



---

CAT304 - GROUP INNOVATION PROJECT AND STUDY FOR  
SUSTAINABILITY FINAL PROJECT REPORT

PUSAT PENGAJIAN SAINS KOMPUTER

---

UNIVERSITI SAINS MALAYSIA  
SEMESTER I 2020 / 2021

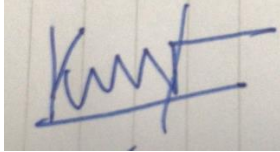
**Project Title:** JomTrace

**Group No:** 33

No.	Name	Matric No	Email Address
1.	Sharvin A/L Kogilavanan	148056	sharvin11@student.usm.my
2.	Katheeravan A/L Balasubramaniam	147744	katheeravan305@student.usm.my
3.	Jeevitan	146863	jeevittankrishnan@student.usm.my
4.	Divenesh A/L Shamugam	146311	diveneshbeez@student.usm.my

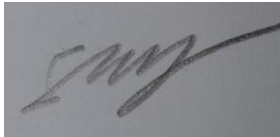
## Declaration Page

“We declare that all this submitted work is entirely our own except for those sources which we have referenced, and that it has not been previously submitted for assessment in any course”.



---

Katheeravan A/L Balasubramaniam



---

Sharvin A/L Kogilavanan



---

Jeevitan



---

Divenesh A/L Shamugam

## Table of Content

No	Content	Page
<b>1</b>	<b>Introduction</b>	
	Project Background	4 - 5
	Problem Statements	5 - 6
	Proposed Solution	6
	Project Objectives	6
	Benefit or Impact of The Proposed Solution	9 – 10
	Uniqueness of Proposed Solution	10
	Contribution	11
	Organization of the report	12
<b>2</b>	<b>Background Study &amp; Related Work</b>	
	Existing System	13
	Features Comparison of the Existing System	13 - 14
	Existing Technique / Algorithm / Method	14
<b>3</b>	<b>System Requirement and Analysis</b>	
	Status of project development	15
	Scope of proposed solution	15
	System Capabilities & Limitations	15
	Project Management	15 -16
	Development Methodology	16 -17
	Analysis of Proposed Solution / Project	17
	UML Diagram	18 - 24
	Technology Deployed	25 - 26
<b>4</b>	<b>System Design</b>	
	System Architecture	26 - 27
	System Components/Modules	27

	Database and Class Diagram	28 - 29
	Interface Design	29 - 33
	Input and Output Design	34
<b>5</b>	<b>System Implementation</b>	
	Module Implementation	35 - 45
	System Integration	45
<b>6</b>	<b>Testing and Evaluation</b>	
	System Testing	45 - 46
	Test Case	46
	System Evaluation	47
<b>7</b>	<b>Conclusion and Future Work</b>	47
<b>8</b>	<b>Appendix and References</b>	48

## **1.0 Introduction**

### **1.1 Project Background**

Many of us did not hear the word pandemic before until Covid-19 affected the whole world. Covid-19 is a viral infection caused by the SARS-CoV-2 virus. This virus spreads to humans through the air or through close contact with someone infected with Covid-19. This virus took tons of life around 5 million worldwide and have recorded more than 250 million confirmed cases. It prompted many governments around the world to consider ways to limit the spread of this virus. Almost every government in all the countries announced movement control orders with some restrictions but, the cases never dropped and increased in upcoming waves. It was because movement control order only focuses on not letting people go out, but still, one person from a family who goes out to buy groceries might meet a person who has covid and eventually spread to the rest of the family.

Thus, governments started paying more attention to contact tracing in the pandemic to identify the spreading users who spreads the virus, getting users who may get infected by Covid-19 by someone who has it, the super spreader, the chain of spreading etc. But instead of executing the conventional contact tracing method, the governments focused on digitalizing contact tracing, most importantly automating the process since we had hundreds of thousands of cases reporting every day [1]. Many countries started their research and development in building a contact tracing app using various technologies since there is no exact way to trace and track the disease. Due to the multiple research methods, there were variability in terms of the architecture of the contact tracing app in terms of infrastructure of the app, sensors used in mobile phones to track users' activity, and some made their applications open source as the rest remained as closed source [2]. Many countries have succeeded, and some failed in this league.

Firstly, in terms of the app's architecture there were two types of approaches, centralized and decentralized approaches [3]. In a centralized approach, if a user gets infected by Covid-19, the individual must upload his anonymous ID and anonymous IDs of people he met to the server to perform all the contact matching processes and pushing alerts. On the other hand, in a decentralized approach, if the person gets infected by covid he only must upload his anonymous ID to the server as the app will frequently download the list of anonymous IDs of infected people and does the contact matching in the app without uploading all the IDs of the people we have met throughout the time. Many countries have been developing their app based on decentralized approach. As for the others, they have been retaining with a centralized approach.

Secondly, the method is to track users' activity. Since the contact tracing app is a mobile solution, every phone comes with vital sensors such as GPS, Bluetooth, Gyroscope, accelerometer and etc. Before diving deep into the sensors used by the application, another important thing that needs to be known is the type of tracking method. Currently, there are two methods available one is location tracking and the second is proximity tracing. For location tracking, the GPS in mobile device is used to track the places a person has visited. If an individual is tested positive, people around the individual at the place and time will be notified to take a covid test. However, some contact tracing apps avoided the usage of GPS to track the places a person has visit. Hence, they used another approach which is called the check-in system. So, when a person visits a location, there will be a QR code in the entrance or somewhere around the visiting place that allows the people to scan to label them that they have visited that place. The check in method is widely used by Malaysians in an app called MySejahtera. The final tracking method used is proximity tracing using Bluetooth. When there are two persons, Person A and B meet the app on their phones exchange a temporary anonymous ID via Bluetooth signal. The temporary anonymous ID will only be broadcasted for 15 minutes after that it expires and generate a new temporary anonymous ID to broadcast.

If person A tests positive for Covid-19, he must upload his information to the server, which is where the app architecture comes into play. If the application is built based on the centralized approach, the user must upload all his past temporary anonymous ID and the anonymous IDs of people he met. If it's a decentralized approach, the infected person only must upload his past temporary anonymous ID to the server, and the app on person B side will download the list of infected people's temporary anonymous IDs and perform the contact matching in the device. Singapore was the first country to be ventured in the Bluetooth proximity tracing where they developed their own Bluetooth tracing protocol called BlueTrace that used in the contact tracing app called TraceTogether [4]. Many were amazed at this invention and some countries adopted their BlueTrace protocol for their contact tracing app since the protocol was open source. However, in later months Google and Apple joined their hands to create a privacy- preserving Bluetooth tracing protocol that can provide extensive support to iOS users which previously was a failure with the BlueTrace protocol.

Furthermore, a few governments have made their app open-sourced due to time-critical situations, they needed many supports to build a robust solution, so they made their codebase publicly available. So that people who found a bug in the app can report immediately by opening an issue or pushing the fix directly. However, some countries keep their application

closed source and only reveal the overall functionality of the app.

Many researchers had proposed many ways to find the chain of spread with in contact tracing details gathered by the contact tracing apps. But one popular methodology suggested by many researchers was performing network analysis with contact tracing details. Many have proposed some effective methods in performing network analysis. However, there is no clear evidence on how the governments found the chain of spread whether it is done using applications or manually.

## **1.2 Problem Statements**

Even though many countries have done many implementations to contact tracing, they still have room to improve. Many researchers and netizens have raised privacy-related issues that may potentially cause by contact tracing apps. For instance, contact tracing app that uses GPS to track people movement could save our digital footprint. It may look commonly, but a digital footprint can reveal so much of a person that intruders to invade someone's privacy.

However, some had leapt over this issue by implementing a QR code check-in system. But this further raised more problems. The first problem is, how will we know, the time spent by every person at a place was also visited by someone infected with Covid-19. Since most of the system only has the function to check into place but not check out. Even if some applications have that, people do not check out when they leave. Apart from that, what if a person met a person tested with Covid-19 somewhere around the road where the QR code is not available.

To solve this problem Singapore had developed their application-based proximity level contact tracing using Bluetooth. Even though Bluetooth based contact tracing protocol they use sounds doable, but it does not work as expected in iOS, leaving behind all the iPhone users in the contact tracing ecosystem [5]. The BlueTrace team worked hard to make it work in iOS, but due to the software architecture of iOS, it will not allow the app to run in the background. It became an unsuccessful try for them.

However, to solve this issue Exposure Notification was created by Google and Apple together. This framework did provide extensive support to iOS users to perform Bluetooth scanning in the background. Other countries who initially adopted BlueTrace technology had made a transition to this framework. But things do not go as we all expected, which is Exposure Notification only able to tell you that you met a person who tested positive for Covid. However, it will not reveal who that person is and with its contact tracing details we also cannot find the chain of spread [6].

Apart from all the issues stated above, it comes back to the square one as many citizens had argued over their countries app architecture which focuses on a centralized manner where all their data will be controlled at one central location. For example, China had it mandatory for its citizens to use their local contact tracing app. It tracks users' location using GPS and stores it in a central database. It had opened a debate among the citizens and government asking about some of their privacy concerns. Also, on the other hand, some countries claimed that their application is decentralized. But no papers or evidence shows the degree of decentralization of their system.

### **1.3 Project Objectives**

Our main target of this application is to do contact tracing during pandemic to curb the spreading of the deadly Covid-19 disease.

- Find close contacts via Bluetooth
- Alert users to maintain social distance in crowded place
- Perform network analysis to get more insights on chain of disease spreading
- Uses data analysis algorithm to intervene frequently meeting people

### **1.4 Proposed Solution**

In order to address the aforementioned difficulties, we have devised a new solution. It will be called JomTrace. During pandemics, JomTrace will employ Bluetooth technology to track public movement. As a result, the proposed solution is built on the Mystie 10-10 framework, with a special emphasis on the **Medical and Healthcare Industry. Sensor Technology and Augmented Analytics and Data Discovery** are the project's drivers. The proposed solution will be developed for cross platform usage. This is to make sure that everyone can use this application without having to worry about what device they are using. There will be two types of users on JomTrace: public and medical authority.

The proposed approach mainly utilizes Bluetooth technology to track public movements during pandemics. During a pandemic, knowing who has come into contact with us is critical. This is because it will be easy to trace and warn people if we become infected or someone we meet becomes infected. They can be contacted sooner, and medical authorities can instruct them on how to perform the necessary tests to determine if they are infected. This helps to contain pandemics more effectively. This application will assign each user a unique id that will



be generated automatically but it would not be based on the user's personal information. When they meet someone, their unique id will be transmitted to the person they met, while the user will receive the unique id of the person they encountered. User will receive a unique id for everyone they meet when they go out and it will be stored inside their phone together with their other information.

Aside from that, the application we propose will be partially decentralised. Instead of immediately saving data in a central database, the user's phone will keep all their information about close contacts. Only after a user is confirmed positive their close contact and check in details information, be sent to a central server.

Check-in is another function that will be incorporated in the application. The check-in feature requires the user to choose a location from the list before entering a location. This functionality was added to obtain a user's geometrical data. Network analysis is the final feature. The use of networks and graph theory to examine social structures is known as network analysis. Data from the public will be used to generate all the networks and graph theory. This tool is solely for medical authorities to use, and it will help them analyse, forecast, and make decisions during a pandemic. The two-mode technique will be used to store all data in this application. It is a way in which the user's personal information, such as name and IC number, is saved separately from data, such as the location and time of visit. Network analysis can only be done properly if data is stored in this manner.

### **1.5 Benefits/Impacts and Significance of the Project**

Most of the contact tracing apps utilize Global Positioning System (GPS) to identify the transmission of disease. The data obtained through this feature might be inaccurate as it cannot trace the close contact of two individuals. Hence, we introduce our contact tracing app which uses Bluetooth feature to identify the contact among individuals more accurately. We also provide a check-in which to track visits of user. The significance of using both Bluetooth and check in is that it will help to distinguish two users who is visiting two different locations located nearby. Although they are not close to each other or there is an obstacle between them the Bluetooth will still label them as close contact, but check-in enhances the close contact identifying process.

Our users also get notified earlier once the individual is approached by the suspected disease carrier. By this feature, our users can discover the presence of the disease earlier and

isolate them from others. This is way more efficient and faster than contacting each person one by one. We also provide a self-reporting feature where users can report themselves by sending a capture of their self-test kit. On the other hand, the data analyzed in the server side such as network analysis report and the self-quarantine report can be reviewed by the health authorities which can used to observe the changes in the transmission of the disease. Our focus is maintaining the privacy the of the people. Hence, we implement the concept of the partially decentralized system and will store necessary information in the database.

Although our Malaysian Government introduced MySejahtera, it is not considered as contact tracing app as it doesn't detect the approach of two individuals. Hence, we promise that through our app, we can easily identify the contact among people via the aid of Bluetooth. Moreover, the infected individuals can be isolated from the rest by providing geo-fence and tracking their movement. Some of the contact tracing apps did not maintain the privacy of their users where the infected one will be identified easily by others. Even our MySejahtera by Malaysian Government is storing the information of the users in a centralized database which might cause privacy issues. Hence, to overcome it we are developing a partially decentralized system and store only necessary information in the database.

### **1.6 Uniqueness of Proposed Solution**

There are several factors that makes our proposed solution unique compared to others. Firstly, our proposed solution uses Bluetooth technology to track public movement during pandemic. Most of the current contact tracing apps does not use this technology. They only use manual check-in method where users have to scan QR code each time they visit a place. Manual check-in method is not a very good method as data quality might be affected. This is because not all users will check-in properly, some might forget to check-in the first place or some might forget to check-out right after they leave a place. So, data quality will be affected and it will cause a problem for pandemic analysis process. The proposed solution will use Bluetooth technology which make the contact tracing automatic and data analysis process will not be affected.

Next, our proposed solution is also partially decentralized. Most apps like MySejahtera stores user's details straight away inside database. In JomTrace, data of user will only be stored in database when a user is infected and until then user details will be stored inside their phone itself. This makes our application more unique compared to others. User details will be stored inside database using two-mode method. Most of the contact tracing application use one-mode

method where all user data are stored together in one category. Two-mode method in the other hand, will store personal data separately and geometrical data separately. This eases the process of analysis.

Another feature that makes our application unique is Network Analysis feature. Network analysis will be built using all the data collected from the user. It is mainly for the usage of medical authorities. Medical authorities will have access to a non-editable dashboard where they can see all the analysis related to the pandemic. Matrices like out-degree and betweenness will be used to create the network analysis. Some example analysis will be global social network analysis graph as well as personal risk analysis. Personal risk analysis can be used to classify an infected patient to stages or even can be used to check the probability of someone getting infected.

## **1.7 Contribution**

<b>Group Member</b>	<b>Task Allocation</b>
Sharvin A/L Kogilavanan  (Programming the Bluetooth Tracing System Setup and Design API and Database)	<ul style="list-style-type: none"> <li>• Responsible for implementing Bluetooth Tracing in Mobile App which includes programming the background process for the Bluetooth Tracing to execute</li> <li>• Setup and Create Python Backend. Design APIs to be used by mobile app and dashboard</li> <li>• Setup and maintain database</li> </ul>
Katheeravan A/L Balasubramaniam (Programming of Health Authority Dashboard, Creating Algorithm for Two Mode Network Analysis)	<ul style="list-style-type: none"> <li>• Created algorithm for two mode network analysis, which used to find people who are in high risk and mild risk</li> <li>• Develop the health authority dashboard using Streamlit</li> <li>• Create specific APIs for dashboard</li> </ul>

Jeevitan Design and Develop Mobile App	<ul style="list-style-type: none"> <li>• Design wireframe for mobile app and dashboard</li> <li>• Develop mobile app based on the design created</li> <li>• Implements check in and check out in mobile app</li> <li>• Implements authentication in the app</li> </ul>
Divenesh A/L Shamugam Creating algorithm for clustering, association rules and Develop Health Authority Dashboard	<ul style="list-style-type: none"> <li>• Created algorithm to find clusters using KMeans</li> <li>• Created algorithm to find frequently meeting people based on maximum cliques using apriori algorithm</li> <li>• Develop the health authority dashboard</li> </ul>

*Table 1 : Contribution*

### **1.8 Organization of the report**

For the following chapters, we will be focusing on the Bluetooth Contact Tracing that we will be developing. In the chapter 2, we will be comparing the existing system with the system that will be developed. By using this idea, we could develop a better contact tracing application which helps all of use to be safe in this pandemic. In chapter 3 the analysis and system requirement are discussed where development methodology, system capabilities and limitations, project management, as well as other possibility evaluations and improvement strategy will be additionally discussed. Chapter 4 is about system design which includes system architecture diagram, overall module diagram database design and class diagram. We do explain our subsystem, interface design and also input/output design here. Chapter 5 discussed about system implementation which fully covered of each of our module. Chapter 6 is testing and evaluation. We do test in 3 level which are unit, integration, and system testing. System evaluation is where we explain about how we evaluate our system since start.

## **2.0 Background Study & Related Work**

### **2.1 Existing System**

Our main source of inspiration is the local applications “*rHAIS*”, “*MySejahtera*” which is used to monitor user’s health risk against Covid-19 *and* “*MyTrace*” which is Bluetooth contact tracing application which is recently replaced with MySejahtera Bluetooth Tracing.

*rHAIS* is an app developed by students of Computer Sciences, USM. It’s a cloud-based, real-time integrated smart health information system consisting of a mobile application, web application, and smart thermometer. The application gives the functionalities to monitor, record, and track the users within the USM’s campuses. *rHAIS* intends to offer complete, dependable, vigorous, and effective monitoring and management system to combat COVID-19 cases within the university.

*MySejahtera* is an application developed by the Malaysian Government to monitor and manage the breakage of Covid-19. It uses the Check-In and Bluetooth tracing (MYSJ) technology replacement of *MyTrace* to identify close contacts and notify users in risk of Covid-19 virus. Through the BLE tracing the chain of Covid-19 can be easily broken when all the involved users are alerted. Besides educational contents on the application keeps the user follow best practises in pandemic.

### **2.2 Features Comparison**

<b>rHAIS</b>	<b>MySejahtera</b>	<b>JomTrace (Our System)</b>
No contact tracing and notification alert to close contacts.	Uses Bluetooth technology to perform contact tracing and notify the close contacts and notify them if user is tested positive.	
-	Educates user with informative materials.	1m distance alert to keep users safe with best pandemic practices.
-	-	Data analytics to predict the risk and severity of user and

		provides useful analytical information in dashboard for health authorities.
Self-report health status		
-	Can Check-In & Check-Out when visit a location	Can Check-In & Check-Out when visit a location
-	-	Displays network analysis graphs in web dashboard for health authorities.

*Table 2 : Features Comparison*

### **2.3 Existing Techniques**

The most common technique used to monitor and manage the Covid-19 pandemic is by a contact tracing app by using Bluetooth and location. Some of the famous approaches are Google Apple Exposure Notification, Trace Together, CovidSafe etc. Most of the application uses Bluetooth as their base of contact tracing but implements it differently. Besides there are also application which monitors the pandemic using the Wi-Fi, GPS and QR codes or some even through the daily surveys. However, Bluetooth is the most accepted method as it consumes low energy, and it is practical to be used in daily life. Additionally, cloud notification is used to notify users that is identified as close contacts.

### **3.0 System Requirements and Analysis**

#### **3.1 Status of project development**

This type of project already exists but we are developing an enhanced version of it with added functionalities which helps the community to be safe from pandemics. The project is under testing phase.

#### **3.2 Scope of proposed solution**

Our primary achievable scopes of our proposed system are as follows:

1. User able to sign up and log in the system, all the data been saved into centralised database.
2. The tracing details will be stored locally until the user is declared as positive carrier.
3. Once user declared as positive, the tracing details been saved into centralised database.
4. The system able to process network analysis to find the degree of centrality, maximum cliques, clusters, and apriority.
5. The system able to perform two-mode network graph for geographical analysis.
6. Once a user is declared as positive, the system able to send push exposure notification to the other users in contact.
7. System able to store check-in and check-out activity of all users.
8. System able to alert the user when in contact with other user within 1 meter distance.

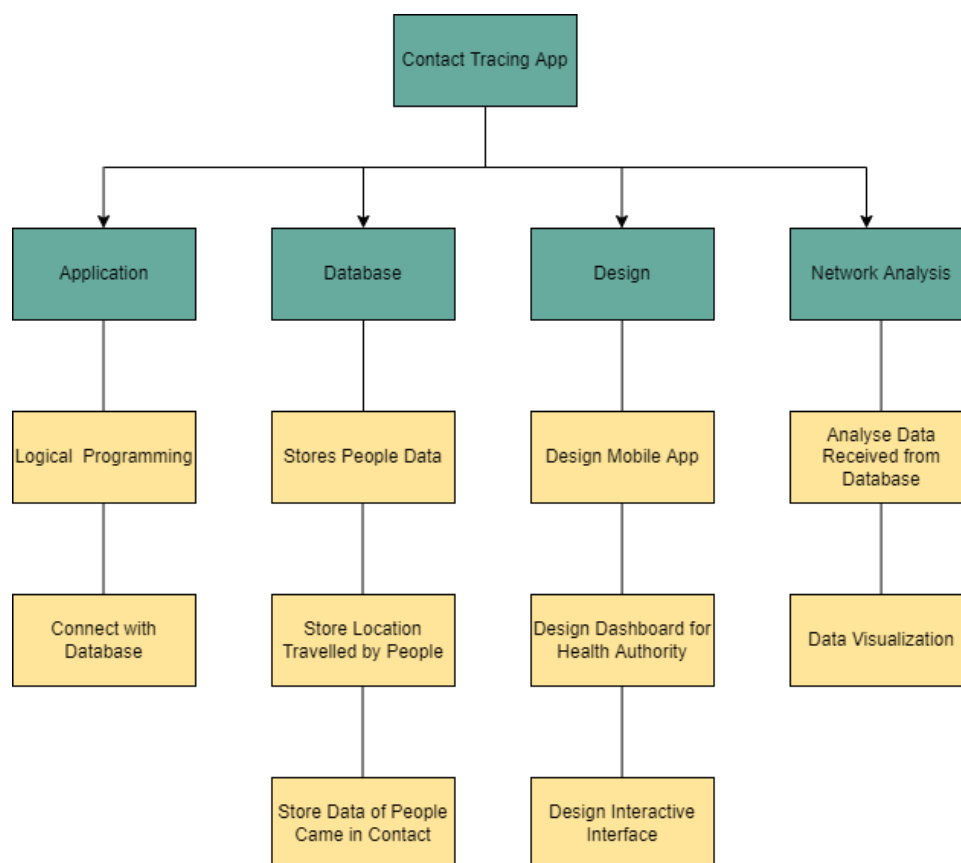
#### **3.3 System capabilities and limitations**

Since our entire system was composed of three different parts which are the mobile app, backend server and analysis dashboard. They do have their own capabilities and limitations. First the mobile app can store Bluetooth information of their close contact through Bluetooth tracing mechanism. This Bluetooth tracing mechanisms can run in background. Even if the mobile phone is turned off or restarts the app able to start automatically in the background and start the Bluetooth tracing mechanisms. The mobile app can also be able to handle check in and check out of location and display the number of visitors currently being visited the location. So the user who wants to go to a particular place and plan to go there can know early about the crowd status. However, the limitation of this feature is that the user must have a constant internet connection to get real-time data of visitors.

Meanwhile, our backend server serves a lot of purpose. The first function is to perform data query and data storing to database. This is because we design the system in a way where the backend server is the only part in our system that can directly communicate to the database instead of the mobile app and dashboard communicate with database separately. The second function that the backend server capable to perform is network analysis task. The server can provide the individual their risk exposure through the computation of degree centrality and provide some information from the data analysis. However, the limitation of the backend server is that it has to be hosted in freemium cloud hosting service and it is not a compute engine. Hence this makes the backend server to run slow when performing network analysis task which will lead the mobile app to have longer screen loading time.

### **3.4 Project Management**

#### **1) Work Breakdown Structure**



*Figure 1 : WBS Structure diagram*



## 2) Gantt Chart

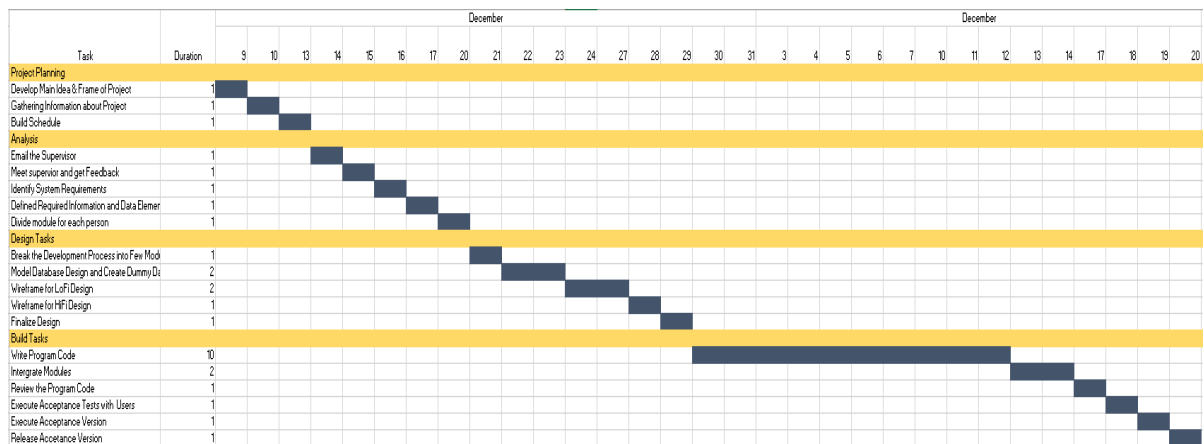


Figure 3 : Gantt Chart

## 3) SWOT Analysis

Strengths	Weakness
<ul style="list-style-type: none"> <li>The developments cost for the project is low.</li> <li>Data storage of user data is very convenient.</li> </ul>	<ul style="list-style-type: none"> <li>Bluetooth must always be turned on in user's phone.</li> <li>User must have constant internet connectivity for check-in and check-out feature to make API calls.</li> </ul>
Opportunities	Threats
<ul style="list-style-type: none"> <li>Can be used to contain pandemic situations better in future.</li> </ul>	<ul style="list-style-type: none"> <li>Possible security threats through Bluetooth.</li> </ul>

Table 3 : Swot Analysis

### 3.5 Development methodology

In this project, we are using agile methodology. We have break down our project development into several phases. Phases that we had went through for this entire project development was planning, designing, develop and deploy. We have broken down our work into smaller parts in every phase to speed up the development process. Planning phase was the time where we gathered user requirements and identified the features that we need to include in our system. We prioritized these features and create a work breakdown flow to develop our system. In development stage, we have practiced continuous integration by

integrating our codes from various branches to the master branch routinely in git system. Even though, we could not call for daily standup but we managed have a weekly standup to discuss on the work that we have completed and pitfalls we faced during the development process.

### **3.6 Analysis of Proposed Solutions/ Project**

#### **Identify user's requirements**

##### **Functional Requirements**

- Users can do contact tracing by using the Bluetooth beacon advertiser
- User able to view their individual risk value and severity reports
- User receives cloud notification when identified as close contacts
- User able to view no of hotspot locations visited

##### **Non- Functional Requirements**

- User Interface
  - User friendly interface with nice look and easy navigation.
- Reliability & Availability
  - User can still perform contact tracing without internet connectivity
  - User can check-in / out without internet connection
- Efficiency
  - Contact tracing will be kept running exchanging IDs from nearby device in the background
  - Users receive notification when identified as close contacts
- Usability
  - Provide clear easy to follow guides to use the application.

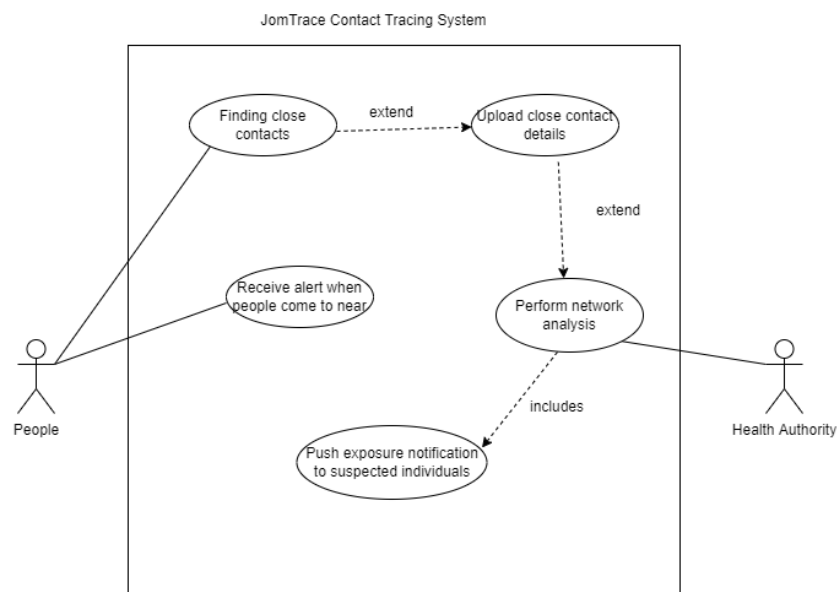
#### **Gathering requirements methods**

In the stage of identifying and gathering user requirements we used several methods such as casual interviews with users from different age groups, web articles, suggestions from social media pages such as *Developer Kaki*. From the interview session and web articles we found that users are very much concerned about their privacy when contact tracing application requires location access where users typically think the application tracks their activity.

Therefore, we identified that having an application that is transparent of what type of information that is collected will make the users feel safer. Besides, users also expect the application to have individual risk values to determine if it's safe to go out.

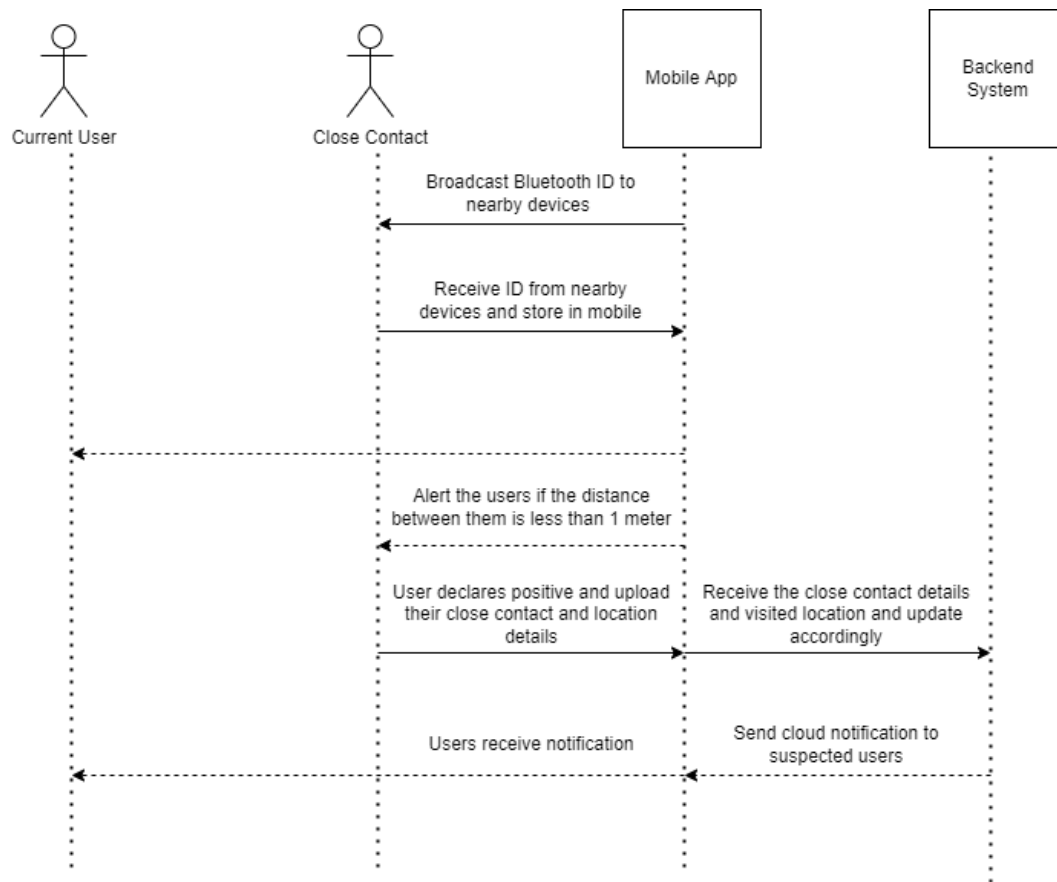
### **3.7 UML / Design Diagrams**

#### **3.7.1 Use Case Diagram**



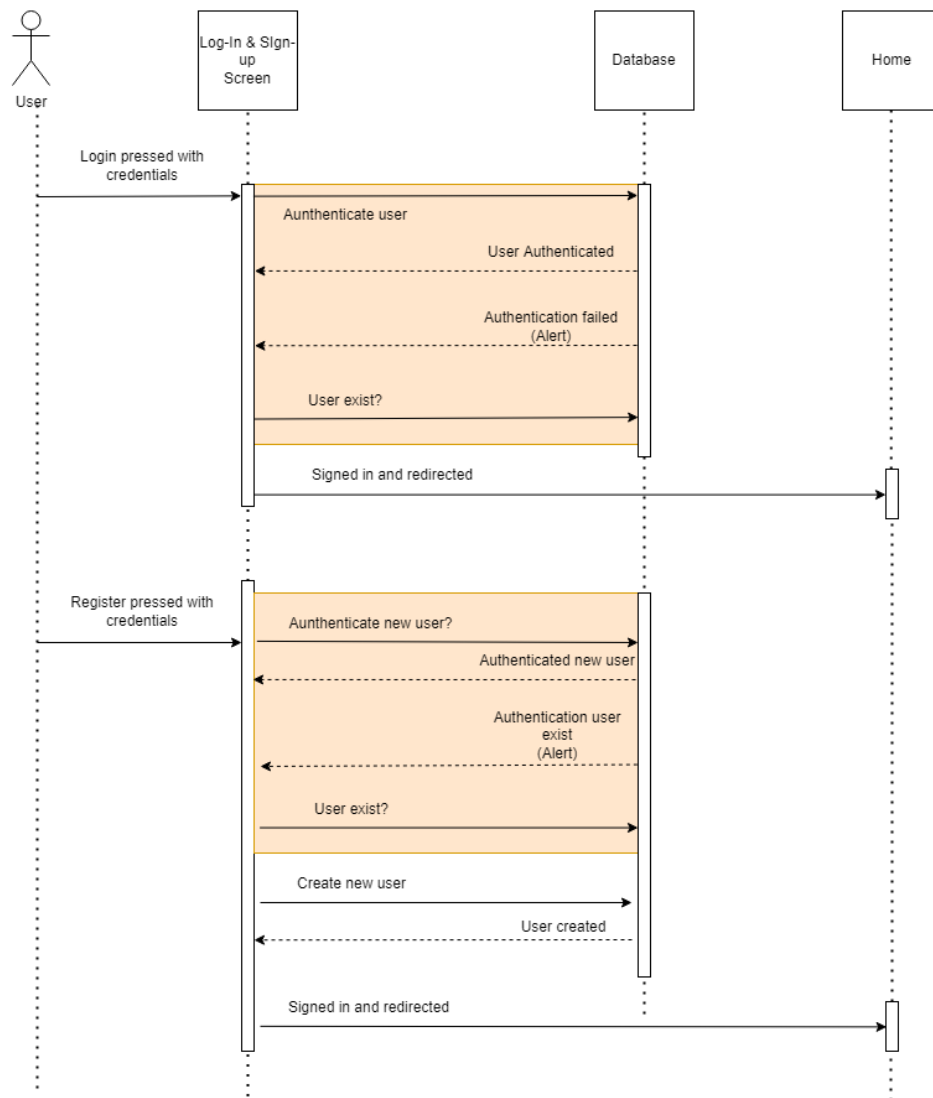
*Figure 4 : Use Case Diagram*

There are some use cases can be defined from the user requirements. The first one would be allowing the users device to find the close contact without compromising the privacy. For this we would be using Bluetooth. Another use case for the user would be sending them alert when they are too near to someone. To achieve these two uses cases, the users need to make sure their Bluetooth is turned on. Next would be analysing close contact details and take decision to curb the spread. For this scenario, there will be two use cases that involve health authority as Actors. First, performing network analysis and data analysis to find frequently meeting people and clusters. For this, the user has to upload their tracing details as precondition. The next use case will be pushing exposure notification to suspected individuals. For this use case the precondition will be the analysis result from network analysis. Because from there we get to know about the users .



*Figure 5 : Contact Tracing Sequence Diagram*

The above sequence diagram shows the contact tracing mechanism in the mobile app. First, the app will broadcast their unique Bluetooth ID to nearby devices. At the same time the app will also receive the unique Bluetooth ID of their neighbouring device. This ID will then be stored in their mobile device. When two devices come closer and if the distance is less than 1.5 metres both mobile devices will receive a notification as an alert to maintain their distance. Once a user found himself infected by covid. He or she will upload their close contact details and their visited location details to the server. The server then sends notifications to suspected individuals to alert that they may be exposed and required to upload their close contact details.

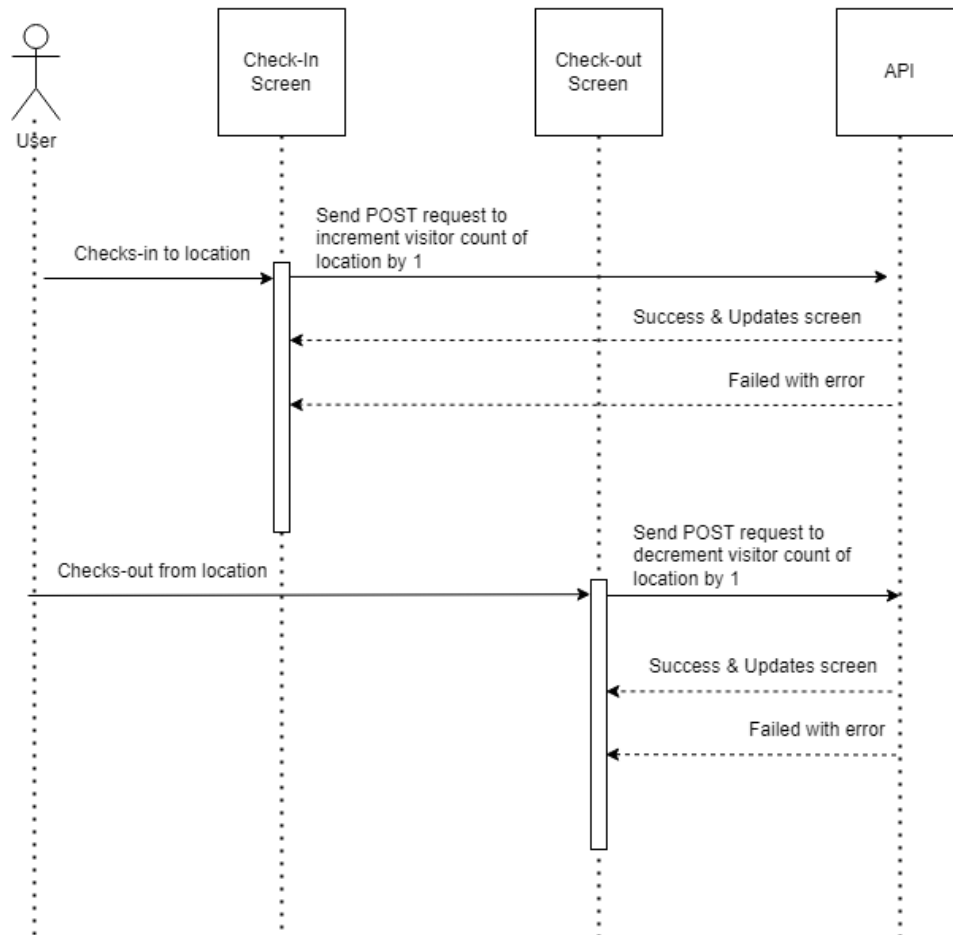


*Figure 6 : Log-In and Register Sequence Diagram*

The above sequence diagram shows the user registration and Log-In mechanism in the mobile application. The user can choose to Register or Log-In and its respective mechanism flow will be executed accordingly. If Log-In is clicked with user credentials, the credentials are authorized in the database using the auth method. If user is authenticated a successful authentication message is sent else failed authentication failed message is sent. These actions will be looped until a valid credentials are entered or register method is invoked.

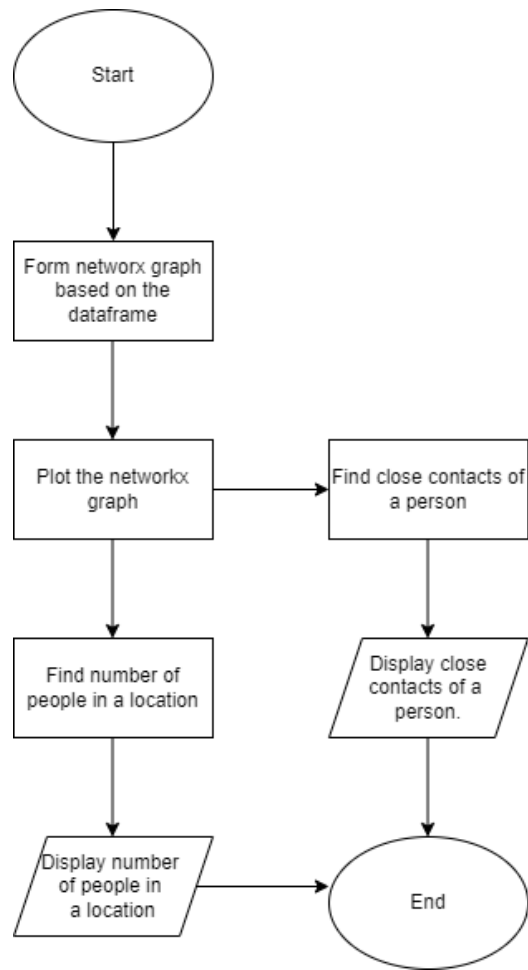
Whereas when register is clicked with respective credentials, the credentials is sent over the database to authenticate if the credentials already exist. If authenticated new user a success message is sent else failed message is sent. These actions are looped until success message is

received or login is clicked. Once the success message is received for both login and registration loop, the user logged in and navigated to the homepage.



*Figure 7 : Check-In and Check-Out Sequence Diagram*

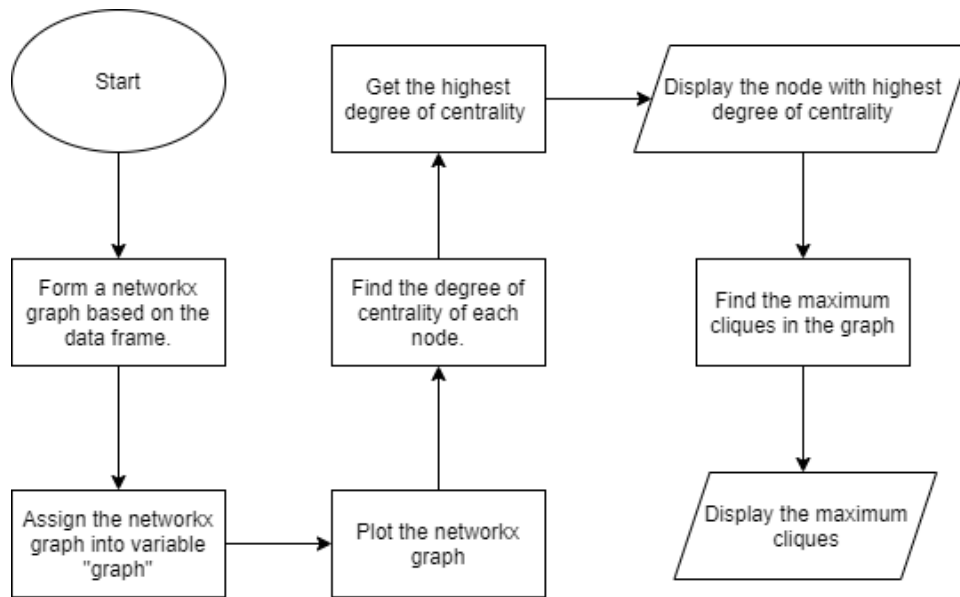
The above sequence diagram shows the Check-In/Out mechanism in the mobile application. Firstly, when user navigates to the Check-In page, the GET method will be executed to get list of locations from API. When user clicks any one location, the POST method will be executed using the location Id to increment visitor count by 1. A feedback message will be sent on success or failure of the POST method. The failure may be due to internet connectivity to connect with API. Once the success message received, the value of visitor will be updated on screen. Whereas for the Check-out have the same exact mechanism with only one changes which is the POST method called is to check out from given location Id and the visitor count is decremented.



*Figure 8: Flowchart of two mode network analysis*

Explanation:

After we get data, we will first plot two mode graph because only then we can do the other features. The graph can be plotted using the networkx library. In the graph, connection between people and people as well as people and location will be shown. After that using that graph, we will be able to find the number of people visited a location as well as close contacts of a particular person.



*Figure 9: Flowchart of one mode network analysis*

#### Explanation

At first, once the dataset is imported, the network graph has been formed and appended into graph variable using networkx library. Then, the graph has been plotted to visualise one mode graph of the nodes. Then, the degree of centrality is calculated to find the connection of each node to other adjacent nodes. Based on the degree of centrality, the risk of each node can be easily determined. Then, the node of highest value of degree centrality will be displayed to find the super spreader. Lastly, the maximum cliques have been found to find the maximal number of nodes can be formed in a single clique. Then, the maximum clique will be displayed.

### **3.8 Technology Deployed**

In our JomTrace system the mobile application is the primary part that records and collects user location check-in details and close contact details via Bluetooth. Hence these features require the users to fulfill some hardware specification needs. First for the close contact tracing functionality the user needs to have their Bluetooth and GPS turned on. The Bluetooth connection is required since we will detect our close contacts via Bluetooth Low Energy (BLE) technology. Even though, we do not collect any live location data from the users through GPS. Users who are using android device with version 5.0 and above required to turn their GPS to allow the BLE technology in their device to work. Finally, we require the users to have an active internet connection to give them live data of the visitors who are currently visiting a



specific location. Meanwhile, for health admin who monitors the movement of people should have an active internet connection to view the dashboard.

Even though, the application we have built can run cross-platform but in our current development we only focused on bring out the most from the application for only android users. Users who running android version 5.0 and above can run our application without running into any issues. Meanwhile, the dashboard which is visible to the health admin has been hosted as a web application. So, the health admin just needs to have an active internet connection and latest browsers with latest version to access the web application.

Hardware	Specification
Mobile Phone	Bluetooth 4.0 WIFI / Mobile data GPS
Dashboard	Personal Computer with WIFI Connectivity and web browser

*Table 4 : Hardware Specification*

We have been using two programming languages to build this entire tracing system. The first language is JavaScript and the second one is Python. We use JavaScript to develop the mobile app since we use a framework called React Native. The reason why we use this framework to build our application because it helps us build mobile applications cross-platform and decreases the development time. Secondly, we use python language to build our backend. The reason why we used python for the backend is because we will perform some analysis on the data based on individuals. To perform this analysis, we will be using libraries such as NetworkX API, Pandas, Numpy which is only available in python language. This has made us to choose python as the language for our backend. To build our backend we used a framework called Flask in python. The reason why we used this framework is because it will remove the overhead of complex designing and creation of APIs process and only need smaller size of code base. Moreover, it is highly flexible and compatible when it comes to integrating our whole system. The final tool we used will be streamlit.io. Streamlit.io is a python web framework that allows people to create dashboards to perform data analysis.

Software	Programming Languages and Tools
Mobile Phone	Javascript React Native Framework Support only android
Web Server	Python Flask
Database	MongoDB
Dashboard	Python Streamlit.io

Table 5 : Software Specification

## 4.0 System Design

### 4.1 System architecture

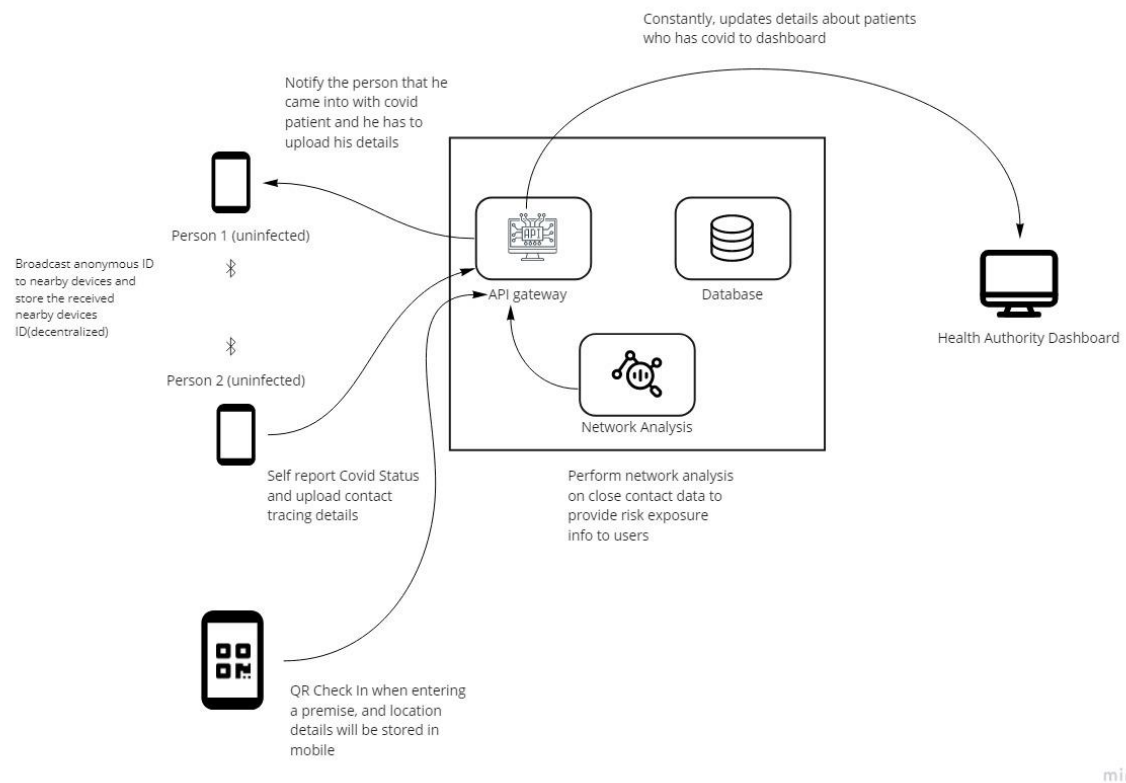


Figure 10: System Architecture

Our JomTrace system built on top of three layers. The first layer is the mobile application. The mobile application collects users' close contact details via Bluetooth and store them in the device. Then it will also log location check in details in the mobile. The mobile application will be connected to the backend server via API gateway. The mobile must communicate to the API to send their close contact details and location check in details once if they are tested positive. On the other hand, suspected users also receive exposure notification pushed by the backend and the users will then be urged to upload their contact details. As what we have mentioned earlier the server acts as a layer to store and retrieve data from database. But the function of the server does not stop there, it will also push exposure notification to suspected individuals and perform light network analysis on information based on individuals to provide them information about their risk exposure based on the people they daily meet. Finally, the health authority dashboard is a data science web application used to analyze the suspected individuals, possible clusters and frequently meeting people using network analysis. The dashboard gets updated every time when it was opened as new data will brought in by the API from database.

#### **4.2 System components / Modules**

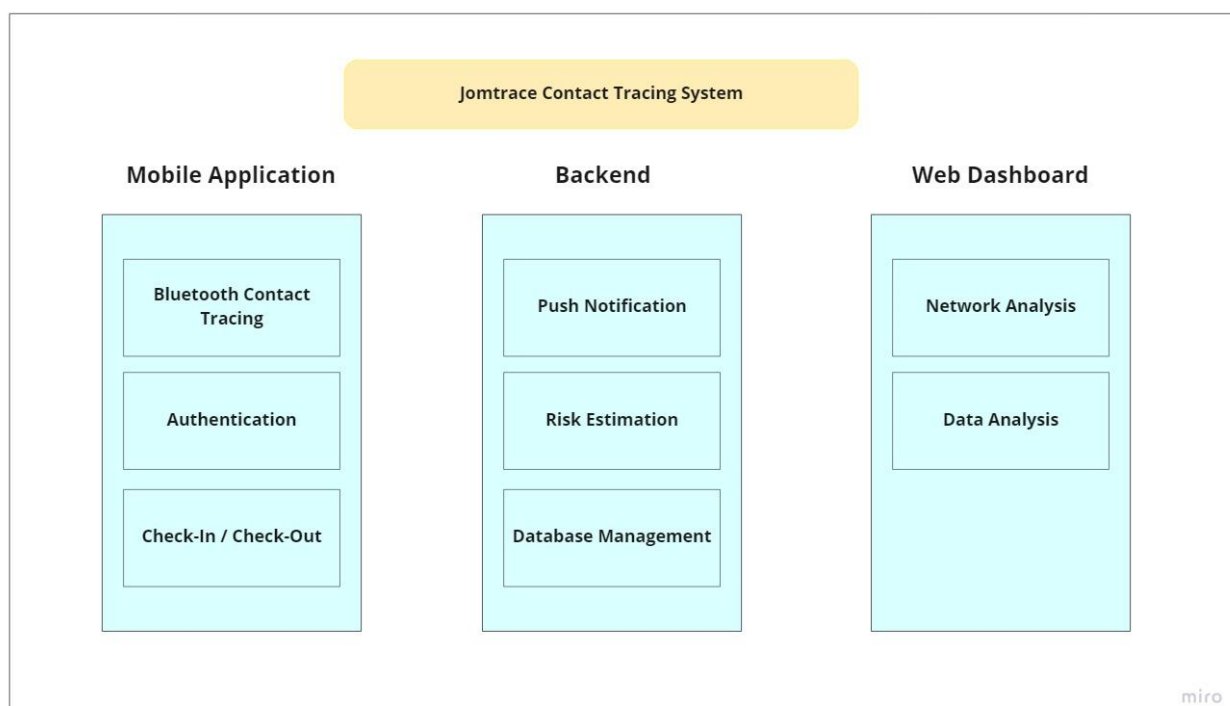


Figure 11 : System Components/ Modules

### 4.3 Database Design

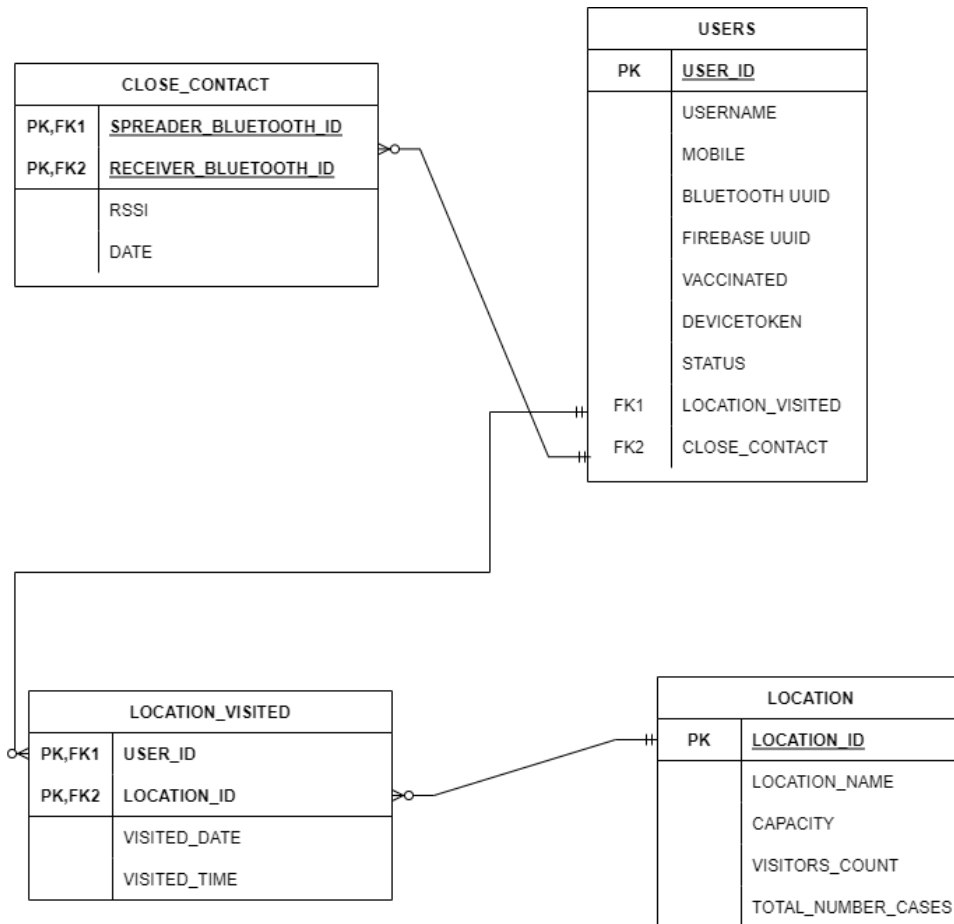


Figure 12 : Database Design

Based on the figure above we can see that there will be 4 main entities in our database. The first entity is user. In the user entity we store some necessary information about the users such as their username, mobile number, and their vaccination status. Apart from that we do have status field to indicate whether a user is positive or negative for covid. To push notifications to user's device, we need their device token which will also stored in the database. The second entity will be location. This entity contains all relevant information about a location. The capacity field in this table refers to the allowed number of people to visit that location. Meanwhile, visitors count used to store the number visitors currently visiting that location. The close contact entity refers to the person who met us based on Bluetooth tracing mechanism. In this table, we store spreader and receiver Bluetooth ID as primary keys. We also store their RSSI, and date came into contact for further analysis. For the location visited entity we store logs of the location visited by a covid 19 patient. The close

contact entity and location visited entity will only be added if a user has covid. The user and location have many-to-many relationship because many users may visit many locations. Hence for that we created a bridge entity that is called location visited. Meanwhile, many users may have many close contacts. For that we created a bridge entity called close contact

#### 4.4 Interface Design

JomTrace have a total of 6 pages in mobile each with its own functionalities as shown below:

Mobile Application		
No	Page Name	Page Description
1.	Register/Sign-In screen	Allows user to register and login using email and password credentials. Error alert is shown if invalid credentials are entered.
2.	User information's form	Users enter information such as username, phone number and vaccination status for interactive experience and for the functionalities of application.
3.	Home dashboard	Displays important analytical values of user such as Hotspot Visited, Risk estimation and Severity reports.
4.	Check-In	Users check in to a location with time, date and location being recorded.
5.	Live close contacts	Displays no of people that is currently in contact (within Bluetooth range) with users.
6.	Profile	Page to display all user information and allows user to update their health status.
7.	Health Authority Dashboard	Dashboard page where users can view all the network analysis and data visualization.

*Table 6 : Mobile Application*

# JomTrace



Email

Password

Log-In

Register

Figure 13 : User Log-In and Register Screen

Let us know  
more about you.

username

Handphone Number

Vaccinated?

☐ Yes (At least 2 doses)

☐ No

Submit

Figure 14 :

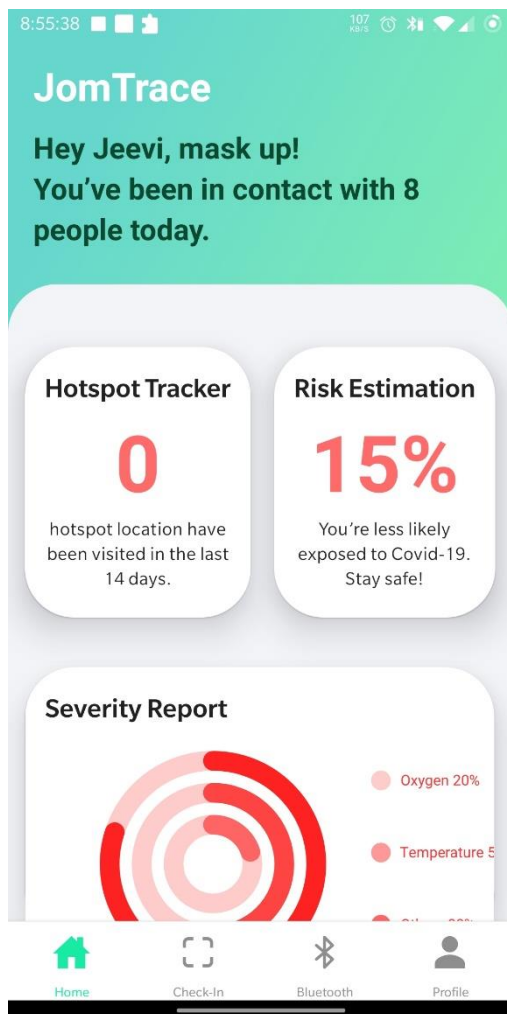


Figure 15 : JomTrace Home Dashboard

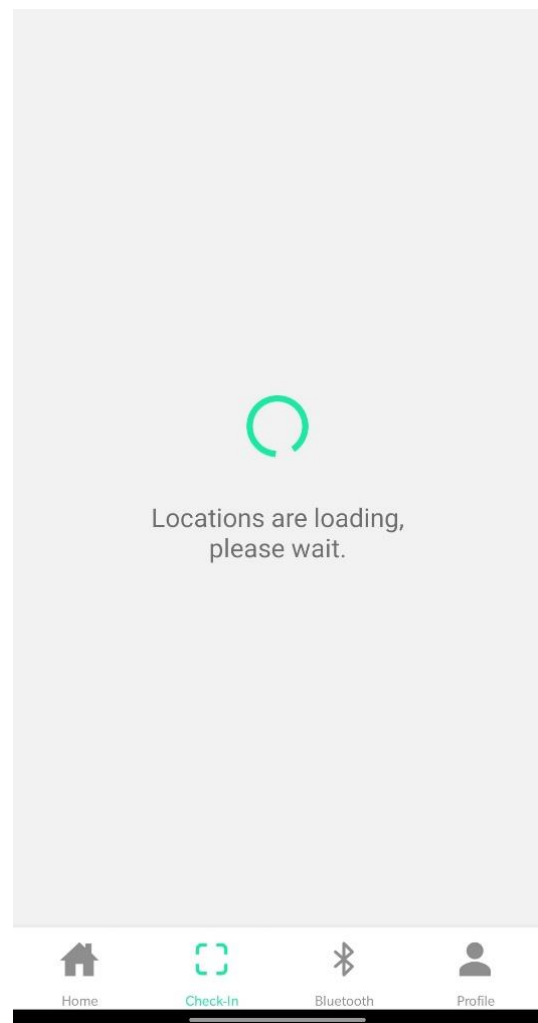


Figure 16:  
Check-In page

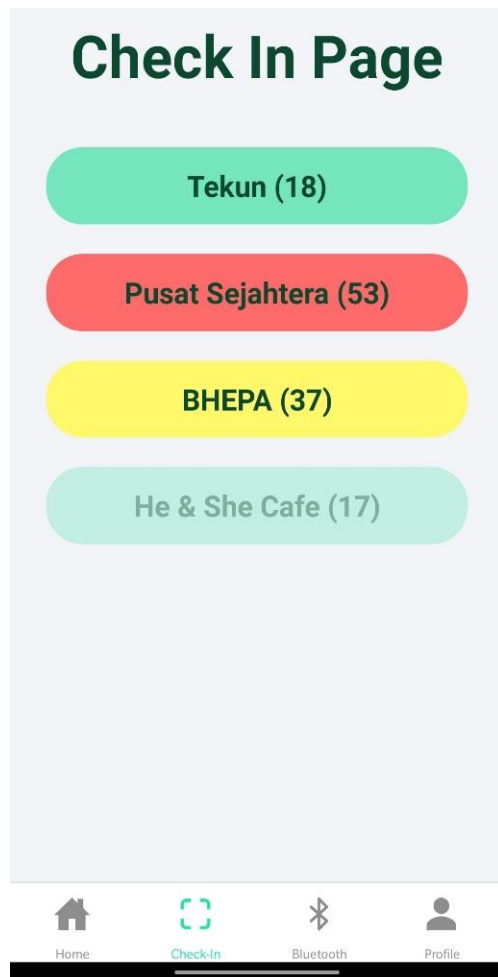


Figure 17:

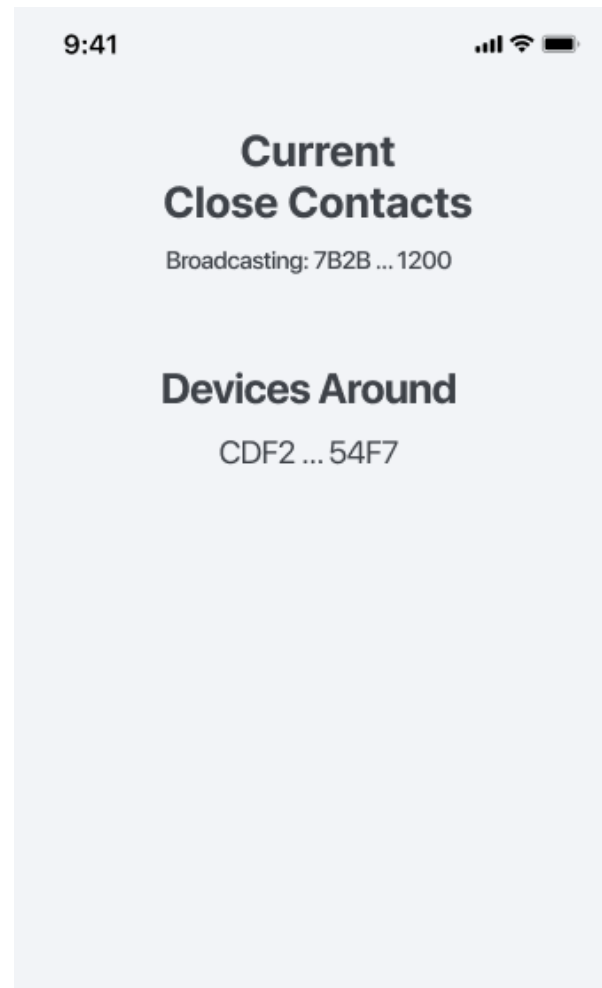


Figure 18

Bluetooth page (prototype)



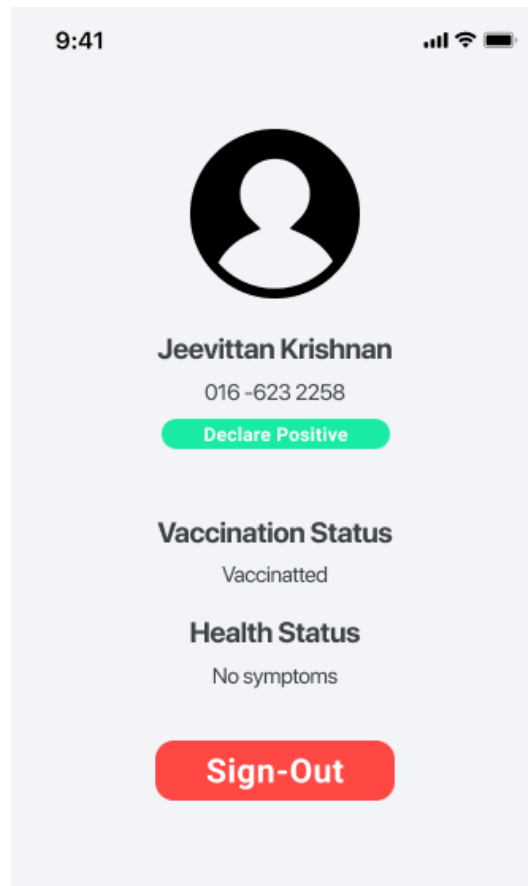
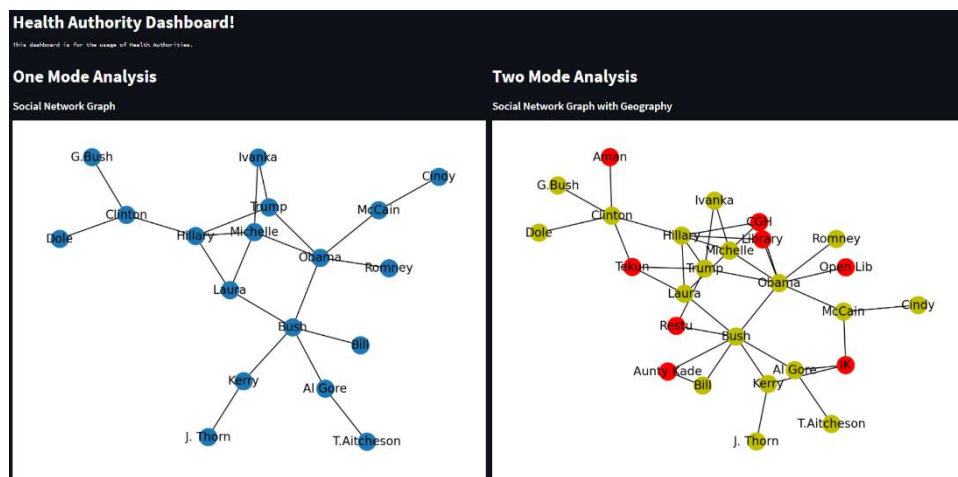


Figure 19: User Profile page  
(prototype)



## **4.4 Input & Output Design**

### **Input Design**

Input Methods	Input Integrity Controls
Text-fields	Text requirements is defined to reduce human errors such as typing error. Authentication is also made for some inputs such as login credentials to make sure each text field matches with its properties. Besides text fields with type password should be hidden.
Buttons to submit form and trigger subsequent actions such as check-in, check-out, update status, sign in, register, sign out.	Button performs action as its displayed.

*Table 7 : Input Design*

### **Output Design**

Output Type	Output Integrity Controls
Data fetched from local storage such as from async storage and data fetched from database / API	Data displayed is not sensitive and trustable.
Data output in different representation/ visualization such as charts and percentages.	-

*Table 8 : Output Design*

## 5.0 Module Implementation

### Implementation of Modules

#### Contact Tracing Module

**Member:** Sharvin A/L Kogilavanan

**Contribution:** Mobile App and Web Server

```
static async start() {
  console.log('[BLEService] Starting BLE service');
  this.clearListener();
  this.emitBroadcastingStatus('Starting');
  this.emitScanningStatus('Starting');

  this.onDeviceFoundListener = this.eventEmitter.addListener(
    'onDeviceFound',
    event => {
      if (event.serviceUuids) {
        for (let i = 0; i < event.serviceUuids.length; i++) {
          if (event.serviceUuids[i]) {
            console.log('[BLEService]', 'onDeviceFound', event);
            this.addDevice(
              event.serviceUuids[i],
              event.deviceName,
              event.rssi,
              new Date(),
            );
            this.checkDistance(event.rssi);
          }
        }
      }
    },
  );

  this.onBluetoothStatusListener = this.eventEmitter.addListener(
    'onBTStatusChange',
    bluetooth => {
      this.emitBluetoothStatus(bluetooth.enabled ? 'On' : 'Off');
    },
  );
}
```

*Figure 21 : Contact Tracing Module*

The above figure shows how the Bluetooth Low Energy technology starts broadcasting its Bluetooth ID and listens to the Bluetooth ID of the nearby people after it's the initialization. The onDeviceFoundListener function will receive the Bluetooth ID of the nearby devices as an event. Later we can destructure the event to find the Bluetooth ID and the rssi value of the nearby people. The details then will be stored in the asynchronous storage of the mobile

phone using the addDevice function. The checkDistance function is the one that checks the distance between two people based on the rssi value and alert them if there are too near

```
@app.route("/uploadContactDetails", methods=['POST'])
def upload_details():
    _json = request.json
    _uuid = _json['uuid']
    _closeContact = _json['closeContact']
    _locationVisited = _json['locationVisited']

    if _uuid and _closeContact and _locationVisited and request.method == 'POST':
        id = mongo.db.user.find_one_and_update({'uuid': _uuid}, {
            "$set": {"closeContact": _closeContact, "locationVisited": _locationVisited}})

        notifyUsers = []
        for contact in _closeContact:
            notifyUsers.append(contact.uuid)

        suspectedIndividuals = mongo.db.user.find(
            {"uuid": {"$in": notifyUsers}})

        suspectedDeviceToken = []

        for individuals in suspectedIndividuals:
            suspectedDeviceToken.append(individuals.deviceToken)

        message_title = "Exposure Notification"
        message_body = "You have been suspected, please upload your details"
        result = push_service.notify_multiple_devices(
            registration_ids=suspectedDeviceToken, message_title=message_title, message_body=message_body)

        resp = jsonify("Contact details added")
        resp.status_code = 200
        return resp
```

Figure 22 : Notification function

The above figure shows how the notification will be sent to suspected individuals after a covid patient uploads his contact tracing details to the server. From the POST requests we will receive the Bluetooth UUID, closeContact and locationVisited details of the user. Then from it we will query the database to find the deviceToken of close contacts in database. Once we gathered all the deviceToken we will push the notification to the close contacts asking them to upload their contact details. The figure below shows the data of a user with covid positive after he uploaded them to database

```
{
  "_id": ObjectId("61f596ce97ce364b8cbf0108"),
  "uid": "X0ox6MKRntdCXJNyV663qWpW25v2",
  "uuid": "3d7675c4-d0ff-4d00-b360-93ff497e0326",
  "username": "Jeevitan",
  "mobile": "011-36055713",
  "vaccinated": "y",
  "deviceToken": "cm3w8bo3RZlWGDkQ0bHK1NS:APA91bH862BtjsseT5troCi5-b33h50D9Lg01bt40vzv-Pk...",
  "status": "positive",
  "closeContact": Array
    ✓ 0: Object
      UUID: "8bf9b13c-e585-4d88-87f3-1d4cca19bb39"
      rssi: "-70"
      date: "2022/01/28"
  "locationVisited": Array
    ✓ 0: Object
      id: "61ec53ff27e6193c09be4c97"
      locationName: "Tekun"
      date: "2022/01/20"
}
```

Figure 23: Database Record

**Member : Jeevittan**

**Contribution : Mobile Application**

## **Log-In & Register Module**

```
44 // SIGN IN AUTHENTICATION METHOD
45 const handleSignIn = async () => {
46   try {
47     const user = await auth().signInWithEmailAndPassword(email, password);
48     const userDetails = await getUser(user.user.uid);
49     storeData('my_bluetooth_uuid', userDetails.uuid);
50     navigation.replace('Home');
51   } catch (error) {}
52 };
53
54 // SIGN UP AUTHENTICATION METHOD
55 const handleSignUp = () => {
56   auth()
57     .createUserWithEmailAndPassword(email, password)
58     .then(() => {
59       console.log('User account created & signed in!');
60       navigation.replace('UserForm');
61     })
62     .catch(error => {
63       if (error.code === 'auth/email-already-in-use') {
64         alert('That email address is already in use!');
65         console.log('That email address is already in use!');
66       }
67
68       if (error.code === 'auth/invalid-email') {
69         alert('That email address is invalid!');
70         console.log('That email address is invalid!');
71       }
72
73       console.error(error);
74     });
75 };
```

*Figure 24 : Log-In and Register code snippet*

The above code snippet shows the Sign-In/ Log-In method and Sign-Up/ Register method. The Sign-In method authenticates the user credentials using the `auth().signInWithEmailAndPassword(email, password)` method which authorize the credentials entered in the database system which is firebase. The `await` in line 47 waits for feedback messages and user is signed in if success message is received. The line 48 & 49 stores the user UUID in the asynch storage for the background tasks and Bluetooth tracing.

The Sign-Up Register function starts at line 55 where `auth().createUserWithEmailAndPassword(email, password)` is used to authenticate if submitted credentials are valid and unique from the existing users credentials in the database. If the authentication is successful, then user account is created and redirected to next page. While, if there is error, alert box with error will be displayed.

## Check-In Module

```
38 | // CHECK-IN BY CREATING CHECK-IN OBJECT AND CALL POST METHOD TO API
39 | const checkInSuccessful = index => {
40 |   var date = moment().format('DD/MM/YYYY');
41 |   var time = moment().format('HH:mm:ss');
42 |   var loc = users[index].locationName;
43 |   var id = users[index]._id.$oid;
44 |   // CREATE CHECK-IN OBJECT
45 |   const checkInObj = {
46 |     date: date,
47 |     time: time,
48 |     loc: loc,
49 |     id: id,
50 |   };
51 |   // CALLING POST METHOD API
52 |   axios.post(
53 |     'https://jom-trace-backend.herokuapp.com/checkIn',
54 |     {
55 |       location: id,
56 |     },
57 |     {
58 |       headers: {
59 |         'Content-Type': 'application/json',
60 |         //other header fields
61 |       },
62 |     },
63 |   );
64 |   // TRIGGERING STATE CHANGES TO MAIN DASHBAORD (CHECK-OUT CARD)
65 |   dispatch(checkInLocation(checkInObj));
66 |   console.log(checkInObj);
67 |   getLocationDetails();
68 |   console.log(loc);
69 |   console.log(id);
70 |   // CREATING CHECK-IN ALERT
71 |   createCheckInAlert(date, time, loc);
72 | };
73 |
```

Figure 25: Check-In code snippet

The above code snippet shows how Check-In is done from line 39. Firstly, when a location is pressed the date, time location name and location Id is stored in an object known *checkInObj*. Then the POST API method is called to update visitors count value of location of given id. This action requires internet connectivity for successful POST method. Followed by the *dispatch(checkInLocation(checkInObj))* function which stores the check-in data in redux and triggers update in the home dashboard page of checked in card which displays data of currently checked in location.

## Check-Out Function

```
onPress={() => {  
  dispatch(checkOutLocation(location.id));  
  getLocationDetails;  
  axios.post(  
    'https://jom-trace-backend.herokuapp.com/checkOut',  
    {  
      location: location.id,  
    },  
    {  
      headers: {  
        'Content-Type': 'application/json',  
        //other header fields  
      },  
    },  
  );  
});
```

*Figure 26: Check-Out code snippet*

The above code snippet shows the execution of Check-Out in home dashboard. The `dispatch(checkoutLocation(location.id))` removes the `location.id` object from redux. Followed by the POST API method to update the visitor count in API.

**Member: Katheeravan**

**Contribution: Two-Mode Analysis and Dashboard in Streamlite**

```
with col2:
    st.subheader("Social Network Graph with Geography")
    people1 = list(df['From'])
    people2 = list(df['To'])
    place = list(df['Location'])

    def create_from_edgelist(top,bottom,middle):
        B = nx.Graph()
        for i in range(len(top)):
            B.add_node(top[i],bipartite=0)
            B.add_node(bottom[i],bipartite=1)
            B.add_node(middle[i],bipartite=2)
            B.add_edge(top[i],bottom[i])
            B.add_edge(bottom[i],middle[i])
        return B

    B = create_from_edgelist(place,people1,people2)

    f = plt.figure(1,figsize=(8,6),dpi=400)
    pos = nx.spring_layout(B)
    colors = {0:'r',1:'y',2:'y'}
    shapes = {0:"s",1:"d",2:"d"}
    nx.draw_kamada_kawai(B, with_labels = True,node_color=[colors[B.nodes[node]['bipartite']]for node in B])
    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()
```

*Figure 27 : Two-Mode Graph Snippet*

The above code snippet shows code of lines that will create a two-mode network analysis graph. For that, networkx library is used. It plots all the people and people contact as well as people to their visited location.



```

user_input = st.text_input("Enter a Positive Patient's Name:", Name)

if user_input:

    ple = []
    for i in range(len(people1)):
        ple.append([people1[i], place[i]])

    print(ple)

    ple2 = []
    for i in range(len(people1)):
        ple2.append([people1[i], people2[i]])

    people_cont = []
    place_cont = []
    warn = []
    no_warn = []
    counter = 0

    def warning(name):

        global counter
        for i in range(len(ple2)):
            if name == ple2[i][0]:
                people_cont.append(ple2[i][1])
            elif (name == ple2[i][1]):
                people_cont.append(ple2[i][0])

        for i in range(len(ple)):
            if name == ple[i][0]:
                place_cont.append(ple[i][1])

        for i in range(len(people_cont)):
            for j in range(len(place_cont)):
                counter = 0
                for x in range(len(ple)):
                    if(place_cont[j] == ple[x][1]):
                        if(people_cont[i] == ple[x][0]):
                            counter = counter + 1

                if counter > 0:
                    warn.append(people_cont[i])
                else:
                    no_warn.append(people_cont[i])

        output_df = pd.DataFrame({'People Close Contact' : [warn],
                                   'Not Close Contact' : [no_warn]}, columns = ['People Close Contact', 'Not Close Contact'])
        st.write(output_df)
    warning(user_input)

```

*Figure 28: Finding Close Contacts Snippet*

This code snippet is to find all the close contacts of a particular person. People who are close contact will be divided into people were in the same location and people who are not in the same location. These values will be then displayed in a table format in Streamlite.

```

st.subheader("Searching for Number of People Visited a Place...")
Name2 = ""
user_input2 = st.text_input("Enter a Location Name:", Name2)
if user_input2:
    circles_size, people_active = bipartite.degrees(B, people1)

    def people_in_place(place):
        st.write('Number of People Visited the People: ', circles_size[place])

    people_in_place(user_input2)

```

Figure 29: Finding Number of People in a Location Snippet

This code snippet is used to find the number of people who visited a particular place. User can insert a locations name and it will display the number of people who have visited the place.

**Member: Divenesh**

**Contribution: One-Mode Data & Network Analysis**

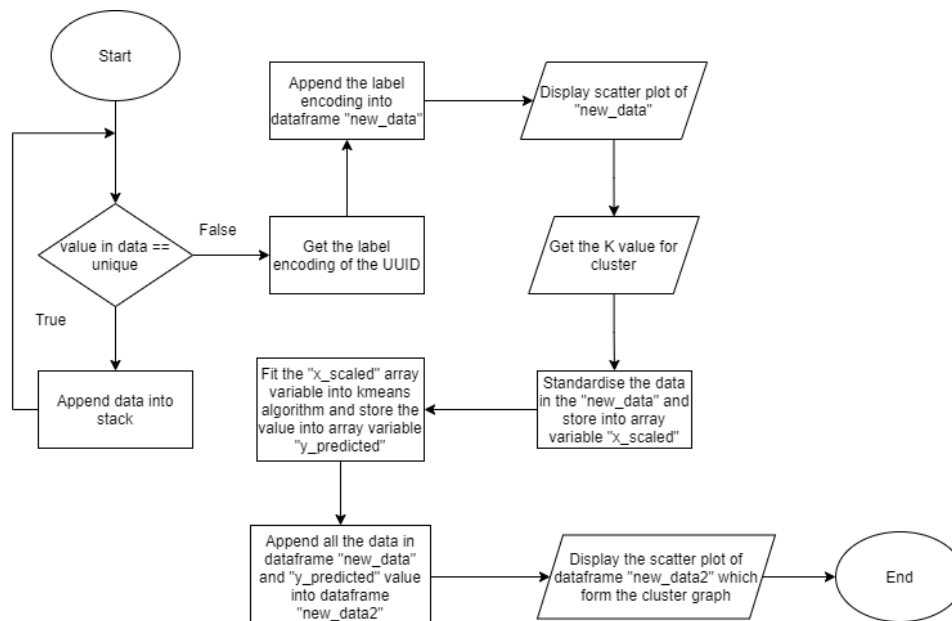


Figure 30 :Flowchart of Clustering

## Explanation

The flowchart above represents the clustering in network analysis. Based on the flowchart, first and foremost, the information in the main data frame called 'data' is checked whether the information is unique and appended into stack. Once all the unique information is appended, the system will undergo label encoding to change the string values to numerical values as the

clustering needs numerical values to calculate the Euclidean distance. The label encoding is appended into a new data frame called “new\_data”. Then, a scatter plot is visualized which will plot the nodes. Then, the health authority need to input the K value which will be used as number of clusters to be found in further clustering process. A new array variable is declared called “x\_scaled” which will be appended with the standardised information of the data frame “new data”. The x-scaled array is fitted with K-means algorithm and stored into another array variable called “y\_predicted”. Then, all the information in data frame new\_data which consists the label encoding, and y\_predicted value array which consists the group of cluster each node represent will be stored into another data frame called “new\_data2”. At last, display the scatter plot and centroid of the clusters. Each cluster will be colorized with different colour.

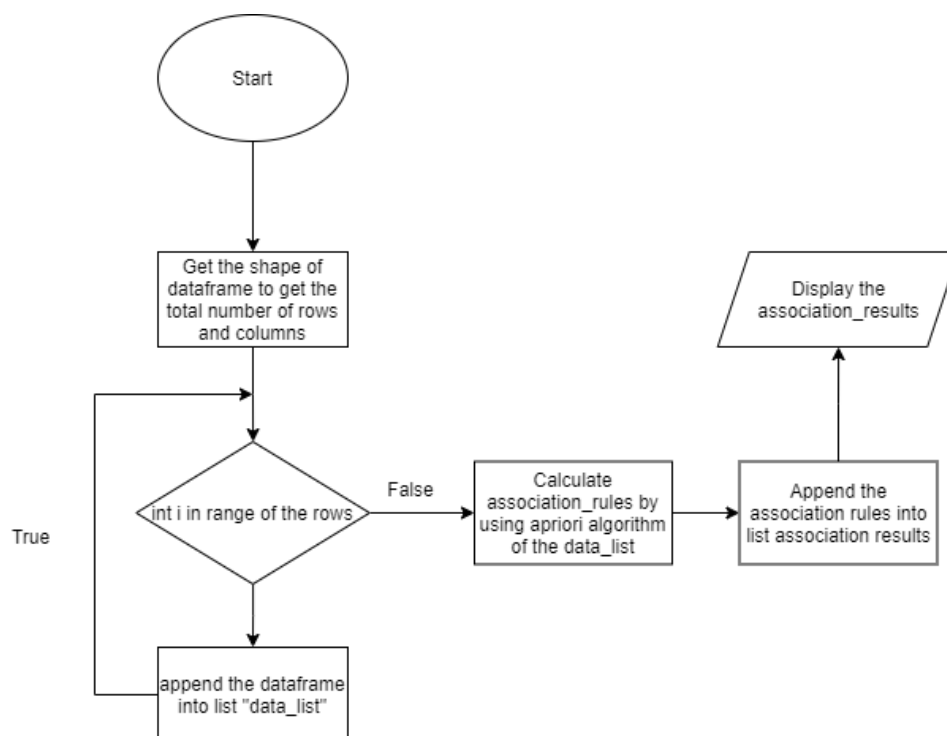


Figure 31: Flowchart of Apriori

## Explanation

The flowchart above represents the clustering in network analysis. Based on the flowchart, first and foremost, get the shape of the data frame. It will display the total number of rows and columns in the data frame. The for loop is carried out to the total number of rows, where all the data frame information will be stored into data list. Once all the values are appended, the

association rules are calculated using apriori algorithm. All the values are then stored into another list called “association results”. Then, the association results will be displayed in tabular form.

```

39 #####
40 st.text("")
41 st.text("")
42 st.text("")
43 st.subheader("Apriori Analysis")
44 data_list = []
45 for i in range(0,21):
46     data_list.append([str(data.values[i,j]) for j in range(0,2)])
47 association_rules = apriori(data_list, min_support=0.003, min_confidence=0.7, min_lift=1.3, min_length=2)
48 association_results = list(association_rules)
49
50 def inspect(association_results):
51     lhs = [tuple(result[2][0][0][0] for result in association_results)]
52     rhs = [tuple(result[2][0][1][0] for result in association_results)]
53     support = [result[1] for result in association_results]
54     confidence = [result[2][0][2] for result in association_results]
55     lift = [result[2][0][3] for result in association_results]
56     return list(zip(lhs, rhs, support, confidence, lift))
57
58 output_DataFrame = pd.DataFrame(inspect(association_results), columns = ['Left_Hand_Side', 'Right_Hand_Side', 'Support', 'Co
59 st.write(output_DataFrame)
60 #####

```

*Figure 32: Apriori Algorithm Snippet*

The above code snippet shows code of lines that will perform association by using apriori algorithm. This code will gives output of the two nodes which highly attached to each other. At first, all the association results will be calculated and displayed into tabular form that will display two highly connected nodes.

```

st.text("")
st.text("")
st.text("")

st.subheader("Clustering in One Mode Analysis")
st.write("Scatter Plot before Clustering")

country_map = {country:i for i, country in enumerate(data.stack().unique())}
new_data=data.copy()

new_data['From'] = new_data['From'].map(country_map)
new_data['To'] = new_data['To'].map(country_map)
plt.scatter(new_data["From"],new_data["To"])
st.pyplot()

st.write("Scatter Plot after Clustering")
kmeans = KMeans(n_clusters=2)
x_scaled = preprocessing.scale(new_data)
y_predicted=kmeans.fit_predict(x_scaled)

new_data2=pd.DataFrame(x_scaled, columns=list('xy'))
new_data2['cluster'] = y_predicted
df1 = new_data2[new_data2.cluster==0]
df2 = new_data2[new_data2.cluster==1]
df3 = new_data2[new_data2.cluster==2]

plt.scatter(df1.x,df1["y"],color='green')
plt.scatter(df2.x,df2["y"],color='red')
plt.scatter(df3.x,df3["y"],color='black')
plt.scatter(kmeans.cluster_centers_[0],kmeans.cluster_centers_[1],color='purple',marker='*',label='centroid')

plt.xlabel('From')
plt.ylabel('To')
plt.legend()
st.pyplot()

```

*Figure 33: Clustering Snippet*

The above code snippet shows code of lines that will perform clustering. This cluster will segregate nodes to the several group. At first, it takes all the nodes from the data frame and plot it without differentiate the cluster. Once, user determined the total value of K, which is total cluster, then it will segregate the nodes to different colour of group.

## **System Integration**

We will integrate our mobile application and web dashboard to the server over the API. Our backend server that builds on top of python will be deployed to Heroku. Heroku is a freemium cloud hosting service. We will host our server in Heroku because to test the mobile application we could not communicate over the localhost. Our dashboard will also use the backend server to fetch the data. Hence, the dashboard also will make HTTP requests to fetch data and push notifications to the user. So, the connection between the mobile app and dashboard will be bridged by the backend.

## **System Testing**

Level of Testing	Explanation
Unit Testing	In the testing we focus more on determining whether each component in our application is fully functional.
	Example: Checking whether Bluetooth is working in background by checking the backlog.
Integration Testing	This is done after all of the components are tested individually. In this testing, we check whether each methods in a subsystem can interact with each other with no issues.
	Example: <ul style="list-style-type: none"> <li>• After user created an account, they will be given a new Bluetooth UUID created by the server.</li> <li>• Once a user tested positive, they can upload their contact tracing</li> </ul>

	details from the mobile to server, then the server sends the data over to dashboard
System Testing	Testing phase where a subsystem is integrated with another subsystem and check whether interaction goes smoothly.
	Example: The application is integrated with MongoDB for data retrieval and handling.

*Table 9: System testing*

### **Test Case**

Few relevant test cases were able to identified in order to determine our complete application functionality. Below are the identified test cases and test scenarios:

Test Scenario	Test Case
Check Login Functionality	<ol style="list-style-type: none"> <li>1. Check login with valid UID.</li> <li>2. Check login with invalid UID.</li> </ol>
Check Bluetooth Functionality	<ol style="list-style-type: none"> <li>1. Able to find nearby devices Bluetooth</li> <li>2. Able to run Bluetooth tracing in the background</li> <li>3. Able to start the app and run Bluetooth tracing after the device restarts or boot up</li> </ol>
Check application connectivity to MongoDB	<ol style="list-style-type: none"> <li>1. Backend server able to communicate to database</li> </ol>
Check push notification in mobile	<ol style="list-style-type: none"> <li>1. Mobile app able to receive alert notification in background</li> </ol>
Data fetching in dashboard	<ol style="list-style-type: none"> <li>1. Able to fetch data from the database using API</li> </ol>

*Table 10: Test Case*

## **System Evaluation**

Our application is evaluated using these criteria:

- 1) Performance of system.
- 2) Usability of system.
- 3) Portability of system.
- 4) Accuracy of Bluetooth feature.
- 5) Reliability of system.
- 6) Standard requirements of system.

## **Conclusion and Future Work**

JomTrace is a social contact tracing application which is made up of both mobile and web application. It has a great influence on people especially on health authority. Public will be able to be safer in pandemic situation while health authorities will be able to contain pandemics more efficiently. If this application is deployed and developed well, it will be a life changing application for both public and health authority. They will be able to overcome any type of pandemics better and faster.

In future, if our project is integrated with more functionalities and designed better, we hope this application can be useful not only to us but all of the people in the world. It will be very rewarding if this application is used to save peoples' life in the future. We want to add more functionalities especially in the network analysis side which will be very useful for authorities as they will be able to handle situations better.

## Appendix

### 1) Mobile Application Link

= <https://github.com/Sharvin1106/BluetoothTracingApp.git>

### 2) Dashboard Link

= [https://github.com/jom-trace/dashboard\\_streamlit](https://github.com/jom-trace/dashboard_streamlit)

### 3) Jom Trace Backend Link

= <https://github.com/jom-trace/JomTrace-Backend>

## Reference

1. Budd, J., Miller, B.S., Manning, E.M. et al. Digital technologies in the public-health response to COVID-19. *Nat Med* 26, 1183–1192 (2020). <https://doi.org/10.1038/s41591-020-1011-4>
2. Buchanan, W. J., Imran, M. A., Ur-Rehman, M., Zhang, L., Abbasi, Q. H., Chrysoulas, C., Haynes, D., Pitropakis, N., Papadopoulos, P. (2020, January 1). Review and critical analysis of privacy-preserving infection tracking and contact tracing. *Frontiers*. Retrieved November 13, 2021, from <https://doi.org/10.3389/frcmn.2020.583376>.
3. Kelion, C. C. L. (2020, May 7). Coronavirus contact-tracing: World split between two types of APP. *BBC News*. Retrieved November 13, 2021, from <https://www.bbc.com/news/technology-52355028>.
4. Asher, S. (2020, July 4). Tracetogether: Singapore turns to wearable contact-tracing covid tech. *BBC News*. Retrieved November 13, 2021, from <https://www.bbc.com/news/technology-53146360>.
5. Warwick, S. (2020, June 6). Singapore contact tracing app isn't mandatory because it won't work on IOS. *iMore*. Retrieved November 13, 2021, from <https://www.imore.com/singapores-contact-tracing-app-isnt-mandatory-because-it-doesnt-work-ios#:~:text=Singapore's%20TraceTogether%20contact%20tracing%20app,of%20the%20app%20is%20restricted.>



6. Chen, Y. D., Chen, H., & King, C. C. (2010). Social Network Analysis for Contact Tracing. *Infectious Disease Informatics and Biosurveillance: Research, Systems and Case Studies*, 27, 339–358. [https://doi.org/10.1007/978-1-4419-6892-0\\_15](https://doi.org/10.1007/978-1-4419-6892-0_15)

7. YouTube. (2021, July 8). *Analysis of two-mode networks with Python / Demival vasques filho*. YouTube.

Retrieved January 30, 2022, from

<https://www.youtube.com/watch?v=CcOfX4n4pWg&list=WL&index=17>

8. Real Python. (2021, January 8). *K-Means Clustering in Python: A Practical Guide*. K-Means Clustering. <https://realpython.com/k-means-clustering-python/>

6. Government, S. (2020, June 29). Two reasons why Singapore is sticking with Tracetoegether's protocol. Government Technology Agency. Retrieved November 13, 2021, from <https://www.tech.gov.sg/media/technews/two-reasons-why-singapore-sticking-with-tracetoegether-protocol>.
7. (Zhang, 2021). The Coronavirus Is Here Forever. This Is How We Live With It. Retrieved August 18, 2021, from [How We Live With the Coronavirus Forever - The Atlantic](#)
8. (Israel Edem Agbehadji, 2020). Review of Big Data Analytics, Artificial Intelligence and Nature-Inspired Computing Models towards Accurate Detection of COVID-19 Pandemic Cases and Contact Tracing. July 24, 2021 from, [Review of Big Data Analytics, Artificial Intelligence and Nature-Inspired Computing Models towards Accurate Detection of COVID-19 Pandemic Cases and Contact Tracing \(nih.gov\)](#)
9. Chen, Y.-D., Chen, H., & King, C.-C. (2010, July 27). *Social network analysis for contact tracing*. Infectious Disease Informatics and Biosurveillance: Research, Systems and Case Studies. Retrieved November 13, 2021, from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7121135/>.