

RMIT University

Cloud Computing

COSC2626

Assignment 3

Web App url: <https://tracex.link/>

GitHub: <https://github.com/harindukodi/TraceX>

S3763840 - Harindu Kodituwakku
S3767707 - Anson Go Guang Ping

13/06/2021

Table of Contents

Table of Contents	2
Contribution Agreement	3
1. Summary	4
2. Introduction	5
3. Related Work	6
COVIDSafe app	6
NHS COVID-19	7
4. System Architecture	8
1. Amazon Elastic Beanstalk (EB)	9
2. Amazon Elastic Compute Cloud (EC2)	9
3. AWS Lambda	9
4. Amazon Simple Storage Service (S3)	9
5. Amazon Relational Database Service (RDS)	10
6. Amazon API Gateway	10
7. Amazon Simple Email Service (SES)	10
8. Amazon Quicksight	11
9. Amazon Route 53	11
10. Amazon CloudFront	11
10. AWS Certificate Manager	12
Description of the data structure	12
5. Developer Manual	14
6. User Manual	26
User Functions	26
Administrator Functions	29
References	32

Contribution Agreement

Student Name	Anson Go Guang Ping	Harindu Kodituwakku
Student ID	S3767707	s3763840
Contributions	<ol style="list-style-type: none">1. Project idea formulation2. Frontend development3. Writing the report	<ol style="list-style-type: none">1. Frontend development2. Backend development3. Development of AWS services4. Development of UI5. Writing the report
Contribution Percentage	40%	60%
By signing below, I certify all information is true and correct to the best of my knowledge.		
Signature	Anson	HarinduKodi
Date	13/06/2021	13/06/2021

1. Summary

The COVID-19 pandemic has surprised the world with its highly infectious abilities and high death rates of its patients. A non-negligible fraction of the infected individuals exhibit hidden symptoms thus can be a hidden source of transmissions at any given time, forcing drastic changes to daily life, from the implementation of stay-at-home orders to mandating facial coverings and limiting in-person gatherings. Although these practices can contain the spread of the virus, it comes with severe social and economic consequences. To automate the difficult task of tracing all recent contacts of newly identified infected individuals One promising yet controversial solution to this dilemma is the implementation of contact-tracing apps [1].

In response, the proposed project is to develop a contact tracing app which quickly identifies and informs users who may have encountered an infected person on Victorian public transports (Trains, Trams, Busses). Aim of this project is to use cloud platform services to develop an application that was highly scalable and had a distributed architecture. During the development of the project, it is planned to use a myriad of services which increase user engagement and provide data analysis. The overall objective of this report is to present the idea and structures that will support the development and deployment of digital contact tracing within an established program.

TraceX would be an essential tool for contact tracing conducted by the health authorities and governments.



<https://tracex.link/>

2. Introduction

Cloud computing increases sustainability and flexibility for applications by allowing delivery of services over the internet. Our project was created by developing and deploying our responsive web application over the cloud infrastructure, including storing data and utilizing cloud services. In the development of the application, utilizing the use of Amazon Web Service which enabled the access to a variety of services to incorporate into the application. As a result, AWS services were useful to create an infrastructure for the application and integrated them throughout the project to add functionality.

Our inspiration comes from the current global pandemic outbreak and the resulting needs of users to adapt to the situation. The COVID-19 pandemic has surprised the world with its highly infectious abilities and high death rates of its patients. A non-negligible fraction of the infected individuals exhibit hidden symptoms thus can be a hidden source of transmissions at any given time, forcing drastic changes to daily life, from the implementation of stay-at-home orders to mandating facial coverings and limiting in-person gatherings. To automate the difficult task of tracing all recent contacts of newly identified infected individuals One promising yet controversial solution to this dilemma is the implementation of contact-tracing apps [1]. In response, we decided to develop a contact tracing app which quickly identifies and informs users who may have encountered an infected person on public transports in Victoria (Trains, Trams, Busses).

We created '**TraceX**' to allow tracing of users' trips with public transport, maximising the efficiency of contact tracing. Users can sign up to the website and register their email address, then pursue the check in with their current or previous route by entering their public transport records (Tram, train, bus) with its date and time. Records are stored in a database, allowing us as developers to analyse and retrieve user data. Admins can view these records formulated by the application and if there is a covid patient, admins can notify users who encountered the patient through using the same public transport. Alerts would then be shown in users' dashboard and sent to emails immediately.

This app is required by users as it alerts users' contacts to the possibility of infection, so early countermeasures can be performed to interrupt ongoing transmission and reduce the spread of

an infection. It improved workflow for staff and patients, and during an outbreak, gathering that information can be as simple as logging into your application and downloading the reports you need. TraceX is extremely useful as analysis and communication are more simple and potential exposure lists can be provided to healthcare and government organizations, so that monitoring and treatment are properly targeted.

3. Related Work

The proposed project is similar to other applications which provide similar functionalities of contact tracing.

COVIDSafe app

The COVIDSafe app is part of the Department of Health's work to slow the spread of COVID-19. The COVIDSafe app helps state and territory health officials to quickly identify and contact people who may have been exposed to COVID-19 (called 'close contacts').



Users who download the app will receive a confirmation text message to complete the installation. COVIDSafe generates an encrypted reference code based on the information on that phone. COVIDSafe uses Bluetooth® to look for other devices that have the app installed and performs a digital handshake when a contact occurs. It securely logs the other user's encrypted reference code and the date, time, Bluetooth® signal strength and proximity of the contact on the user's phone and notes the phone model.

If you are tested positive for COVID-19, a state or territory health official will collect your consent to upload your digital handshake information to the National COVIDSafe Data Store. The uploaded information enables state and territory health officials to call close contacts to advise them on what to do. [2, para. 3-17]

NHS COVID-19

The NHS COVID-19 app is the official contact tracing app for England and Wales. The app runs on proven software developed by Apple and Google, designed so that nobody will know who or where you are. And you can delete your data, or the app, at any time. It has several features [3, para. 7-9]:

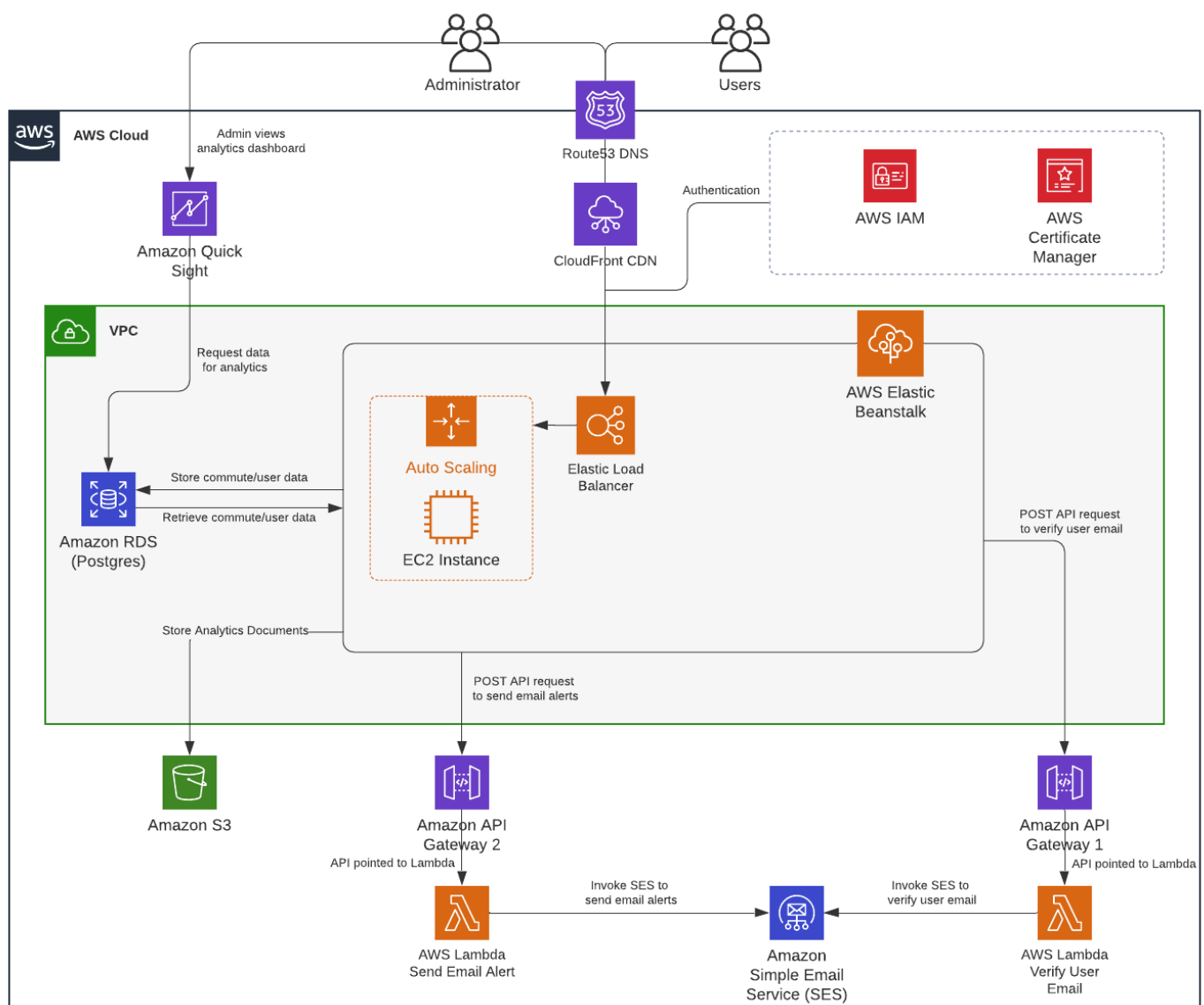


- Trace: find out when you have been near other app users who have tested positive for coronavirus.
- Alert: lets you know the level of coronavirus risk in your postcode district.
- Check-in: get alerted if you have visited a venue where you may have come into contact with coronavirus, using a simple QR code scanner. No more form filling.
- Symptoms: check if you have coronavirus symptoms and see if you need to order a test.
- Test: helps you order a test if you need to.
- Isolate: keep track of your self-isolation countdown and access relevant advice.

4. System Architecture

This section describes the high-level system design and the architecture of the TraceX application. TraceX is an Amazon Web Services based web application which is partly deployed and uses serverless architecture. TraceX is designed to handle high throughput of requests if required. Since the backend of TraceX is written in Python using the Django framework, the official AWS SDK for Python which is 'Boto3' is being used. The below software architecture diagram represents the overall design of the system.

Software Architecture Diagram



System Components

1. Amazon Elastic Beanstalk (EB)

Amazon EB is able to deploy, manage and scale web applications by employing EC2 Auto Scaling and Elastic Load Balancing features. Using EB, developers can deploy their applications without having to worry about the infrastructure running the application as it provides tools to automate background tasks and monitor the health of deployed applications [4]. This project TraceX uses Elastic Beanstalk to deploy the Python based web application to the AWS infrastructure, interacting and deploying through the EB Command Line Interface.

2. Amazon Elastic Compute Cloud (EC2)

Initially, the system was tested by directly deploying the application to the EC2 instance rather than using EB. Yet, in the development phase, it was realized that using AEB it would be useful to get features such as auto scaling and elastic load balancing. In the final version of the application, Amazon EC2 instances are a part of the AWS resources created by Elastic Beanstalk to support its architecture and application hosting.

3. AWS Lambda

AWS Lambda is a serverless compute service that enables applications to run code without provisioning or managing servers. Using AWS Lambda service, TraceX was able to make several services of the system serverless which makes the system more secure and easy to maintain. Two of our functions which are to verify the user email using Amazon Simple Email Service (SES) and to send emails of COVID-19 alerts to users were implemented using Lambda functions. Both Lambda functions were written in Python and connected to API Gateways as the triggers for the Lambda services. Necessary permission was provided to the Lambda execution roles making the security of the system is guaranteed within the AWS internal services.

4. Amazon Simple Storage Service (S3)

Amazon Simple Storage Service is a highly scalable, durable storage that makes it simple and practical to collect, store, and analyze data regardless of format and type at massive scale. In the development of the TraceX application, S3 was used to upload the zipped version of the

application which then could be used to deploy to the AWS Beanstalk environment. Furthermore, the administrator can upload dashboard analytics documents to the S3 using the file upload feature implemented in the system.

5. Amazon Relational Database Service (RDS)

The TraceX application uses a relational database to store the commute details, user tracking information, user subscriptions details. The decision for implementing a relational database for the TraceX application was taken after thoroughly analyzing the data that would be generated and used in the real life related to the application. Since most of the information would be related and not a massive amount of data such as Big data, the necessity was to implement a relational database rather than a No-sql type database such as AWS DynamoDB. A Postgres engine was used with the AWS RDS instance which was created using the AWS EB and connected with the Elastic Beanstalk environment to make the database instance elastically scale with the requirements.

6. Amazon API Gateway

The TraceX application has 2 RESTful APIs which were developed using the Amazon API Gateway. Amazon API Gateway is a fully managed service that makes it easier to create, publish, maintain, monitor, and secure APIs at any scale. Both API Gateway RESTful APIs were integrated with Lambda functions. In TraceX, both API Gateways are used as Triggers for the serverless lambda functions enabling thousands of potential concurrent API calls, traffic management and API version management seamless.

7. Amazon Simple Email Service (SES)

Amazon SES service was used in the TraceX application to send critical alerts for the users in the system. Since the TraceX application is a time sensitive application as it is related to the COVID-19 based information provider, the system is required to provide instant alerts for the user who might be exposed to COVID-19 traces while using public transportation. Amazon Simple Email Service (SES) is a cost-effective, flexible, and scalable email service that enables TraceX to send mail from within the application. SES is sandboxed with the TraceX application meaning the users are required to provide their email address when they register a new account.

When a user registers with their email, the system automatically sends a verification email to the user. When the user verifies the email, the system is capable of sending real-time alerts for the users.

8. Amazon Quicksight

Amazon Quicksight was used in the TraceX application to provide a quick overview of the COVID-19 affected public commutes and the users in a dashboard-based visualization. This feature enabled the administrators of the TraceX to identify potentially significant commutes, the number of potential COVID-19 infected people. The Amazon Quicksight dashboard was directly linked with the AWS RDS in the application which was able to receive data realtime and monitor them in a serverless architecture. Amazon QuickSight is a scalable, serverless, embeddable business intelligence service built for the cloud.

9. Amazon Route 53

Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. Amazon Route 53 effectively connects user requests to infrastructure running in AWS, such as Amazon EC2 instances, Elastic Load Balancing load balancers, or Amazon S3 buckets and can also be used to route users to infrastructure outside of AWS [8]. Amazon Route 53 was used in TraceX to register the domain name (<https://tracex.link/>) using a A record which is required to use HTTPS protocol for the web application deployed in the Elastic Beanstalk.

10. Amazon CloudFront

Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data. In the TraceX, the Elastic load balancers of the Beanstalk deployed application were integrated with the Amazon CloudFront. Then the CNAME records of Route 53 were also submitted to the CloudFront distribution. This enabled TraceX to serve customers globally with low latency, high transfer speeds.

10. AWS Certificate Manager

AWS Certificate Manager allows the provision, management, and deployment of public and private Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services and the internal connected resources. ACM was used in TraceX to obtain the SSL certification for the web application.

Description of the data structure

Throughout the TraceX application, a relational database which is a Postgres using the AWS RDS was used. All the database tables were created using the Django models feature in the Django Framework. The following is the schema of the tables used in the application.

user_tracking_table

Contains information on user subscribed commutes.

```
5 class user_tracking_table(models.Model):
6     user_email = models.CharField(max_length=100)
7     commute_type = models.CharField(max_length=100)
8     commute_name = models.CharField(max_length=100)
9     commute_year = models.CharField(max_length=20)
10    commute_month = models.CharField(max_length=60)
11    commute_day = models.CharField(max_length=20)
12    commute_hour = models.CharField(max_length=20)
13    commute_minutes = models.CharField(max_length=20)
14    commute_ampm = models.CharField(max_length=20)
15    commute_alert = models.CharField(max_length=50)
16
17    class Meta:
18        db_table = "user_tracking_table"
```

user_info_table

Contains information on user registered users.

```
5 class user_info_table(models.Model):
6     user_email = models.CharField(max_length=100)
7     user_password = models.CharField(max_length=100)
8     user_access_level = models.CharField(max_length=100)
9
10    class Meta:
11        db_table = "user_info_table"
```

commute_table

Contains information on the public transports added by the administrator.

```
5 class commute_table(models.Model):
6     commute_type = models.CharField(max_length=60)
7     commute_name = models.CharField(max_length=100)
8     commute_year = models.CharField(max_length=20)
9     commute_month = models.CharField(max_length=60)
10    commute_day = models.CharField(max_length=20)
11    commute_hour = models.CharField(max_length=20)
12    commute_minutes = models.CharField(max_length=20)
13    commute_ampm = models.CharField(max_length=20)
14    commute_alert = models.CharField(max_length=50)
15    commute_passenger_count = models.CharField(max_length=30)
16
17    class Meta:
18        db_table = "commute_table"
```

1. Programming Languages:

1. Backend framework - Django Web Framework using Python.
2. Frontend framework - AngularJS
3. Frontend tools and other libraries - CSS, Bootstrap, jQuery

2. Database - Postgres

3. AWS SDK - Boto3

5. Developer Manual

1. Amazon Elastic Beanstalk (EB)

The Elastic Beanstalk was developed referring to the following documentation provided by the AWS.

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-django.html>

1.1 Initially the Django project was set up locally and created a virtual environment.

1.2 Configuring the Django application with Elastic Beanstalk.

To configure the Django application with the Elastic Beanstalk, a directory named '.ebextensions' was created and a file named 'django.config' was created. The followings are the content of the django.config

```
1 option_settings:
2   aws:elasticbeanstalk:container:python:
3     WSGIPath: traceXapp.wsgi:application
4   aws:elasticbeanstalk:environment:proxy:staticfiles:
5     /static: staticfiles
6     /images: staticfiles
```

1.3 Deploy the web application using the EB CLI.

Using the EB CLI a Beanstalk environment was created to deploy the application.

```
(venv) C:\Users\datam\Documents\Harindu\Other\RMIT\Cloud_Computing\Assignment III\code\TraceX>eb create TraceXNewDockerEnvCC2
Creating application version archive "app-43f1-210607_155058".
Uploading TraceX/app-43f1-210607_155058.zip to S3. This may take a while.
Upload Complete.
Environment details for: TraceXNewDockerEnvCC2
  Application name: TraceX
  Region: us-east-1
  Deployed Version: app-43f1-210607_155058
  Environment ID: e-dcub6muycm
  Platform: arn:aws:elasticbeanstalk:us-east-1::platform/Python 3.7 running on 64bit Amazon Linux 2/3.3.0
  Tier: WebServer-Standard-1.0
  CNAME: UNKNOWN
  Updated: 2021-06-07 05:51:16.477000+00:00
Printing Status:
2021-06-07 05:51:15 INFO createEnvironment is starting.
2021-06-07 05:51:16 INFO Using elasticbeanstalk-us-east-1-681989621175 as Amazon S3 storage bucket for environment data.
2021-06-07 05:51:37 INFO Created security group named: sg-0dcf3bdfd5a68bfd1
2021-06-07 05:51:53 INFO Created load balancer named: awseb-e-d-AWSEBLoa-1QNJ4BRW0BP8I
```

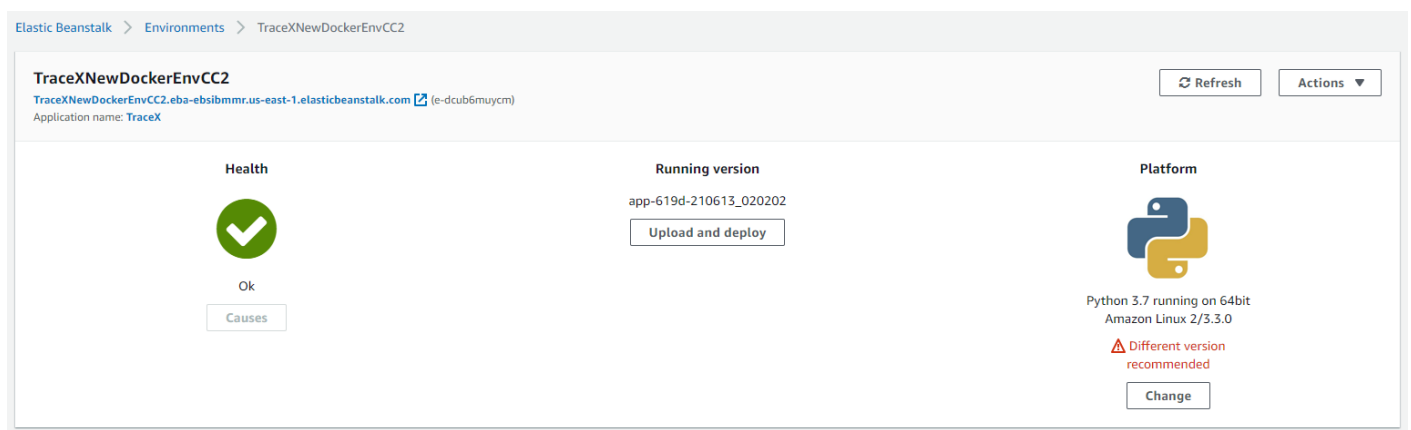
Once the environment was deployed. The application was deployed using the 'eb deploy' command.

```
(venv) C:\Users\datam\Documents\Harindu\Other\RMIT\Cloud_Computing\Assignment III\code\TraceX>eb deploy
Creating application version archive "app-0ac9-210607_165326".
Uploading TraceX/app-0ac9-210607_165326.zip to S3. This may take a while.
Upload Complete.
2021-06-07 06:53:41    INFO    Environment update is starting.
2021-06-07 06:53:46    INFO    Deploying new version to instance(s).
2021-06-07 06:53:50    INFO    Instance deployment successfully generated a 'Procfile'.
2021-06-07 06:53:59    INFO    Instance deployment completed successfully.
2021-06-07 06:54:04    INFO    New application version was deployed to running EC2 instances.
2021-06-07 06:54:04    INFO    Environment update completed successfully.
```

1.4 Update the Django Project

Run the Django migrations and re-deploy the application.

Deployed Beanstalk Environment



Elastic Beanstalk > Environments > TraceXNewDockerEnvCC2

TraceXNewDockerEnvCC2
TraceXNewDockerEnvCC2.eba-eb5lmmr.us-east-1.elasticbeanstalk.com (e-dcub6muycm)
Application name: TraceX

Refresh Actions

Health
Ok
Causes

Running version
app-619d-210613_020202
Upload and deploy

Platform
Python 3.7 running on 64bit Amazon Linux 2/3.3.0
Different version recommended
Change

2. Amazon Relational Database Service (RDS)

The RDS was created and integrated referring to the following documentations.

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-rds.html>

Since the application was deployed using the AWS Beanstalk, the RDS instance needed to be configured and integrated with the Beanstalk environment. This was created by adding a Database instance from the Beanstalk environment Configurations. A db.t2.micro instance

with PostgreSQL engine was used in the TraceX. Since the RDS was created through the Beanstalk environment, a VPC security group was added with the RDS instance.

Database	Endpoint: aa1ft0t22fh6m6e.cbolvjxtjubd.us-east-1.rds.amazonaws.com:5432 Availability: Low (one AZ) Engine: postgres Instance class: db.t2.micro Retention: Create snapshot Storage: 10 Username: harindu
----------	--

Since the same RDS instance was used for the local development, the following Inbound rules were configured.

Inbound rules			
Inbound rules (3)			
Type	Protocol	Port range	Source
PostgreSQL	TCP	5432	0.0.0.0/0
PostgreSQL	TCP	5432	::/0
PostgreSQL	TCP	5432	sg-0a2458739cadea61f / awseb-e-dcub6muycm-stack-AWSEBSecurityGroup-157BY26192MYT

The following configurations were done in the Django project to connect with the RDS instance.

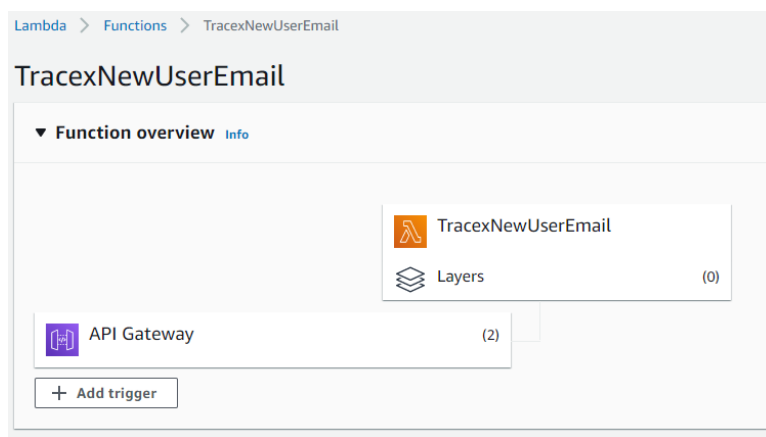
```
96 DATABASES = {
97     'default': {
98         'ENGINE': 'django.db.backends.postgresql',
99         'NAME': 'postgres',
100         'USER': 'harindu',
101         'PASSWORD': 'XXXXXXXXXXXXXXXXXXXX',
102         'HOST': 'aa1ft0t22fh6m6e.cbolvjxtjubd.us-east-1.rds.amazonaws.com',
103         'PORT': '5432',
104     }
105 }
```


3. AWS Lambda

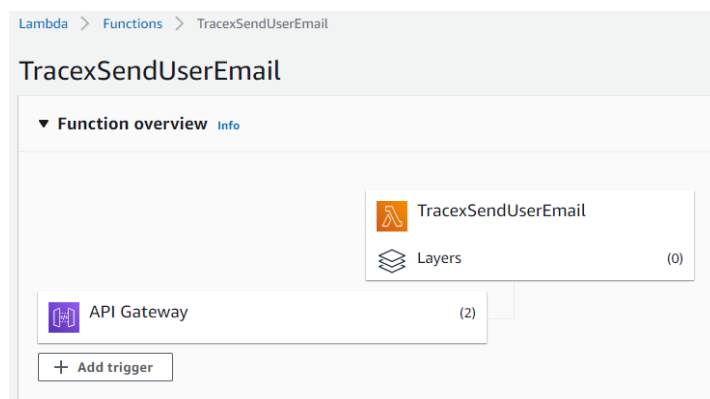
There are 2 lambda functions that are being used in the TraceX application. Development was conducted referring to the following documentation.

<https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html>

1. Created a Lambda function named 'TraceXNewUserEmail'. This Lambda is triggered using an API Gateway, which then would send the email of a new user that's being registered with the system. Then the Lambda function would use the Amazon Simple Email Service to send a verification email to the user. Which then will be used to send alerts. The Lambda function was written in Python.



2. The second Lambda function is named 'TracexSendUserEmail'. This Lambda is triggered using an API Gateway when an Administrator of the TraceX changes the Alert status of a commute in the system. Emails of all the users who are subscribed to that particular commute then would be sent to the Lambda function and using SES, alert emails would be sent to the users. The Lambda function was written in Python.



4. Amazon API Gateway

Amazon API gateway was used as the triggers for both Lambda functions. Two API gateways for each lambda were created. Both API types were REST API using POST methods.

Lambda > Add trigger

Add trigger

Trigger configuration

API Gateway
api application-services aws serverless

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. [Learn more](#)

API
Create a new API or attach an existing one.

Create an API

API type

☐ HTTP API
Create an HTTP API.

☒ REST API
Create a REST API.

Security
Configure the security mechanism for your API endpoint.

IAM

Use IAM permissions to authorize access to your API. Users are required to include authentication signatures in their HTTP calls.

► Additional settings

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel Add

Following is the API Gateway named *‘TracexSendUserEmail-API’* created for the TracexSendUserEmail lambda function.

API Gateway: TracexSendUserEmail-API
arn:aws:execute-api:ap-southeast-1:681989621175:pnoe7c6ci9/*/*POST/TracexSendUserEmail

▼ Details

API endpoint: <https://pnoe7c6ci9.execute-api.ap-southeast-1.amazonaws.com/default/TracexSendUserEmail>

API type: **REST**

Authorization: **NONE**

Method: **POST**

Resource path: **/TracexSendUserEmail**

Stage: **default**

Amazon API Gateway | APIs > TracexSendUserEmail-API (pnoe7c6ci9) > Stages > default

APIs

Custom Domain Names

VPC Links

Stages

default

Create

default Stage Editor

Invoke URL: <https://pnoe7c6ci9.execute-api.ap-southeast-1.amazonaws.com/default>

Following shows how the *TracexNewUserEmail-API* was used in the Django application.

```
url = "https://59pq4rfmoi.execute-api.ap-southeast-1.amazonaws.com/default/TracexNewUserEmail" \
      "?user_email=" + str(reg_user_id)

auth = AWS4Auth('AKIAZ5SN7YW33BSZ2MML', 'SixER2DjIxvC00N7S47FKkHX6XCzH9940zuapSYI', 'ap-southeast-1',
                'execute-api')

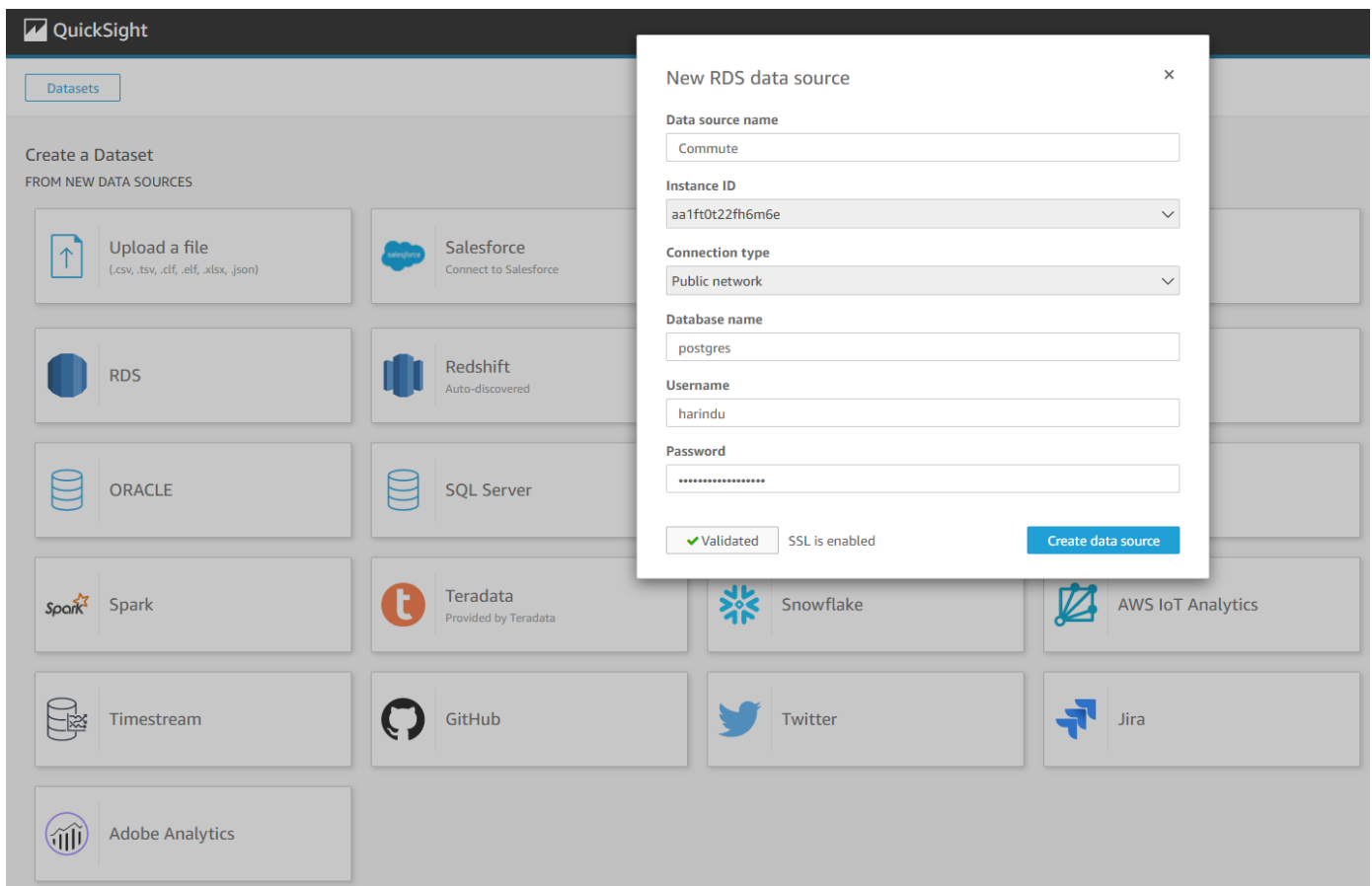
response = requests.post(url, auth=auth)
print(response)
```

5. Amazon Quicksight

The following documentation was used to integrate Amazon Quicksight with the TraceX application.

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/quicksight.html>

Since the TraceX data is stored in the RDS instance, Quicksight has the option of connecting a RDS as a data source.



Since the Quicksight dashboard needs to be integrated with the TraceX Admin Dashboard, application URLs need to be whitelisted through the Domains and Embedding section of the Quicksight. An Enterprise edition version of Quicksight was used since the dashboard needed to be embedded to an external website.

Manage domains for embedded dashboards

Embedded dashboards only work if they originate from domains you explicitly allow.

Domain

Enter a domain name (https://example.com or https://www.example.com or https://example.com:8779)

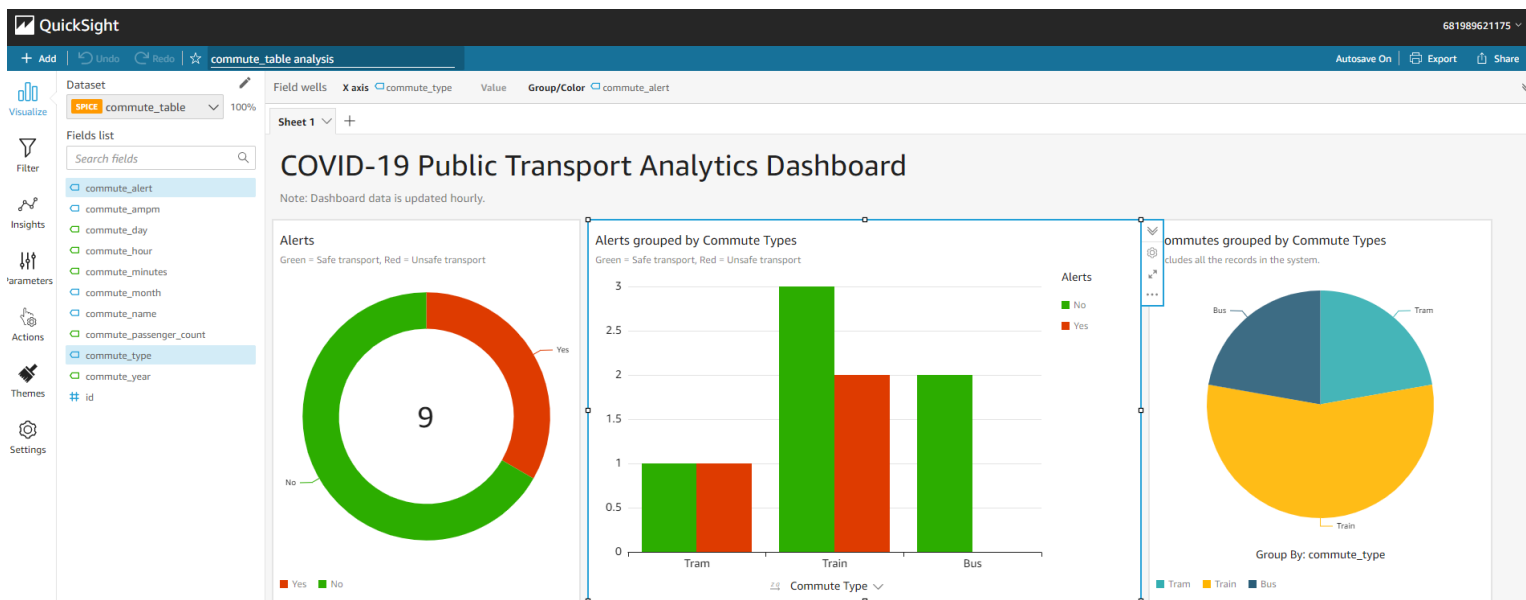
☒ Required

☐ Include subdomains ⓘ

Add

Domain	Include subdomains
https://d758cqe2bs24d.cloudfront.net	No
https://d142airwlb5xio.cloudfront.net	No
https://tracex.link	No
https://www.tracex.link	No

The dashboard was developed as follows.



After the Quicksight dashboard was published, an embedding URL was retrieved using the following code written in the Django application.

```
# session = boto3.session.Session()
quicksight_client = session.client('quicksight', region_name='us-east-1')
# quicksight_client = session.resource('quicksight')
# client = session.create_client("quicksight", region_name='us-east-1')
response = quicksight_client.get_dashboard_embed_url(AwsAccountId="681989621175",
                                                    DashboardId="2624b7cd-c80a-4bea-96ff-ea0416a64315",
                                                    IdentityType="IAM", SessionLifetimeInMinutes=100,
                                                    ResetDisabled=True,
                                                    UndoRedoDisabled=True)
```

The following code demonstrates how the published dashboard was embedded to the administrator page in the frontend.

```
var dashboard;

function embedDashboard() {
    var containerDiv = document.getElementById("embeddingContainer");
    var options = {
        // replace this dummy url with the one generated via embedding API
        url: "https://us-east-1.quicksight.aws.amazon.com/embed/8560e36e5cbc4",
        container: containerDiv,
        scrolling: "no",
        {#height: "700px", #}
        {#width: "1000px", #}
        footerPaddingEnabled: true
    };
    dashboard = QuickSightEmbedding.embedDashboard(options);
}
```

6. Amazon Route 53

The following documentation was used in this section.

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/getting-started.html>

The domain (tracex.link) for the application was bought using the AWS Route 5. Then a hosted zone was created with reference to the domain name and the Name servers. In order to connect the Beanstalk application with the domain name, 'A' record was created in the hosted zone.

Records (5) [Info](#)
Automatic mode is the current search behavior optimized for best filter results. [To change modes go to settings.](#)

Type ▼
Routing policy ▼
Alias ▼

<input type="checkbox"/>	Record name ▼	Type ▼	Routin... ▼	Differ... ▼	Value/Route traffic to
<input type="checkbox"/>	tracex.link	A	Simple	-	tracexnewdockerenvcc2.eba-ebisbmmr.us-east-1.elasticbeanstalk.com.
<input type="checkbox"/>	tracex.link	NS	Simple	-	ns-1470.awsdns-55.org. ns-849.awsdns-42.net. ns-1564.awsdns-03.co.uk. ns-308.awsdns-38.com.
<input type="checkbox"/>	tracex.link	SOA	Simple	-	ns-1470.awsdns-55.org. awsdns-hostmaster.amazon.com. 1 7200 900 1
<input type="checkbox"/>	_2cdd38807d59078f2669ef8c57c...	CNAME	Simple	-	_9ed9c8690ba3fa8c6c8ef82abc01d16f.xrchbtpdjs.acm-validations.aws
<input type="checkbox"/>	_593792bfa7f3d5209604e48991...	CNAME	Simple	-	_23bbcd05b921be6b5947a2f59d0f81fa.xrchbtpdjs.acm-validations.aws

7. Amazon Certificate Manager

ACM was used to obtain the SSL certifications for the purchased domain.

<input type="checkbox"/>	Name ▼	Domain name ▼	Additional names	Status ▼	Type ▼
<input type="checkbox"/>	-	tracex.link	www.tracex.link	Issued	Amazon Issued

Status

Status Issued
Detailed status The certificate was issued at 2021-06-12T09:13:17UTC

Domain	Validation status
▶ tracex.link	Success
▶ www.tracex.link	Success

Classic Load Balancer listener
×

Listener port
443

Listener protocol
The load balancer transport protocol to use for routing.
HTTPS ▼

Instance port
The port on which the instance server is listening.
80

Instance protocol
The protocol to use for routing traffic to backend instances. This must be at the same internet protocol layer as the listener protocol. It also must have the same security level as any other listener using the same instance port as this listener.
HTTP ▼

SSL certificate
tracex.link - 7c6608eb-360e-485c-bb4a-acae... ↻

Cancel Save

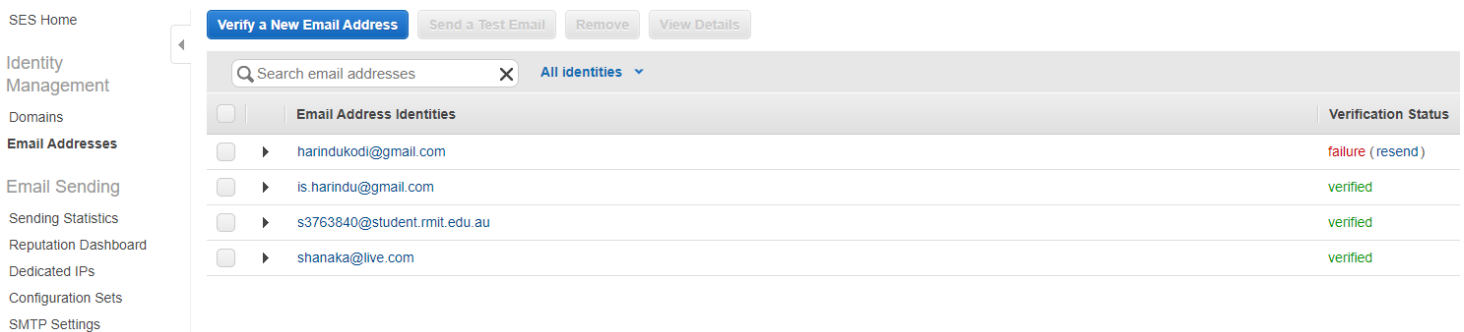
After the SSL certification was issued by ACM. It was needed to be connected with the Load Balancer of the Beanstalk environment.

8. Amazon Simple Email Service

The implementation of the SES was conducted by referring to the following documentation.

<https://docs.aws.amazon.com/ses/latest/dg/send-email-getting-started-migrate.html>

Since the SES is a sandboxed version in TraceX, when the users get registered with the system, an email would be sent out to them using a Lambda function. Once the user verifies the email, alerts would be sent out for the users.



Following code is written in the Lambda function to verify the email of a user.

```
def verify_email_identity(user_email):
    ses_client = boto3.client("ses", region_name="ap-southeast-1")
    response = ses_client.verify_email_identity(
        EmailAddress=user_email
    )
    print(response)
```

Following code is written in the Lambda function to send Email alerts for the users.

```
def send_plain_email(email, email_text):
    ses_client = boto3.client("ses", region_name="ap-southeast-1")
    CHARSET = "UTF-8"

    response = ses_client.send_email(
        Destination={
            "ToAddresses": [
                email,
            ],
        },
        Message={
            "Body": {
                "Text": {
                    "Charset": CHARSET,
                    "Data": email_text,
                }
            },
            "Subject": {
                "Charset": CHARSET,
                "Data": "TraceX COVID-19 Alert",
            },
        },
        Source="s3763840@student.rmit.edu.au",
    )
```

8. Amazon CloudFront

The following documentation was used in integrating CloudFront with TraceX.

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.cloudfront.html>

To integrate the CloudFront with the Beanstalk deployed application a CloudFront distribution was created by connecting the Elastic load balancer of the Elastic beanstalk environment with the Origin Domain Name and Path of the CloudFront distribution.

CloudFront Distributions > E3PZ505YCL9FS1

General | **Origins and Origin Groups** | Behaviors | Error Pages | Restrictions | Invalidations | Tags

Origins

Create Origin | Edit | Delete

	Origin Domain Name and Path	Origin ID
<input type="checkbox"/>	awseb-e-d-AWSEBLoa-1QNJ4BRW0BP8I-1235674701.us-east-1.elb.amazonaws.com	ELB-awseb-e-d-AWSEBLoa-1QNJ4BRW0BP8I-1235674701

The following shows the configurations of the CloudFront distribution of the TraceX.

Distribution ID	E3PZ505YCL9FS1
ARN	arn:aws:cloudfront::681989621175:distribution/E3PZ505YCL9FS1
Log Prefix	-
Delivery Method	Web
Cookie Logging	Off
Distribution Status	Deployed
Comment	-
Price Class	Use All Edge Locations (Best Performance)
AWS WAF Web ACL	-
State	Disabled
Alternate Domain Names (CNAMEs)	-
SSL Certificate	tracex.link (7c6608eb-360e-485c-bb4a-acae3362be4a)
Domain Name	d142airwlb5xio.cloudfront.net
Custom SSL Client Support	Clients that Support Server Name Indication (SNI) - (Recommended)
Security Policy	TLSv1.2_2019
Supported HTTP Versions	HTTP/2, HTTP/1.1, HTTP/1.0
IPv6	Enabled
Default Root Object	-
Last Modified	2021-06-13 00:34 UTC+10
Log Bucket	-

9. Amazon S3

Administrators can upload the dashboard analytics CSV files to the S3 for future references for specific dates if necessary. Administrators can upload the CSV files using the file upload feature implemented in the system. These files then would be saved in the S3 location media folder.

Amazon S3 > elasticbeanstalk-us-east-1-681989621175 > media/

media/

```
775 try:
776     file_path = 'media/' + name_new
777     S3_BUCKET = 'elasticbeanstalk-us-east-1-681989621175'
778     path = 'media/'
779     key = path + name_new
780     s3_client.upload_file(file_path, S3_BUCKET, key)
781     os.remove(file_path)
```

Setting up the project

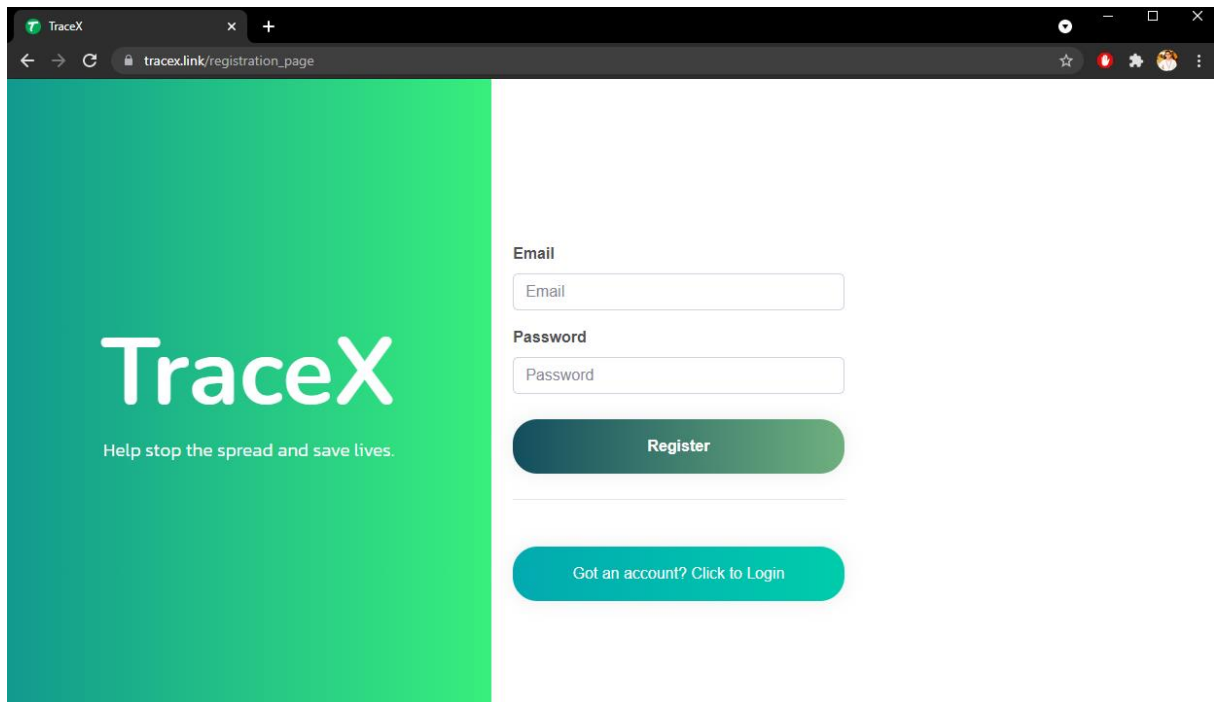
1. Clone the project from <https://github.com/harindukodi/TraceX>
2. Create a Python virtual environment.
3. Install the requirements.txt
4. Run the project.

6. User Manual

User Functions

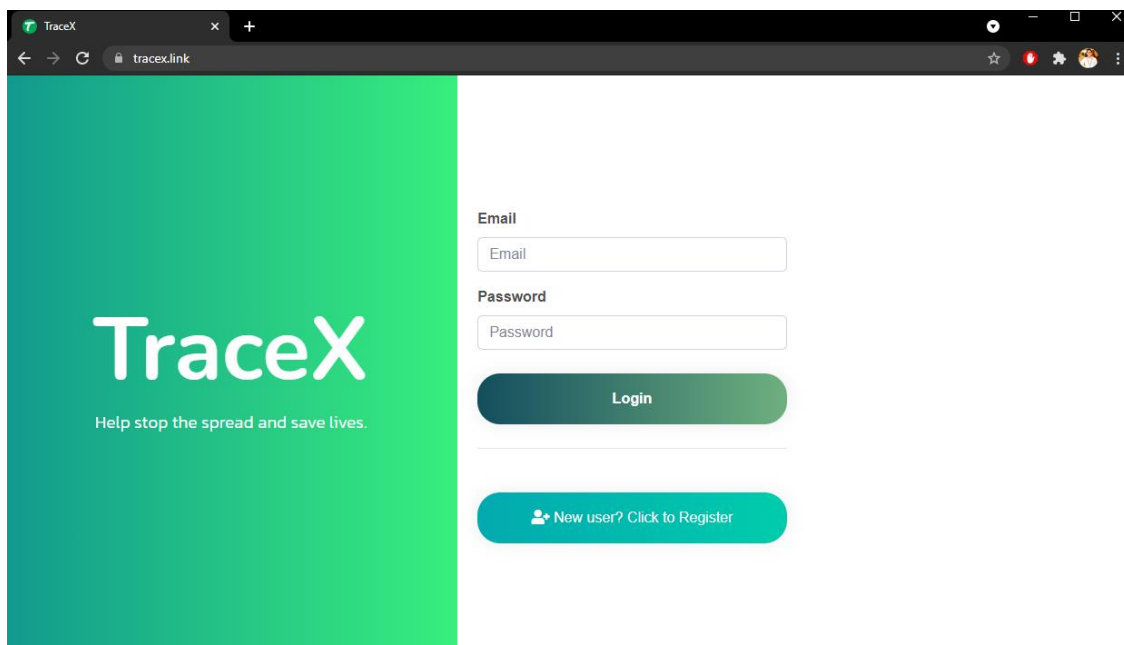
1. Registration

Users can get registered from this page. Once the user gets registered, an email would be sent to the user email inbox to verify the email to send out application alerts.



The screenshot shows a web browser window with the URL `tracex.link/registration_page`. The page features a large green banner on the left with the 'TraceX' logo and the tagline 'Help stop the spread and save lives.' On the right, there is a registration form with two input fields: 'Email' and 'Password'. Below these fields is a green 'Register' button. At the bottom of the form, there is a link that says 'Got an account? Click to Login'.

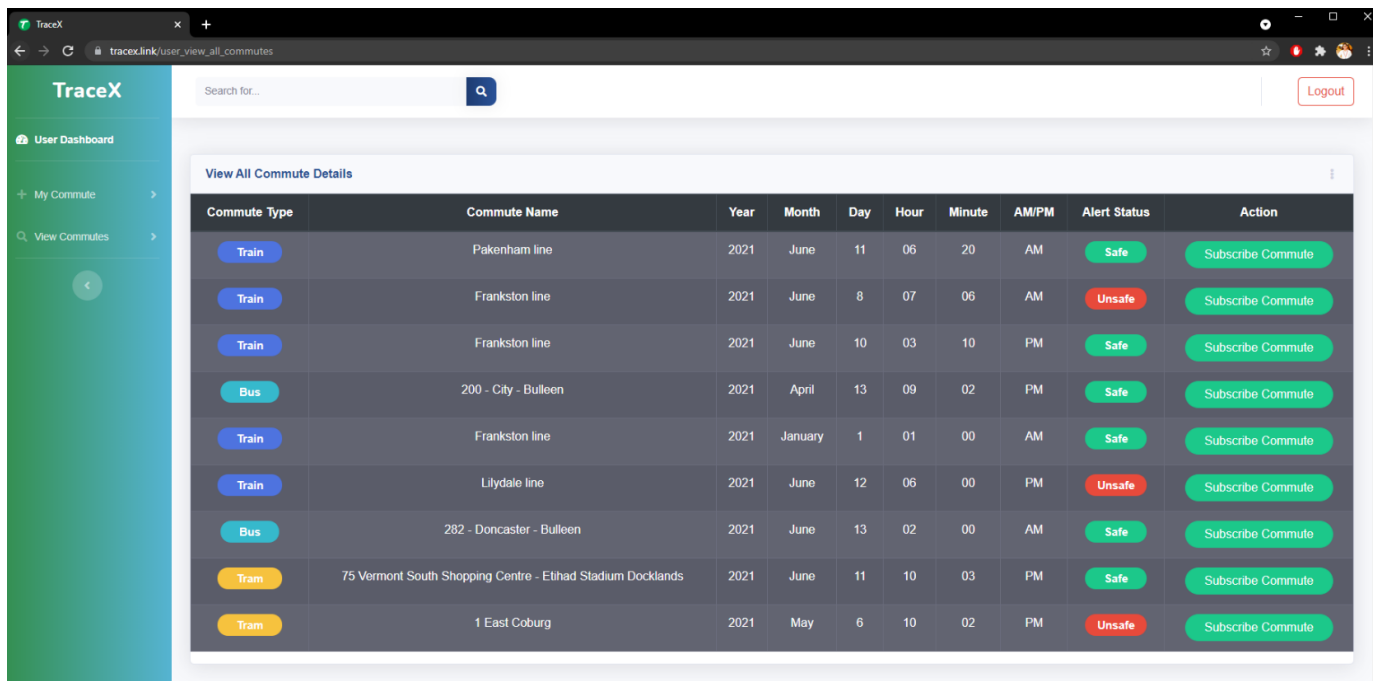
2. Login – Users can login to the system using this page.



The screenshot shows a web browser window with the URL `tracex.link`. The page features a large green banner on the left with the 'TraceX' logo and the tagline 'Help stop the spread and save lives.' On the right, there is a login form with two input fields: 'Email' and 'Password'. Below these fields is a green 'Login' button. At the bottom of the form, there is a link that says 'New user? Click to Register'.

3. View and search all the transports

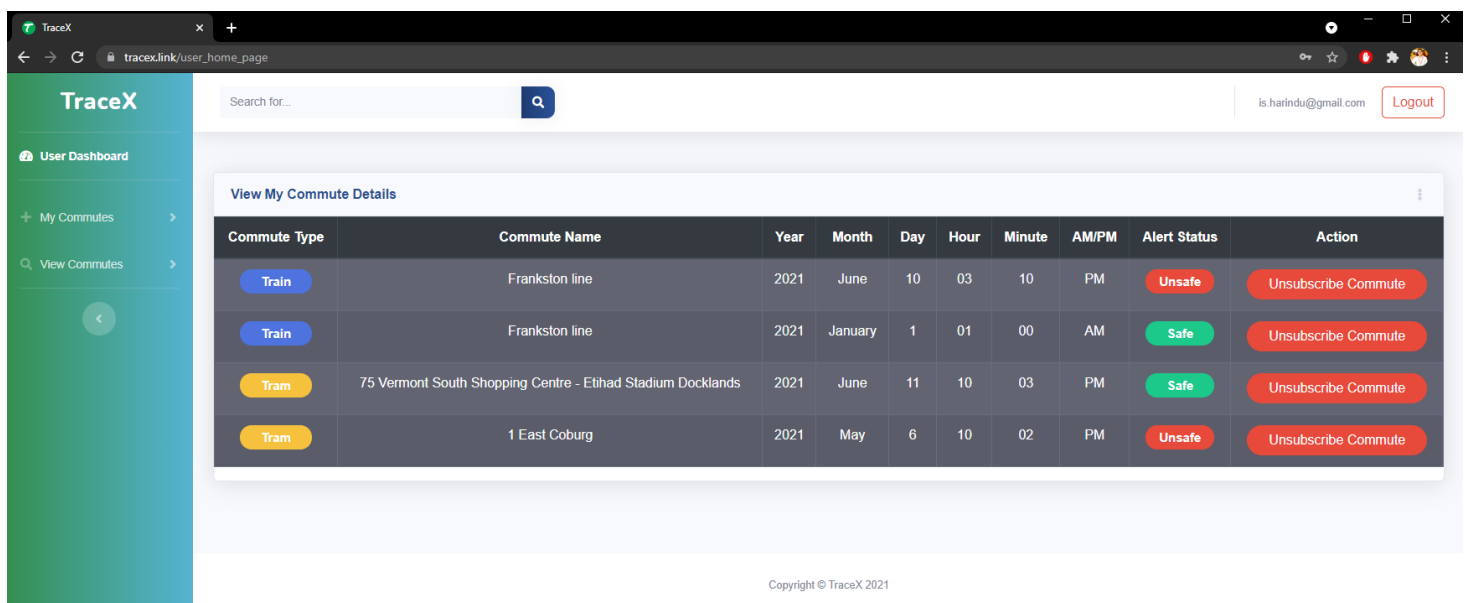
Users can view or search all the transports available in the system and click ‘**Subscribe Commute**’ button to get alerts for those transports if the user has commuted in that transport.



Commute Type	Commute Name	Year	Month	Day	Hour	Minute	AM/PM	Alert Status	Action
Train	Pakenham line	2021	June	11	06	20	AM	Safe	Subscribe Commute
Train	Frankston line	2021	June	8	07	06	AM	Unsafe	Subscribe Commute
Train	Frankston line	2021	June	10	03	10	PM	Safe	Subscribe Commute
Bus	200 - City - Bulleen	2021	April	13	09	02	PM	Safe	Subscribe Commute
Train	Frankston line	2021	January	1	01	00	AM	Safe	Subscribe Commute
Train	Lilydale line	2021	June	12	06	00	PM	Unsafe	Subscribe Commute
Bus	282 - Doncaster - Bulleen	2021	June	13	02	00	AM	Safe	Subscribe Commute
Tram	75 Vermont South Shopping Centre - Etihad Stadium Docklands	2021	June	11	10	03	PM	Safe	Subscribe Commute
Tram	1 East Coburg	2021	May	6	10	02	PM	Unsafe	Subscribe Commute

4. View my commute details

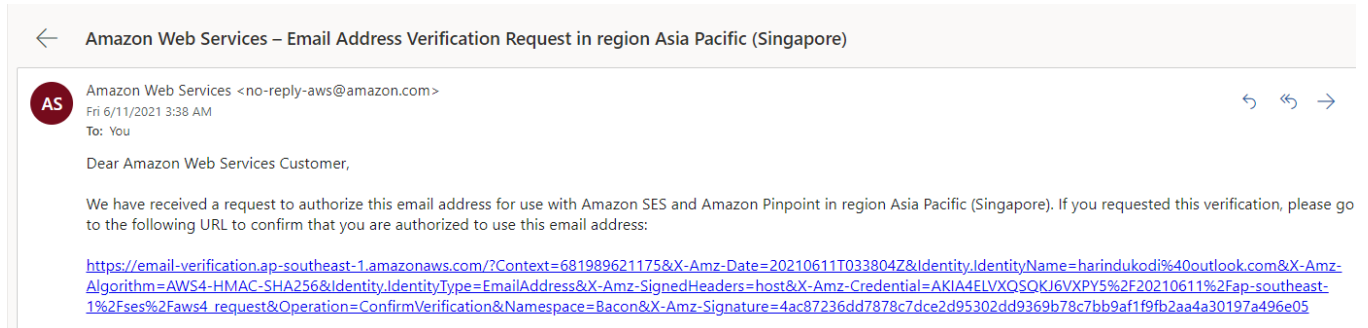
Users can view the subscribed transports. This only shows the transports that the user has taken. The user also can click ‘**Unsubscribe Commute**’ to remove a transport.



Commute Type	Commute Name	Year	Month	Day	Hour	Minute	AM/PM	Alert Status	Action
Train	Frankston line	2021	June	10	03	10	PM	Unsafe	Unsubscribe Commute
Train	Frankston line	2021	January	1	01	00	AM	Safe	Unsubscribe Commute
Tram	75 Vermont South Shopping Centre - Etihad Stadium Docklands	2021	June	11	10	03	PM	Safe	Unsubscribe Commute
Tram	1 East Coburg	2021	May	6	10	02	PM	Unsafe	Unsubscribe Commute

5. User email verification

The following email will be sent to the user email inbox from the AWS Simple Email Service once a user gets registered with the system.

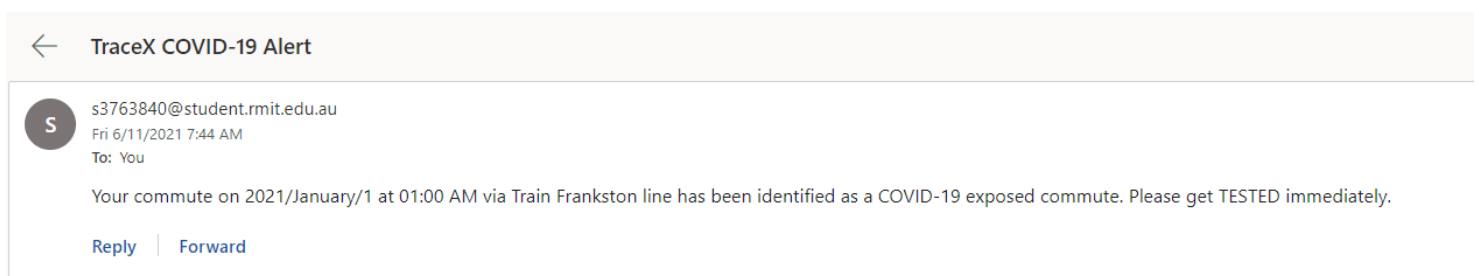


6. User email alerts

The following email alert would be received to a user, if the user has been on a public transport which has been identified as an COVID-19 exposed commute by the health authorities. Once the administrator changes the status of the transport, all the users who are subscribed to that transport would get this email.

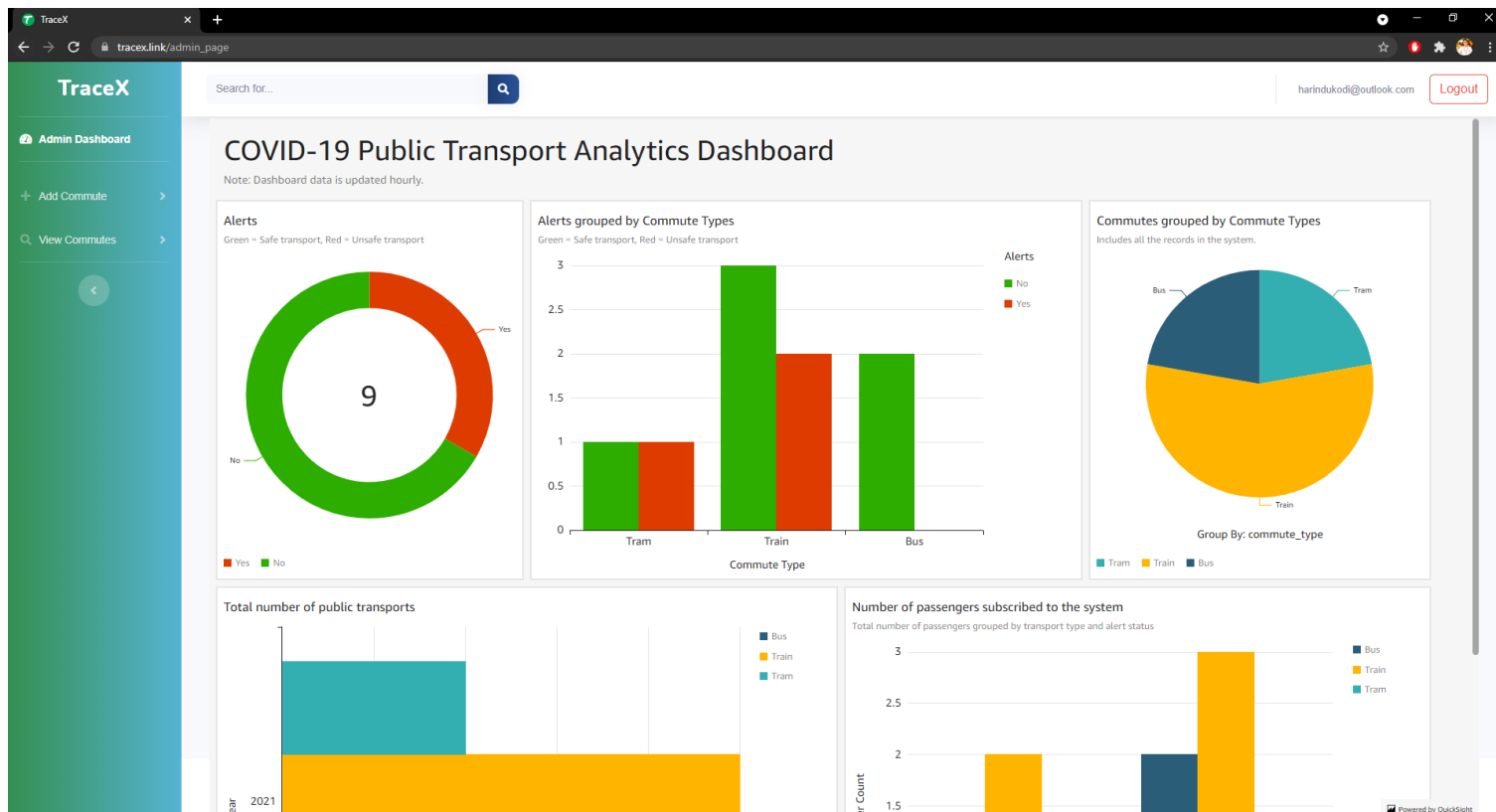


The following email would be received to an user, if the user has been on a COVID-19 exposed transport and the authorities have confirmed the transport is COVID-19 free.



Administrator Functions

1.Admin Dashboard View



The TraceX administrator would be able to view the above dashboard of analytics from the data in the system. This dashboard displays the summary of the data related to the COVID-19 public transports and commutes. Dashboard displays the numbers of transports which are identified as COVID-19 exposed commutes or not. Alerts grouped by different types of commutes, number of users registered and taken the public transports according to different transport types etc.

2.Add Commute Details

Admins can add commute details by clicking on the 'Add Commute' item at the left sidebar. Admins need to specify all the details in the form below before clicking on the 'Add Commute' button. A message that notifies the commute has been added successfully will be displayed.

The screenshot shows the 'Add Commute Details' form in the TraceX admin dashboard. The form is titled 'Add Commute Details' and contains the following fields:

- Commute Type:** A dropdown menu with 'Train' selected.
- Commute Name:** A dropdown menu with 'Frankston line' selected.
- Month:** A dropdown menu with 'January' selected.
- Day:** A dropdown menu with '1' selected.
- Year:** A dropdown menu with '2021' selected.
- Hour:** A dropdown menu with '01' selected.
- Minute:** A dropdown menu with '00' selected.
- AM / PM:** A dropdown menu with 'AM' selected.
- Subject to Alert:** A dropdown menu with 'No' selected.

At the bottom of the form is a green button labeled 'Add Commute'.

3.View commute details

Admins can view all commute details by clicking on the 'View Commutes' item at the left sidebar. If there are COVID cases on board, admin can alert affected commute session and passengers who ride those commutes will receive an alert email.

The screenshot shows the 'View All Commute Details' table in the TraceX admin dashboard. The table has the following columns: Commute Type, Commute Name, Year, Month, Day, Hour, Minute, AM/PM, Alert Status, Passenger Count, and Action.

Commute Type	Commute Name	Year	Month	Day	Hour	Minute	AM/PM	Alert Status	Passenger Count	Action
Train	Pakenham line	2021	June	11	06	20	AM	Safe	0	Mark Unsafe
Train	Frankston line	2021	June	8	07	06	AM	Unsafe	0	Mark Safe
Train	Frankston line	2021	June	10	03	10	PM	Safe	1	Mark Unsafe
Bus	200 - City - Bulleen	2021	April	13	09	02	PM	Safe	0	Mark Unsafe
Train	Frankston line	2021	January	1	01	00	AM	Safe	1	Mark Unsafe
Train	Lilydale line	2021	June	12	06	00	PM	Unsafe	0	Mark Safe
Bus	282 - Doncaster - Bulleen	2021	June	13	02	00	AM	Safe	0	Mark Unsafe
Tram	75 Vermont South Shopping Centre - Etihad Stadium Docklands	2021	June	11	10	03	PM	Safe	1	Mark Unsafe
Tram	1 East Coburg	2021	May	6	10	02	PM	Unsafe	1	Mark Safe

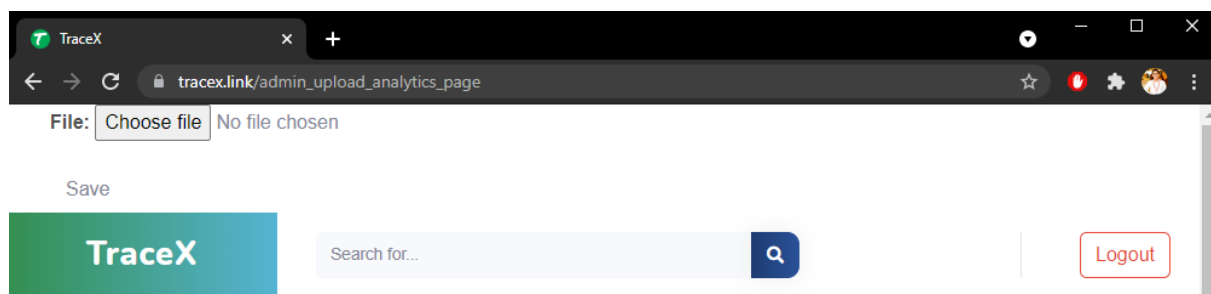
4. Make a transport safe or unsafe

Administrators can make the transport Safe or Unsafe depending on the COVID-19 status of that transport.

View All Commute Details										
Commute Type	Commute Name	Year	Month	Day	Hour	Minute	AM/PM	Alert Status	Passenger Count	Action
Train	Pakenham line	2021	June	11	06	20	AM	Safe	0	Mark Unsafe
Train	Frankston line	2021	June	8	07	06	AM	Unsafe	0	Mark Safe
Train	Frankston line	2021	June	10	03	10	PM	Safe	1	Mark Unsafe
Bus	200 - City - Bulleen	2021	April	13	09	02	PM	Safe	0	Mark Unsafe
Train	Frankston line	2021	January	1	01	00	AM	Safe	1	Mark Unsafe
Train	Lilydale line	2021	June	12	06	00	PM	Unsafe	0	Mark Safe
Bus	282 - Doncaster - Bulleen	2021	June	13	02	00	AM	Safe	0	Mark Unsafe
Tram	75 Vermont South Shopping Centre - Etihad Stadium Docklands	2021	June	11	10	03	PM	Safe	1	Mark Unsafe
Tram	1 East Coburg	2021	May	6	10	02	PM	Unsafe	1	Mark Safe

5. Upload analytics reports

Administrators can download the analytics relevant for a specific time frame from the dashboard into a CSV file and then upload to a location in the cloud for future analysis.



References

- [1] Ferretti, L., Wymant, C., Kendall, M., Zhao, L., Nurtay, A., Abeler-Dörner, L., Parker, M., Bonsall, D., & Fraser, "Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing", *Science*, vol. 368, pp. 1, May 2020, DOI: 10.1126/science.abb6936
- [2] Department of Health, "COVIDSafe app", <https://www.health.gov.au/resources/apps-and-tools/covidsafe-app> (accessed June 6, 2021)
- [3] NHS, "NHS COVID-19", <https://www.nhs.uk/apps-library/nhs-covid-19/> (accessed June 6, 2021)
- [4] AWS. "Amazon EC2 Secure and resizable compute capacity to support virtually any workload." aws.amazon.com. <https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc> (accessed June 11, 2021).
- [5] AWS. "Amazon S3 Object storage built to store and retrieve any amount of data from anywhere." aws.amazon.com. <https://aws.amazon.com/s3/> (accessed June 11, 2021).
- [6] AWS. "Amazon API Gateway Create, maintain, and secure APIs at any scale." aws.amazon.com. <https://aws.amazon.com/api-gateway/> (accessed June 11, 2021).
- [7] AWS. "Amazon QuickSight Scalable, serverless, embeddable, ML-powered BI Service built for the cloud." aws.amazon.com. <https://aws.amazon.com/route53/> (accessed June 11, 2021).
- [8] AWS. "Amazon Route 53 A reliable and cost-effective way to route end users to Internet applications." aws.amazon.com. <https://aws.amazon.com/route53/> (accessed June 11, 2021).
- [9] <https://docs.aws.amazon.com/quicksight/latest/user/refreshing-imported-data.html>
- [10] <https://docs.aws.amazon.com/quicksight/latest/user/embedding-overview.html>
- [11] <https://docs.aws.amazon.com/quicksight/latest/user/embedded-analytics-dashboards-with-anonymous-users-step-3.html>
- [12] <https://medium.com/@jbesw/how-to-add-ssl-certificates-to-elastic-beanstalk-and-cloudfront-452f68591ca5>
- [13] <https://botocore.amazonaws.com/v1/documentation/api/latest/tutorial/index.html>

