proposal
Optimizing simulation results with websockets / Testing code
Implement code onto C++ / Debug code
Begin implementing interactive simulation
Debug code and perform tests
Finalize code and clean up repository
Work on Progress Report
Practice for Oral Presentation
Work on Draft Final Report
Work on Final Report
miro

Figure 2: Updated Project Timeline