Introduction

The main objective of this project is to create a proof of architecture prototype of a contact tracing website for the University of Leeds to use on campus to mitigate the spread of coronavirus.

Overview of Technical Design

Development Tools

- The web application is made with PHP, MySQL (5.6) and HTML
- GNU Nano is used as the editor
- Amazon Web Services EC2 for hosting the files
- Amazon Web Services RDS for the hosting database
- Apache Server (2.4.46 Amazon)
- Web-Browser: Google Chrome, Safari (Desktop and Mobile)
- Terminal for SSH to the EC2 instance

System Software:

Client side:

- OS supporting web browsing
- Web Browser

Server side:

- Web Server
- PHP, MySQL

Implementation diagram

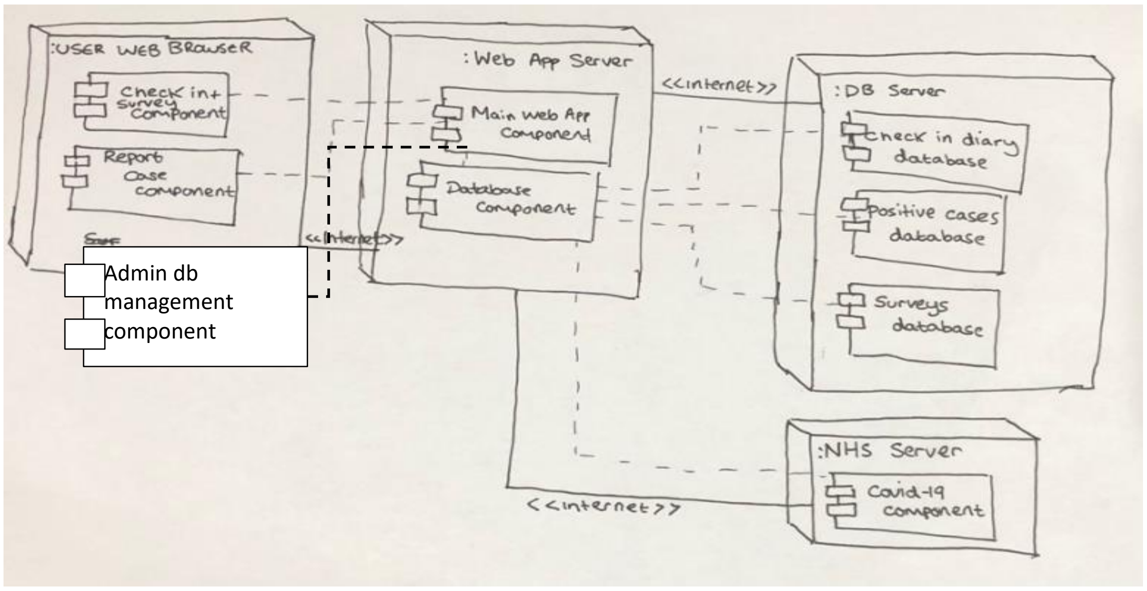


Figure 1 Implantation diagram of system design to be tested by the prototype

Figure 1 shows the structure of the hardware and software.
The backend databases which are stored on an AWS RDS instance are accessed over the internet. There are separate tables for 'check ins', coronavirus cases and surveys. All database interaction is managed by a database component within the web server.

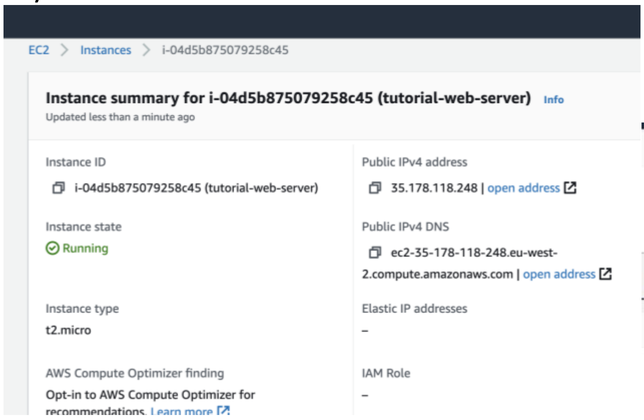
The web application is hosted on an AWS EC2 instance and is accessible by the user over the internet on a web browser.
The web application allows users to check in, answer surveys, report cases and perform admin through a web browser. Should it be required the system connects to NHS systems over the internet.

Testing scenario

Real life testing will require accessing the website URL. This should display the simple mock-up check-in screen. From here, the system should be able to take a user ID and location and store it centrally. Such that it is retained when the testing device is offline. This can be checked by closing the web page or browser or using another device then accessing the admin page to check the database table content. The system should also be tested the same way for reporting cases.

Evidence of development

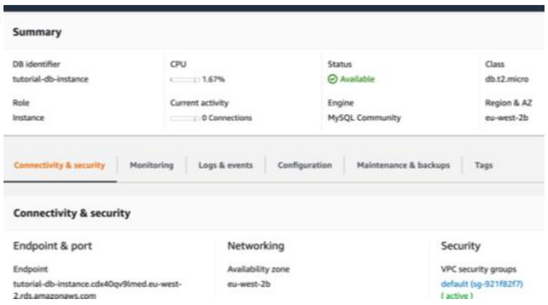
The provisioning of an EC2 instance for the web application and an RDS instance for the MySQL database:



EC2 > Instances > i-04d5b875079258c45

Instance summary for i-04d5b875079258c45 (tutorial-web-server) Info
Updated less than a minute ago

Instance ID	Public IPv4 address
i-04d5b875079258c45 (tutorial-web-server)	35.178.118.248 | open address
Instance state	Public IPv4 DNS
Running	ec2-35-178-118-248.eu-west-2.compute.amazonaws.com | open address
Instance type	Elastic IP addresses
t2.micro	–
AWS Compute Optimizer finding	IAM Role
Opt-in to AWS Compute Optimizer for recommendations. Learn more	–



Summary

DB identifier	CPU	Status	Class
tutorial-db-instance	1.67%	Available	db.t2.micro
Role	Current activity	Engine	Region & AZ
Instance	0 Connections	MySQL Community	eu-west-2b

Connectivity & security

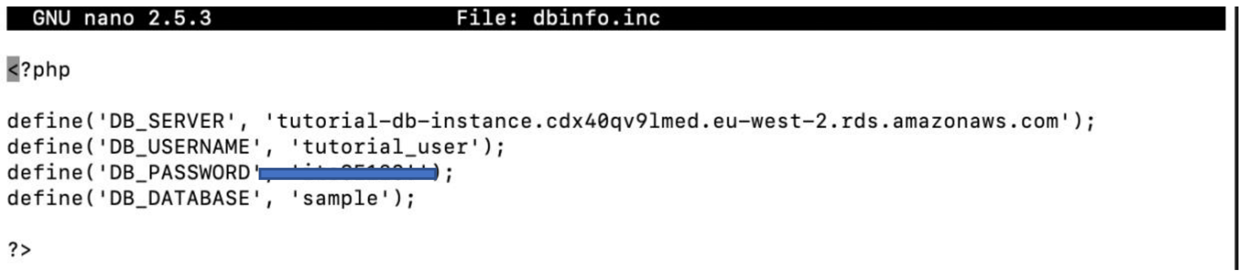
Endpoint & port	Networking	Security
Endpoint	Availability zone	VPC security groups
tutorial-db-instance.cdx40qv9lmed.eu-west-2.rds.amazonaws.com	eu-west-2b	default (sg-921f82f7) (active)

Figure 2 Amazon EC2 instance

Figure 3 Amazon RDS database instance

I used SSH to access the EC2 instance, updated the Linux image and installed Apache webserver.

The database connection and authentication details are shown in figure 4. It is good practice to store the database connection definition in a separate file to the web app files.



```
  GNU nano 2.5.3                    File: dbinfo.inc

<?php

define('DB_SERVER', 'tutorial-db-instance.cdx40qv9lmed.eu-west-2.rds.amazonaws.com');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD'[redacted]);
define('DB_DATABASE', 'sample');

?>
```

Figure 4 dbinfo.inc

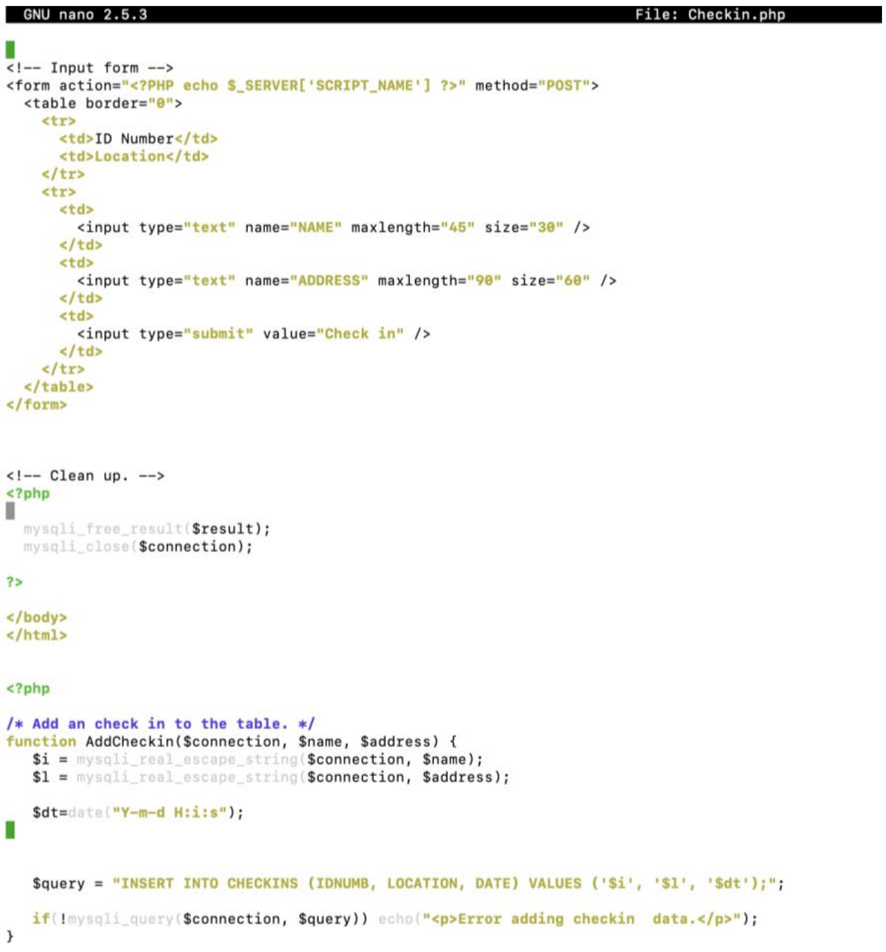
This database has tables for

- “CHECKINS” – to record each check in to a location

- "CASES" – to record each reported positive case

The respective php files (Checkin.php/Report.php) of the web app handle creating the table if it doesn't exist already in the database

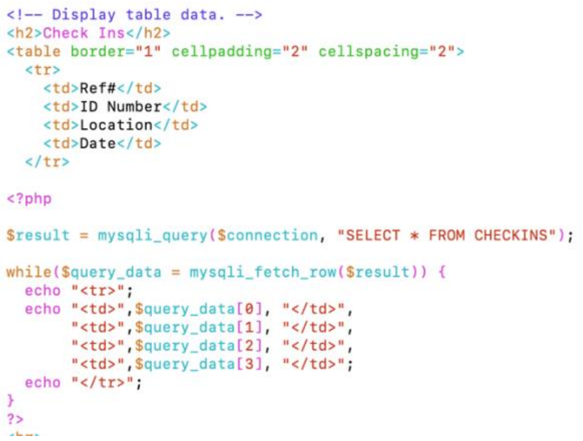
Frontend is handled by hosting PHP webpages on the AWS EC2 web server. These web pages form the web application. They render the components for the users to interact with.



```
GNU nano 2.5.3                                                    File: Checkin.php

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>ID Number</td>
      <td>Location</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
        <input type="submit" value="Check in" />
      </td>
    </tr>
  </table>
</form>


<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>

</body>
</html>


<?php

/* Add an check in to the table. */
function AddCheckin($connection, $name, $address) {
   $i = mysqli_real_escape_string($connection, $name);
   $l = mysqli_real_escape_string($connection, $address);

   $dt=date("Y-m-d H:i:s");



   $query = "INSERT INTO CHECKINS (IDNUMB, LOCATION, DATE) VALUES ('$i', '$l', '$dt');";

   if(!mysqli_query($connection, $query)) echo("<p>Error adding checkin  data.</p>");
}
```

Figure 5 nano editor of Checkin.php

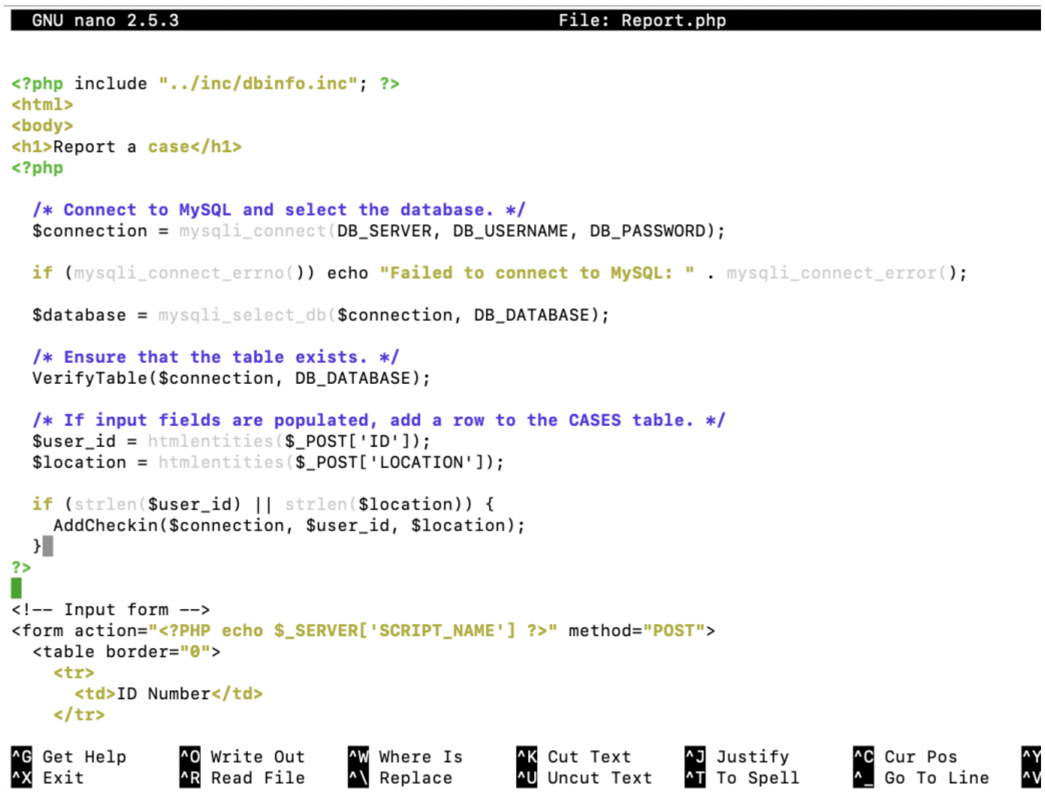
Figure 5 shows the structured HTML for creating the input form and the PHP which adds a check in to the database using SQL.



```
<!-- Display table data. -->
<h2>Check Ins</h2>
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>Ref#</td>
    <td>ID Number</td>
    <td>Location</td>
    <td>Date</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM CHECKINS");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>",
       "<td>",$query_data[3], "</td>";
  echo "</tr>";
}
?>
```

Figure 6 vim editor of Admin.php

Figure 6 shows the PHP for the admin page displaying the data on screen for the administrators to see



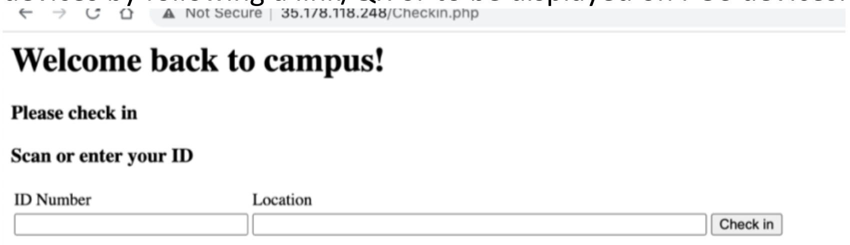
```
  GNU nano 2.5.3                                  File: Report.php

<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Report a case</h1>
<?php

  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " . mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the table exists. */
  VerifyTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the CASES table. */
  $user_id = htmlentities($_POST['ID']);
  $location = htmlentities($_POST['LOCATION']);

  if (strlen($user_id) || strlen($location)) {
    AddCheckin($connection, $user_id, $location);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>ID Number</td>
    </tr>

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     ^Y
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  ^V
```

Figure 7 Report.php

Figure 7 shows the PHP for the Report page. It shows establishing the connection to the database and errors for failure.

Evidence of deployment

Figure 8 shows the web page that all users use to check in to a location. Usable on personal devices by following a link/QR or to be displayed on POS devices.



Not Secure | 35.178.118.248/Checkin.php

Welcome back to campus!

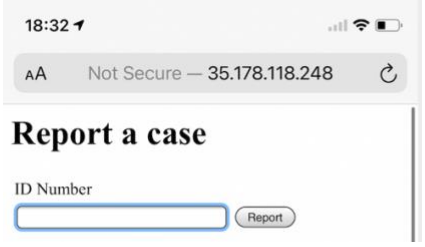
Please check in

Scan or enter your ID

ID Number Location

Figure 8 Checkin.php on a PC (For POS example)

Figure 9 shows the web page users will access to report a positive COVID-19 case.



18:32

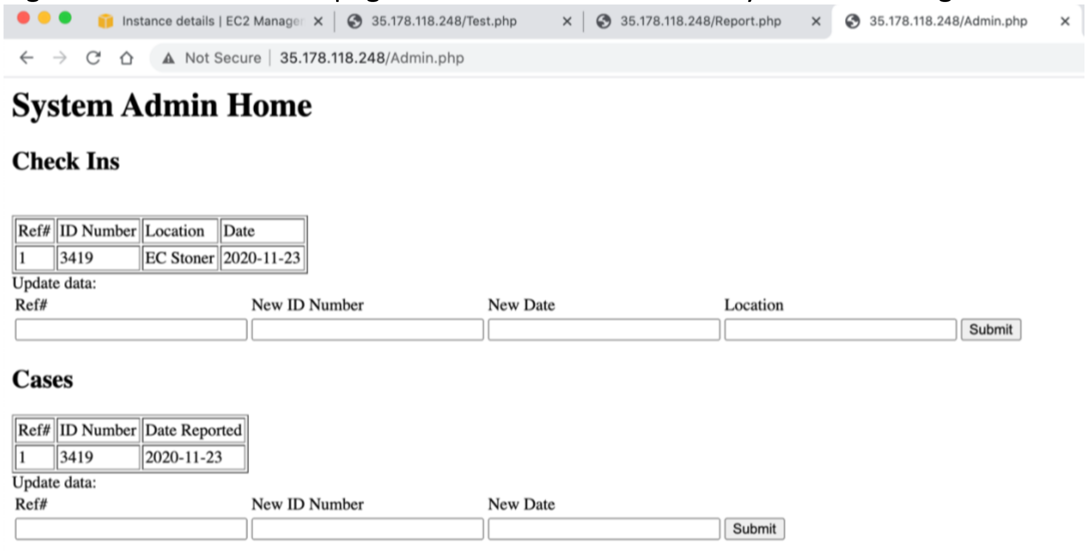
AA Not Secure — 35.178.118.248

Report a case

ID Number

Figure 9 Report.php on a mobile phone

Figure 10 shows the web page for administrators of the system to manage the data


Instance details | EC2 Manage
35.178.118.248/Test.php
35.178.118.248/Report.php
35.178.118.248/Admin.php
Not Secure | 35.178.118.248/Admin.php

System Admin Home

Check Ins

Ref#	ID Number	Location	Date
1	3419	EC Stoner	2020-11-23

Update data:

Ref#	New ID Number	New Date	Location	
				Submit

Cases

Ref#	ID Number	Date Reported
1	3419	2020-11-23

Update data:

Ref#	New ID Number	New Date	
			Submit

Figure 10 Admin.php on a PC

Evaluation

This proof of architecture prototype uses Amazon Web Services to host a coronavirus track and trace system with a backend database and frontend for users to interact with.
The prototype is lightweight and effective with the test data submitted be stored and manipulated correctly. It should also be scalable using AWS. Meaning the system should continue to be effective with a larger load.
The system is simple to use, works on a large range of devices and is inexpensive as the AWS platform charges depending on the load.
This prototype fails to address security features as it is only testing the feasibility of the system structure. However, the use of a VPC to isolate the database and a login page for the administrators would increase security. Fortunately, these are features AWS can help implement. Additionally, security groups can be altered to restrict traffic to specific ports and IP addresses.
Unfortunately, this prototype does not successfully implement email confirmation for users checking in or surveys. This would require some kind of notification service perhaps AWS SNS but would also require a survey database and access or copies of email addresses corresponding to user IDs.
To improve this prototype, it could use a more relevant domain name and have better validation on inputs such as locations. In the real system the POS will automatically fill in location data.
A challenge I encountered was setting up a centralised database. I have used AWS before, but I haven’t worked with online databases. I researched tutorials and documentation before having to learn some PHP in order to implement my prototype.