



IFS4205: Information Security Capstone Project

Group 1

Semester 1 AY2022/2023

Final Project Report

Authors:

Lee Kai Wen, Aloysius

Tan Jia Le

Tay Zheng Yao, Schuyler

Tan Juen Woo

Neo Ken Hong, Kelvin

Table of Contents

1.	Introduction	1
1.1.	Overview	1
1.2.	Main Services	1
1.2.1	Tracing Mobile Application	1
1.2.2	Contact Tracing Portal.....	1
1.2.3	Research Portal	1
1.2.4	Links to Services	1
2.	Code Repository.....	2
3.	Tools and Software	2
4.	System Architecture.....	4
4.1.	User-facing Architecture	5
4.2.	Continuous Integration and Continuous Deployment Architecture.....	5
4.3.	Logging Architecture	5
5.	Security and Privacy Claims	5
6.	Subsystem 1 – Close Contact Tracing	10
6.1.	Overview	10
6.2.	Assumptions.....	10
6.3.	Tools and Software	10
6.4.	User Interaction	10
6.4.1	Request for Permissions	10
6.4.2	Detecting Other Nearby Users.....	12
6.4.3	Uploading Close Contacts	12
6.5.	Implementation	13
6.5.1	Permission Requests	13
6.5.2	Registration of User Account	13
6.5.3	Temporary Identifier	14
6.5.4	Close Contact Scanning.....	14
6.5.5	Upload of Close Contacts	16
6.5.6	Verification of Uploaded Close Contact Data	16
6.6.	Security Considerations	17
6.6.1	Security of Temporary IDs.....	17
6.6.2	Secure Storage of Close Contact Data	17
6.6.3	Upload of Close Contact Data	17
6.6.4	Disabling Bluetooth Services.....	17
7.	Subsystem 2 – Building Access.....	17
7.1.	Overview	17
7.2.	Assumptions.....	18
7.3.	Tools and Software	18
7.4.	User Interaction	18
7.4.1	Building QR Code Generation	18
7.4.2	Submission of Building Entry.....	19
7.5.	Implementation	20
7.5.1	Building QR Code	20
7.5.2	Submission of Building Entry.....	20
7.6.	Security Considerations	20
7.6.1	Physical Replacement of Building's QR Code.....	20

7.6.2	Screenshot of Approval Screen	21
8.	Subsystem 3 – Contact Tracer Portal	21
8.1.	Overview	21
8.2.	Assumptions.....	21
8.3.	Tools and Software	21
8.4.	User Interaction	22
8.4.1	Infected Users and Close Contacts.....	22
8.4.2	Building Access.....	23
8.4.3	People Access.....	24
8.4.4	Additional Pages.....	25
8.5.	Implementations.....	26
8.5.1	Tracing Infected Users and Close Contacts.....	26
8.5.2	Tracing Potential Infection Clusters by Location	26
8.6.	Security Considerations	26
8.6.1	Data Privacy and Leakage	26
9.	Subsystem 4 – Researcher Web Portal	27
9.1.	Overview	27
9.2.	Assumptions.....	27
9.3.	Tools and Software	27
9.4.	User Interaction	28
9.4.1	Understanding the Data.....	28
9.4.2	Query of Research Data	29
9.4.3	Cached Data	30
9.4.4	Data Refresh Schedule	30
9.5.	Implementation	31
9.5.1	Anonymisation	31
9.5.2	Getting Data and Pushing Data	32
9.5.3	Frontend Query and Caching	32
9.6.	Security Considerations	32
9.6.1	SQL Injection Attack on Anonymisation Process	32
9.6.2	Compromised Researcher Web Portal.....	32
9.6.3	Background Knowledge Attack	33
10.	Subsystem 5 – Secure Authentication	33
10.1.	Overview	33
10.2.	Assumptions.....	33
10.3.	Tools and Software	33
10.4.	User Interaction	33
10.4.1	User Registration and Login	33
10.4.2	Time-based One-time Password Registration and Input.....	35
10.5.	Implementation	36
10.5.1	Password Complexity Checks and Storage	36
10.5.2	Refreshable JSON Web Tokens	36
10.5.3	Two Factor Authentication	37
10.5.4	Role-Based Authentication	37
10.6.	Security Considerations	37
10.6.1	Stolen Tokens from Phishing Attacks.....	37
10.6.2	Transmitting Sensitive User Data Securely	37
11.	Subsystem 6 – Database Management	37

11.1.	Overview	37
11.2.	Assumptions.....	37
11.3.	Tools and Software	38
11.4.	User Interaction	38
11.5.	Implementation	38
11.5.1	Managing Roles Permissions.....	38
11.5.2	Creating Tables for the Databases	38
11.5.3	Generating Mock Data	38
11.5.4	Secure Encryption of Data Stored in the Database.....	39
11.5.5	Managing Data Encryption in Transit.....	39
11.6.	Security Considerations	39
11.6.1	Compromised exposed database roles	39
11.6.2	Copying data from the database.....	39
12.	Subsystem 7 - System and Security Operations.....	39
12.1.	Overview	39
12.2.	Tools and Software	40
12.3.	Developer Interaction	40
12.4.	Operator Interaction	41
12.5.	Implementation	42
12.5.1	Continuous Integration Testing	42
12.5.2	Automated Release of Artifacts	42
12.5.3	Regular Scans of Code Repositories.....	42
12.5.4	Continuous Deployment to Production Servers	42
12.5.5	Security Hardening of Virtual Machines	43
12.5.6	Automated Operations	43
12.6.	Security Considerations	43
12.6.1	Preventing Hard-coded Secrets in Source Code	43
13.	Subsystem 8 - Logging.....	43
13.1.	Overview	43
13.2.	Assumptions.....	43
13.3.	Tools and Software	44
13.4.	Implementation	44
13.4.1	NGINX Reverse Proxy Logs	44
13.4.2	Django Application Logs.....	44
13.5.	Security Considerations	46
13.5.1	Offsite Access to Logs.....	46
13.5.2	Privacy of User Data	47
14.	Subsystem 9 - Secrets Management and Public Key Infrastructure	47
14.1.	Overview	47
14.2.	Tools and Software	47
14.3.	Implementation	47
14.3.1	Storage of Secrets based on Criticality	47
14.3.2	Public Key Infrastructure.....	48
14.3.3	Mutual Transport-Level Security (mTLS) between services.....	48
14.3.4	Dynamically Generated Database Credentials	48
14.4.	Security Considerations	48
14.4.1	Principle of Least Privilege	48
14.4.2	Passing Secrets Securely	48

14.4.3	Providing Encryption in Transit	48
15.	References	50

1. Introduction

1.1. Overview

TraceIT aims to be a prototype national pandemic tracing system that utilises information technology to monitor the outbreak and spread of a disease. TraceIT is designed to be deployed in the early phases of a pandemic to facilitate tracing of infected individuals, safe access into public buildings and providing valuable data for research purposes.

Much like in the case of Singapore in the early stage of a pandemic, it is likely that individuals can only be tested and diagnosed with the disease at health facilities such as clinics and hospitals. Hence, TraceIT has been designed around such scenarios, where healthcare facilities would transfer a list of individuals diagnosed to be positive to TraceIT administrators daily to be inserted into the system. The public would also be highly encouraged to download the TraceIT app, a contact tracing and building access mobile application, to align with the strict movement regulations likely to be declared by the government.

1.2. Main Services

TraceIT contains three main services which are integrated with one another to enable efficient contact tracing between individuals as well as extracting valuable data for research.

1.2.1 Tracing Mobile Application

Members of the public would be encouraged to download the TraceIT app on their mobile phones to facilitate contact tracing. The app will record the presence of other individuals who also have the app installed on their device, allowing for the upload of the collected close contact data, and record safe and responsible access into buildings and facilities.

1.2.2 Contact Tracing Portal

The contact tracing portal is an internal web portal that contact tracers can retrieve the contact information of infected individuals and inform them of any health protocols, as well as to request for their close contact data which has been collected on TraceIT app to be uploaded. The contact tracers would then analyse the close contacts of the infected individual to determine who might have a high risk of being infected and contact them as well.

As contact tracers need to handle users' sensitive data, they are likely to have gone through a security screening process and signed non-disclosure agreement. Thus, they have been deemed trustworthy and expected not perform any malicious activities with their access to users' personal information.

1.2.3 Research Portal

The research portal provides a variety of valuable data to assist researchers, such as epidemiologists and immunologists, in their study of the pandemic. The data in the research database are anonymised to prevent the identification of individuals and to protect their privacy.

1.2.4 Links to Services

Table 1: Links to TraceIT Services

Service	Link
TraceIT App APK	https://github.com/IFS4205-TraceIT/traceit-app-for-IFS4205-testing/releases/tag/vlatest
Tracer Web Portal	https://traceit-04.comp.nus.edu.sg/tracer
Research Web Portal	https://traceit-04.comp.nus.edu.sg/research

2. Code Repository

TraceIT comprises of 10 repositories available at <https://github.com/IFS4205-TraceIT>.

Table 2: Code Repositories

Repository	Description
ansible-management	Github Action workflows and Ansible playbooks that automate the deployment and teardown of the TraceIT infrastructure.
dast-scan	GitHub Action workflows and API definitions that are used to perform DAST scans against TraceIT's backend API services.
database-configurations	This repository contains the database configuration used for TraceIT.
tracer-backend	Django source code of the contact tracer web frontend.
tracer-frontend	Vue3 source code of the contact tracer web backend.
contact-backend	Django source code of the close contact web backend.
traceit-app	Flutter source code of the TraceIT mobile app.
research-backend	Django source code of the research web frontend.
research-frontend	Vue3 source code of the research web backend.
kanonymity	Python source code for anonymising user data for researchers.

3. Tools and Software

This section details the tools and software used in building TraceIT's various components.

Table 3: Tools and Software Used in TraceIT Systems

System	Tools and Software
Source Code Control	Git – Used to manage versioning of code and features GitHub – Software for sharing codebase and infrastructure configuration files
Issue Management	GitHub Issues and Project – Provides tracking for tasks yet to be completed and bugs to resolve
Secrets Management	GitHub Action Secrets – Securely store secrets in an encrypted form, that can only be read by GitHub Action runners Vault – Provides secure storage of secrets and can be configured with secret engines that can generate X.509 certificates, dynamic database credentials and TOTP.
Deployment Automation	Ansible – Enables Infrastructure-as-Code for easier configuration, installation and application deployment. GitHub Actions – CI/CD platform that allows developers to create automation pipelines that build, test or deploy code changes.
Server Hardening	Uncomplicated Firewall (UFW) – Creates firewall configurations and restrict incoming and outgoing traffic
Database	PostgreSQL TDE – Relational database management system that support querying via the use of the Structured Query Language (SQL). This patched version of PostgreSQL supports Transparent Data Encryption (TDE) which encrypts the database files at rest.
Web Frontend	Vue 3 with Nuxt3 – Modern JavaScript framework that allows reusable components and layouts for the frontend.

	TailwindCSS – Clean and minimal CSS framework to provide styling for the frontend web portals.
Web Backend	Django and Django REST Framework – Provides a full suite of tools and libraries such as Object Relational Mapping and View Mapping to ease development of backend endpoints.
Mobile App	<p>Android Device – Android mobile device running Android 5.0 (API level 21) or higher.</p> <p>Flutter – an open-source user interface toolkit for both iOS and Android. Flutter allows for the quick design of screens as well as providing versatility to develop in platform-native code to extend device-specific functionality.</p>
Logging	<p>Grafana Cloud – The cloud platform suite that include Grafana Loki and the dashboards to view the health of the services.</p> <p>Grafana Loki – Log collecting endpoint for Django backend to send their logs to.</p> <p>Fluentbit – Collects logs from NGINX and forwards to Grafana Loki.</p> <p>Grafana Dashboard – The web page to view and aggregate logs collected from the Django backend and NGINX proxy.</p>
Testing Framework	unittest – A testing library built-in to the Python Standard Library
Pentesting Framework	<p>OWASP Zed Attack Proxy (ZAP) – A web application scanner that test for web vulnerabilities and misconfigurations. It has a script that is tuned for scanning against APIs.</p> <p>Semgrep – A SAST tool that utilises a series of predefined rules to scan and detect for vulnerable patterns and misconfigurations in code.</p> <p>PyCharm Python Security plugin – A SAST tool that scans specifically for common security vulnerabilities in Python code</p>

4. System Architecture

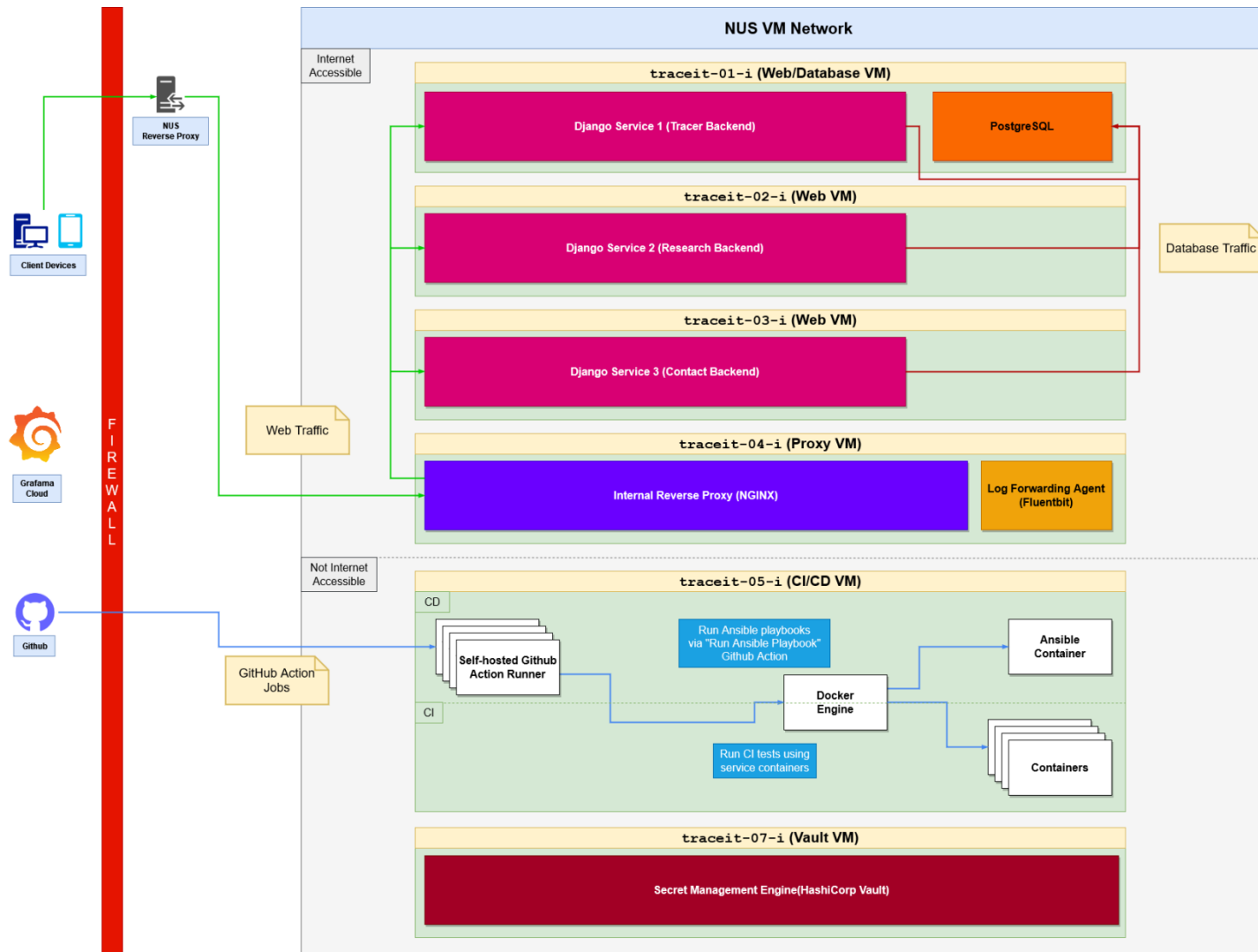


Figure 1: System Architecture of TraceIT

4.1. User-facing Architecture

All user-facing web interactions will pass through the NGINX reverse proxy in traceit-04-i. After which, the traffic is routed to the relevant backends that hosts the API endpoints within TraceIT. The frontend pages are compiled and hosted by NGINX directly as static pages. This means that the user will not be exposed to services they are not expected to interact with directly, such as Vault.

4.2. Continuous Integration and Continuous Deployment Architecture

A few instances of GitHub Action runners run on traceit-05-i. They reach out to GitHub and listen for any new GitHub Action jobs. When a new job for continuous integration testing is initiated, the runners may spawn new Docker containers to act as isolated environments for testing. For deployment or teardown jobs, a runner will spawn a dedicated Ansible container that remotely deploys configurations and applications to the virtual machines.

4.3. Logging Architecture

The logging agent Fluent Bit is installed on traceit-04-i to forward web access logs to Grafana Cloud. Similarly, user actions to the backend applications in traceit-01-i to traceit-03-i are logged to Grafana Cloud. The details of the logging infrastructure can be found in [Subsystem 8 - Logging](#).

5. Security and Privacy Claims

Table 4: Security and Privacy Claims of TraceIT

S/N	Category	Claim	Description
1	Confidentiality	An adversary cannot stalk users who are scanning for close contacts using the TraceIT app.	Temporary IDs which only have a 15-minute lifetime are exchanged during close contact scanning, which prevents an adversary from continuously correlating an ID to a user or location. Refer to section 6.6.1 on the security of temporary IDs.
2	Confidentiality	Close contact data stored on a user's device cannot be exported without the user's intervention.	Users will also need to manually trigger the upload of the close contact data on their device, preventing accidental upload and ensuring data privacy. Refer to section 6.6.3 on the security when uploading close contact data.
3	Confidentiality	An adversary cannot identify individuals from the anonymised records stored in the research database without any background knowledge attack.	The research data is anonymised to achieve k-anonymity before it is inserted into the research database and cannot be traced back to a specific individual without any background knowledge attack and homogeneity attack. Refer to section 9.5.1 on the anonymisation process.
4	Confidentiality	Even if a copy of the source code is obtained, adversaries will not be able to obtain any useful	No secrets such as encryption keys and database credentials were hard coded in the source code. Therefore, even if adversaries obtain a copy of the source code, they will

		information (such as secrets, credentials or keys) that allow them to pivot to other hosts or access privileged information.	<p>need to figure out how and what were the secrets passed to the services at run-time.</p> <p>Refer to section 14.4.2 on the secret management and public key infrastructure.</p>
5	Confidentiality	An adversary cannot retrieve data that is at rest.	<p>The database used in the TraceIT infrastructure is a patched version of the PostgreSQL software that supports Transparent Data Encryption (TDE). This means services that access the database can still do so, even while the data remains securely stored on disk in an encrypted format. Adversaries will not be able to read the contents directly from disk without decrypting it with the correct decryption key.</p> <p>The close contact data are encrypted locally using AES256-CBC, then stored in the app's data directory which can only be accessed by the app itself.</p> <p>Refer to section 6.6.2 on the secure storage of close contact data and section 11.5.4 on PostgreSQL TDE.</p>
6	Integrity	An adversary has minimal impact when replaying close contact Bluetooth messages.	<p>Temporary IDs which only have a 15-minute lifetime are exchanged during close contact scanning, which prevents an adversary from replaying the temporary ID after it has expired.</p> <p>Refer to section 6.6.1 on the security of temporary IDs.</p>
7	Integrity	An adversary cannot modify or forge temporary IDs to appear legitimate.	<p>Temporary IDs contain the user information that has been encrypted by the TraceIT server using AES256-GCM, which cannot be modified by an adversary. Modified or forged temporary IDs will be corrupted as its authentication tag cannot be verified by the TraceIT server, hence invalidating the forged close contact record.</p> <p>Refer to section 6.6.1 on the security of temporary IDs and section 6.5.6 on close contact validation.</p>
8	Integrity	An adversary tampering on the research data cannot affect the original user data in the main database.	<p>Any data tampering done on the research web portal does not affect the data shown in the tracer portal.</p> <p>Any data tampering or modification on the research database will be refreshed after every midnight where new data are</p>

			<p>anonymised from the main database to the research database.</p> <p>Refer to section 9.5.2 on the data processes.</p>
9	Confidentiality and Integrity	An adversary cannot retrieve plaintext data that are in transit within the TracelT infrastructure.	<p>All web and database traffic that relates to the TracelT infrastructure is encrypted with Transport-Layer Security (TLS) to ensure confidentiality and integrity of data sent between services such as web servers, database, or Vault.</p> <p>Refer to section 14.3.2 on the public key infrastructure and section 14.3.3 on mutual TLS.</p>
10	Access	Users are only authorised to access the respective web applications they are registered to.	<p>While the authentication is unified, the role-based authentication only permits the user to access the backends in which they have registered. This ensures that users can only view or modify data that are permitted by their role.</p> <p>Refer to section 10.5.4 on role-based authentication.</p>
11	Access	An adversary will have limited access to the backends if the user's password is leaked unless the adversary has access to the user's personal device.	<p>As part of multi-factor authentication, all users are required to enrol a device with a time-based one-time passcode application. This ensures that even when the user's password is compromised, adversaries require the device of the user to have full access to the backends.</p> <p>Refer to section 10.5.3 on time-based one-time passcode implementation.</p>
12	Access	Even if adversaries gain a foothold on any machine, they will not be able to obtain the SSH private key necessary to remotely login to the other machines.	<p>During deployment and teardown of the architecture, the CI/CD machine uses Ansible to remote manage all the virtual machines, which requires the use of a SSH private key. This SSH private key is stored in the GitHub Action Secrets and is only passed to the GitHub Action runners in the CI/CD machine when they receive a job. After the job is done, the SSH private key is eradicated from the CI/CD machine. Additionally, the GitHub Action runners are also not able to request for the SSH private key from GitHub Action Secrets.</p> <p>Refer to section 14.3.1 on secrets management pertaining to GitHub Action Secrets.</p>

13	Networking	<p>TraceIT machines within the NUS internal network will not accept incoming connections from an adversary unless allowed by the firewall rules.</p>	<p>This access control has been enforced by firewall rules.</p> <ul style="list-style-type: none"> • The firewalls are configured to allow only communications between specific machines within the internal network. • Restrict the connections to only allow the specified incoming communications. <p>Refer to deploy firewall.yml in the ansible-management repository for the firewall configuration, and section 12.5.5 on the hardening of VMs.</p>
----	------------	--	--

6. Subsystem 1 – Close Contact Tracing

6.1. Overview

The close contact tracing subsystem involves detecting other individuals within proximity who also have the TracelT app installed on their mobile devices. The TracelT app uses the mobile device's Bluetooth capabilities to exchange messages with other nearby users and records them as close contacts locally on the device. The tracing protocol is based on BlueTrace, an open-source protocol that facilitates digital contact tracing of users during a pandemic (Bay, et al., 2020).

6.2. Assumptions

Users are assumed to own an Android smartphone with Bluetooth and camera capabilities. Bluetooth and location services (Android 11 and lower) are always assumed to be enabled when using the app. It is also assumed that the user has installed a third-party two-factor authentication app such as Google Authenticator or Authy.

6.3. Tools and Software

Mobile app:

- An Android smartphone
 - Running Android 5.0 Lollipop (API Level 21) or higher
 - Has Bluetooth and camera capabilities
- Flutter and Kotlin

To simplify development and testing, the TracelT app has only been built to run on Android devices, and there are no plans for iOS devices. Thus, the app has been developed in Flutter and Kotlin for Android-specific functionality.

6.4. User Interaction

6.4.1 Request for Permissions

When a user first launches the TracelT app, it will prompt the user to allow permissions required for the app's functionality. Bluetooth scanning and camera access are requested on devices running Android 12 (API level 31) and higher while location service and camera access are requested on Android 11 (API level 30) and lower.

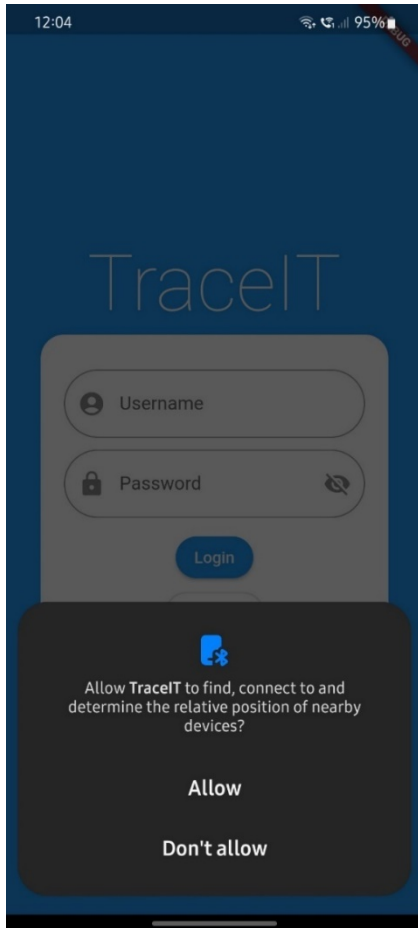


Figure 2: Request for Scanning Permission on Android 12 and Higher

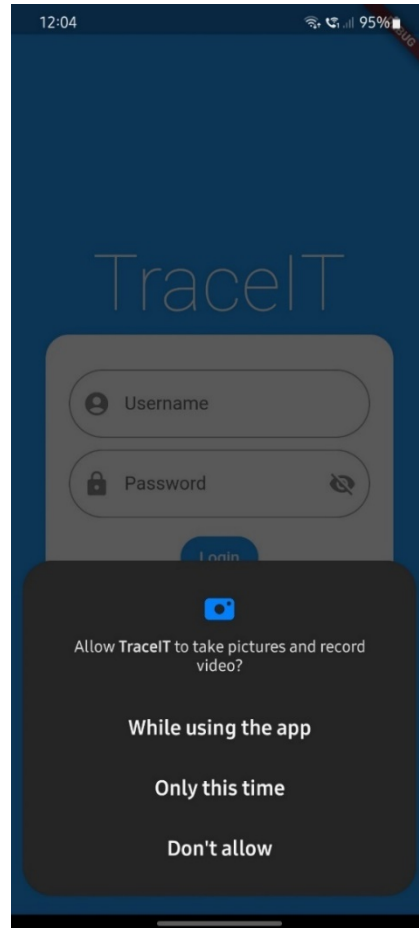


Figure 3: Request for Camera Permission on Android 12 and Higher

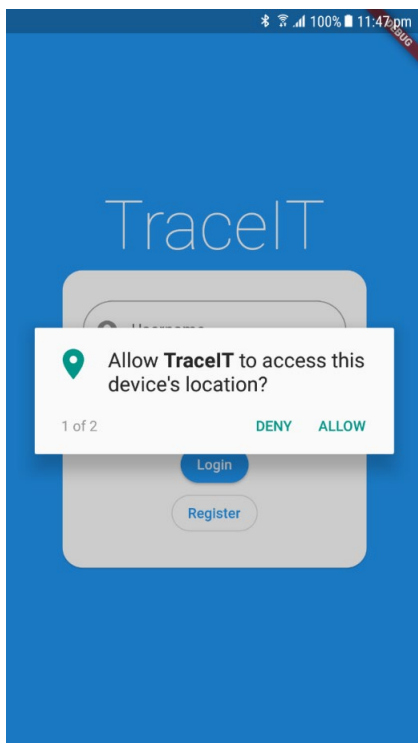


Figure 4: Request for Location Permission on Android 11 and Lower

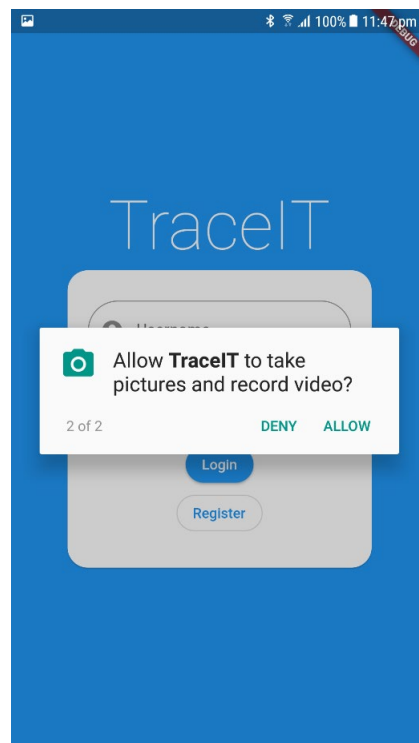


Figure 5: Request for Camera Permission on Android 11 and Lower

6.4.2 Detecting Other Nearby Users

After logging in, users will be shown the tracing screen with helpful information. The information displayed are the user's contact status (Positive/Negative/Close Contact), daily close contacts detected, current tracing mode and the device's support for Bluetooth Low Energy (BLE) advertisement.

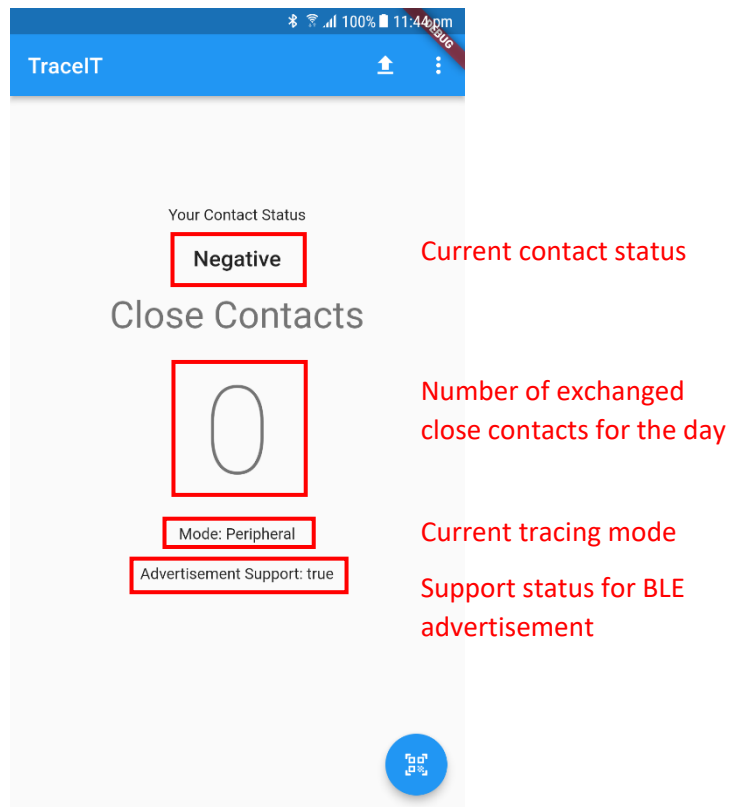


Figure 6: Tracing Screen

6.4.3 Uploading Close Contacts

If a user's contact status is "Positive", an official contact tracer may require the information of other users in close contact. If so, the user will be contacted via their phone number by contact tracers to guide them to upload their close contact data.

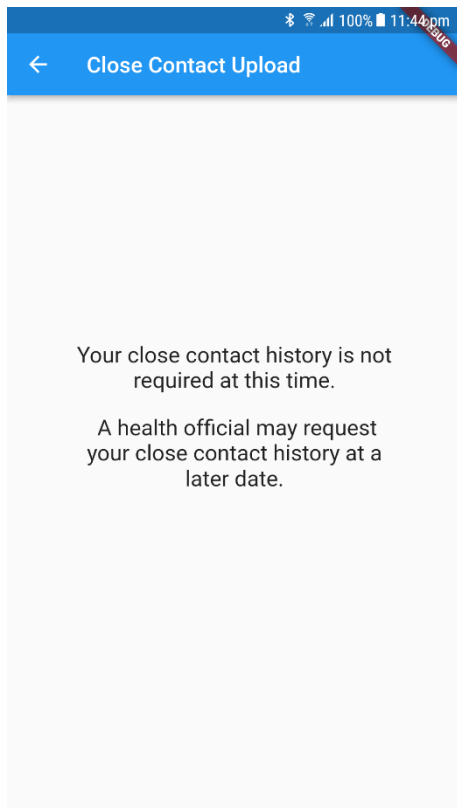


Figure 7: Upload of Close Contact Not Requested by Contact Tracers

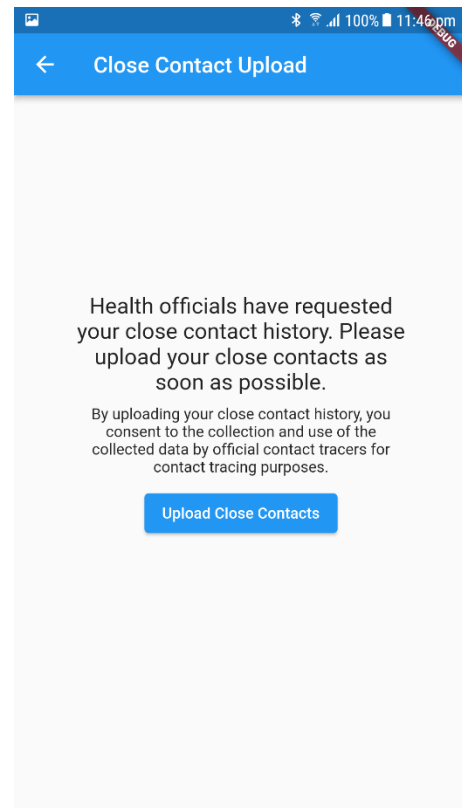


Figure 8: Upload of Close Contact Requested by Contact Tracers

6.5. Implementation

6.5.1 Permission Requests

Permissions for the TracelT app depend on the version of Android running on the device. Location permission is requested on Android 11 and lower instead of nearby scanning as on Android 12 and higher due to Bluetooth Low Energy (BLE) scanning functionality being grouped under location services. Hence, devices on Android 11 and lower will need to enable location and Bluetooth services for the app to function as intended.

6.5.2 Registration of User Account

New users will need to provide their intended login credentials and certain personal details when registering for a user account. The following are the information collected from new users:

- Username
- Password
- Email address
- Phone number
- National identity registration (NRIC) number
- Full name
- Date of birth
- Gender
- Home address
- Postal code

In the early stages of a pandemic, it is crucial for individuals who have been infected to be identified and isolated in a timely fashion to prevent the spread of the disease. The personal details collected from users are to provide contact tracers with multiple methods of contacting the infected user, especially if the user does not respond via phone call. Although users may be concerned that providing such personal information may compromise their privacy, TraceIT has placed more priority on a timely response to prevent a widespread outbreak in the event of a highly infectious disease and for research purposes.

6.5.3 Temporary Identifier

A temporary identifier that changes every 15 minutes and contains encrypted information will be sent during message exchange with other users over Bluetooth. The app will request a list of temporary IDs from the backend server, which then the server will generate 24 temporary IDs and return them to the client. As the server returns a list of 24 temporary IDs, the app can continue to perform close contact tracing via Bluetooth without internet access for 6 hours.

Each temporary ID contains the user's universally unique ID (UUID), the start and end time of the temporary ID's validity period, encrypted in AES256-GCM with the server's secret key. The initialisation vector used in the encryption and the generated authenticated tag is then appended to encrypted contents and encoded in base64 to form the temporary ID.

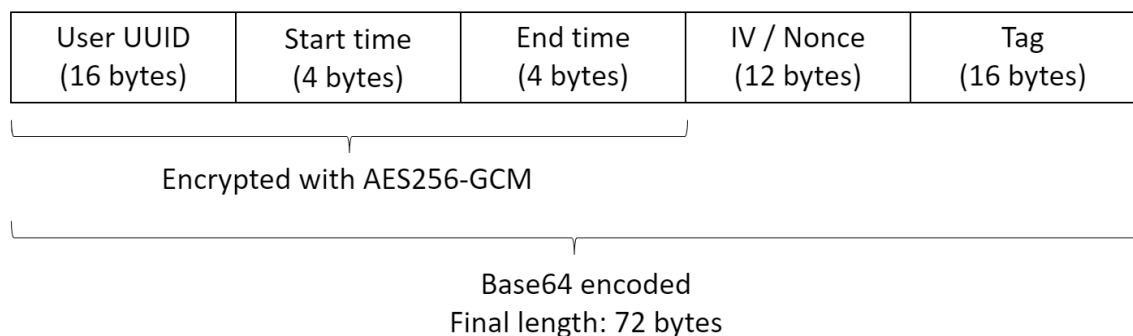


Figure 9: Temporary ID Format

6.5.4 Close Contact Scanning

Two devices must exchange messages over Bluetooth to reliably record each other as close contacts. Bluetooth communication functions similarly to server-client communication, Peripheral is the server and Central is the client. The TraceIT app configures every device to function as both a Peripheral and a Central, but not simultaneously. Each device will be randomly assigned 12 random one-minute periods to function as a Peripheral, which is 80% of a 15-minute period, and assigned to function as a Central for the remaining three one-minute periods, which is 20% of the same period.

When a device runs as a Peripheral, the following events occur. The device will start the BLE service advertisement and a BLE generic attribute (GATT) server, then wait for Central devices to connect to its GATT server. When a Central device is connected, it reads the Peripheral's GATT characteristic to obtain the Peripheral's temporary ID, followed by writing to the characteristic to send its temporary ID and the signal strength of the connection to the Peripheral. The connection is closed after both parties have exchanged their temporary IDs, and the Peripheral resumes listening for connections from other Central devices.

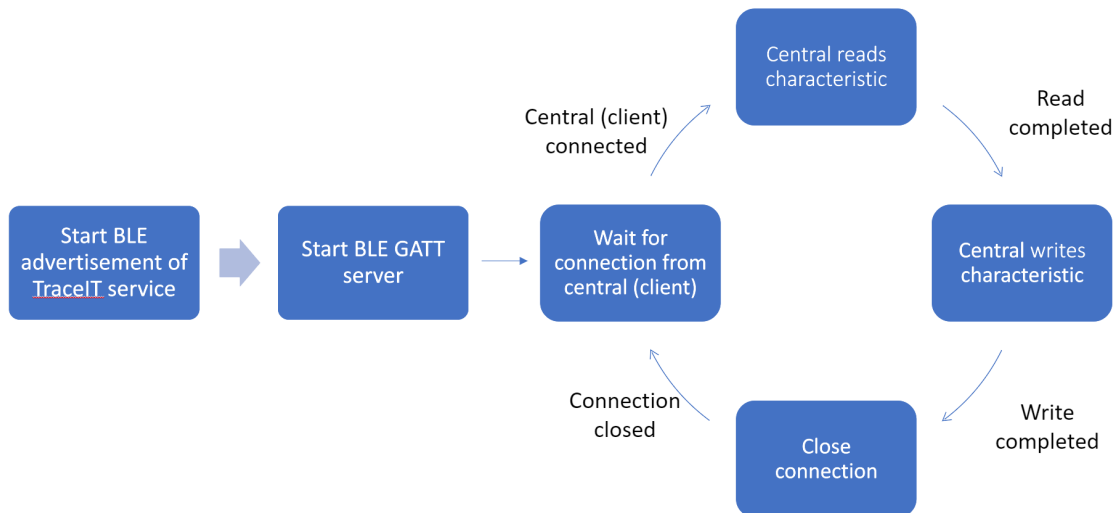


Figure 10: Peripheral Device Behaviour

A device running as a Central will first start scanning for BLE service advertisements. When a service advertisement is discovered, the Central device will stop scanning for advertisements before connecting to the discovered Peripheral's GATT server. The Central device will read the GATT characteristic to retrieve the Peripheral's temporary ID, then write to the characteristic to send its temporary ID and signal strength of the connection. The connection to the GATT server is closed and the Central device restarts its scan for BLE service advertisements.

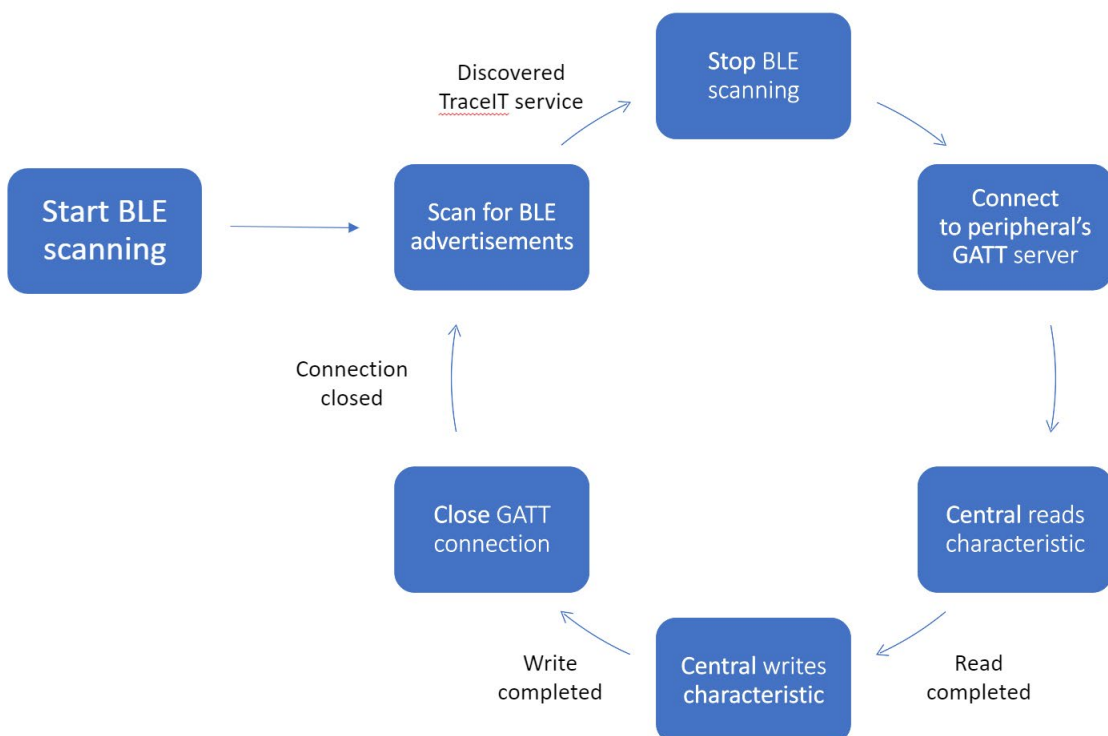


Figure 11: Central Device Behaviour

```
{
  "id": "eD0xHwe1GEa2k/PfUpn5uRA8GthaNApweyaEdwHG0iyy6ttQ1g5q8x2wurNqmxYq+FPW4g=="
}
```

Figure 12: Example of Read Characteristic Data

```
{
  "id": "XXCuAbC6AByzbnz6U3nkCEtEwHwFmOdQrSjn3mSFkK/AsZItbH2wvS8H78KGWewX+iKg6Q==",
  "rssi": -74
}
```

Figure 13: Example of Write Characteristic Data

After closing the GATT server connection on both Peripheral and Central devices, the received temporary ID, signal strength and current timestamp are stored on the device's local storage.

6.5.5 Upload of Close Contacts

The TraceIT app will check with the server on its status to upload close contact data before enabling the upload functionality for the user. If the close contacts of a user have been requested, the close contact upload functionality will be enabled, and the user will need to tap a button to manually upload the close contact data. This ensures that the user has consented to provide the requested data stored on their device.

The payload uploaded to the server will be the data stored during close contact scanning, which contains the temporary ID of the other user, the signal strength of the connection and the timestamp when the close contact was detected.

```
[
  {
    "temp_id": "eD0xHwe1GEa2k/PfUpn5uRA8GthaNApweyaEdwHG0iyy6ttQ1g5q8x2wurNqmxYq+FPW4g==",
    "rssi": -55,
    "contact_timestamp": 1666764747
  },
  {
    "temp_id": "zRFu9MNmhC8SI4I2+aNy6oaXt/SiaR3SJsQvajf5p2burSiy3RmGjeMCu7q10GPol82Eiw==",
    "rssi": -60,
    "contact_timestamp": 1666778543
  },
  ...
]
```

Figure 14: Example Payload of Close Contacts

6.5.6 Verification of Uploaded Close Contact Data

Upon receiving the close contact data uploaded by the app, the TraceIT server will decrypt each temporary ID to retrieve the user's UUID and the start and end validity timestamp. Close contact entries with their recorded contact timestamp outside of their start and end validity period are invalid and will be discarded. The server will insert the user UUID, signal strength and contact timestamp of valid close contact entries into the database.

6.6. Security Considerations

6.6.1 Security of Temporary IDs

As the user's UUID in the temporary ID is encrypted by the server, adversaries will not be able to decrypt it to obtain the UUID. Moreover, modification to the temporary ID will corrupt its encrypted contents and render it invalid. Temporary IDs do not expose any sensitive information, and tampered IDs will be invalidated, hence preventing modification of the ID and ensuring anonymous close contact scanning among users.

The rotation of temporary IDs every 15 minutes ensures that adversaries cannot track users. As users' temporary IDs are not stagnant, adversaries who may have the ability to sniff Bluetooth messages are unable to continuously correlate a user's ID with a location for more than 15 minutes. Therefore, using temporary IDs minimises the likelihood of users being tracked or stalked by adversaries.

Adversaries may attempt replay attacks by retransmitting captured Bluetooth messages with temporary IDs. However, since temporary IDs only last 15 minutes before it is changed, a Bluetooth message with a particular temporary ID cannot be replayed beyond its 15 minutes lifetime as the temporary ID would have expired. Even within the 15-minute valid period, the adversary would need to be close to the victim to capture the Bluetooth message then replay the temporary ID that the victim would already be exchanging with other users anyway. Thus, temporary IDs significantly reduces the impact of replay attacks by adversaries.

6.6.2 Secure Storage of Close Contact Data

Close contacts recorded during scanning will be encrypted in AES256-CBC with PKCS7 padding using a secret key generated on the device, which is stored in the Android Keystore. The close contact data is written to a file in the app's data directory which can only be accessed by the app itself and not by other applications. Hence, the close contact data is securely stored locally on the device.

6.6.3 Upload of Close Contact Data

A user must have a positive infection status and have their close contact data requested by a contact tracer on the tracer web platform before the user can upload it, thus ensuring that they do not accidentally upload their close contact data when it is not required. Users must also manually tap a button to upload their close contact data, as this would confirm that they have read the consent disclaimer and consented to uploading their data.

6.6.4 Disabling Bluetooth Services

Scanning of close contacts cannot function if Bluetooth services are disabled on the device. Adversaries can disable Bluetooth services and ignore prompts by the app to enable Bluetooth to obstruct the scanning functionality. Unfortunately, this is an end-user issue and cannot be completely mitigated. Thus, it is assumed that the user will always have Bluetooth services enabled on their device when using the TraceIT app.

7. Subsystem 2 – Building Access

7.1. Overview

The building access subsystem involves recording users' entry into buildings such as shopping malls or offices. Users would need to use the TraceIT app to scan the QR code at the entrance of a building to submit their entry before entering the building. The app would also indicate whether the user has been allowed or denied entry into the building, which can be verified by security staff on premises.

7.2. Assumptions

Every individual owns an Android smartphone with camera capabilities and camera permissions have been allowed for the TracelT app. It is also assumed that most buildings have been enrolled in this building access system and their respective QR codes are placed at every entrance of the building.

7.3. Tools and Software

Mobile app:

- An Android smartphone
 - Running Android 5.0 Lollipop (API Level 21) or higher
 - Has camera capabilities
- Flutter

Backend:

- Django
- Django Rest Framework
- PostgreSQL

7.4. User Interaction

7.4.1 Building QR Code Generation

The QR code generation is done by the building owner and is not managed by TracelT. When a building owner registers a building with the TracelT administrator, the administrator should provide the UUID of the building to the owner. The owner simply must convert the UUID into QR Code and the QR Code is ready for building access tracking. However, for testing purposes, a page in the tracer portal, `/buildingqrcode` can be used to generate QR Codes. Note that the `/buildingqrcode` page is out of scope and should not be attacked or be used as any part of an attack.

To access the `tracer/buildingqrcode`, visit the tracer portal and login as a tracer. You will be presented with this page when visiting `tracer/buildingqrcode`.

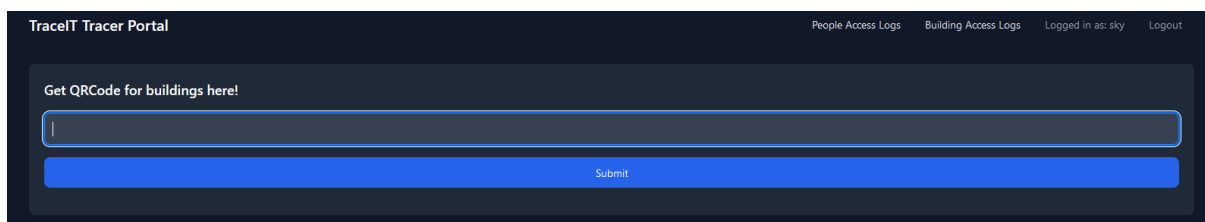
The screenshot shows the 'TracelT Tracer Portal' interface. At the top, there are navigation links: 'People Access Logs', 'Building Access Logs', 'Logged in as: sky', and 'Logout'. The main content area has a dark background with a light blue box containing the text 'Get QRCode for buildings here!'. Below this text is a large, empty text input field. At the bottom of the light blue box is a blue button labeled 'Submit'.

Figure 15: Generating QR Code for Testing

Type in the name of the building to generate the QR code and press submit. Note that the name is case sensitive.

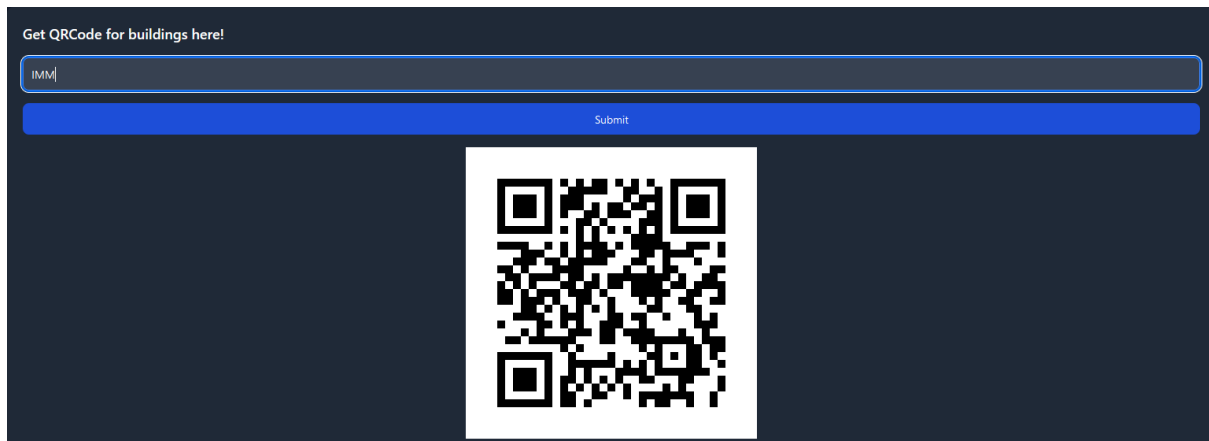


Figure 16: Sample Building QR Code

You can save the QR Code as a picture and print out to scan it with TracelT app.

7.4.2 Submission of Building Entry

Users can tap the floating action button at the bottom right of the tracing screen to open the TracelT app's scanner. Once on the scanner screen, users can point the device's camera at a building's QR code to submit their entry into the building.

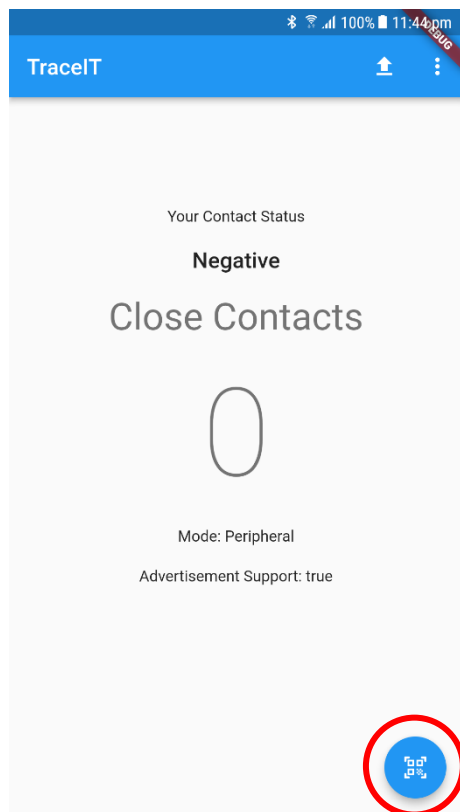


Figure 17: Button to Open In-App Scanner



Figure 18: Scanning a Building's QR code

After a user has scanned a building's QR code, the app will display whether the user's entry has been allowed or denied depending on their infection status. The user can then show the entry status to security staff to verify their status before entering the building.



Figure 19: User Allowed Entry into a Building



Figure 20: User Denied into a Building

7.5. Implementation

7.5.1 Building QR Code

Every building registered in the building access system is assigned a UUID which will be encoded in its respective QR code. Although the generated QR code can be scanned with a regular QR code scanner, it is intended to be scanned with the TraceIT app's built-in scanner.

7.5.2 Submission of Building Entry

As a building's QR code only encodes the text data of the building's UUID, the submission of a user's entry into a building cannot be done with any typical QR code scanner app but only through the scanner in the TraceIT app. When a user scans a building's QR code, the app will read the building's UUID from the QR code and then sends a web request to the server along with its JWT, which contains the user's UUID, to submit the user's building entry record.

Upon receiving a user's building entry submission, the server checks for the user's infection status. If the user infection status is "positive", the server will deny the user's entry and is not recorded in the database. However, if the user infection status is "negative" or "close contact", the user is allowed entry into the building and the entry is recorded in the database. The user's entry status will also be returned to the app and clearly displayed. In the case when an invalid building UUID is scanned and submitted, the server will respond with an invalid building error which is reflected on the app.

7.6. Security Considerations

7.6.1 Physical Replacement of Building's QR Code

The QR Codes are meant to be scanned by users before entering a building to record the user's access. However, this does not prevent adversaries from placing a building's QR code at a different

building or saving the QR code as a picture to scan the QR code at a later time. One possible way to mitigate this problem is to ensure the building's security staff prevents fraudulent QR codes from being placed at the entrances of buildings. Another possible way to reduce the abuse of scanning QR Codes inappropriately is by having the government enforce some legislation to avoid the misuse of the scanning saved QR codes. These problems are difficult to solve with only technological means and require human intervention to ensure that QR Codes are being scanned legitimately.

7.6.2 Screenshot of Approval Screen

An infected user may use a previously captured screenshot of the allowed entry screen and use it to enter a building when they would have otherwise been denied. A building's security staff can request users to prove the legitimacy of the entry submission by scanning the QR code under the staff's supervision or by tapping the device's screen and navigating in the app. It is still possible for adversaries to forge an allowed entry screen even if screenshots have been blocked on the app, making this issue difficult to completely mitigate through technical means. A solution would be to pass a government policy that considers display of forge entry screens to be a crime, thus deterring such actions.

8. Subsystem 3 – Contact Tracer Portal

8.1. Overview

The contact tracer portal subsystem aims to provide contact tracers with the key functionality required to perform their jobs of tracking the close contacts to infected people. A list of functionalities that the portal provides:

- List infected individuals
- Allowing an infected user to upload their close contact information
- Identify close contacts of an infected individual
- List buildings and their user entry logs
- List the buildings accessed by an individual

8.2. Assumptions

Contact tracers are trustworthy users of the platform and will not perform malicious activities.

It is also assumed that there will be no duplicate infections. For example, an individual who is infected with the disease and is considered to be positive cannot be reported again while they are still positive.

While the platform is used to obtain information relating to infected individuals, communication with infected and close contact individuals are done by the contact tracer via phone calls, email or even SMS.

Although the platform is intended for internal use by official contact tracers within an intranet network, the web portal will be publicly exposed for ease of testing.

8.3. Tools and Software

The contact tracer web portal has been designed with two key components, a frontend to provide easy interaction with the system and the backend which communicates with the database to retrieve the data to be displayed on the frontend. The following software has been used to develop the web portal.

Frontend:

- Vue.js/Vue 3
- Nuxt3 with typescript
- TailwindCSS

Backend:

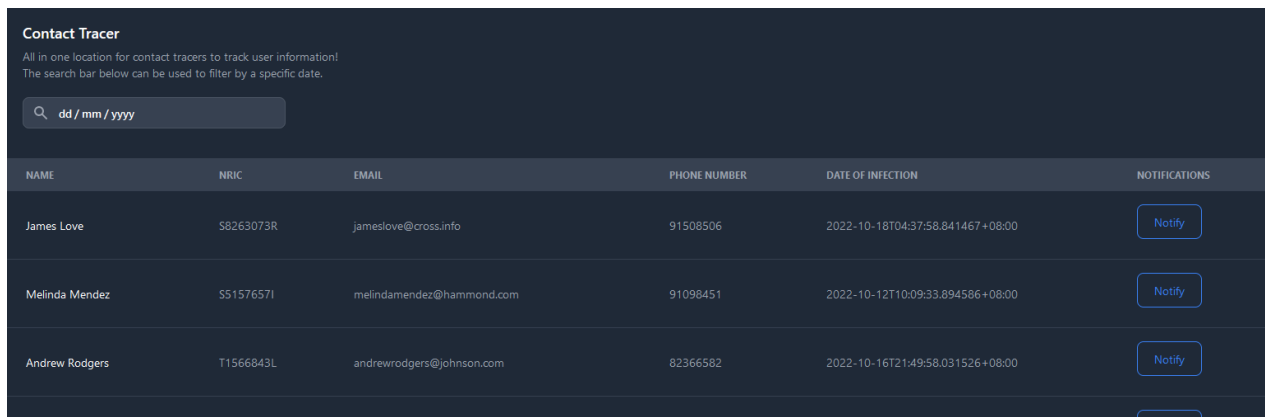
- Django
- Django Rest Framework
- PostgreSQL

8.4. User Interaction

8.4.1 Infected Users and Close Contacts

Contact tracers must be registered and logged in before they can interact with the web portal.

The main page lists the infected users within the past 14 days along with their personal information. The search bar can be used to filter for infected users from a specified date. For example, setting the search date to 25/10/2022, will yield all records of infected users on 25/10/2022 and the previous 14 days inclusive. When a date is not specified, the records from the current date will be displayed.



Contact Tracer
All in one location for contact tracers to track user information!
The search bar below can be used to filter by a specific date.

Search: dd / mm / yyyy

NAME	NRIC	EMAIL	PHONE NUMBER	DATE OF INFECTION	NOTIFICATIONS
James Love	S8263073R	jameslove@cross.info	91508506	2022-10-18T04:37:58.841467+08:00	<button>Notify</button>
Melinda Mendez	S5157657I	melindamendez@hammond.com	91098451	2022-10-12T10:09:33.894586+08:00	<button>Notify</button>
Andrew Rodgers	T1566843L	andrewrodgers@johnson.com	82366582	2022-10-16T21:49:58.031526+08:00	<button>Notify</button>

Figure 21: Sample Home Page to Show Infected Users

After a contact tracer has contacted an infected user to update them on their infection status and provide information on current health protocols, they can update the user's notification status. This will change the status of the infected user to notified and they will be able to upload their close contact information.

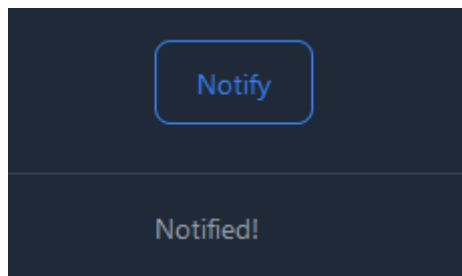


Figure 22: Different Notification Statuses

A contact tracer can only obtain more information about an infected user's close contacts after the user has uploaded their close contact information. Contact tracers can then view the information of the close contacts by clicking on the name of the infected user.



Figure 23: Status of a Notified User

For example, clicking on the user “Peter Lynch” after they have been notified and uploaded their close contact information shows their close contacts.

Close Contacts					
NAME	NRIC	EMAIL	PHONE NUMBER	RSSI	CLOSE CONTACT DATE
Monica Roberts	S4645171L	monicaroberts@palmer.net	84031802	-22	2022-10-22T08:10:51.489889+08:00
James Todd	T0954314P	jamestodd@day-patterson.com	90886385	-70	2022-10-20T03:45:25.306600+08:00
Nicholas Stephens	S6029292J	nicholasstephens@shelton.com	84639398	-44	2022-10-10T14:33:42.573230+08:00

Figure 24: Sample Screen for a Close Contact of an Infected User

Personal information of close contacts is displayed so that contacts tracers can inform the close contacts by phone call or email to isolate and follow any health protocols. RSSI, the Bluetooth signal strength when the close contact is recorded, can assist contact tracer in estimating the relative distance the close contact was from the infected user, which can be essential in prioritising who to contact first depending on the type of disease in a pandemic.

8.4.2 Building Access

Contact tracers can also view the building access logs by clicking on “Building Access Logs” in the navigation bar at the top right of the web page.

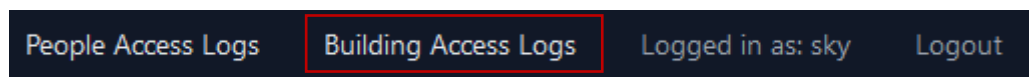


Figure 25: Link to Building Access Logs in the Navigation Bar

The page shows the list of buildings enrolled in the building access programme and their postal code.

Building Access	
Contract tracers can find building access entries here. Click on the building name to find check the access logs for each building.	
BUILDING NAME	LOCATION
Bugis Junction	188024
Far East Plaza	228213
ION Orchard	238801
Bedok Mall	467360
NEX	556083
JCube	609731
Rivervale Mall	545082
Paya Lebar Square	409051
Rochester Mall	138639
IMM	609601

Figure 26: List of Buildings in Building Access Logs

Contact tracers can view the user entry logs of a building by click on any of the building names. They will be able to view the building access logs containing personal information if they wish to contact the user.

Building Access logs

Building: ION Orchard

Contract tracers can view all the access logs

The search below will show all entries to a building on that date.

Q

dd / mm / yyyy

NAME	NRIC	GENDER	EMAIL	PHONE NUMBER	BUILDING ENTRY TIME
Heather Taylor	S3979717Z	Female	heathertaylor@allison.biz	82799438	2022-10-25T12:35:19.984249+08:00

Figure 27: Sample of a Specific Building's Logs

By default, the logs shown will be from the current date that the log is accessed. Access logs of a particular date can be retrieved by searching for that date using the search bar. The purpose of the building access log is for contact tracers to track entry into a building that may become a potential infection cluster.

8.4.3 People Access

Contact tracers can also view the building accessed by a specific person under the People Access Logs.

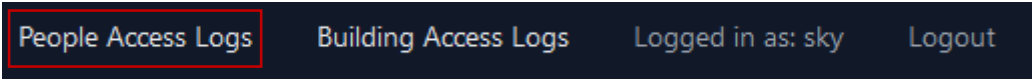


Figure 28: Link to People Access Logs in the Navigation Bar

On this page, contact tracers can view the building access log of a specific individual. However, by default, the page will not show any records as it requires an NRIC to display the specific user information.

Building Access

Contract tracers can find building access entries here for a specific user.

Q

dd / mm / yyyy

Search for NRIC

Search

BUILDING NAME	LOCATION	ACCESS DATETIME
No records found		

Figure 29: People Access Log Page

When an NRIC is provided, the page will display all the buildings that the infected user has accessed before, even if it is more than a year ago. When an NRIC is provided, with the page will display all the buildings that the infected user has accessed before, even if it is more than a year ago.

Building Access
Contract tracers can find building access entries here for a specific user.

BUILDING NAME	LOCATION	ACCESS DATETIME
ION Orchard	238801	2020-09-17T18:55:15.810173+08:00
ION Orchard	238801	2021-02-27T02:21:26.935694+08:00
JCube	609731	2020-10-19T09:53:12.101479+08:00
JCube	609731	2022-01-20T12:41:12.950636+08:00

Figure 30: Sample of Find Building Access Logs with NRIC

Contact tracers can also provide a date in addition to the NRIC to obtain the buildings accessed on that specific date. However, providing a date without a NRIC will yield no records.

Building Access
Contract tracers can find building access entries here for a specific user.

BUILDING NAME	LOCATION	ACCESS DATETIME
JCube	609731	2022-01-20T12:41:12.950636+08:00

Figure 31: Example of Further Filtering Done with Date

The purpose of this page is to help contact tracers track a user's entry across multiple buildings to understand where the infected user has visited.

8.4.4 Additional Pages

There are two other pages that the contact tracer can access to add or access information specific to the web portal. However, these pages are not within the scope of this project and are placed here to conveniently add data and ease demonstration. These pages should not be attacked or used as any part of an attack.

The infections page can be used to add infection records, while in reality, the records are meant to be added by the administrator.

Add Infection History Records
Declare a list of users as infected with their NRICs.

NRICs (separated by line)

Figure 32: Add Infection Records

The building QR code page can be used to easily obtain a building's QR code that users have to scan before entering.

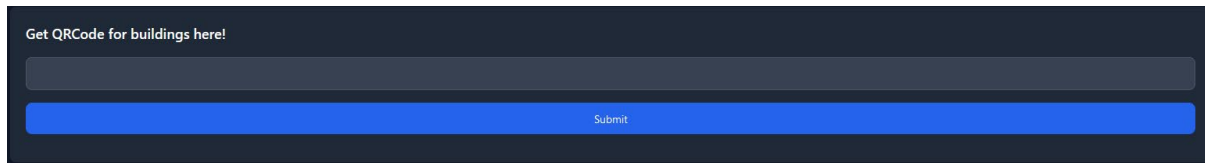


Figure 33 Get QR Code for Building

8.5. Implementations

8.5.1 Tracing Infected Users and Close Contacts

The TraceIT web application allows tracers to manage the people who have tested positive for an infection. The web frontend calls the routes from the Django backend API within the `/api/infections` routes. The API provides a functionality to list all the users that have had been tested positive and reported to the system administrator. TraceIT defines a positive infection as having been infected by the pandemic within the last 14 days. By default, the API will list all users with an infection record from a specified date till 14 days ago. If a date is not specified, the date of accessing the API is used instead.

After identifying the infected users, the tracer can notify the infected user. Once the user has uploaded their close contact information, the tracer can request to view a user's close contact information. The backend will provide all the close contacts related to its current infection for a specific user, and their personal details so that the tracer can contact them.

8.5.2 Tracing Potential Infection Clusters by Location

The tracer can also view the building access logs for a specific building, or the building accesses made by a particular user. These API can be found in `/building` routes. The building access logs will show all the users that have access to the building on a particular date that the tracer can specify. By default, the date of accessing API would be used.

The people access log provides the functionality of displaying all the building access made by a particular user identified by their NRIC. Additionally, a date can be specified to show access records related to a specific user on a chosen date.

8.6. Security Considerations

8.6.1 Data Privacy and Leakage

While the tracer accessing the portal must have signed the NDA, it is still possible for the tracer to abuse the system. The tracer could leak the personal information of the users stored within the database or stalk users using building access logs. However, due to the information needed by the tracer, the tracer inevitably requires personal information to perform their job. TraceIT has attempted to reduce the amount of data the tracer can read by showing them a need-to-know basis by lowering the information displayed for each query. For example, the infected person logs only show infections within the past 14 days, while the building access logs only show the records for a single day. Limiting the records makes data leaks more difficult for an adversary.

While removing this security concern is impossible, TraceIT attempts to reduce the possible information leak.

9. Subsystem 4 – Researcher Web Portal

9.1. Overview

This subsystem aims to provide a web portal for researchers, such as epidemiologists and immunologists, to gather data regarding the pandemic situation. These researchers could study the data collected from the main database, but they should not have access to users' personal information which could be used for identification. The data provided by the web portal are instead anonymised with k-anonymity property to prevent data re-identification. This web portal allows researchers to view anonymised data collected from the tracer portal and query specific fields to filter the data in the research portal.

Homogeneity attack refers to gathering a general classification for a group of quasi-identifiers with similar sensitive data.

Background knowledge attack refers to using existing knowledge regarding the user to link an anonymised data to that user.

9.2. Assumptions

Although the platform is intended for internal use by approved researchers within an intranet network, the web portal will be publicly exposed for ease of testing.

An adversary is assumed to have no access to the contact tracer portal. Otherwise, the adversary can retrieve user's information which renders the research data irrelevant to the adversary.

9.3. Tools and Software

There are three parts to the researcher web portal. Firstly, the k-anonymity process which processes the data from the tracer database to the researcher database. Secondly, the researcher web backend communicates and pull data from the researcher database. Finally, the researcher frontend which displays the data from the researcher database while providing certain querying features on the displayed data.

Frontend

- Vue.js/Vue 3
- Nuxt3 with typescript
- TailwindCSS

Backend

- Django
- Django Rest Framework
- PostgreSQL

Anonymisation

- Mondrian k-anonymity by kaylode
- Python scripting
- PostgreSQL

9.4. User Interaction

9.4.1 Understanding the Data

TracelT Researcher Portal Logged in as: woo Logout

Researcher DB

Search for vaccines Search for postal Search for dob Search for total infection Please select gender

Search

DATE OF BIRTH	GENDER	POSTAL CODE	LIST OF VACCINES	LATEST CLOSE CONTACT	LATEST INFECTED DATE	TOTAL INFECTION	TOTAL CLOSE CONTACT AS INFECTED	TOTAL CLOSE CONTACT WITH INFECTED
50-59	Male	*	Pfizer-BioNTech	2022-10-12	None	0	0	85
50-59	Male	*	Pfizer-BioNTech/Moderna/Sinovac/Novavax	2022-10-21	2020-06-26	1	192	76
50-59	Male	*	Pfizer-BioNTech/Pfizer-BioNTech	2022-10-10	None	0	0	73
50-59	Male	*	Pfizer-BioNTech/Moderna	2022-10-12	2021-10-05	1	148	75
50-59	Male	*	Pfizer-BioNTech/Moderna/Sinovac/Novavax	2022-10-21	2020-06-26	1	192	76

Figure 34: Researcher Main Web Page

After login into the researcher web portal, a researcher will see a table containing research data, which can be categorised as quasi-identifiers or sensitive data.

Firstly, sensitive data:

Table 5: Sensitive Data in Researcher Web Portal

Column Name	Description	Reasoning
List of vaccines	<p>Display a full list of vaccines taken by that specific user separated by the delimiter “/”.</p> <p>If a vaccine contains the same delimiter in its name, the anonymisation process will filter them out. For example, if there is a vaccine called Pf/zer, the anonymisation process will treat it as Pfizer.</p> <p>For a user who has never been vaccinated before, the data will be empty.</p>	A researcher might be interested in finding the effectiveness of vaccines and vaccine combinations that reduces the risk of infection.
Latest close contact	<p>Display the latest date in which the specific user has a close contact with an infected individual.</p> <p>For a user who has never been in close contact before, the data will show “None”.</p>	A researcher might be interested in finding the incubation period of the virus.
Latest infected date	<p>Display the latest date in which the specific user is diagnosed as infected.</p> <p>For a user who has never been infected before, the data will show “None”.</p>	A researcher might be interested in finding the incubation period of the virus and the current rate of infection.
Total infection	Display the total amount of times in which the user has been diagnosed as infected.	A researcher might be interested in determining which group of people are at higher risk of being infected.
Total close contact as infected	Display the number of times in which the user has close contact with while being diagnosed as infected.	A researcher might be interested in determining which group of people have a higher exposure rate.

Total close contact with infected	Display the number of times in which the user has been close contact with any infected individual.	A researcher might be interested in determining which location has a higher exposure rate and which group of people is at higher risk of getting infected.
--	--	--

Secondly, quasi-identifiers:

Table 6: List of quasi-identifiers in researcher web portal

Column Name	Description	Generalisation Scheme
Date of birth	Display the age of the user	From least generalised to most generalised: <ul style="list-style-type: none"> • 1951-01-04 • 1951-01 • 71 • 70-79 • *
Gender	Display the gender of the user	From least generalised to most generalised: <ul style="list-style-type: none"> • Male / Female • *
Postal code	Display the user's postal code	From least generalised to most generalised: <ul style="list-style-type: none"> • 657843 • 65784* • 6578** • 657*** • 65**** • 6***** • *

With the provided quasi-identifiers and sensitive data provided, the researcher can draw an accurate finding regarding specific groups of people in relation to the ongoing pandemic situation.

9.4.2 Query of Research Data

Researcher DB

Figure 35: Researcher Web Page Query Feature

The researcher web portal also provides tools for researchers to filter existing data. The type of filter the researcher could query on are keywords of vaccines, keywords of postal code, date of birth of user, number of total infections of user, and the gender of user.

Table 7: Query Fields

Filter label	Description	Example
Search for total infection	Search for a specific value for the total infection.	For example, if the researcher input "0" into the total infection, only records with total infection of 0 will be displayed on the webpage after clicking the search button.

Search for vaccines	Search for certain keywords in the list of vaccines. The keyword search is case sensitive.	For example, if the researcher search for “Moderna/Sinovac”, only records that have the string “Moderna/Sinovac” in its list of vaccines will be displayed after clicking the search button.
Search for postal	Search for records that contain the specific postal code.	For example, if the researcher searches for “*”, only records that contain “*” in its postal code will be displayed. For example, if the researcher searches for “1”, only records that contains “1” in its postal code will be displayed.
Please select gender	Filter for records that has the specified gender. There are three options, any to display (male, female, *), male to display (male), and female to display (female).	-
Search for dob	Filter for records that contain the specified date of birth of the user. There are two options to use this filter. First, search by decade keyword. Secondly, search by exact age.	For example, in the case of search by decade. The researcher should search for “50-59” or any other value of “X0-X9” where X is an integer between 0 to 13 inclusive. This will return records that contain the value in its date of birth, which means any specific age where 52 is displayed will not be shown. For example. In the case of search by age. The researcher should search for a specific number. This will return any records that match the age and any records with decade as date of birth that includes the specified age. If the researcher searches for “22”, any records with “22” or “20-29” as the date of birth will be shown.

With the filtering function provided, researchers could mix and match the queries to gather specific results for analysis. In the case of no input for the querying fields, the filter would not filter that specific field. As a result, if all the querying fields are empty, the search result will return the unfiltered list of the research data.

9.4.3 Cached Data

When the researcher logs into the research web portal, the data shown will be cached. In the case of gathering updated data, the researcher should always refresh the webpage in order to get the new set of data from the research database.

9.4.4 Data Refresh Schedule

Data from the main database will undergo anonymisation and replace the data in the research database regularly at midnight. As a result, the researcher should expect the same set of data of the during the entire day.

9.5. Implementation

9.5.1 Anonymisation

The anonymisation process is done using k-anonymity with the algorithm of Mondrian. Mondrian is a simple and efficient greedy approximation algorithm for several general-purpose quality metrics.

The k-anonymity will process our 3 quasi-identifiers which are (date of birth, gender, postal code), which means that for any distinct groups of quasi-identifiers, there can never be any less than k members for a distinct group. The current k-anonymity for the anonymisation process is set to 3 due to the reduced size of records in the current database as well as to display less generalized records for demo purposes.

The Mondrian k-anonymity process is done through a third-party package by kaylode. And our generalization schemes are:

Table 8: Generalization scheme used for k-anonymity

Column Name	Generalization Scheme
Date of birth	From least generalised to most generalised: <ul style="list-style-type: none">• 1951-01-04• 1951-01• 71• 70-79• *
Gender	From least generalised to most generalised: <ul style="list-style-type: none">• Male / Female• *
Postal code	From least generalised to most generalised: <ul style="list-style-type: none">• 657843• 65784*• 6578**• 657***• 65****• 6*****• *

The anonymisation process will attempt minimal generalisation for the distinct groups of quasi-identifiers. This means that there might be different groups of quasi-identifiers with different levels of generalisation while maintaining the aspect of k-anonymity.

The k-anonymity level set for the anonymised data is 3.

The level of anonymisation concludes at k-anonymity without going further into l-diversity. This is to provide a no loss data coverage of the data collected from the tracer portal, which means that there should be the same number of records in the research database as the number of user records in the main database. Furthermore, the sensitive data provided could be similar for a particular quasi-identifiers group as it is vital to draw conclusion for the ongoing pandemic. Therefore, the anonymisation process will be only using the standard of k-anonymity.

9.5.2 Getting Data and Pushing Data

With the anonymisation algorithm in place, this implementation details the process of gathering data from main database and process it to the research database. This is done using python psycopg2 to connect to the database and to perform queries.

Firstly, gathering data from the main database is done by collecting data from a SQL view. This SQL view is created to gather data for the researcher web portal with the relevant column name and joint operations. When the data has been gathered, it will be stored in a CSV file for later usage during the anonymisation process.

Secondly, generating of generalisation schema. With the gathered data from the main database, a generalisation scheme will be generated for the anonymisation process and written into different CSV file for the 3 different quasi-identifiers.

Thirdly, the execution of the anonymisation process, where a set of anonymised data will be generated based on the data gathered from the main database and the generalization schemes. This data will be saved as a CSV file for auditing purposes and to push into the research database.

Lastly, a connection will be established with the research database and the anonymised data will be pushed using psycopg2 with proper filtering to prevent certain SQL injections.

9.5.3 Frontend Query and Caching

The researcher web portal also provides a caching mechanism to reduce waiting between each filtering query due to the large sum of data. Upon entering the researcher web portal, the research data will be gathered once and cached. Hence, a refresh is required on the researcher web portal to update the data from the research database.

The querying is done entirely on the frontend of the researcher web portal. This is to prevent long waiting times, as mentioned earlier and to reduce the load of handling requests on the researcher backend and research database.

9.6. Security Considerations

9.6.1 SQL Injection Attack on Anonymisation Process

The anonymisation process is not directly accessible by the public or other user roles (tracer or researcher). However, SQL injection still pose a risk such as n-th order SQL injection. As a result, the anonymisation process has been strengthened by sanitising database queries using the python package psycopg2.

9.6.2 Compromised Researcher Web Portal

While the researcher web portal provides a layer of data security through anonymised data, researcher web portal is at risk of compromise where an adversary might gain entry rights into the researcher web portal through social engineering. In this case, data security measures have been put in place such that an adversary would not be able to link any research data to any specific user due to the k-anonymity standard, assuming the adversary does not have any background knowledge.

An additional security measure is the isolation of the research database from the other databases. This minimises the impact of an adversary gaining full control of the research database as any queries will remain in the research database. Furthermore, the transfer of data is unidirectional, where it is retrieved from the main database, anonymised then inserted into the research database.

9.6.3 Background Knowledge Attack

In the event that an adversary has background knowledge of users, they could identify an individual based on their unique sensitive data. Unfortunately, this is an unsolved problem regarding data privacy and TraceIT cannot mitigate this issue.

10. Subsystem 5 – Secure Authentication

10.1. Overview

The secure authentication subsystem aims to provide a consistent and secure authentication experience through the numerous applications found in the project. An overview of the subsystem is as follows:

- Password complexity checks
- JSON Web Tokens with expiring rotating tokens
- Two Factor Authentication with Time-based One Time Passcode (TOTP)
- Role-based authentication between the applications

10.2. Assumptions

Typically, in a secure environment, the contact tracers and researchers will have administrators that would register new accounts in a separate portal as part of the onboarding process. However, as part of allowing testing in the numerous web applications, the registration endpoint and page will be enabled to provide the testers with accounts to access. As such, the registration endpoints may not be strictly hardened and should not be extensively tested on. This assumption does not include the two-factor authentication.

10.3. Tools and Software

The secure authentication subsystem is built on the same backends as the various subsystems stated above. The following software is used as part of the development of the authentication subsystem.

Backend

- Django
- PostgreSQL
- Vault

Frontend

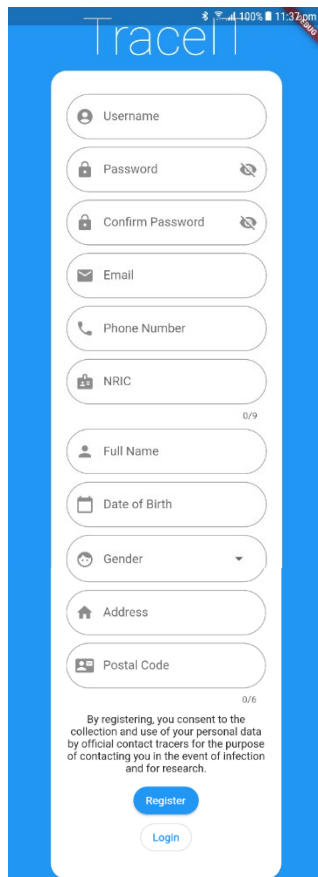
- Flutter (Contact Tracing Application)
- Nuxt 3 with Typescript (Tracer and Researcher Portal)
- TailwindCSS (Tracer and Researcher Portal)

To manage the secret keys used by the Two Factor Authentication, Vault's built-in TOTP feature is used to generate and manage the one-time passcodes.

10.4. User Interaction

10.4.1 User Registration and Login

New users will first need to register a user account and log in before using the application or portals. The information new users will need to provide are their username, password, email address, phone number, national identity registration (NRIC) number, full name, date of birth, gender, home address and postal code. The backend will ensure a high password complexity is met to prevent common password attacks that may occur on the user accounts.



Tracell

Username

Password

Confirm Password

Email

Phone Number

NRIC

0/9

Full Name

Date of Birth

Gender

Address

Postal Code

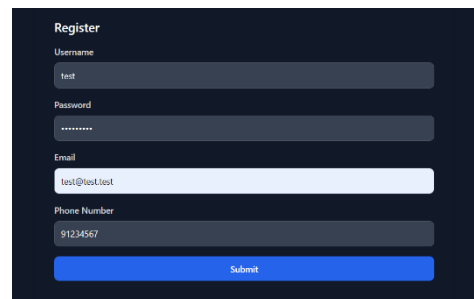
0/6

By registering, you consent to the collection and use of your personal data by official contact tracers for the purpose of contacting you in the event of infection and for research.

Register

Login

Figure 36: User Registration Page in Application



Register

Username

test

Password

.....

Email

test@test.test

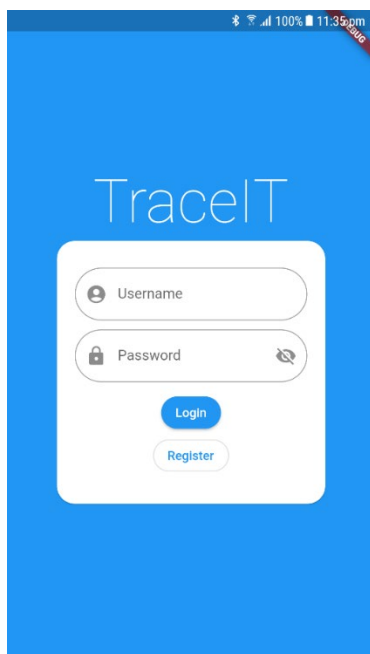
Phone Number

91234567

Submit

Figure 37: User Registration Page in Portals

After registration, new and existing users can log in with their username and password and will be directed to complete their two-factor authentication.



Tracell

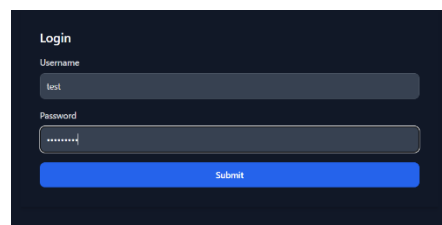
Username

Password

Login

Register

Figure 38: Login Screen for Application



Login

Username

test

Password

.....

Submit

Figure 39: Login Screen for Portals

10.4.2 Time-based One-time Password Registration and Input

New users who have logged in for the first time will be prompted to generate a time-based one-time password (TOTP) QR code to be registered with their 2FA app. Once generated, the user can use a secondary device to scan the QR code and register on that device or, more commonly, register the TOTP on the current primary mobile device.

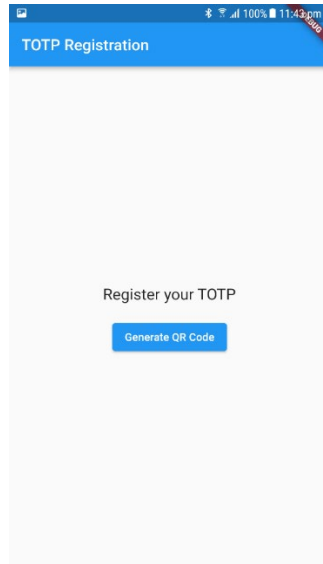


Figure 40: Preparing to generate QR code on Application

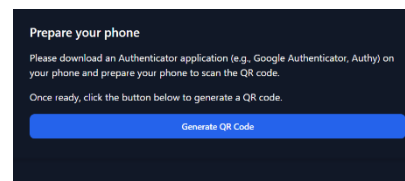


Figure 41: Preparing to Generate QR Code on Portals

Once the user is ready, they may press the “Generate QR Code” button, which will call the backend to provide a QR code that contains the shared secret key between the server and the user. The user will then scan the QR code with their authenticator application of their choice and press login when they are ready to enter their TOTP. Do note that the user only has a single chance of generating the QR code, and if they were to close or reload the page, their new account would be locked out with no recovery access.

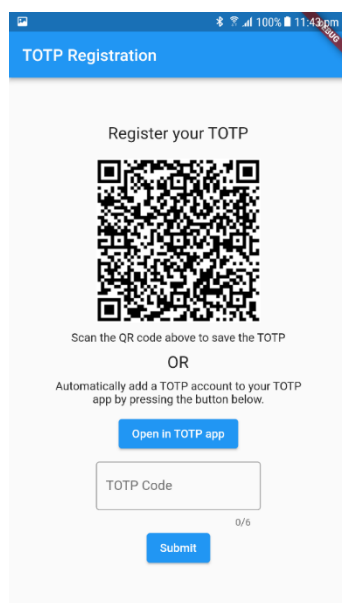


Figure 42: TOTP QR Code for Application

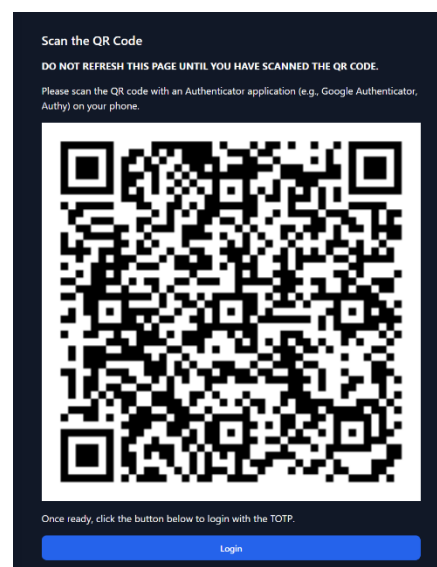


Figure 43: TOTP QR Code for Portals

The user will then key in their TOTP generated from their phone to complete the authentication process.

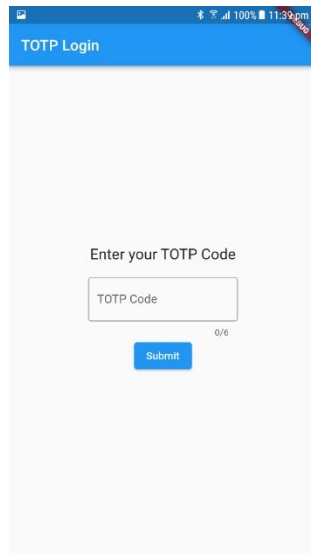
A mobile application interface for TOTP login. At the top, a blue header bar contains the text "TOTP Login". Below this, the text "Enter your TOTP Code" is centered. Underneath is a text input field labeled "TOTP Code" with a character count "0/6" to its right. A blue "Submit" button is positioned below the input field.

Figure 44: TOTP Input Page for Application

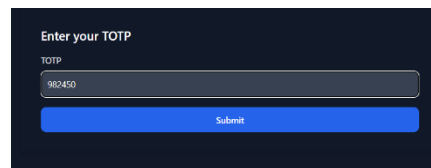
A web portal interface for TOTP login. It features a dark background. At the top, the text "Enter your TOTP" is displayed. Below it, a label "TOTP" is followed by a text input field containing the value "982450". A blue "Submit" button is located at the bottom of the form.

Figure 45: TOTP Input Page for Portals

The user may choose to log out of the portal to invalidate any of their session details to ensure no party steals their access tokens.



Figure 46: Logout in Navigation Bar

10.5. Implementation

10.5.1 Password Complexity Checks and Storage

To ensure passwords are not too simple, Django provides built-in tools to ensure passwords do not contain simple words or values found in the user's email address or username. This ensures that the adversaries are not able to perform educated guesses on their passwords.

All passwords are also hashed and salted with the Argon2 algorithm in the database.

10.5.2 Refreshable JSON Web Tokens

After successful authentication, the server replies with an access token and a refresh token. The access token is a short-lived (5 minutes) token that can be used in the Authorization header of HTTP requests to authenticate the user for future requests. Once the access token expires, the web frontends or the application will use the refresh token to request a new access and refresh token from the server without requiring the user to log back into the application. The refresh token will automatically expire in a day and will be blacklisted when the user logs out or requests a new token pair.

The refreshable tokens ensure that if the user has their access tokens compromised through man-in-the-middle attacks, the access tokens will be short-lived. They would still require the refresh tokens to maintain a persistent connection with the applications. The tokens are signed with a secret key managed by Vault and GitHub Actions using the HMAC-SHA256 algorithm to ensure an adversary cannot spoof the tokens.

10.5.3 Two Factor Authentication

After the user keys in their username and password, a temporary JSON Web Token is provided to the client to allow the user to key in their one-time passcode from their authenticator application. The temporary token expires in 5 minutes and is not refreshable to ensure that the user provides a valid one-time passcode within a small window of time. The two factor authentication secrets are generated and stored in Vault, with the user UUID as their identifier. Vault will also verify all one-time passcodes sent from the user through the Django backend services.

10.5.4 Role-Based Authentication

To ensure the separation of concerns between the multiple applications, role-based authentication ensures that authorized users are only allowed to access the portals they have registered from. Each user is added to a specific table in the database depending on the role that they are required to perform. For example, when a new user registers from the mobile application, a new row is added to the User table, which will be referenced when the user logs into the application.

10.6. Security Considerations

10.6.1 Stolen Tokens from Phishing Attacks

While refreshable JSON web tokens reduce the window of opportunity for adversaries to reuse access tokens, the subsystem does not prevent against attackers from stealing tokens through phishing attacks. The secure authentication subsystem attempts to reduce this risk by only exposing the refresh token when the user must refresh their access tokens and when they initially complete the authentication flow and setting access token validity to a short duration of a few minutes. However, if an adversary dumps the token from session storage through phishing or direct physical access, they would be able to utilize the refresh token to generate new tokens. This form of attack is hard to mitigate through technology. However, it can be mitigated by educating the platform users not to allow access to their devices and to log out of the sessions once they are done with their activities, especially on the contact tracer and researcher portals.

10.6.2 Transmitting Sensitive User Data Securely

To ensure that the user's passwords and one-time passcodes do not get sniffed through man-in-the-middle attacks, TLS is employed through the portions of the network that TracelT manages. This portion is further explained within Subsystem 9 – Secrets Management and Public Key Infrastructure. However, as part of the demonstration, the traffic between the NUS reverse proxy and TracelT reverse proxy may not be using any form of encryption. As such, there may be a man-in-the-middle channel between the two proxies, which is unavoidable but can be mitigated if TracelT is deployed as a standalone application without the NUS reverse proxy.

11. Subsystem 6 – Database Management

11.1. Overview

This subsystem involves the management of all the database used in the system as well as generating large amounts of data for testing purposes. In addition, confidentiality will be implemented in the form of encryption of database data stored using TDE.

There will be a total of 3 different databases, and they are databases that handles web authentication, tracing data, and research data respectively. These 3 databases are separated logically.

11.2. Assumptions

The systems that implement the mentioned components are not user-facing and can only be observed/accessed by the developers and operators of this project.

11.3. Tools and Software

The tables in the databases are generated using Django migration feature whereas the databases and roles related to the database are generated through Continuous Deployment (CD) using Ansible.

The contents of the tables are generated using a mix of python scripts and PostgreSQL scripts. The python scripts generate the primary tables while the PostgreSQL scripts generate the secondary relations that draw relations between the primary tables' data.

The list of tools used are:

- Creating tables for each database
 - Django migrations
- Generating Mock data
 - Python scripts
 - PostgreSQL scripts

11.4. User Interaction

This subsystem does not provide any direct interaction with the user.

11.5. Implementation

11.5.1 Managing Roles Permissions

In order to minimise the impacts of a compromised web portal, minimal access rights have been implemented to the roles used by the web backends that is communicating with the database. As such, the roles are not able to perform any superuser actions such as creating roles, databases, etc. In addition, these roles are contained within the default schema of the logical database and the default schema have restricted connections rights as compared to a normal default schema.

A full permission granted to the roles are select, insert, update, and delete on tables from the default schema (public) for their respective databases.

These settings are done directly through ansible using certain SQL commands that could grant and revoke permissions to schema and roles.

11.5.2 Creating Tables for the Databases

The tables for the databases are managed through the migration feature provided by Django. As a result, any table dependencies will be installed correctly without any manual creation of these tables which may be prone to errors if done manually.

Routers have also been defined that handles the separation of these 3 databases and their respective tables. The result allows an easier migration with the Django manage.py migrate command for each of the databases.

11.5.3 Generating Mock Data

Using python faker package, data can be generated for the primary tables that closely simulates real world data. These data include a user's name, user's email, user's date of birth, etc.

After the population of the primary tables, a PostgreSQL script will be used to generate relations for the secondary tables. This is done through PostgreSQL due to the dependencies for each relation of the tables which is highly unique for the database.

The entire process of the mock data generation is as follows using a single python script. Firstly, it establishes connection with the main database. Secondly, it generates the data for the primary tables

and insert them accordingly. And lastly, it run the PostgreSQL script using `psycopg2` which will generate the rest of the data for the secondary tables.

11.5.4 Secure Encryption of Data Stored in the Database

PostgreSQL TDE has been used to manage data encryption at rest. The TDE feature available for PostgreSQL is provided as a patch by CyberTEC. The key used by TDE to encrypt the database data will be stored in vault.

The GitHub action will automatically build the PostgreSQL with the TDE patch applied.

11.5.5 Managing Data Encryption in Transit

Communication between web server (Django) and the database are important areas to cover. Attacks could be carried out in which data are leaked unencrypted to an internal attacker using network sniffing techniques on the same network. However, the database traffic in the system is not exposed publicly.

To add an additional layer of security to manage data encryption at transit, SSL has been used to encrypt these data. For more information, refer to subsystem 9 regarding public key infrastructure (PKI) settings.

11.6. Security Considerations

11.6.1 Compromised exposed database roles

If any web backend has been compromised, an attacker could gain control of the database role managing the communication between the database and the web backend. As such, security measures have been implemented to limit the permissions given to these roles. This provides a layer of protection of the data stored in the database, where the limited roles are not able to perform any superuser actions.

In addition, database isolation has also been put in place such that for any compromises in any databases, the damages will be retain in it.

11.6.2 Copying data from the database

Database data stored on disk may be at risk of being extracted by an attacker. This data compromises sensitive data of user information. An adversary could have file access to the database server where it allows them to extract all files stored. With the implementation of PostgreSQL TDE, even if an adversary were to extract data from the disk of the database server, the data will not be accessible.

12. Subsystem 7 - System and Security Operations

12.1. Overview

The subsystem encapsulates around the various aspects of the DevSecOps lifecycle. The components of the subsystem are as follows:

- Automated Continuous Integration (CI) testing
- Regularly scheduled Static Application Security Testing (SAST)
- Dynamic Application Security Testing (DAST)
- Vulnerability management of project software dependencies
- Automated Continuous Deployment (CD) to production environment
- Server Hardening

12.2. Tools and Software

Continuous Integration and Deployment

- GitHub Actions
- Docker
- Self-hosted GitHub Action runners
- Ansible

Security tools

- Semgrep
- PyCharm Python Security Plugin
- GitHub Dependabot
- OWASP ZAP

GitHub Actions is an online CI/CD platform that can be used to deploy pipelines that automate testing and deployment of code. Pipelines on the platform can be further customized by actions shared on the GitHub Marketplace, which are reusable modules that wrap around commonly performed steps.

GitHub Action runners are agents that is administered from GitHub and execute jobs when code is pushed or when manually a job is manually dispatched. Self-hosting GitHub Action runners from within NUS's infrastructure would allow jobs that require resources only available in the local network, such as the virtual machines provisioned by NUS Tech Services, to be executed normally without having to request for any firewall changes. As the repositories were set to private, there is also an added benefit that the number of billable minutes for running workflows is 0, as compared to running workflows of private repositories on GitHub-hosted runners, which have a limit of 3000 billable minutes per month.

Docker is utilised by GitHub Action runners to spawn ephemeral execution environments for code as well as other dependent services during continuous integration tests, such as PostgreSQL databases or Vault servers, so that developers can test their changes more quickly and without causing any permanent impacts to the production environment.

Ansible is used to remotely manage the provisioned virtual machines by pushing Ansible playbooks that perform automated configuration and execution of services, as well as security hardening of the provisioned virtual machines.

The code repositories are scanned using SAST tools such as Semgrep and PyCharm Python Security Plugin, that can detect vulnerable coding patterns and configurations based on a set of pre-defined rules. GitHub Dependabot also scans the code repositories and alerts when a new vulnerability is detected in the project dependencies. OWASP ZAP, a DAST tool, was also used to scan a development instance of the TraceIT architecture to detect for common web vulnerabilities and misconfigurations.

12.3. Developer Interaction

When a developer pushes new changes to a code repository, multiple jobs are kickstarted by GitHub, such as continuous integration tests and SAST scans. The results from these jobs provide other developers with information on the correctness and security of the new changes and may approve them if they pass the tests and scans.

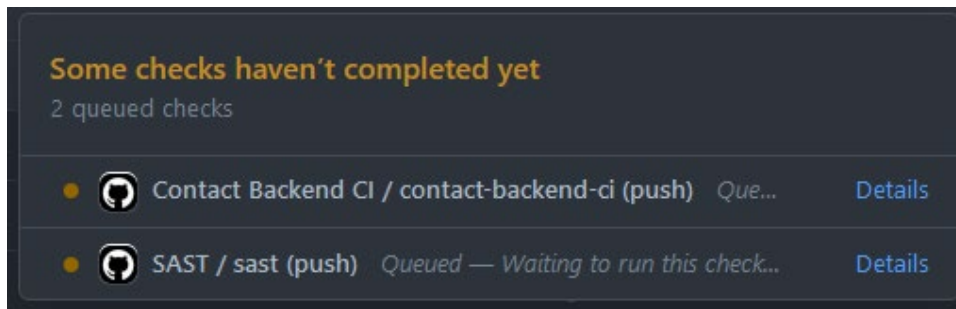


Figure 47: Pending CI Operations on GitHub

12.4. Operator Interaction

Operators can use the GitHub Actions page to centrally manage the deployment and teardown processes to the environments. They simply need to indicate which environment that they wish to run the deployment/teardown process in, and the GitHub Actions pipeline will manage the rest.

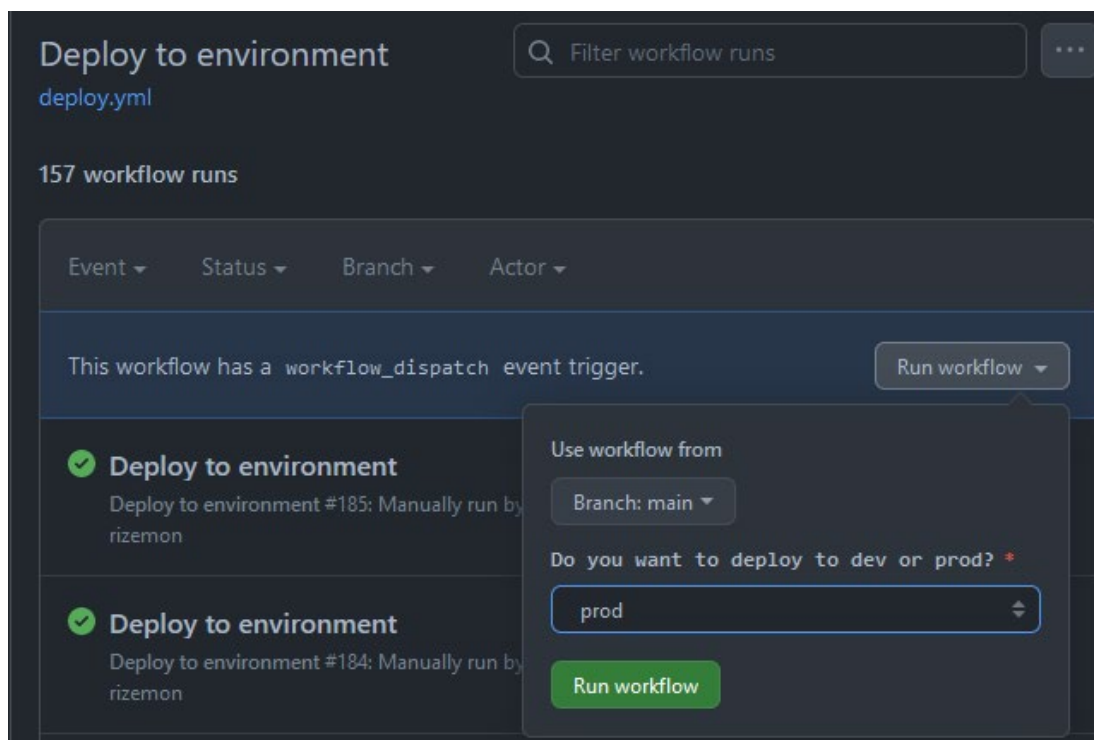


Figure 48: Environment Selection for Deployment

Whenever a teardown is kickstarted or when a deployment is completed, the team is notified via an alert is sent via Telegram, which is an instant messaging platform.

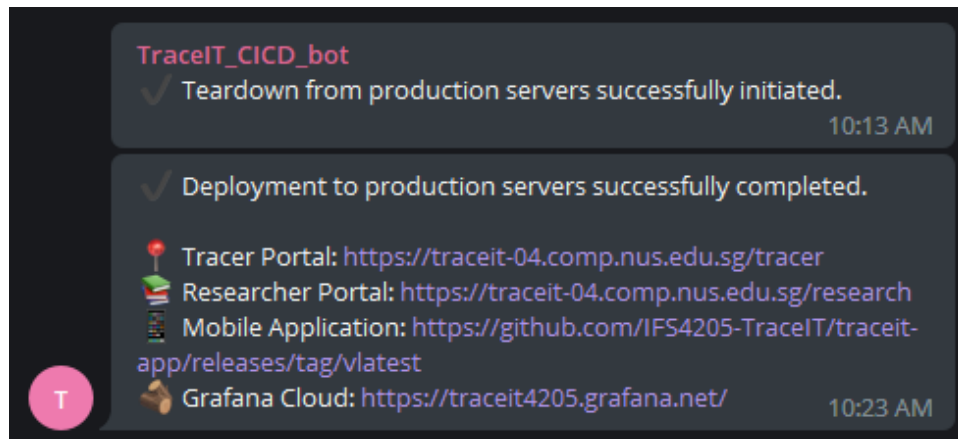


Figure 49: Telegram Notification on Deployment Progress

12.5. Implementation

12.5.1 Continuous Integration Testing

For the relevant code repositories that require CI testing, GitHub Action workflows are programmed specifically for each of them, but they share the following steps:

- Spawning of Docker containers that act as the execution environment or host dependent services such as databases
- Cloning of code repository from GitHub
- Restoring of cached dependencies
- Installing of third-party dependencies and packages
- Executing of unit tests

The GitHub Action workflows are scheduled to execute when new changes are pushed to a branch or when a branch is merged into the the main branch. This ensures that the code is tested regularly, and any errors are detected as soon as possible.

12.5.2 Automated Release of Artifacts

To speed up and simplify the deployment process, GitHub Action workflows were also designed to build the source code and release the resulting artifacts on GitHub. This includes static web content generated by frontend frameworks and a patched version of PostgreSQL that supports Transparent Data Encryption (TDE). Therefore, during the deployment process, the released artifacts will simply need to be retrieved from GitHub and deployed without going through any compilation of source code.

12.5.3 Regular Scans of Code Repositories

Aside from the workflows used for continuous integration, a separate workflow performs a scan of the relevant code repositories using the mentioned SAST tools. This workflow is executed upon code pushes as well as daily. If a vulnerable third-party dependency is being used or a new vulnerability is discovered, GitHub Dependabot will automatically create a pull request containing changes that will upgrade it to a newer version.

12.5.4 Continuous Deployment to Production Servers

Two GitHub Action workflows were designed specifically for managing the infrastructure:

- Deployment workflow
- Teardown workflow

Both workflows mainly consist of steps that utilise the “Run Ansible playbook” action to remotely execute Ansible playbooks on the provisioned virtual machines. The deployment workflow ensures that configurations are performed in the correct order, whereas the teardown workflow follows the reverse order and ensures that no residual data remain which could affect subsequent deployments.

12.5.5 Security Hardening of Virtual Machines

As part of the deployment workflow, a series of hardening steps were taken on the virtual machines to reduce its attack surface. The steps include:

- Utilising a more hardened SSHd config
- Configuring of firewall rules that restrict the type of connections between machines
- Hardening of the Linux kernel via the use of sysctl to configure kernel parameters such as IPv4 forwarding and use of IPv6

12.5.6 Automated Operations

Two additional GitHub Action workflows were created to automate internal processes in the TracelT Architecture:

- Web Uptime workflow
- Anonymization workflow

The “Web Uptime” workflow check at 30 minutes intervals to see if any of web services are down and attempts to redeploy them, whereas the “Anonymization” workflow performs the anonymization process at midnight daily on the current dataset and pushes the results to the research database.

12.6. Security Considerations

12.6.1 Preventing Hard-coded Secrets in Source Code

By embedding secrets in the source code, they become easy targets for adversaries to discover and exploit. Once they know what the embedded secrets are, they can potentially keep using it to circumvent protections that are in place, even across various deployments of the TracelT architecture. Therefore, developers are required to write their code in a way that retrieve the secrets from the environment variables that are passed via continuous deployment process. To monitor for any secrets committed to the repositories, a workflow has been configured to scan for hard-coded secrets and produce a warning if it finds one.

13. Subsystem 8 - Logging

13.1. Overview

To ensure TracelT’s web services operate as intended, the logging subsystem allows the team and operators to view the status and responses logged by the NGINX reverse proxy and the Django backends. The logs are sent to Grafana Cloud to offload the log storage requirements from the TracelT virtual machines to a third-party provider. These logs would not only allow operators to check if the web services are under regular use but also assist them in investigating attacks launched against the web services by an adversary, such as identifying the source and vector of attack.

13.2. Assumptions

The logging subsystem assumes that the system administrators, who are the users of this subsystem, are in positions where they are properly vetted by the relevant authorities and have signed non-disclosure agreements to ensure that the data seen within the logging platform is kept confidential, similar to the contact tracer portal.

13.3. Tools and Software

Log Management:

- Grafana Cloud
 - Loki
 - Grafana Dashboard

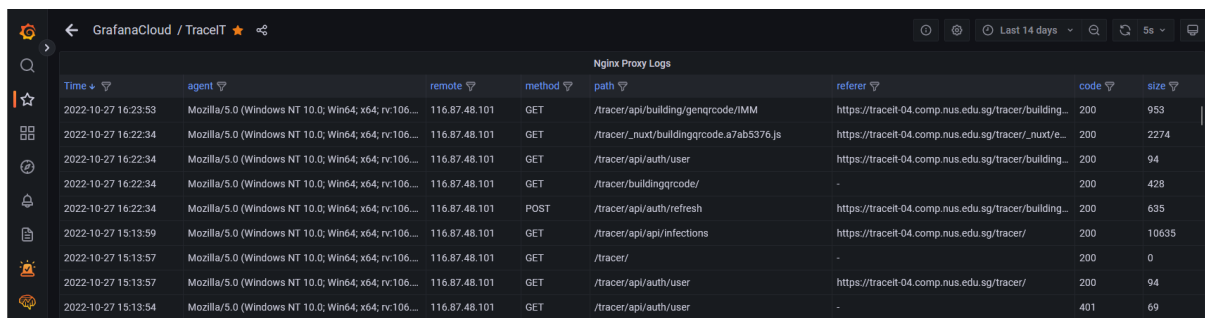
Log Forwarding:

- Fluent Bit

13.4. Implementation

13.4.1 NGINX Reverse Proxy Logs

Fluent Bit is installed on the NGINX reverse proxy server to collect and forward the logs generated by the NGINX web service. When a client sends a request to the TraceIT web service, the request will be routed by the NGINX reverse proxy to the appropriate backend server. These requests will be logged by NGINX and written to a local log file which will then be read by Fluent Bit and forwarded to Loki on Grafana Cloud and then represented in a dashboard.



Time	agent	remote	method	path	referer	code	size
2022-10-27 16:23:53	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106...	116.87.48.101	GET	/tracer/api/building/genqrcode/1MM	https://traceit-04.comp.nus.edu.sg/tracer/building...	200	953
2022-10-27 16:22:34	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106...	116.87.48.101	GET	/tracer/_nuxt/buildingqrcode.a7ab5376.js	https://traceit-04.comp.nus.edu.sg/tracer/_nuxt/e...	200	2274
2022-10-27 16:22:34	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106...	116.87.48.101	GET	/tracer/api/auth/user	https://traceit-04.comp.nus.edu.sg/tracer/building...	200	94
2022-10-27 16:22:34	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106...	116.87.48.101	GET	/tracer/buildingqrcode/	-	200	428
2022-10-27 16:22:34	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106...	116.87.48.101	POST	/tracer/api/auth/refresh	https://traceit-04.comp.nus.edu.sg/tracer/building...	200	635
2022-10-27 15:13:59	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106...	116.87.48.101	GET	/tracer/api/api/infections	https://traceit-04.comp.nus.edu.sg/tracer/	200	10635
2022-10-27 15:13:57	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106...	116.87.48.101	GET	/tracer/	-	200	0
2022-10-27 15:13:57	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106...	116.87.48.101	GET	/tracer/api/auth/user	https://traceit-04.comp.nus.edu.sg/tracer/	200	94
2022-10-27 15:13:54	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106...	116.87.48.101	GET	/tracer/api/auth/user	-	401	69

Figure 50: NGINX Reverse Proxy Logs

The log details collected by NGINX are:

- Time of client request
- User agent
- Client IP address
- Web request method
- URL of the web request
- Referrer of the web request
- HTTP status code
- Size of the returned response

13.4.2 Django Application Logs

The Django backends have been configured to send the generated logs directly to Loki on Grafana Cloud without any intermediary forwarder like Fluent Bit. The logs generated by Django have been customised to focus on the actions carried out by users on the platforms for auditing proper execution of tasks and access of data by the appropriate users.

The log details collected by all three Django backend services are:

- Time of the user action
- Severity of the log (info/warning/error)

- Detail message of the action
- Action of the user
- Web request method and route
- User ID that performed the action
- Building ID accessed by the user (only on tracer backend)
- Exception error message

Table 9: Actions Logged on All Backends

Action	Details
User Authentication	<ul style="list-style-type: none"> • Registration attempt • Registration success • Registration failed • Login attempt • Login success • Login failed • No permission to web portal • Logout
User 2FA TOTP	<ul style="list-style-type: none"> • Registration • Validation • Invalid token • Not yet verified • No permission to web portal
Authentication Exceptions	<ul style="list-style-type: none"> • Exception class name

Time	severity	message	action	request	user_id	exception
2022-10-26 14:23:51	info	User accessed building.	building_access	<rest_framework.request.Request: POST /buildings/register>	f52d18f0-e11b-4461-a86a-4ca0bb252c11	
2022-10-26 14:21:25	info	User accessed building.	building_access	<rest_framework.request.Request: POST /buildings/register>	f52d18f0-e11b-4461-a86a-4ca0bb252c11	
2022-10-26 14:21:13	info	User has recent infection history.	building_access	<rest_framework.request.Request: POST /buildings/register>	f55b6c5d-4011-45d2-8f8d-23934cfe892	
2022-10-26 14:09:12	info	User validated TOTP.	validate_totp	<rest_framework.request.Request: POST /auth/totp>	f52d18f0-e11b-4461-a86a-4ca0bb252c11	
2022-10-26 14:09:12	info	User validating TOTP.	validate_totp	<rest_framework.request.Request: POST /auth/totp>	f52d18f0-e11b-4461-a86a-4ca0bb252c11	
2022-10-26 14:08:29	info	User registered TOTP.	register_totp	<rest_framework.request.Request: POST /auth/totp/register>	f52d18f0-e11b-4461-a86a-4ca0bb252c11	
2022-10-26 14:08:29	info	User TOTP registration request.	register_totp	<rest_framework.request.Request: POST /auth/totp/register>	f52d18f0-e11b-4461-a86a-4ca0bb252c11	
2022-10-26 14:08:28	info	User logged in.	login	<rest_framework.request.Request: POST /auth/login>	f52d18f0-e11b-4461-a86a-4ca0bb252c11	
2022-10-26 14:08:28	info	User login request.	login	<rest_framework.request.Request: POST /auth/login>		
2022-10-26 14:08:28	info	Token is invalid.	check_token	<rest_framework.request.Request: POST /auth/login>		
2022-10-26 14:08:27	info	User registered.	register	<rest_framework.request.Request: POST /auth/register>	f52d18f0-e11b-4461-a86a-4ca0bb252c11	
2022-10-26 14:08:27	info	User registration request.	register	<rest_framework.request.Request: POST /auth/register>		
2022-10-26 14:08:27	info	Token is invalid.	check_token	<rest_framework.request.Request: POST /auth/register>		
2022-10-26 14:06:17	warning	Exception: InvalidToken	exception	<rest_framework.request.Request: POST /auth/refresh>		InvalidToken

Figure 51: Contact Backend Logs

Table 10: Actions Logged on Contact Backend

Action	Details
Building Access	<ul style="list-style-type: none"> User accessed a building (user ID is not logged) Building does not exist

Time	severity	message	action	request	user_id	building_id	exception
2022-10-27 16:23:53	info	Generated Building QR code.	generate_building_qr_code	<rest_framework.request.Request: GET /building/genqrcode/IMM>		d213d379-b61e-497f-93...	
2022-10-27 15:13:57	info	List infection request.	list_infection	<rest_framework.request.Request: GET /api/infections>	de3e71c7-f291-4611-a942-fa495e2ff43f		
2022-10-27 15:13:54	warning	Exception: NotAuthenticated	exception	<rest_framework.request.Request: GET /auth/user>			NotAuthenticated
2022-10-27 15:13:54	info	Token is invalid.	check_token	<rest_framework.request.Request: GET /auth/user>			
2022-10-27 15:13:48	info	List infection request.	list_infection	<rest_framework.request.Request: GET /api/infections>	de3e71c7-f291-4611-a942-fa495e2ff43f		
2022-10-27 15:13:47	info	User validated TOTP.	validate_totp	<rest_framework.request.Request: POST /auth/totp>	de3e71c7-f291-4611-a942-fa495e2ff43f		
2022-10-27 15:13:47	info	User validating TOTP.	validate_totp	<rest_framework.request.Request: POST /auth/totp>	de3e71c7-f291-4611-a942-fa495e2ff43f		
2022-10-27 15:13:28	info	User registered TOTP.	register_totp	<rest_framework.request.Request: POST /auth/totp/register>	de3e71c7-f291-4611-a942-fa495e2ff43f		

Figure 52: Tracer Backend Logs

Table 11: Actions logged on tracer backend

Action	Details
Building Access	<ul style="list-style-type: none"> List buildings List user access into buildings
Infection	<ul style="list-style-type: none"> List infected users List close contacts Update upload status

Time	severity	message	action	request	user_id	exception
2022-10-27 12:40:07	info	User validated TOTP.	validate_totp	<rest_framework.request.Request: POST /auth/totp>	1fa9198c-0af6-46ca-ab89-708cbdd0a1ae	
2022-10-27 12:40:07	info	User validating TOTP.	validate_totp	<rest_framework.request.Request: POST /auth/totp>	1fa9198c-0af6-46ca-ab89-708cbdd0a1ae	
2022-10-27 12:39:36	info	User logged in.	login	<rest_framework.request.Request: POST /auth/login>	1fa9198c-0af6-46ca-ab89-708cbdd0a1ae	
2022-10-27 12:39:35	info	User login request.	login	<rest_framework.request.Request: POST /auth/login>	1fa9198c-0af6-46ca-ab89-708cbdd0a1ae	
2022-10-27 11:48:45	warning	Exception: InvalidToken	exception	<rest_framework.request.Request: GET /auth/user>		InvalidToken
2022-10-26 15:11:20	warning	Exception: InvalidToken	exception	<rest_framework.request.Request: POST /auth/refresh>		InvalidToken
2022-10-26 14:30:09	info	User validated TOTP.	validate_totp	<rest_framework.request.Request: POST /auth/totp>	1fa9198c-0af6-46ca-ab89-708cbdd0a1ae	
2022-10-26 14:30:09	info	User validating TOTP.	validate_totp	<rest_framework.request.Request: POST /auth/totp>	1fa9198c-0af6-46ca-ab89-708cbdd0a1ae	
2022-10-26 14:29:35	info	User registered TOTP.	register_totp	<rest_framework.request.Request: POST /auth/totp/register>	1fa9198c-0af6-46ca-ab89-708cbdd0a1ae	
2022-10-26 14:29:35	info	User TOTP registration request.	register_totp	<rest_framework.request.Request: POST /auth/totp/register>	1fa9198c-0af6-46ca-ab89-708cbdd0a1ae	

Figure 53: Research Backend Logs

13.5. Security Considerations

13.5.1 Offsite Access to Logs

To ensure that the team is provided with an overview during an incident, the logs subsystem does not store logs within the local TracelT machines. Instead, the logs are ingested into Loki on Grafana Cloud which allows the team to view the logs of the various subsystems even when the local systems are down. The advantage of this implementation is that even if the adversary could compromise the local

servers, they will not be able to erase the action log they have executed. Thus, this provides the team with an opportunity to respond and fix the vulnerability. The downside of the approach is relying on a third-party provider for the log aggregation service, which requires trust in the vendor to safely store and secure the logs on their servers.

13.5.2 Privacy of User Data

Logging the user's actions may provide an avenue for the administrator of the logs to piece information about specific users, breaching the confidentiality of the users. The logging subsystem aims to reduce as much exposure of user data as possible by only recording what is potentially needed to provide an understanding of an incident. An example would be to omit logging the building ID when a user enters a valid building but logging it when an invalid building ID is submitted. This allows the team to understand the potential cause of issues while ensuring users' locations are not exposed. However, there may be instances where sensitive data is necessary for security operations, such as when a user attempts to upload their close contact data which may reveal the infection status of that user. Therefore, it is assumed that the log administrators have signed a non-disclosure agreement and vetted as trustworthy to minimise the risk of privacy loss.

14. Subsystem 9 - Secrets Management and Public Key Infrastructure

14.1. Overview

As the scale of the applications and subsystems grow, there is a need to manage cryptographic secrets such as private keys and certificates. As such, the secrets management and public key infrastructure (PKI) subsystem aims to standardize and consolidate the secrets within the application stacks found in TraceIT to ensure high confidentiality, integrity, and availability within all other subsystems. The components of the subsystems consist of the following:

- Long-term secret key management
- Public Key Infrastructure
- Dynamically generated database credentials

14.2. Tools and Software

The bulk of the subsystem involves the following software:

- Vault by Hashicorp
- GitHub Action Secrets

14.3. Implementation

14.3.1 Storage of Secrets based on Criticality

GitHub Action Secrets is used to store long-term secrets that are critical to the operations of the infrastructure and are only accessible by the self-hosted GitHub Action runners. These secrets include the following:

- SSH private key (Used by Ansible to remotely login and orchestrate the virtual machines)
- Telegram token and chat ID (Used to alert the team when a deployment or teardown has been initiated and completed)
- GitHub Personal Access Token (Used to access and clone the private repositories)
- Vault Token (Used to read and write secrets to Vault)
- API key for Grafana Cloud API (Used to forward logs to Grafana Cloud)

Vault is used to store short-term secrets that varies from deployment to deployment and therefore, they are not as critical as the one mentioned above. These secrets include the following:

- Django Secret Keys (Used to secure signed data such as JWT tokens)
- Transparent Data Encryption (TDE) keys (Used to encrypt the contents of the databases)
- Database Root credentials (Used to configure the databases)

14.3.2 Public Key Infrastructure

By configuring Vault to act as a certificate authority via its PKI secret engine, certificates can be issued for the various TraceIT services, such as web servers and databases. Before initiating the first deployment workflow, all the virtual machines would have been configured to add the certificate authority's certificate bundle to their respective trust stores and implicitly trust it. During each deployment workflow, the keypair for each service will be re-generated and used for their respective purposes, such as securing their communications with other services.

14.3.3 Mutual Transport-Level Security (mTLS) between services

The services have been configured to securely communicate with each other only via TLS connections as well as configured to require each other to present a valid certificate signed by the certificate authority. These services include:

- Unicorn
- NGINX
- PostgreSQL
- Vault
- Self-hosted Github Action runners

14.3.4 Dynamically Generated Database Credentials

During the deployment workflow, Vault's database secret engine generates a new set of database credentials for each service that needs it. This is done by utilising the database root credentials to create new accounts on the fly that have been assigned the required privileges.

14.4. Security Considerations

14.4.1 Principle of Least Privilege

As Vault is used extensively to manage secrets within the infrastructure, it is important that application can only view or edit the secrets they are assigned to. Therefore, roles are created within Vault to ensure that only specific applications can interact with the secrets they have been assigned to. For example, the Django backends will only be able to view secrets relating to TOTP and contact tracing secret key but will not be able to view the credentials of the databases directly from Vault.

14.4.2 Passing Secrets Securely

To ensure that no secrets are viewable directly from any source code or configuration file, all secret are only provided to the applications within the TraceIT infrastructure only when they are deployed in production. This prevents developers from potentially leaking secrets within their source code and allowing potential adversaries to utilize the secrets to decrypt information or impersonate the services or users within the infrastructure.

14.4.3 Providing Encryption in Transit

To ensure defense-in-depth within the infrastructure of TraceIT, data generated by the services are encrypted by mutual TLS to ensure that adversaries are unable to decrypt the traffic within the network. In addition, applications are required to have a valid client certificate generated by the

internal certificate authority to ensure that the identities of both parties in the TLS connection are validated.

15. References

Bay, J., Kek, J., Tan, A., Sheng Hau, C., Yongquan, L., Tan, J., & Anh Quy, T. (2020). *BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders*. Singapore: Government Technology Agency.