Technical Manual

This project involves the following languages: HTML, CSS, PHP and Javascript. No frameworks, or JS libraries were used. Within psxjb9-20508981_InstallationFiles folder are all the files and folder that makes up the website.

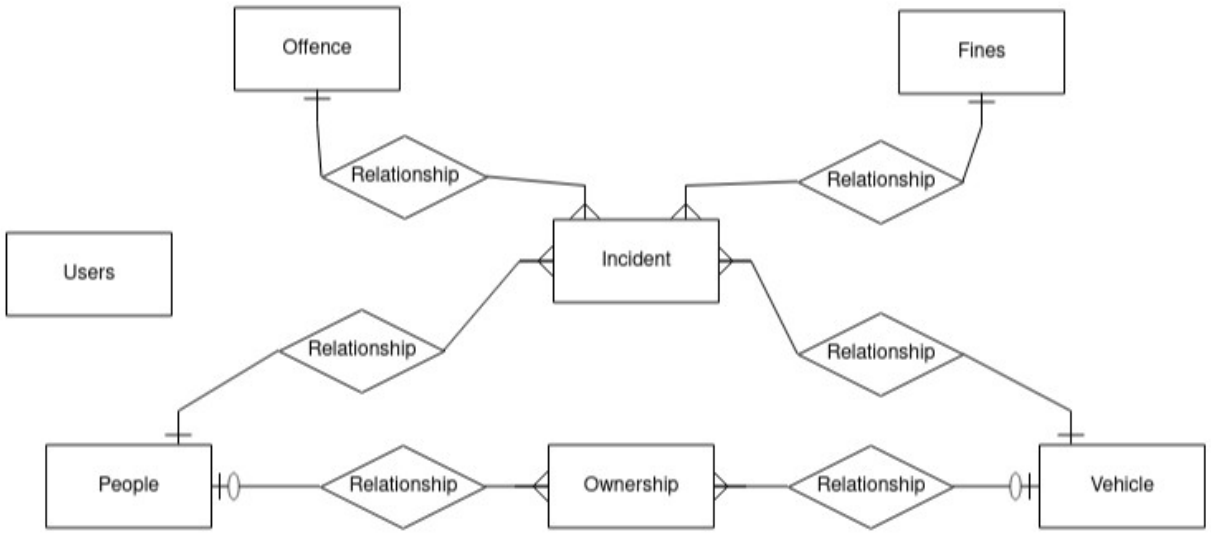
In order to install this project you will require MySQL for the database and PHP for the server side. Under public_html you will find DIS-COMP4039-CW2-psxjb9-20508981. Open this directory and then open the psxjb9-20508981_InstallationFiles in your code editor/IDE (I used vscode) and you will will see all the files displayed. You can access the database from a DBMS, the database script is in the mydb.sql file.

I have used XAMPP on my local computer with vscode in order to work on this project.

The list of files includes:

- config.php
- login.php
- welcome.php
- add_person.php
- add_vehicle.php
- add_fines.php
- change_password.php
- create_accounts.php
- edit_incidents.php
- incident.php
- logout.php
- mydb.sql
- people_search.php
- search_incident.php
- style1.css
- vehicle_search.php

The database design is described in this ERD below:



The Users table is meant to record the various users with their username and

password that can access the system and determine control access with the use of $_SESSION variables. In each file at the top there is a php script that will find the $_SESSION[‘log’] to see if it is set to determine whether the user is logged in otherwise they will be redirected to the login.php page. The Users table has no relation to the other tables. The relationship between the People and Vehicle tables is a many to many so the Ownership separates this creating a one to many relationship. So one person can own one or many vehicles and one vehicle can be owned by one or more people (although in the database there is only one owner per vehicle) but the website allows for many owners per vehicle if need be.

A person can have one or many incidents and a vehicle can also have one or many incidents. A fine can be given out to many incidents but only one find per incident can be given. Similarly with offence where an offence can be given in many incidents but only one offence per incident.

I have also created an Audit_log table for question 7, but I did not have enough time to implement this feature, I created a file for the admin that would allow them to view the users actions with the columns User, Action and Timeline.

The php seen throughout the files is mainly procedural, sometimes using session variables. The php, html and css are usually in the same document, with the php being embedded in the html with inline css styles alter individual html elements.

The config.php file contains the details to connect to the database and this is called in every file that needs to submit a SQL query to the database. session_start() is required for every file that is using the session variables.

Any file that requires interaction with database will have:

```
require(“config.php”);
$conn = newmysqli($servername, $username, $password, $dbname);
```

The common structure/pattern of calling sql statements for example:

```
$sql = “SELECT * FROM People;”;
$result = mysqli_query($conn, $sql);
```

Then to print out results generally in table format (for example):

```
if(mysqli_num_rows($result) > 0)
{
      echo “<table>”;
      echo “<tr><th>Name</th><th>Address</th></tr>”;
}
      while($row = mysqli_fetch_assoc($result))
      {
            echo “<tr>”;
                  echo “<td>”.$row[“People_name”].”</td>”;
                  echo “<td>”.$row[“People_address”].”</td>”;
            echo “</tr>”;
      }
      echo “</table>”;
```

In login.php the sql query on line 64 will select all from users where the post value from the form is equal to the username and the password inputted, both are required and if incorrect an error message will be delayed and the page will be reloaded. Also the sql query includes a BINARY which ensure that the input is case sensitive.

In welcome.php will contain links to all of the various pages available to the user logged in. On line 52 there is a php script that will only show certain results to the admin (daniels) but even if a non admin user enters the url for

the two restricted pages add_fines.php and create_accounts.php then the page will send them back to welcome.php → done through session variables.

In people_search.php a normal SELECT query from People table will search for where name inputted is equal to one on the table. The MySQL is inherently case insensitive and in order to account for only first or last name inputted in the form I used % wildcards representing zero or more characters (line 91). For search for licence it will require the exact characters and only limited to 16 characters in database and form.

In vehicle_search.php the sql statement involves joining three tables (People, Vehicles, Ownership) but since you must account for null values in either People or Vehicles, I have used union to use both right join and left join. The sql query returns the plate number, details of car, person name and licence number. The submitForm() javascript function will remind you with a pop up to fill in the fields in order to submit the form.

The add_vehicle.php allows user to add a vehicle and/or add a person to that new vehicle entry. This involves first an Inserting query into the Vehicle table, once this has been executed a select query will find the vehicle ID of that newly submitted vehicle and store the ID in a session variable called ‘vID’. If the user decides to add a person from the table of existing people in the database then the session variable ‘vID’ along with the people ID from the button using $GET will be inserted into the Ownership table. The user can also choose directly add a person in the add_person.php and after doing this the session variable ‘licence’defined in add_vehicle will get session ‘vID’ which will the automatically add the newly added person to the newly added vehicle into the ownership table. In the add_vehicle.php page after pressing add a JS pop up will ask “Are you sure?” and then a message saying “Person Added to Vehicle”.

The incident.php page uses a form structure as well and also a drop down menu. This is achieved using SELECT * FROM table statement in sql with <select> and <option> html tags interleaving php and html to create dropdown results based on current values I the database. Once the values are set in the form they are sent to $POST, the dropdown menu’s will send their ID of the respective table the select statement used and all of this is inserted into the Incident table to create a new incident (line 144).

The search_incident.php uses driving licence number or vehicle registration/plate number as you may have people with the same names and vehicles that are the exact same. User can enter either, so if driving licence number is entered a select statement for the People table will find People_ID and if vehicle reg number a select statement for vehicle table will find Vehicle_ID. Then using their respective ID’s the correct incident report will be found and displayed in table format as well as an extra button called edit (line 129) which allows the user to press and edit the incident report.

Pressing this button will execute a JS function which will store the incident ID of the chosen row will (through $_GET) will be stored in a $_SESSION[‘I_ID’] and then redirected to edit_incident.php where a similar format to incident.php except the form will be prefilled with the existing results of that incident report pulled from the database. This is achieved with an extra if-else statement within the while loop of the select html tag on line 80, 102, 128 where if the value is equal to the value of the chosen edited incident form chosen by the user a “selected” key word will be placed making it default → this is for the drop down menus:

```
echo “<option selected value=”.$row…
```

The existing date is displayed using the value attribute in the opening input tag and the existing report is displayed with a php script between the opening and closing tags of textarea. The edit_incident.php will use an Update tablename

SET query to modify the specified values in the table. There will also be javascript that pops up asking if you are sure you would like to edit. Then the user can press the reload link to see updated results.

The create_accounts.php uses INSERT with inputed username and password and displays all the user accounts on the system with a SELECT statement ordered by ID (line 87)

The add_fines.php has two input fields and a dropdown menu of incidents to match it with. An INSERT statement into Fines table is used with two input text fields and drop down html menu.

The logout.php does not display any html or css and simply unsets all the session variable, destroys the session and return the user to the login.php page.

The change_password.php allows the user to change their password and ask them to input their old password, new password and confirm their new password. The sql query will involve using UPDATE Users SET Password = inputted_text WHERE the username is equal to $_SESSION[‘log’] username so it changes the password of the right account. A message saying password change will be displayed on completion.

The images folder contains all the background images used throughout the website.