# Technical Manual

## Installation and Setup:

1. Import the SQL file into your MySQL server to set up the database structure and initial data.
2. Update the “db_connect.php” file with your MySQL server details (hostname, username, password, and database name).

## System Structure:

The system is structured around a Model View Controller (MVC) architecture. The main components are:

1. Database (model): The database consists of several tables including “Users”, “People”, “Vehicle”, “Incident”, “Offence”, “Fines”, “Ownership”, and “Audit”. The relationships between these tables are illustrated in the Entity Relationship Diagram (ERD) below.
2. PHP scripts (controller): The PHP scripts handle the business logic of the system. They interact with the databases to retrieve, insert, update, and delete data.
3. HTML pages (view): The HTML pages present the data to the user and provide forms for user input.

## Main Components:

**User Authentication:** The “login.php” and “logout.php” scripts handle user authentication. User roles are stored in the “Users” table and are used to control access to certain pages.

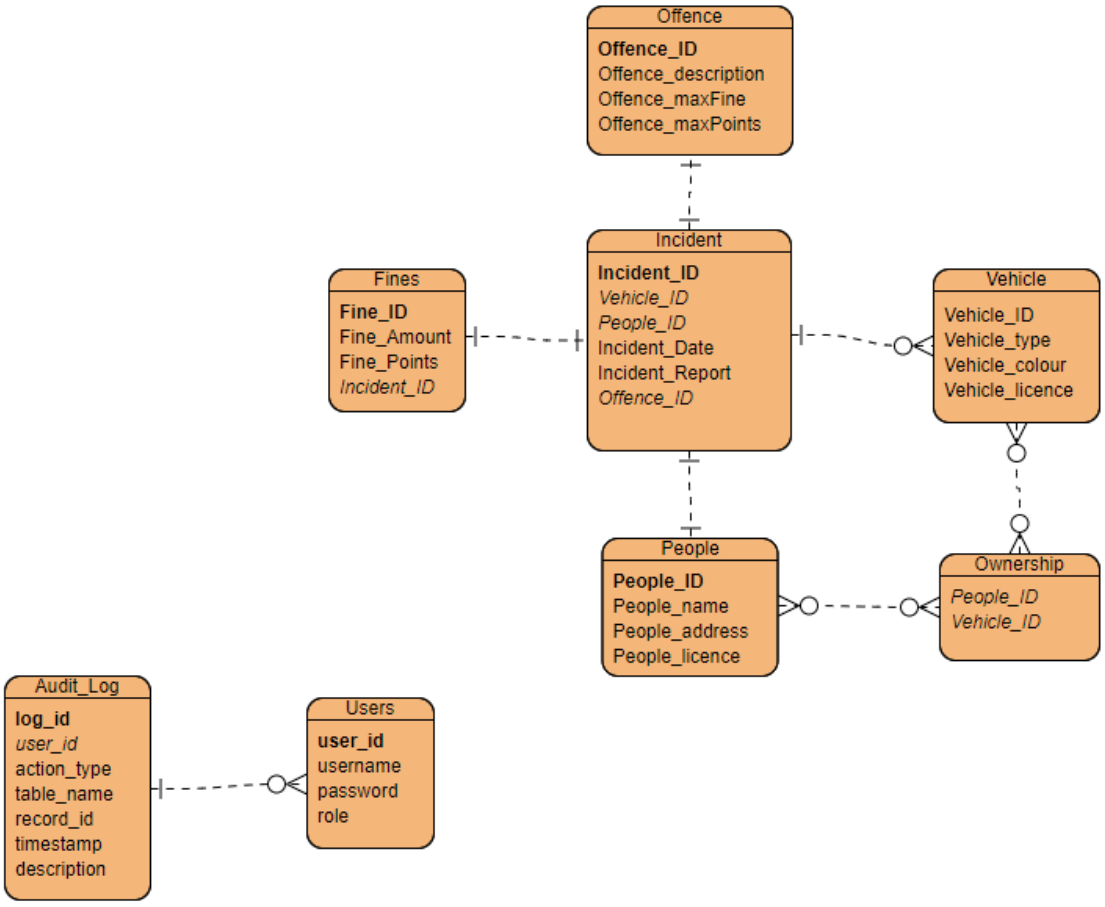
**Audit Logging:** The “audit_log.php” script logs all actions performed in the system. The logs are stored in the “Audit” table.

**Incident Reporting:** The “report_incident.php” script allows users to report incidents. Incidents are stored in the “Incidents” table and can be associated with fines.

**Search Functionality:** The “search_people.php” and “search_incident.php” scripts provide search functionality. They use SQL LIKE queries to search the “People” and “Incident” tables to return records based on a partial match.

**Vehicle Management:** The “add_vehicle.php” script allows users to add vehicles to the “Vehicle” table. If the owner does not exist in the “People” table, a new owner is added.

## Entity Relationship Diagram (ERD):




Primary Keys are shown in bold. Foreign Keys are shown in italic.

## Design Rationale:

The design is kept relatively simple and modular to ensure easy understanding and scalability. The login system is separated from the database connection logic (db_connect.php) to maintain a separation of concerns.

This approach ensures that different aspects of the system, like the navigation bar, database connections, and audit logging, are independently manageable yet cohesively integrated. Consistent use of SQL queries for database interactions and a uniform approach to CSS and HTML for presenting the data further streamline the system's functionality.

Error handling was conducted through conditional statements and database connection checks. If an operation fails (e.g., a SQL query), an error message is stored in a variable and is then displayed to the user. Storing a message in a variable allows for CSS to be used easily to manipulate the appearance of the error (in most cases it displays it in a red block).