Quarantine Overseen
Software Design Specification

Y. Gao (yg), J. Johnson (jj), M. El Jamal (mj), N. Rudolph (nr), T. Wong (tw) - 6-01-2020 - v1.4

Table of Contents

1. SDS Revision History

Date	Author	Description
5-11-2020	nr, tw	Created the initial document outline
5-14-2020	nr, tw	Continued to work on the document - created section 4, 5, 6
5-15-2020	nr, tw	Completed the rest of the document, marked as v1.0
5-27-2020	tw	Overhauled System Overview, marked v1.1
5-29-2020	tw	Overhauled System Architecture with Diagrams, marked v1.2
5-30-2020	nr, tw	Started rewriting Software Modules
5-31-2020	nr, tw	Finished Software Modules, marked v1.3
6-01-2020	nr, tw	Finished Dynamic Models of Operation, marked v1.4

2. System Overview

This system is intended and designed to provide features of contact tracing to all registered clients, and to also provide insight on risks of infection based on proximity and contact history.

The target will be achieved through the ability to collect multiple clients' geospatial data, storing the collected data, determining other clients within close proximity, and allowing clients to report themselves as infected or not infected. The system will then broadcast risk alerts to clients who may have been in close contact with those who reported being infected.

The system will consist of the following modules, developed upon the data collector from Project 1:

1. Web Client with Geospatial Data Collector
    - This component will provide a web UI, which also allows collection of geospatial and time data from multiple clients in a specified time interval in the background.

2. Contact Tracing Data Storage
    - The data storage will be visible to all clients, and holds the ability to store all necessary data for all modules.

3. Proximity Calculator
    - This will provide the calculation of proximity between clients periodically.

4. Contact Risk Alerter
    - The alerter will fulfill its purpose of allowing clients to report themselves as infected, and to also broadcast risk alerts to clients who may have come into contact with clients who reported themselves as infected.

3. System Architecture

3.1 Components

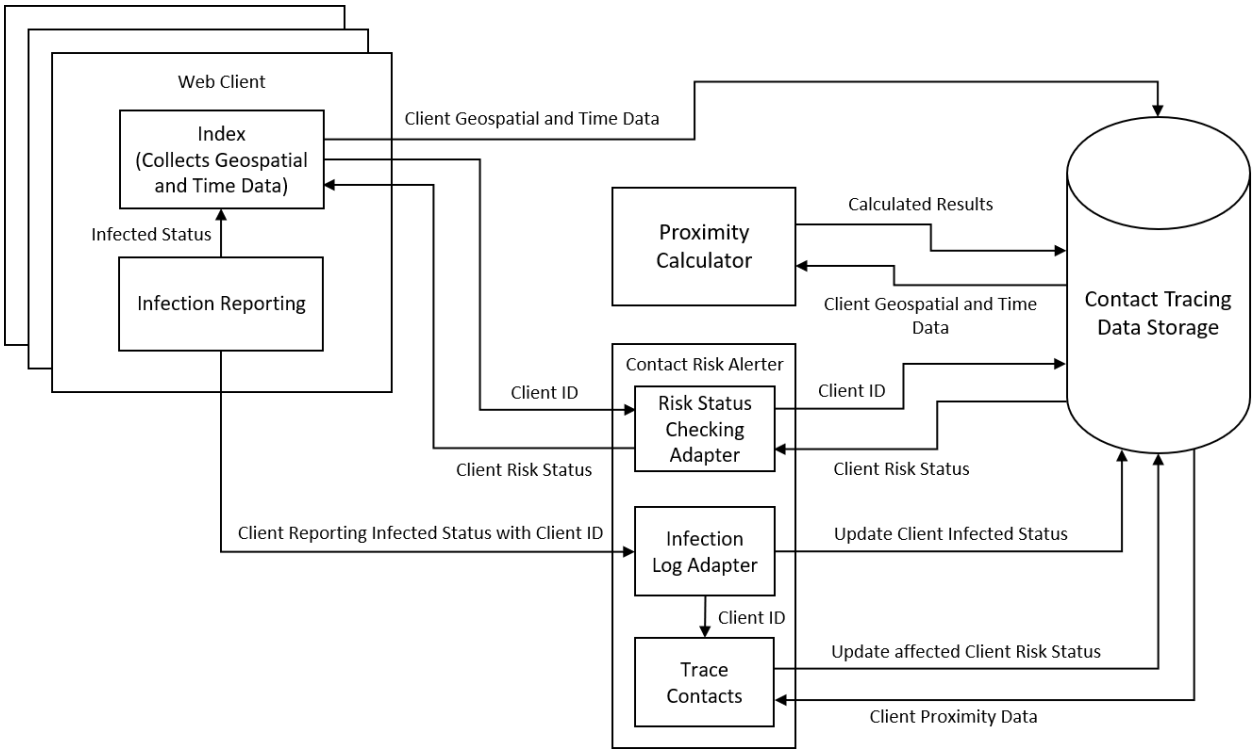

Figure 1 System Architecture Diagram

1. Web Client
    - This component will provide a web client, which also allows the collection of geospatial and time data from multiple clients in a specified time interval in the background.
    - All data collected by this component will be sent and stored in the data storage [2]. The data could be further used by other components for different functionalities.
2. Contact Tracing Data Storage
    - The storage will be accessible for all clients have the ability to store all necessary data for all modules, such as the following:
        - Geospatial and time data of all clients
        - Proximity data between users
        - Clients that were reported infected
        - Clients that are at risk

    - All data stored in this component will be accessible to all clients, and data will be sent and received by all other components for different functions.

3. Proximity Calculator
    - This component will provide the calculation of proximity between clients.
    - Geospatial and time data will be retrieved from the data storage [2] for component's proximity calculations, and results would be sent to the data storage [2].

4. Contact Risk Alerter
    - The alerter will fulfil its purpose of broadcasting risk alerts to clients who may have come into contact with clients who reported themselves as infected.
    - This component will interact with the data storage [2] in order to retrieve and send data related to infected clients, and clients in high risk.

3.2 Rationale

The target of our designed system is to provide functionalities of contact tracing, where it is developed upon our project 1, a geospatial and time data collector incorporating the use of HTML, JavaScript, PHP, and MySQL.

For this system, we are introducing new features, such as the ability for a client to report themselves infected, and broadcasting high risk alerts to clients who had been in close contact with those who reported themselves infected. In order to provide the described features, massive amounts of calculations have to be performed in order to obtain clients' proximity data, where C/C++ would be the optimal choice to implement the calculations for its efficiency.

However, re-implementing the data collector from project 1 with C/C++ could lead to uncertainties, and incorporating new components written in different languages could introduce compatibility issues with already-existing components.

Furthermore, since we already incorporated the use of PHP, this eliminates the unnecessary complexity of incorporating even more programming languages. In addition, where PHP being a server-side scripting language, removes the burden of massive calculations on the clients' end.

Therefore, our team decided to continue implementing the new features upon the use of PHP, and incorporating the use of HTML, JavaScript, and MySQL.

4. Software Modules

4.1. Web Client

Role & Primary Function

The module entails numerous PHP files with inline HTML and JavaScript responsible for retrieving and relaying geospatial data from the user to their MySQL database. This is accomplished all while providing the user with a convenient interface that allows them to switch between the home page and infection reporting page.

4.1.1 Homepage

This page is the first interface the user encounters after initial connection with the web client being established, and provides a clear depiction of their current risk of infection. When an infected user comes in close proximity to our healthy user, the green box that highlights the robust risk status of the individual will swap to red and change the text respectively. This page also offers brief insights on the user's current geospatial and time data alongside their client ID.

In addition, this page also collects geospatial and time data from all clients, which will be sent and stored in the *Contact Tracing Data Storage* module.

Interface Specification

In ten second aggregations, this page acquires the user's geospatial data and forwards this information to the *Contact Tracing Data Storage* module. Furthermore, a navigation toolbar is placed at the top of the page allowing the user to navigate to the "Infected" page with ease.

All collected data will be forwarded towards *Contact Tracing Data Storage* with the following method and format:

```
"INSERT INTO geospatial_time_tracking (client_ID, tracking_time,
              longitude, latitude, time_duration)"
```

4.1.2 Infection Reporting

The infected page allows users to report themselves as infected with COVID-19. This action effectively marks them as contagious to all other users and alters their homepage to encourage the individual to maintain a safe distance from others. In addition, users are also able to report themselves as not-infected.

Interface Specification

Having reported yourself as infected, the webpage prompts a request to the database, using their tracking ID as their key, and changes their status to infected. Although not directly concerned with this module, the action of marking a user as infected allows other components to calculate a user's risk with respect to their proximity to the infected individual(s). Analogous to the homepage, the infected page also places a navigation toolbar at the top of the page for easy navigation to the "Homepage".

Static Model

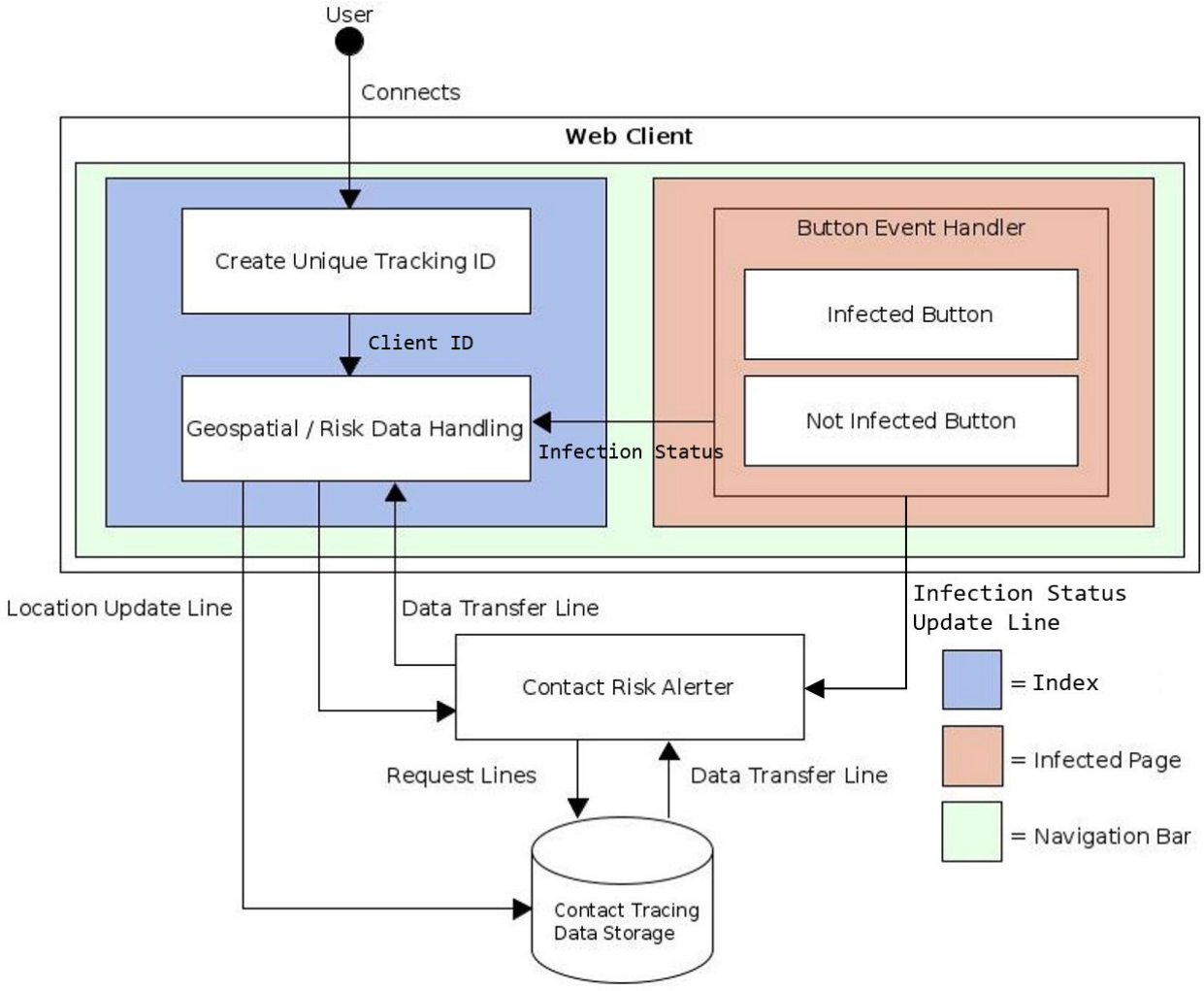

Figure 2 Static Model of Web Client

Dynamic Model

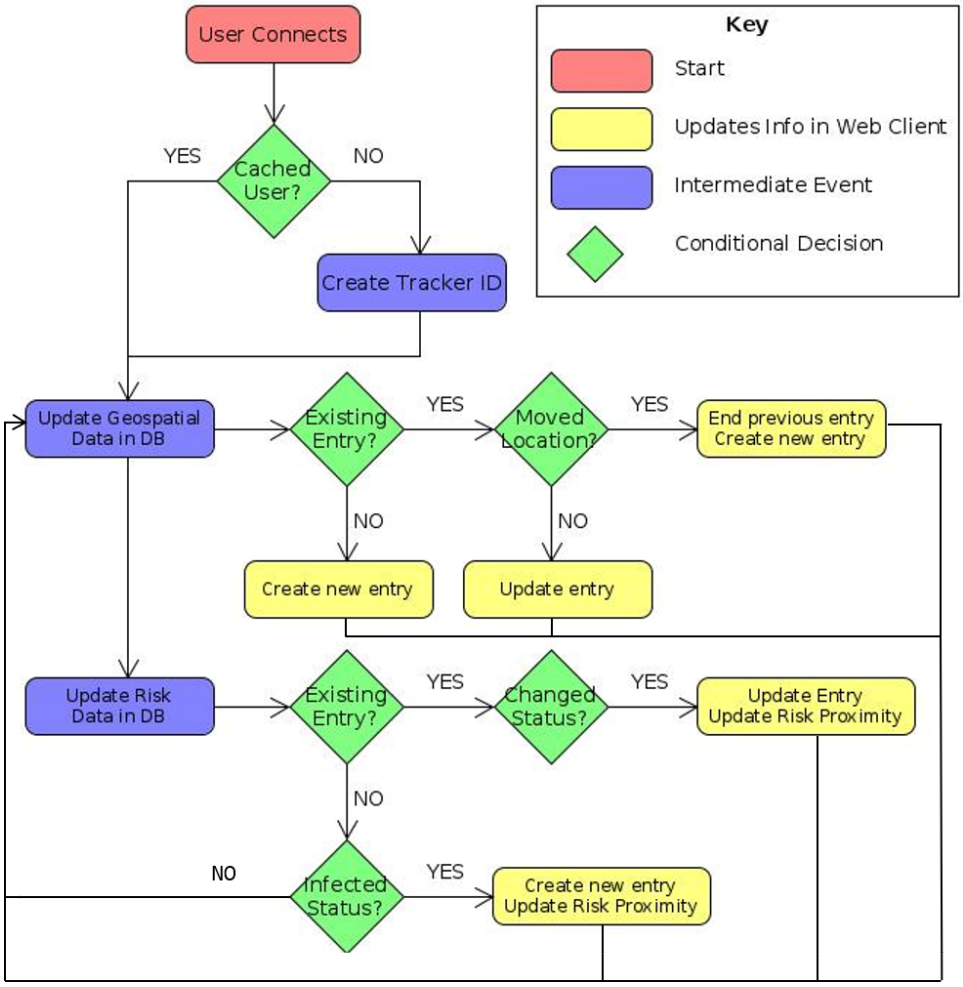

Figure 3 Dynamic Model of Web Client

Design Rationale

Since we are providing 2 major features of data collection, and clients reporting themselves infected; Hence, we decided to create 2 different sub-modules to fulfill each of the respective features. This was chosen to keep the organisation in check for our module, and the ability to perform maintenance or improvements without substantially disrupting other features and creating chaos. This was exceptionally important since we are developing on top of Project 1's data collector.

Alternative Designs

An alternative design we considered was to include all functionalities into one single webpage, without using submodules. However, this creates complexity and branches out too many connections to other modules by itself. As mentioned above, since we are developing on top of Project 1's data collector, over-modifying existing code could be disastrous in the end.

4.2. Contact Tracing Data Storage

Role & Primary Function

The module contains all data in respect of clients' geospatial, time, infected status, and risk status attributes.

As our system is designed to be data intensive, the ability to maintain and implement new entries to our tables is integral to the functionality of our program. Therefore, we have decided to use MySQL services to store, update, and backup all data used within our software. This will be hosted within the user's server at *ix.cs.uoregon.edu*.

4.2.1 Geospatial and Time Data Table

This table is designed to store all clients' geospatial and time data.

Interface Specification

The format and content of stored data within this table should be:

```
TABLE `geospatial_time_tracking` (
       `client_ID`,
       `tracking_time`,
       `longitude`,
       `latitude`,
       `tracking_duration`
)
```

4.2.2 Proximity Data Table

This table is designed to store all clients' calculated proximity data.

Interface Specification

The format and content of stored data within this table should be:

```
TABLE `proximity` (
       `client_ID`,
       `target_client_ID`,
       `distance`,
       `time_of_contact`
);
```

4.2.3 Client Infection Status Table

This table is designed to store data of clients that reported themselves as infected.

Interface Specification

The format and content of stored data within this table should be:

```
TABLE `infected` (
       `client_ID`,
       `time_of_infection_reported`
);
```

4.2.4 Client Risk Status Table

This table is designed to store data of clients who had been in contact with infected clients.

Interface Specification

The format and content of stored data within this table should be:

```
TABLE `at_risk` (
       `client_ID`,
       `infector_client_ID`,
       `distance_at_contact`,
       `time_of_contact`
);
```

Static Model

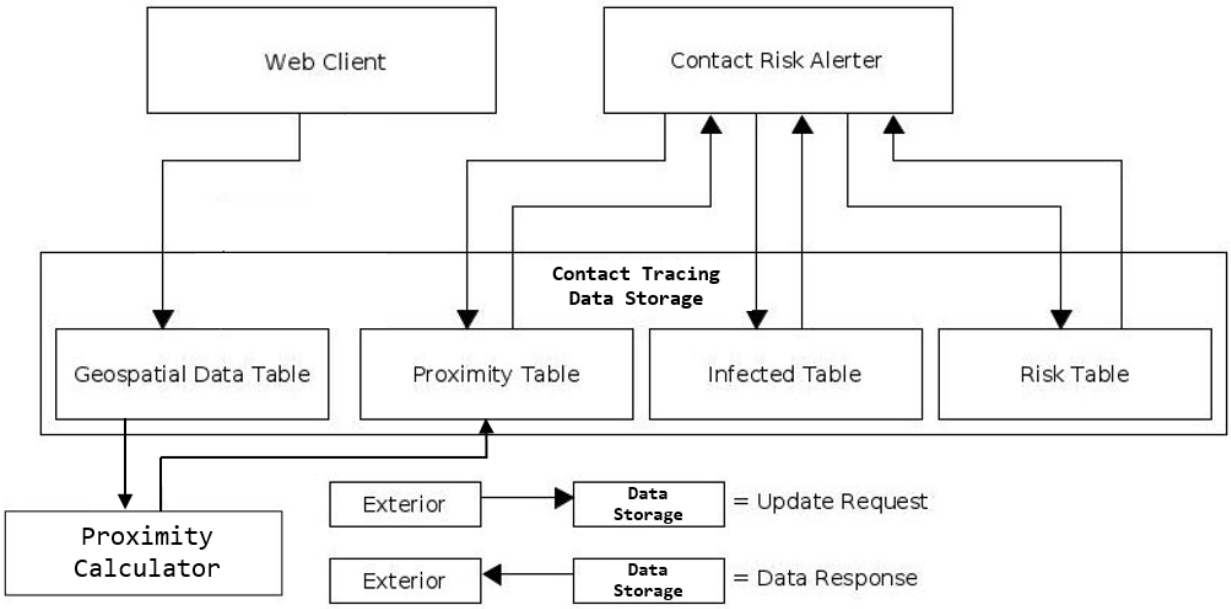

Figure 4 Static Model of Contact Tracing Data Storage

Design Rationale

Since our system is developed on Project 1's data collector using MySQL database for storage, we decided to add different tables into our MySQL database structure, in order to provide storage for newer features and components without massive modifications.

Alternative Designs

An alternative design that we considered, was to store different types of data in several plain text files. This was due to at one point of design, our team considered to rewrite parts of our code in C++ for its efficiency while performing large amounts of calculation. The consideration of plain text files was mainly because C++ requires installation of a MySQL database connector for C++ in the operating system. Since this would tremendously increase complexity, possible compatibility issues, and requires lots of rewriting of existing code; Hence our team retained the use of MySQL database and dropped the adoption of C/C++.

4.3. Proximity Calculator

Role & Primary Function

The purpose of the *Proximity Calculator* is to provide the functionality of calculating the distance between all clients, and all calculations will be done on the server-end, releasing the burden of calculations on the client's device.

This module will be triggered by users' web client periodically, initiating the search for other clients' nearby current location. The search calculates distances by comparing current and all other clients' locations from *Contact Tracing Data Storage*'s geospatial and time data table. All calculations will be stored in *Contact Tracing Data Storage*'s proximity data table, which would be later used by the *Contact Risk Alerter*, in order to determine if another client was within a close proximity.

Interface Specification

Calculated proximity data will be forwarded towards *Contact Tracing Data Storage* with the following method and format:

```
"INSERT INTO proximity (client_ID, target_client_ID, distance,
                        time_of_contact)"
```

Static Model

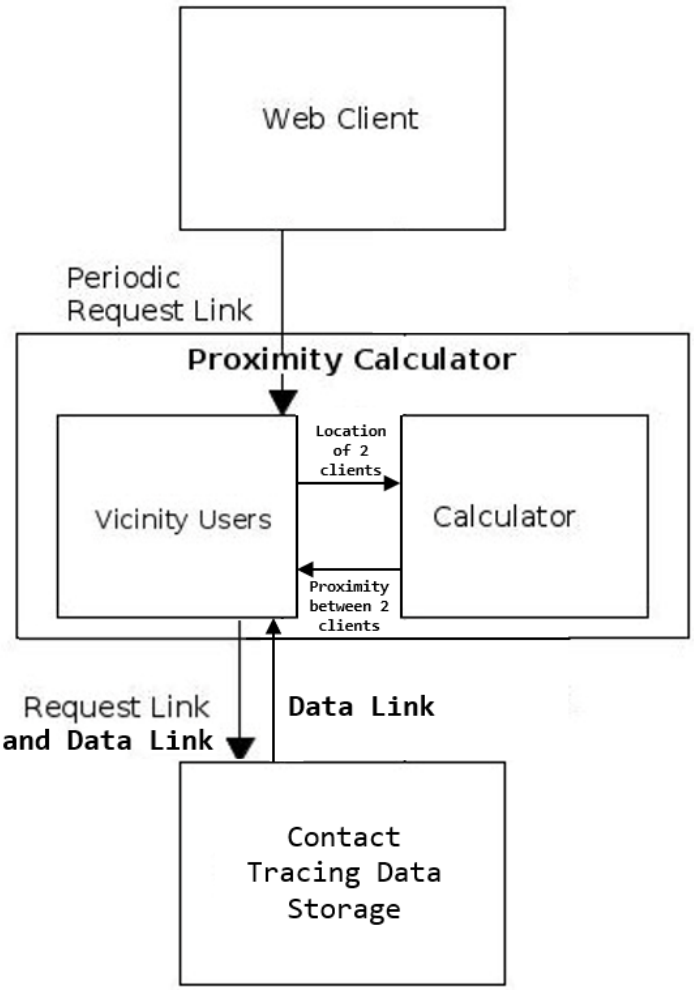

Figure 5 Static Model of Proximity Calculator

Dynamic Model

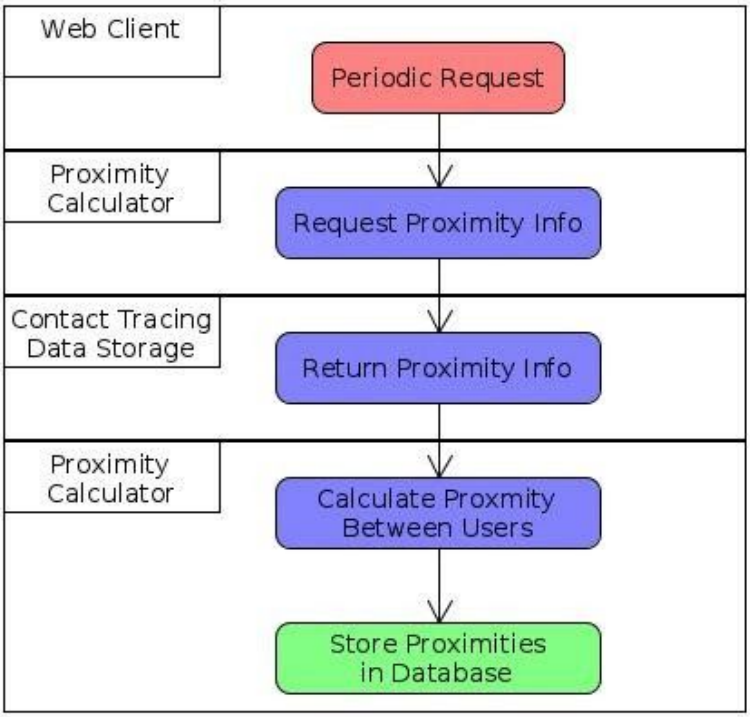

Figure 6 Dynamic Model of Proximity Calculator

Design Rationale

The module is designed using PHP and as an independent module due to the importance of the feature it is providing. Though this module could have been incorporated within other modules as a sub-module, we would like to maintain the organisation of the system, and avoid over-implementing features and executing too many operations concurrently in one sole module. Therefore it is optimal for the feature of calculation proximity to be implemented in the system as an independent module.

Alternative Designs

An alternative design we considered was the implementation of this module incorporating the use of C/C++, which would be ideal and optimal for efficiency while performing large amounts of calculation. However, as described from the *Alternative Designs* from *Contact Tracing Data Storage*, this will require the installation of a MySQL database connector for C++ in the operating system.

For this particular module, this would tremendously increase complexity, and introduce possible compatibility issues, therefore this design was not adopted.

4.4. Contact Risk Alerter

Role & Primary Function

The module is designed for processing infected and risk statuses of clients. The purpose of the *Contact Risk Alerter* is to provide the features of checking clients' *risk status*, and receiving and updating *infected status* of clients that reported themselves as infected or not infected. In addition, the module also traces all clients who had been in close contact with clients that reported themselves as infected, and all tracing will be done on the server-end.

4.4.1 Risk Status Checking Adapter

This sub-module will be triggered by users' web client periodically, requesting the client's risk status. The feature is achieved by receiving the *client_ID* from the *Web Client* periodically, which this sub-module parses the *client_ID* as a data request towards the *Client Risk Status* table in the *Contact Tracing Data Storage*. After the data is received from the *Contact Tracing Data Storage*, if there are no records of risk for the client, integer 0 will be forwarded to the web client. If there exists records of risk for the client, the infectors' client ID and times of contact will be parsed into a list of lists and forwarded to the *Web Client*.

Interface Specification

Risk status requests will be forwarded towards *Contact Tracing Data Storage* with the following method and format as an example:

```
 "SELECT * FROM at_risk WHERE client_id = '{$current_client_ID}'
                          ORDER BY time"
```

If at risk, risk status will be forwarded towards *Web Client* with the following list format:

```
[
      [[target_client_ID], [time_of_contact]],
      [[target_client_ID_2], [time_of_contact_2]],
      …
]
```

If not at risk, risk status will be forwarded towards *Web Client* with an integer "0".

4.4.2 Infection Log Adapter

This sub-module acts as an adapter or controller to update the client's infection status, and to initiate tracing contacts who came into contact with the reported client. The adapter will receive infection status updates with the *client_ID* of the infected clients, through the *Infection Reporting* sub-module of the *Web Client*. The sub-module will either update, or create an entry for recording the *client_ID* and time of report of the infected clients, into the *Client Infection Status* table of *Contact Tracing Data Storage*.

Interface Specification

Infected status updates will be forwarded towards *Contact Tracing Data Storage* with the following method and format:

```
"INSERT INTO infected (client_ID, time_of_infection_reported)";
```

The other sub-module *Trace Contacts* will be initiated through the use of internal function call, with *client_ID* being the sole parameter passed through.

4.4.3 Trace Contacts

After a client reports themselves as infected, this sub-module determines the risk status of other clients who have been in close contact proximity. When clients report themselves as infected, their *client_ID* will be passed from the other sub-module, *Infection Log Adapter*, into this sub-module, *Trace Contacts*. It then retrieves proximity data that contains the *client_ID* of the infected clients, from the *Contact Tracing Data Storage*.

The sub-module will then determine whether or not the distance between the infected clients and all others is within the configured proximity of the infected client. If yes, update the risk status of the other client in the *Client Risk Status* table in *Contact Tracing Data Storage*.

Interface Specification

Client proximity data requests will be forwarded towards *Contact Tracing Data Storage* with the following method and format as an example:

```
        "SELECT * FROM proximity WHERE client_ID =
'{$current_client_ID}' ORDER BY time_of_contact ASCENDING"
```

Client risk status data will be forwarded towards *Contact Tracing Data Storage* with the following method and format:

```
"INSERT INTO at_risk (client_ID, infector_client_ID,
      distance_at_contact, time_of_contact)"
```

Static Model

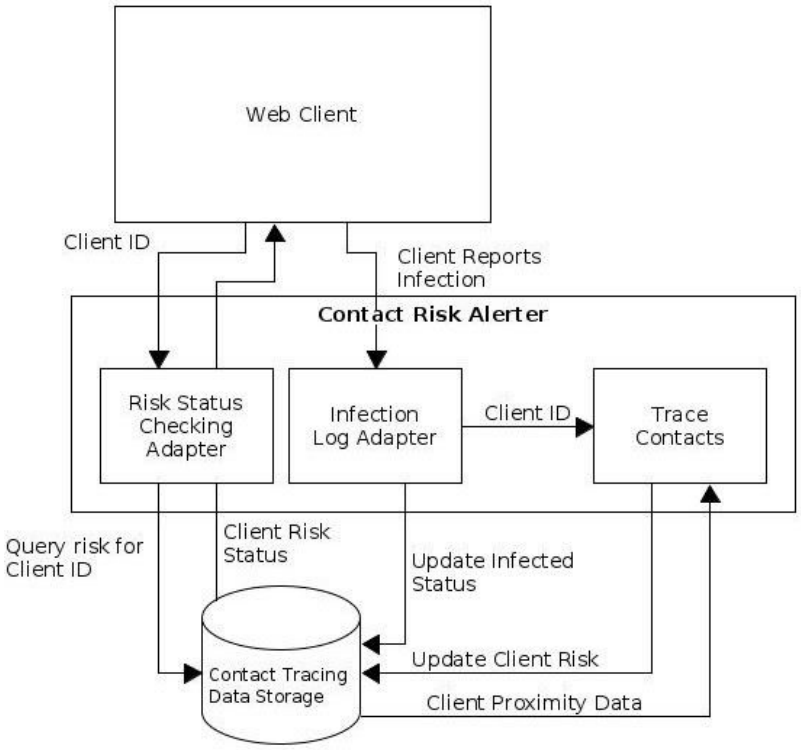

Figure 7 Static Model of Proximity Calculator

Dynamic Model

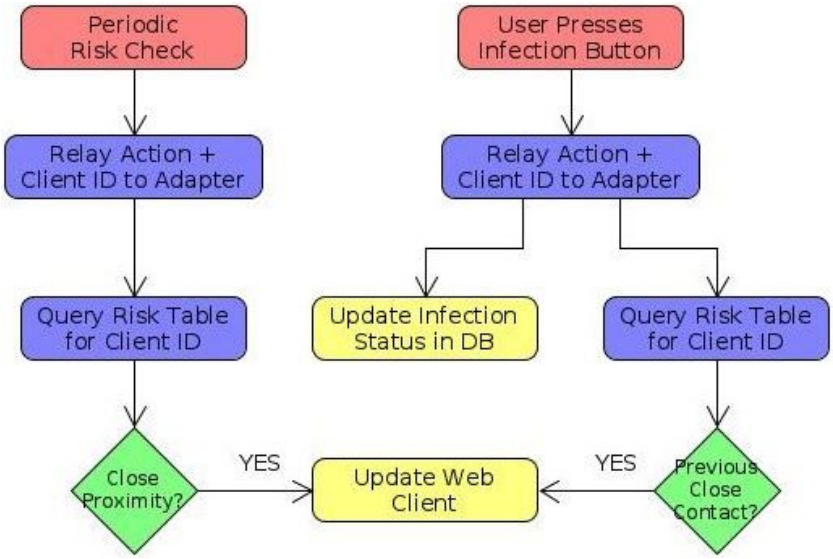

Figure 8 Dynamic Model of Proximity Calculator

Design Rationale

This module is designed using PHP, which fulfills the features of processing infection reports, tracing clients who came into contact with infected clients, and providing risk alerts to those who were in close proximity with infected clients. In our personal evaluating perspective, the division of the module into 3 sub-modules fulfilling 3 features, was a positive effect in terms of organisation, yet not overloading the module.

Alternative Designs

An alternative design in our consideration, is the incorporation of both the sub-modules of *Infection Log Adapter* and *Trace Contacts* to be combined as 1 sub-module. However, this raised problems during the prototype stage, and the complexity and bloatedness of a single sub-module was difficult for us to test and debug. Such as changing the logic of how to trace contracts could break functionalities of the infection log adapter. Therefore, the method of divide and conquer in this case was a better choice.

5. Dynamic Models of Operational Scenarios (Use Cases)

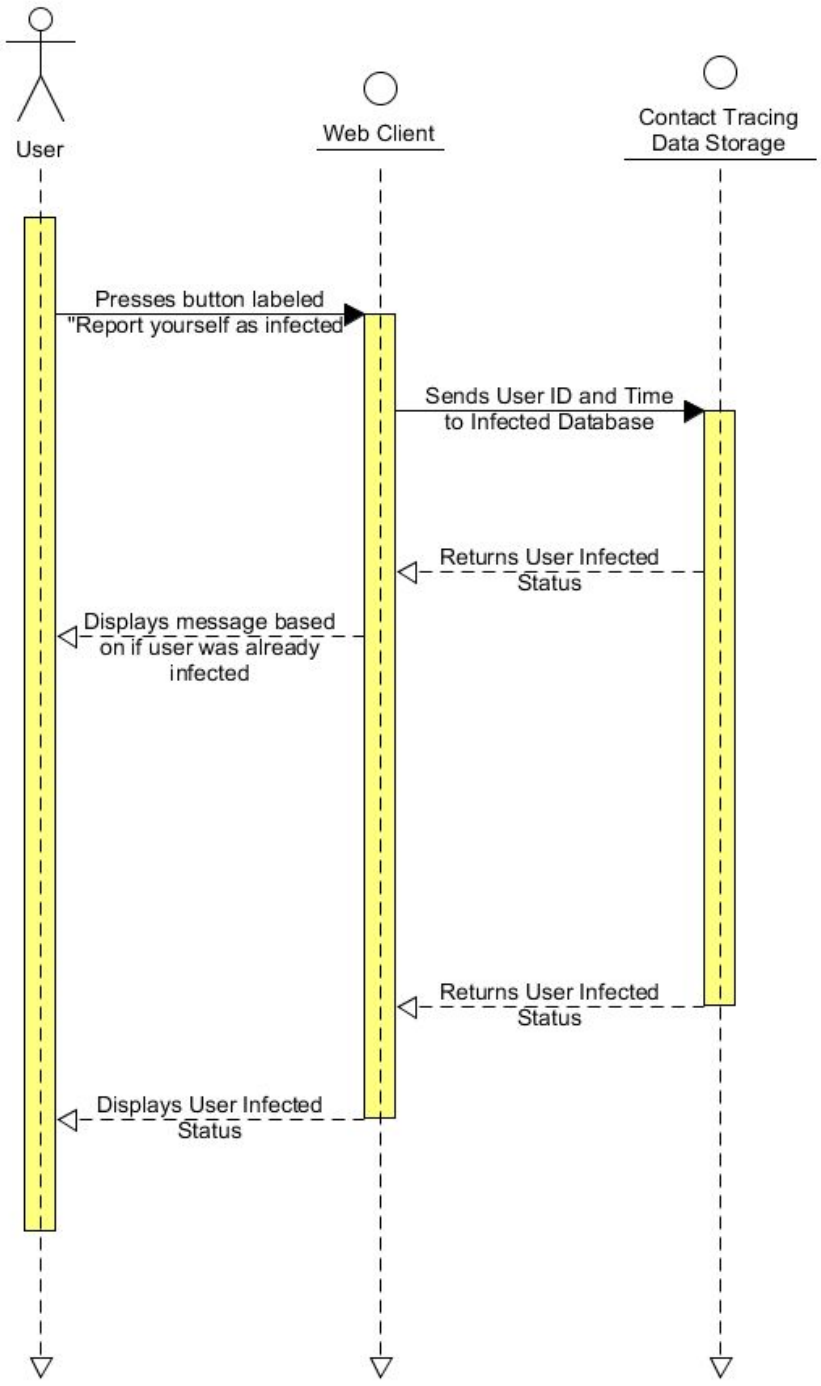

Figure 9 Use Case for Marking Self as Infected

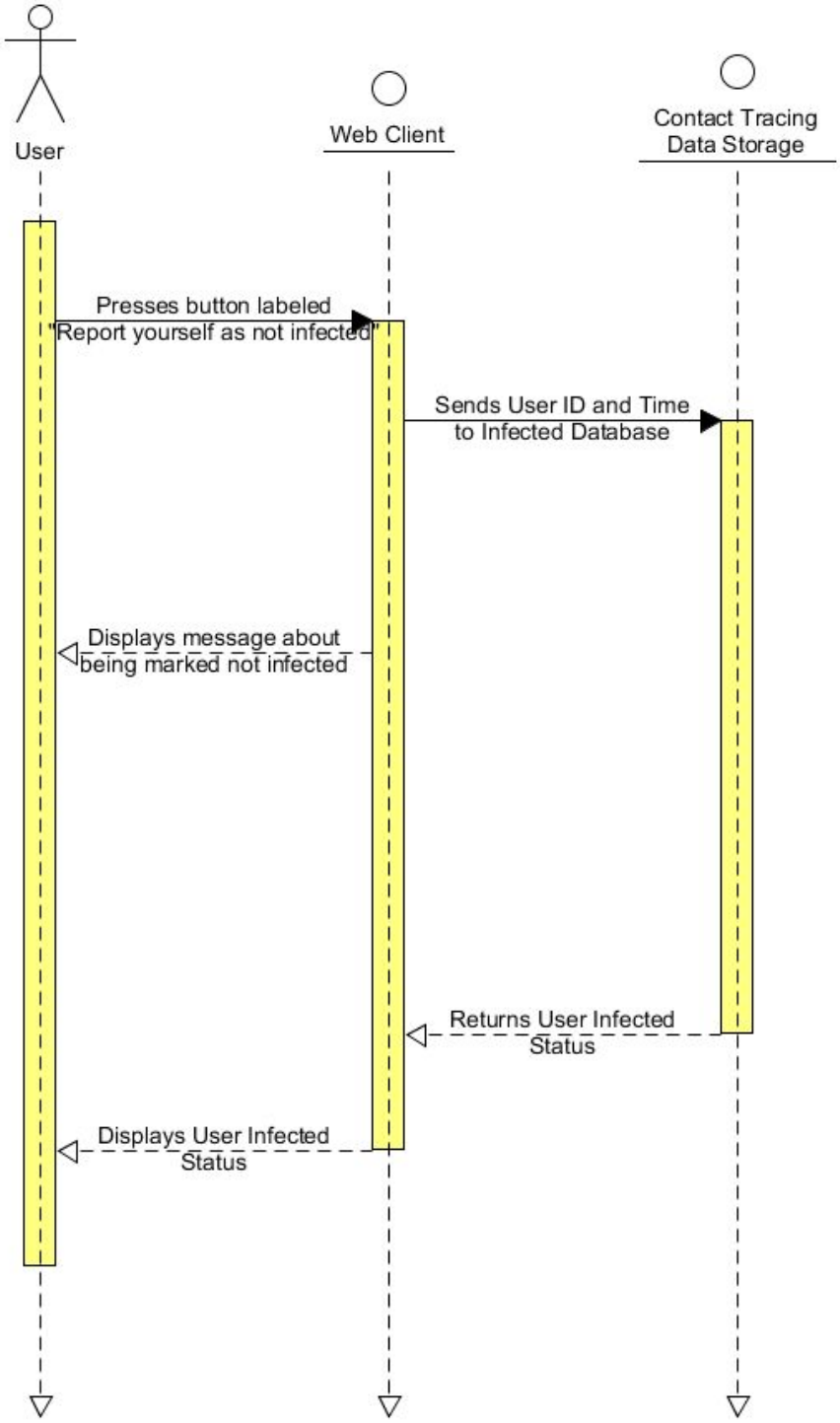

Figure 10 Use Case for Marking Self as Not Infected

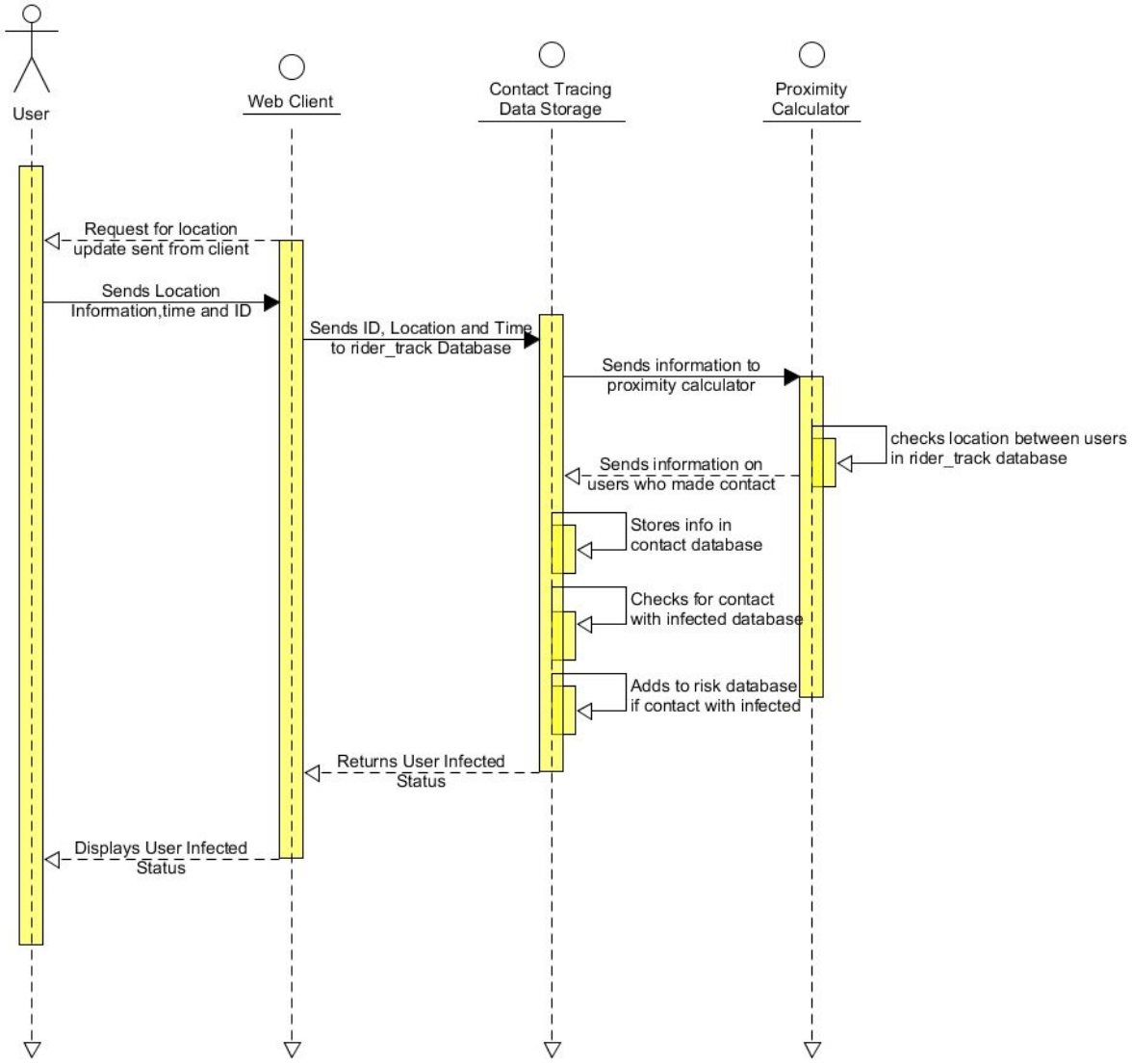

Figure 11 Use Case for User Moving to High Risk Location

6. References

IEEE Std 1362-1998 (R2007). (2007). IEEE Guide for Information Technology–System Definition–Concept of Operations (ConOps) Document. https://ieeexplore.ieee.org/document/761853

Faulk, Stuart. (2013). Understanding Software Requirements. https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf

Hornof, Anthony. (2019). Software Design Specification [Template]. https://classes.cs.uoregon.edu/20S/cis422/Handouts/Template-SDS.pdf

UMLet. (2017). Free UML Tool for Fast UML Diagrams. https://www.umlet.com/