Carleton UNIVERSITY

# SYSC 4907 A, Fall 2020

## *Proposal / A Web-based Platform for the Study and Analysis of COVID-19 Spread*

Confidential

By: Paul Roode (101056469) ✉ Jacob Laboissonniere (101031913) ✉
Chang Qiu (100997188) ✉ Jon Menard (101086242) ✉

Supervisor: Dr. Gabriel Wainer ✉

Date: Nov 2, 2020

# Contents

List of Tables ..... ii
List of Figures ..... ii
1 Requirements and KPIs ..... 1
1.1 Functional Requirements ..... 1
1.2 Nonfunctional Requirements ..... 1
1.3 KPIs ..... 2
2 Background ..... 3
3 Action Items ..... 4
4 Relevance to Authors' Degree Programs ..... 5
4.1 Paul Roode – Software Engineering ..... 5
4.2 Jon Menard – Software Engineering ..... 5
4.3 Jacob Laboissonniere – Software Engineering ..... 5
4.4 Chang Qiu – Electrical Engineering ..... 5
5 Team Skillset ..... 6
6 Methodologies ..... 7
7 Timeline ..... 8
8 Risks and Mitigation Strategies ..... 9
8.1 Network Getting Hacked ..... 9
8.2 DDoS and Injection Attacks ..... 10
8.3 Insecure Data ..... 10
8.4 Overloaded Website ..... 11
9 Required Components and Facilities ..... 12
10 Authors' Contributions to This Proposal ..... 13

Confidential

# List of Tables

Table 1: Risk rating matrix ..... 9
Table 2: Heroku vs. AWS pricing ..... 12
Table 3: Authors' contributions to this proposal ..... 13

# List of Figures

Figure 1: Tentative project timeline ..... 8

Confidential

# 1 Requirements and KPIs

The purpose of the proposed project is to design and develop a web-based platform for the study and analysis of COVID-19 spread. The goal is to create a use case-specific, simulation-based web application to showcase the results of executing COVID-19 simulation models and other Discrete Event System (DEVS) models. To this end, an existing COVID-19 spatial-spread model and other DEVS models—including cellular DEVS (Cell-DEVS) models—will be utilized. The database that will house model data will use the PostgreSQL storage engine for data management. The existing Java/Spring MVC arslab-services codebase will be augmented to support database operations and expose a RESTful API for consumption by the front end.

## 1.1 Functional Requirements

The platform requires developers to first create a PostgreSQL database for storing the simulation models in the Advanced Real-Time Simulation Laboratory (ARSLab). Then developers must implement controller and service classes in the arslab-services codebase for routing and handling create, read, update, and delete (CRUD) operations to be executed on the database. If time permits, developers could implement a front end using HTML, CSS, and JavaScript. It would integrate existing front-end development efforts at the ARSLab. The functional system requirements include the following:

1. Middleware services shall automatically connect to the PostgreSQL database.
2. Users shall be able to download and visualize simulation models.
3. Only admins shall be authorized to view and edit database metadata.
4. Access permissions shall be implemented via authorization role contexts.

## 1.2 Nonfunctional Requirements

The nonfunctional system requirements are as follows:

1. The website shall have an upper user-base limit of 500, 5% (25) concurrent.
2. The cyclomatic complexity of the system shall not exceed three.
3. Unauthorized attempts to mutate model data shall be alerted to security admins.
4. Unsuccessful attempts by users to access model data shall be recorded on an audit trail.
5. Privacy of information, the export of restricted technologies, and intellectual property rights shall be audited.

Confidential

6. Passwords shall never be viewable at the point of entry nor at any other time; only temporary passwords shall be viewable by admins.
7. The platform shall be accessible to all ARSLab personnel.

## 1.3 KPIs

To measure progress toward satisfying the requirements and success criteria, developers will use the agile-scrum project management methodology to reflect on wins and losses and foster continuous improvement. To this end, Slate will be used to build API documentation while sprint backlogs are managed via a Kanban board on GitHub.

Confidential

# 2 Background

The ARSLab is a research laboratory which specializes in advanced modeling and simulations. ARSLab investigates automatic generation with the use of executable models. All their models are based on the DEVS specification.

When a model is created, it needs to be stored so it can be used again in the future. Currently, ARSLab stores the information of a model in an XML file. The XML file is designed to be parsed to extract information about the model. To add to this, when a model is run, the output or results need to be manually recorded in a separate file. While this does work, it is not very efficient. Every time information about the model is needed, the entire file needs to be parsed. Simple information such as who created the model, what the model is about, or what other sub-models are being used, is not easily obtained.

The proposed solution is to implement a database which will store all the information about a model as well as the results. Additionally, new services and controllers will be created to interface with the database using JSON for data serialization. A user will easily be able to query a model and have its execution results available instantaneously. This will not only allow for information about a specific model to be available, but also allow for cross-examination between different models.

Confidential

# 3 Action Items

The team endeavors to retrofit the existing arslab-services codebase to enable the storage of DEVS models, which entails

- devising and implementing a normalized database schema comprising new entities and relationships for DEVS models,
- creating new services for handling CRUD requests for these entities,
- creating new controllers for dispatching the request payloads,
- verifying—through rigorous testing—that KPI success metrics are satisfied by the implementation,
- producing documentation (e.g., API spec, UML diagrams, README) to facilitate the onboarding of future contributors, and
- documenting the aforementioned activities in the team's `blog`.

Confidential

# 4 Relevance to Authors' Degree Programs

## 4.1 Paul Roode – Software Engineering

This is a software development project, and as such, is wholly relevant to Paul's pursuit of a BE in Software Engineering.

## 4.2 Jon Menard – Software Engineering

Being a Software Engineering student means not only working with computers and codebase but also using the knowledge of being an engineer to design and build technology. This project consists of working together to design an optimal solution to ARSLab's DEVS storage/ information problem. To complete this task, computer software must be used to develop a database and a Java project. Using proper programming practices the system will improve output results and create a better solution than the previous generation.

## 4.3 Jacob Laboissonniere – Software Engineering

This project directly relates to the Software Engineering program, as it is a software development project. The skills required to undertake this project such as database design and Java development, are highly relevant to software engineering.

## 4.4 Chang Qiu – Electrical Engineering

As an Electrical Engineering student, I have been working on my programming skill with C++, Java, and Python. Also, I have self-taught SQL for manipulating the database. I have implemented a data structure with the programming language. In this project, I could use my knowledge in SQL to develop the database. And if time permitted, I could use my front end experience to build the front-end development API.

Confidential

# 5 Team Skillset

The Java/Spring MVC codebase is deployed on AWS—Paul and Jacob have some experience using AWS, and worked together in SYSC 3110 (Software Development Project) to iteratively and incrementally develop an event-driven Plants vs. Zombies game from scratch using JavaFX, MVC, and TDD (augmented—via serialization—with a level editor and more). Paul and Jacob also TA'd SYSC 2004 (Object-Oriented Software Development), which is a Java course.

Jacob also has experience designing normalized PostgreSQL database schemas, crafting efficient SQL queries, and developing RESTful applications using Ruby on Rails.

Paul has development experience in the domain of AI trust, having worked as a developer for a year at an AI and consulting firm that partners with governments, technical experts, industry, and entrepreneurs in providing solutions that deliver principled AI to their clients and to global society; more specifically, he helped develop an open and versatile framework for measuring and managing the trustworthiness of machine learning algorithms. Paul may also leverage his chemical engineering experience, having worked as a process specialist for a year at an ethanol plant and as a chemical engineering intern for two summers at a food processing plant.

Jon currently works at Statistics Canada and is on the Database Development team and has gained experience working with and setting up databases. Jon also has lots of experience writing efficient queries with SQL.

Chang has some web dev experience in the front-end using HTML5, CSS3, and JavaScript framework, such as Vue and React. Chang also has experience with the SQL database to retrieve specific data.

Confidential

# 6 Methodologies

The team will use the agile development methodology to develop the database schema and the corresponding CRUD application. Agile is based on the idea of iterative development and will enable the team to rapidly develop iterations on the project's deliverables.

More specifically, the team will use the Scrum subset of agile development. Scrum is a process framework for agile development that increases productivity and enables effective collaboration between team members.

Members of the team that are studying Software Engineering have previously taken SYSC 4106 (Software Product Management), in which the method of agile development was taught.

The team will use GitHub to manage the codebase of the project, and changes made to the codebase will involve a code-review process (also known as peer-review) to ensure code quality.

Team members have previously used GitHub for code reviews in courses involving group projects such as SYSC 3110 (Software Development Project).

Confidential

# 7 Timeline

The team is currently working toward finishing the project proposal which is due on November 2, 2020 at 12:00pm. After the proposal, the team's main goal will be setting up the structure and schema for the database. Determining a proper database structure will be an important task to make sure the right project foundations are set. By the end of November, the database should be set up and focus can be moved to developing a database connection using Java.

In December it is expected progress will slow as the fall term concludes, however, work will continue on connecting the database as well as setting up a test Java project from the existing arslab-services codebase.

At the beginning of January, the creation of test CRUD requests will become the main focus. The team will also be preparing the progress report which is due on January 20, 2021. If everything has gone to plan, a normalized database schema, which can be connected to through Java, will be successfully implemented. The project will be able to insert, update, delete, and run queries using test requests.

After the progress report, working on creating formal queries for the final version will be the main priority. At this time, the project will be mostly complete and writing the final report can begin. The draft final report is due on February 26, 2021. After the draft is submitted, focus will shift to the oral presentation on March 15, 2021. Once the oral presentation is completed, the codebase will be organized, documentation will be completed, and the final project will be exhaustively tested. The project will be finalized and made ready for presentation. Lastly, the final project report will be submitted before April 9, 2021.

The tentative project timeline is summarized in Figure 1 below:

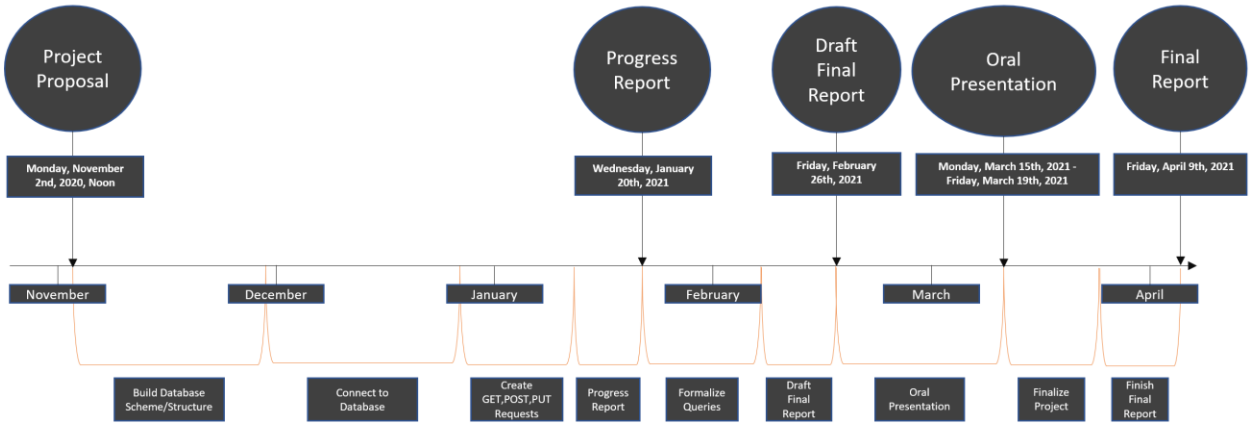


***Figure 1:*** *Tentative project timeline.*

# 8 Risks and Mitigation Strategies

This risk analysis aims to identify all important risks to the cloud-based system and provide comprehensive mitigation strategies so that there is very little residual risk. Note that some mitigation solutions cannot yet be feasibly implemented given the current project scope and budget. Risk ratings were evaluated using Table 1 below:

***Table 1:*** *Risk rating matrix.*

| **Likelihood ↓** | **Impact** | | | |
|---|---|---|---|---|
| | **Insignificant** | **Minor** | **Moderate** | **Major** |
| **Rare** | Very low | Low | Low | Moderate |
| **Unlikely** | Low | Low | Moderate | High |
| **Likely** | Low | Moderate | High | High |
| **Almost certain** | Moderate | Moderate | High | Critical |

## 8.1 Network Getting Hacked

**Risk rating:** High (Unlikely × Major)

**Mitigation:** AWS-provided safeguards:

Already implemented:

- security groups (virtual firewalls) for defending component instances (including relational database service instances) against intrusion
- network access control lists (ACLs) for tightening subnet security

To be considered:

- a network address translation gateway for blocking inbound traffic to private instances while allowing outgoing connectivity for updates
- a hardened bastion host for minimizing the chances of penetration
- AWS Shield for network- and transport-layer protections as well as additional detection and mitigation against distributed denial-of-service (DDoS) attacks

The existing security groups and network ACLs are deemed sufficient given the current project scope and budget—the latter three provisions are for future consideration as the system is scaled out. Time permitting, or if needed, vulnerability scanning and penetration testing of the network can be performed, followed by corrective action to secure the network.

## 8.2 DDoS and Injection Attacks

**Risk rating:** High (Unlikely × Major)

**Mitigation:**

- To prevent SQL injection, users will not be permitted to input SQL statements.
- Unfortunately, there is currently no plan to address DDoS attacks. In the future, the AWS Shield DDoS protection service could be considered for detection and mitigation against DDoS attacks, and could be integrated with a web application firewall for automated protection against web exploits (SQL injection, cross-site scripting, etc.) to preserve system availability and security and to prevent excessive resource consumption.

## 8.3 Insecure Data

**Risk rating:** High (Unlikely × Major)

**Mitigation:**

- IP access to database servers will be restricted to the web server.
- Automized database snapshots to mitigate the risk of database corruption, but there are monetary costs associated with the amount of backup storage used and the amount of backup data restored.
- To avoid account hijacking, user pools can be augmented with two-factor authentication using an authenticator app.
- Application-level encryption can be used to encrypt data before it is stored in a database, thereby preventing snooping and tampering between the application and the encryption routines on the database.

Confidential

## 8.4 Overloaded Website

**Risk rating:** Moderate (Likely × Minor)

**Mitigation:**

- The QA plan will include stress testing, and specific metrics will be established for the break threshold.
- Budget permitting, autoscaling can be used to scale task capacity up or down automatically based on demand and to detect when tasks are unhealthy and replace them automatically to preserve reliability. Load balancing can also be used to distribute incoming traffic across tasks. Basically, autoscaling and load-balancing services can be configured to ensure that a healthy number of tasks (which run component containers) are maintained behind load balancers. Further, these tasks can be placed in multiple availability zones to improve fault tolerance.

An overloaded site is a very relevant risk because of the possibilities of never-ending simulations and students running simulations in simultaneity. Unfortunately, the current budget does not permit redundancy nor load balancers, but the team will endeavor to test system robustness and stability under extreme loads.

Confidential

# 9 Required Components and Facilities

In order to develop and test the new web services and updated database schema, the team requires both a web server and PostgreSQL database. The web server will host the Java application that handles RESTful web requests, and the PostgreSQL database will store data corresponding to the updated schema.

The team plans to use Heroku to host the web server and database, as it provides an easy-to-use platform that will allow the team to focus on development rather than server administration. Heroku offers hobby tier web servers and databases that will be sufficient for the development requirements.

Both Heroku and AWS were considered as candidates to host the servers. While pricing is fairly similar for both, Heroku's features, such as auto deployment from GitHub and automatic updates, make it the preferred option. Prices were evaluated using Table 2 below:

***Table 2:*** *Heroku vs. AWS pricing.*

| **Component** | **Heroku – Hobby Tier (USD/mo)** | **AWS (USD/mo)** |
|---|---:|---:|
| Web server | 7.00 | 3.90 |
| PostgreSQL database | 7.00 | 13.50 |
| **Total** | **14.00** | **17.40** |

Confidential

# 10 Authors' Contributions to This Proposal

The authors' individual contributions to this proposal are shown in Table 3 below. Note that sections *Relevance to Authors' Degree Programs* and *Team Skillset* are omitted from the table, as it should be obvious that each author would contribute to the parts of those sections that are about themself.

***Table 3:*** *Authors' contributions to this proposal.*

| **Author** | **Contributions to This Proposal** |
|---|---|
| Paul Roode | Consolidation/formatting of this document, Action Items, Risks and Mitigation Strategies |
| Jon Menard | Background, Timeline |
| Jacob Laboissonniere | Methodologies, Required Components and Facilities |
| Chang Qiu | Requirements and KPIs |

Confidential