Technical Manual

1.Introduction

This project aims to develop a web-based Traffic Management System, focusing on vehicle and incident management, log tracking, and officer administration. The system interacts with a MySQL database using PHP, allowing users to submit data through forms and perform various database operations.

2. Installation Instructions

Assuming the target environment has MySQL and a web server that supports PHP (such as Apache or Nginx) installed. The content includes:

Docker deployment:

a. Configure the docker-compose.yml file: Navigate to the directory containing the docker-compose.yml file (e.g., the alymh23_20618274_Docker folder) in a terminal.

b. Start command: docker-compose up

c. Visit the browser address: http://localhost/cw2/index.php

d. Configure the db.inc.php file: Modify the database configuration (such as server name, username, password, database name) to ensure it is consistent with the actual environment:

db.inc.php

html > cw2 > admin > db.inc.php > ...

```php
<?php
require './db.php';
$configuration = [
    'servername' => 'mariadb',
    'username' => 'root',
    'password' => 'rootpwd',
    'dbname' => 'coursework2'
];

return new DB(
    $configuration['servername'],
    $configuration['username'],
    $configuration['password'],
    $configuration['dbname'],
    3306
);
```

3. System Architecture Description

a. Overview：

User Interface Layer (HTML, CSS, PHP Front-End)

Provides interactive pages such as login, query, and operation log. Users submit requests via a browser.

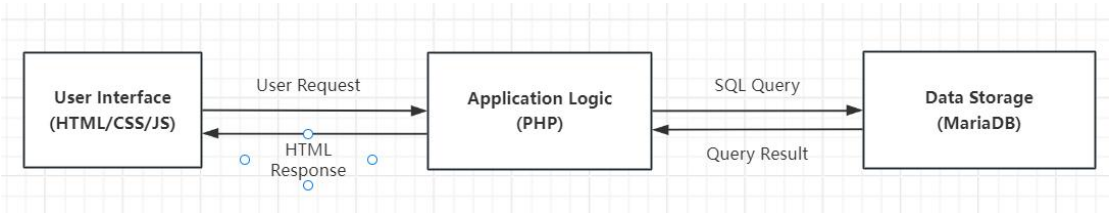
Application Logic Layer (PHP Back-End)

Handles user requests with scripts like login.php:Receives form inputs. Processes logic and interacts with the database. Generates dynamic HTML responses.

Data Storage Layer (MariaDB Database)

Initializes and stores data(e.g.users, vehicles, incidents)using coursework2.sql. Backend interacts with the database via db.inc.php.

b. Architecture Diagram：



c. PHP Components：

(1) Login

The system starts at the login page, where users submit their username and password via POST to login.php. The script connects to the database using db.inc.php credentials and validates the input against the admin table. If the credentials are correct, a session is started, and the user is redirected to index.php.

(2) Search People

The search input is transmitted to SearchPeople.php via a GET request. The input is dynamically appended to an SQL query that searches the people table for matching records. The results, including name and associated data, are displayed on the page. This functionality allows searching by partial or full matches.

(3) Search Vehicle

The search term for vehicle license plates is sent to searchVehicle.php via GET. It is used in an SQL query to search the vehicles table. Matching results, including owner and license details, are displayed.

(4) Add Vehicle

Form data from the create vehicle form is sent via POST to addVehicle.php. The data is validated and inserted into the vehicles table using an SQL INSERT statement. Upon success, the system notifies the user and redirects to a confirmation page.

(5) Report Incidents

Incident data is sent via POST to report.php, which formats the data into an SQL INSERT statement for storing in the incidents table. After successful insertion, the system confirms the operation and may redirect to a summary of reported incidents.

(6) Search Incidents

A search term (incident id) is sent via GET to searchIncident.php, where it is used in an SQL query to search the incidents table. The matching records are displayed in a table, showing details like date, time, location, and involved vehicles.

(7) Officer List

The create form data is sent to officer.php. The form data is then combined

with the SQL statement, and the data is inserted into the database. Finally, the officer list is displayed on the page.

(8) Operate Log

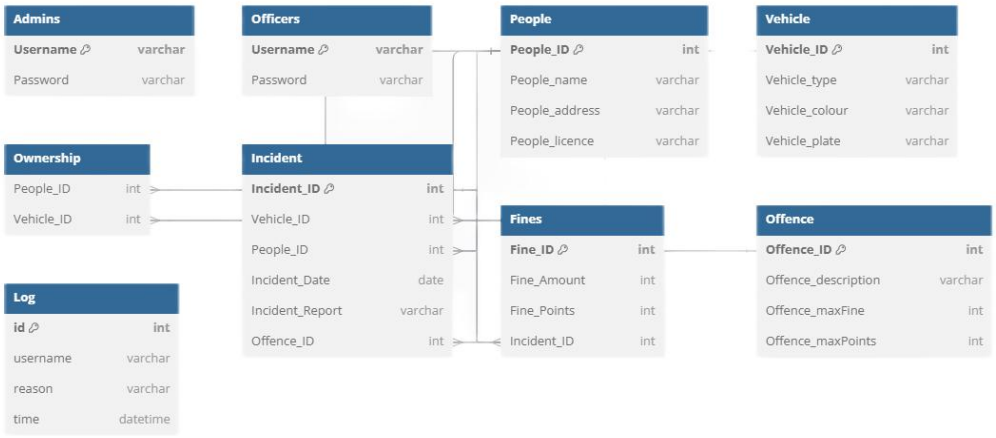
Query the database for all security logs from the logs table in operateLog.php The log records are then retrieved and displayed on the page.

d. Design rationale:

The system uses a modular three-tier architecture (interface, logic, data) for improved maintainability, scalability, and reduced coupling. It ensures data consistency and query efficiency with a standardized database structure, while prioritizing user experience and security through form validation and permission control.

4. Database Design

a. E-R Diagram



b. Relationships and Cardinality Ratios

Ownership (People - Vehicle): One person can own multiple vehicles, but each vehicle is owned by one person (1:N).

Incident - People: A person can be involved in multiple incidents, with each incident involving one person (1:N).

Incident - Vehicle: A vehicle can be involved in multiple incidents, with each incident involving one vehicle (1:N).

Incident - Offence: One offence can be linked to multiple incidents, but each incident is tied to one offence (1:N).

Fines - Incident: Each fine corresponds to exactly one incident, and each incident has one fine (1:1).

Log - Admins/Officers: Admins/officers can have multiple log entries, with each log entry tied to one admin or officer (1:N).