

Audit_Fines		
PK	Audit_Fines_ID	INT(11) AUTO_INCREMENT
	Timestamp	TIMESTAMP NOT NULL
	User_ID	INT(11) NOT NULL
	Action	CHAR(6) NOT NULL
	NEW_Fine_Amount	INT(11) NOT NULL
	NEW_Fine_Points	INT(11) NOT NULL
	Incident_Report	VARCHAR(500)

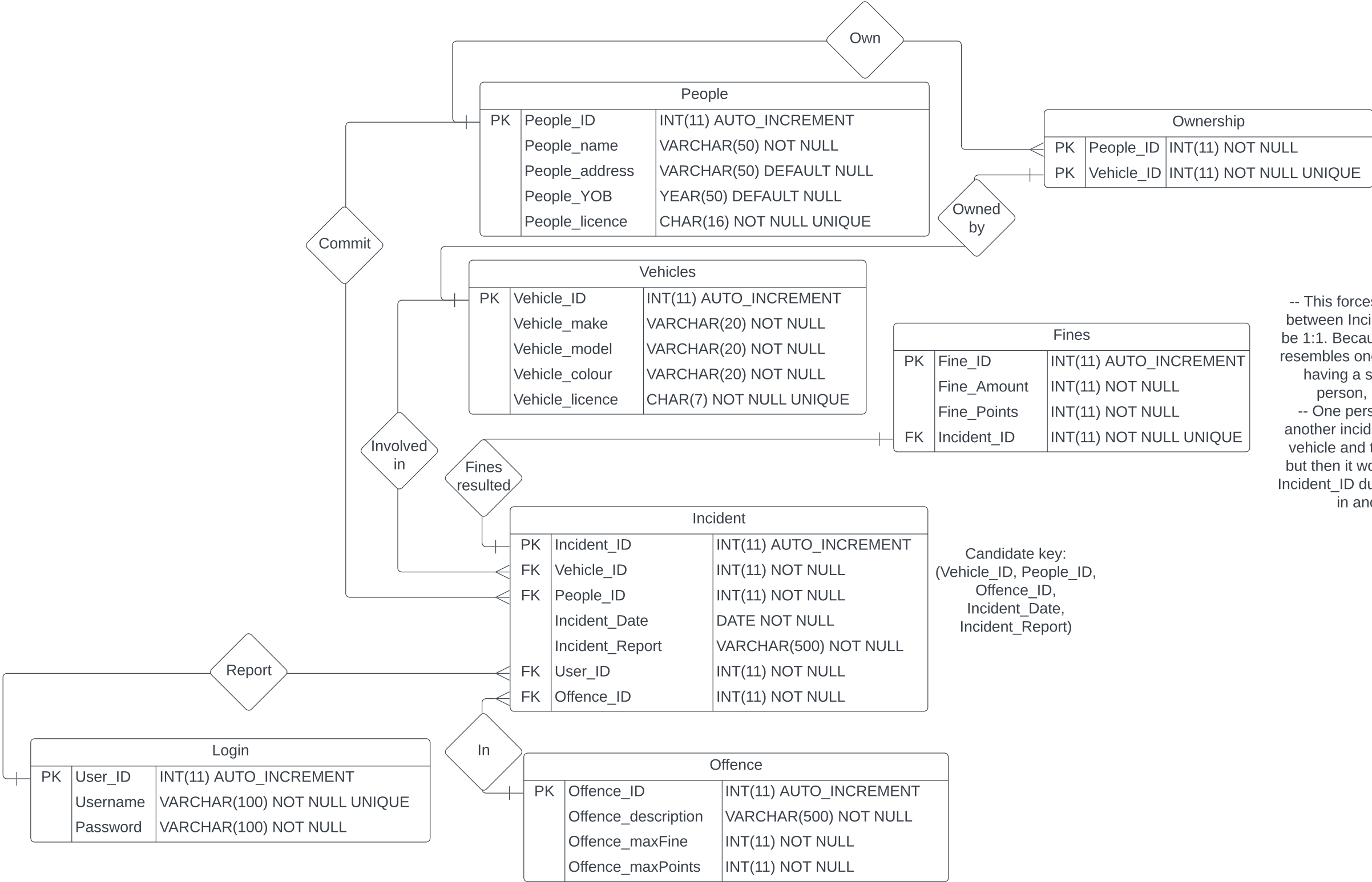
Audit_Vehicle		
PK	Audit_Vehicle_ID	INT(11) AUTO_INCREMENT
	Timestamp	TIMESTAMP NOT NULL
	User_ID	INT(11) NOT NULL
	Action	CHAR(6) NOT NULL
	OLD_Vehicle_make	VARCHAR(20)
	NEW_Vehicle_make	VARCHAR(20)
	OLD_Vehicle_model	VARCHAR(20)
	NEW_Vehicle_model	VARCHAR(20)
	OLD_Vehicle_colour	VARCHAR(20)
	NEW_Vehicle_colour	VARCHAR(20)
	OLD_Vehicle_licence	VARCHAR(20)
	NEW_Vehicle_licence	VARCHAR(20)

Audit_Incident		
PK	Audit_Incident_ID	INT(11) AUTO_INCREMENT
	Timestamp	TIMESTAMP NOT NULL
	User_ID	INT(11) NOT NULL
	Action	CHAR(6) NOT NULL
	OLD_Incident_Report	VARCHAR(500)
	NEW_Incident_Report	VARCHAR(500)
	OLD_Incident_Date	DATE
	NEW_Incident_Date	DATE
	OLD_Vehicle_licence	CHAR(7)
	NEW_Vehicle_licence	CHAR(7)
	OLD_Owner_licence	CHAR(16)
	NEW_Owner_licence	CHAR(16)
	OLD_Offence_description	VARCHAR(500)
	NEW_Offence_description	VARCHAR(500)

Audit_Ownership		
PK	Audit_Ownership_ID	INT(11) AUTO_INCREMENT
	Timestamp	TIMESTAMP NOT NULL
	User_ID	INT(11) NOT NULL
	Action	CHAR(6) NOT NULL
	Owner_licence	CHAR(16)
	Vehicle_licence	CHAR(7)

Audit_Login		
PK	Audit_Login_ID	INT(11) AUTO_INCREMENT
	Timestamp	TIMESTAMP NOT NULL
	User_ID	INT(11) NOT NULL
	Action	CHAR(6) NOT NULL
	OLD_username	VARCHAR(100)
	NEW_username	VARCHAR(100)
	OLD_password	VARCHAR(100)
	NEW_password	VARCHAR(100)

Audit_Owner		
PK	Audit_Owner_ID	INT(11) AUTO_INCREMENT
	Timestamp	TIMESTAMP NOT NULL
	User_ID	INT(11) NOT NULL
	Action	CHAR(6) NOT NULL
	OLD_owner_name	VARCHAR(50)
	NEW_owner_name	VARCHAR(50)
	OLD_owner_address	VARCHAR(50)
	NEW_owner_address	VARCHAR(50)
	OLD_owner_YOB	YEAR(4)
	NEW_owner_YOB	YEAR(4)
	OLD_owner_licence	CHAR(16)
	NEW_owner_licence	CHAR(16)



-- This forces the relationship between Incident and Fines to be 1:1. Because an Incident_ID resembles one specific situation having a specific vehicle, person, offence, etc..
 -- One person could commit another incident with the same vehicle and the same offence but then it would have another Incident_ID due to being present in another date

Police Workforce Database

→ Guide to PHP file linkage

Starting
point

index.php

Homepage



main.php

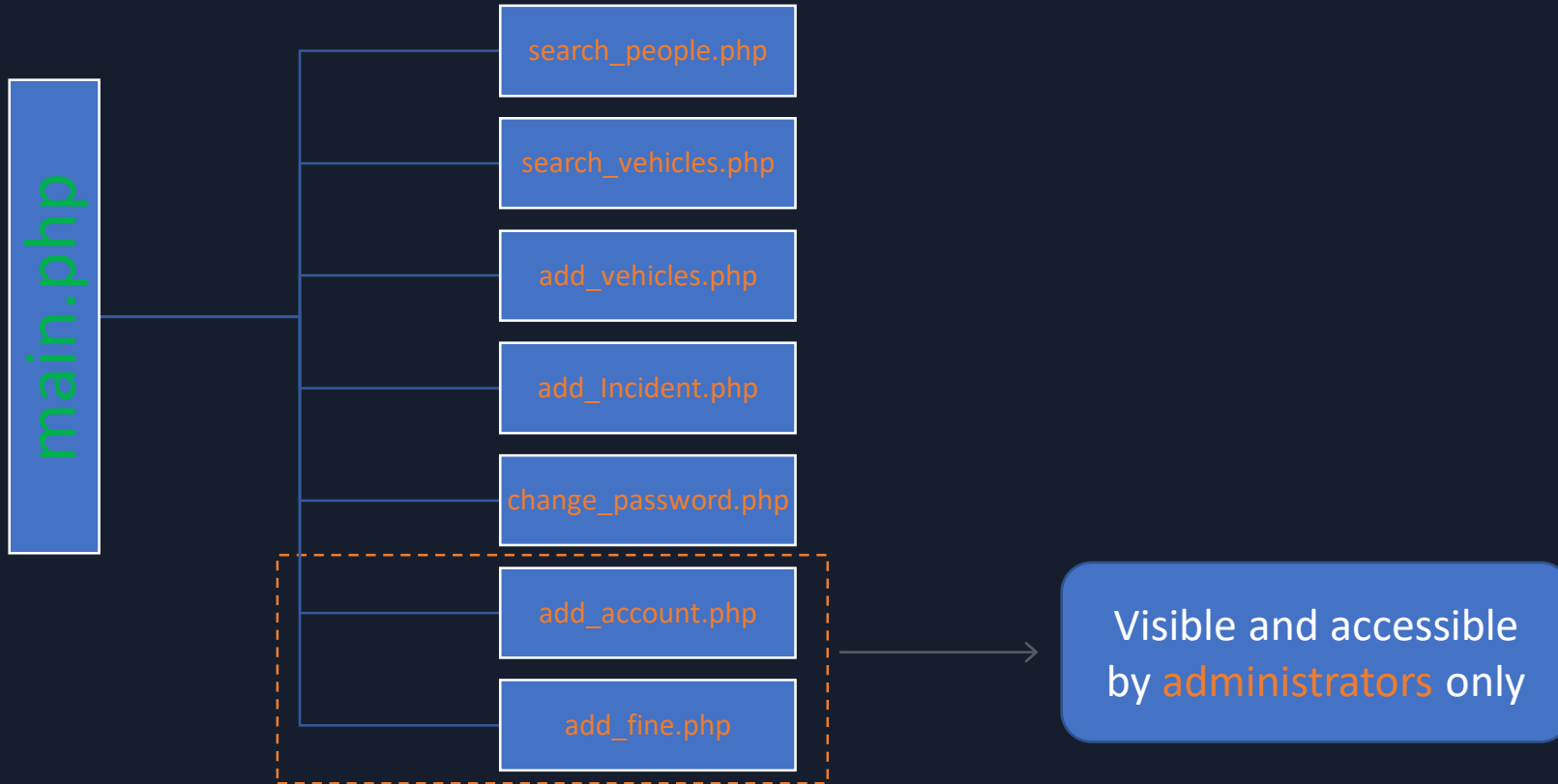
Log out



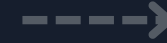
index.php

Starting
point

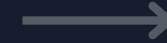
index.php



Processing in same file



Conditional redirect



Unconditional redirect



Present in database







Absent in database


search_people.php


add_account.php


search_vehicles.php


add_fine.php  add_fine.php?add=ID




add_vehicles.php

Vehicle licence, owner licence ✓ ✓
Vehicle licence, owner licence ✓ ✗
Vehicle licence, owner licence ✗ ✓

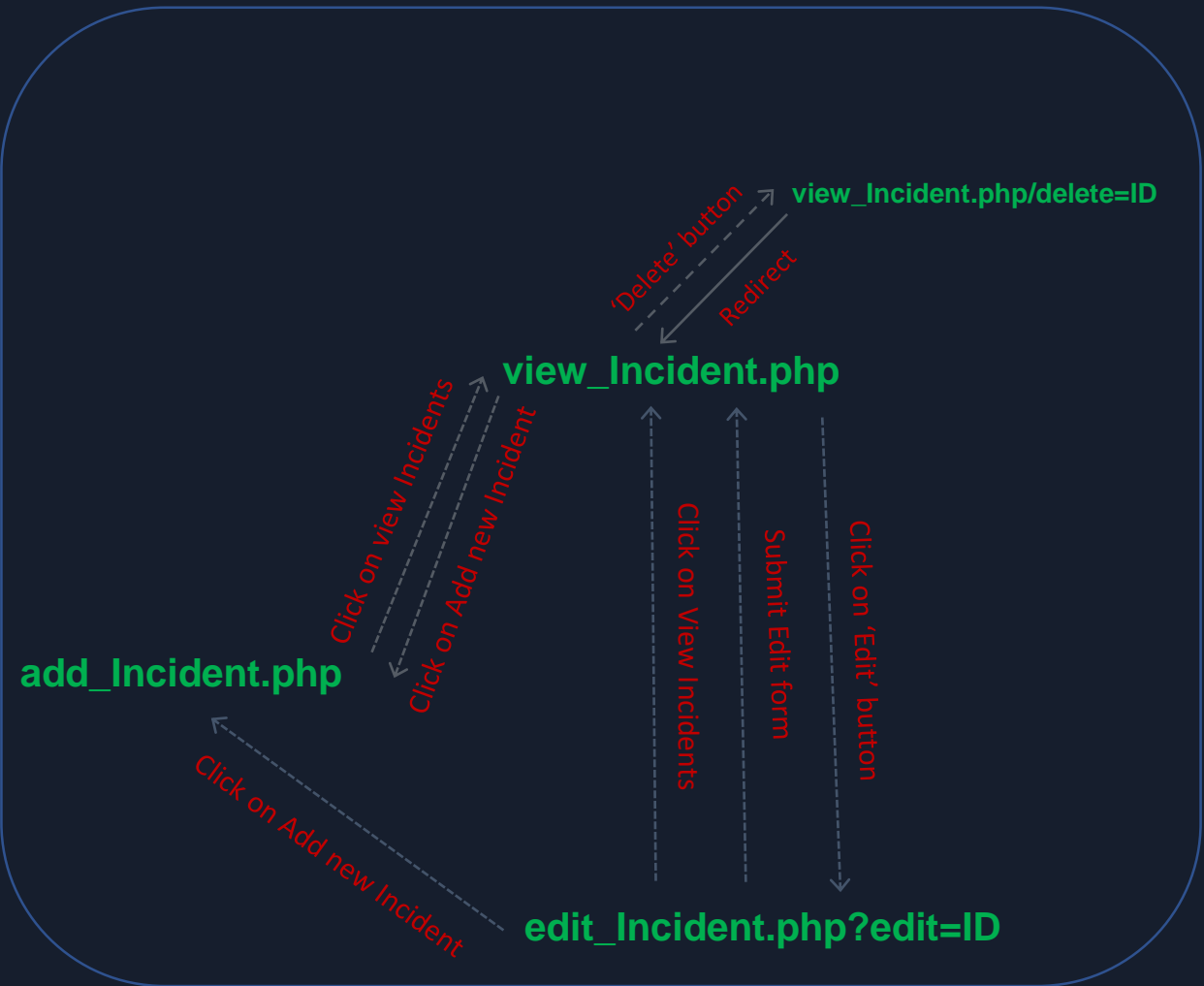
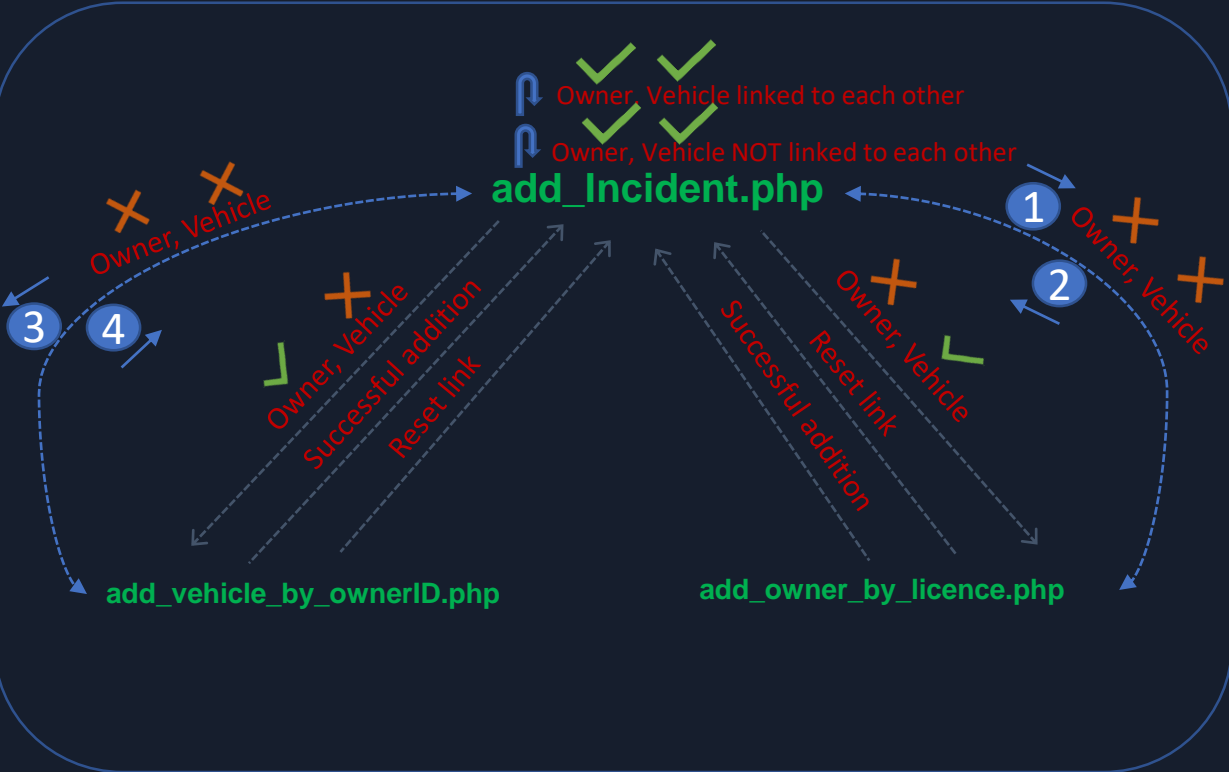
Error
Vehicle already present

owner licence, vehicle licence ✗ ✗

 Submit form


add_owner.php


change_password.php



- Every webpage check if the user is logged in or not which prevents anyone having the URL from accessing the website if he/she doesn't have proper credentials.
- Most of the webpages has homepage and Logout for easier navigation

Index.php ->

- 1) Before login: Contains the actual login form.
- 2) Login failure: produces an invalid credentials error and prompt for another credentials
- 3) After login: Login form disappears and a message displaying the username appears associated with options to:
 1. Go to [Homepage](#)
 2. [Logout](#) -> Redirects to [index.php](#) while removing session variables
 3. [Change password](#)
- 4) Most important SQL queries:
 1. Making sure there is a record with the same username and password in the database
 - `SELECT * FROM Login WHERE username={} AND password={}`

main.php ->

- 1) Any [Homepage](#) link directs the user to this php file
- 2) Contains all the available options for the specified user
 1. [Administrator users only](#) get additional access such as:
 - i. [Creating new police officers accounts](#)
 - ii. [Adding fines](#)
 2. This access is granted only if the username is an administrator username which will then show these available options to the administrator beside the basic privileges like any user
 3. These files are:
 - i. [add_account.php](#)
 - ii. [add_fine.php](#)
 4. A normal user won't see those options and even if he/she tried to access it directly using their URL, it will show a '[Restricted access!](#)' error and prevents them from altering anything
- 3) [Added feature](#): Every option listed for the user has a [number](#) beside it which dynamically resembles the number of appearances in the database.
 1. For example, [Search vehicles](#) option has the number 10 beside which represents the number of available vehicles in the database.
- 4) Most important SQL queries:
 1. `SELECT DISTINCT COUNT(*) 'COUNT' FROM People`
 2. `SELECT DISTINCT COUNT(*) 'COUNT' FROM Vehicle`
 3. .. These commands keep going for other tables in order to count the number of available people, vehicles, etc..

search_people.php ->

- 1) This file has a form inside it containing '[Name](#)' and '[Licence](#)'. The user could [either enter Name or Licence](#) and it will show the available information relating to that/those user/users in the database.

1. The code picks up whether the Name or the Licence has been entered and act upon it.
2. If 'Name', whether full or partial, and 'Licence' are both entered, then the priority will be given to the name.
- 2) Searching for a user who is not present in the database will produce an unavailable error and requests another input prompt to check.
- 3) User could search for partial names.
 1. For example, 'Jo' would search for any name which has 'Jo' as a part of its name like John and any similar names.
- 4) If the user searched for a specific name which is present multiple times in the database relating to different users, it will show all these similar users.
- 5) Most important SQL queries:
 1. Retrieving names similar to the input (partial names as well)
 - `SELECT * FROM People WHERE People_name LIKE "%{name}%"`
 2. Retrieving the specific owner's information using licence
 - `SELECT * FROM People WHERE People_licence = {owner_licence}`

search_vehicles.php ->

- 1) A form will appear where the user should enter the vehicle registration 'licence'.
- 2) It will search for the specified vehicle and list the vehicles' details along with its owner's name and licence.
 - If the owner of this vehicle is unknown, it will write Unknown beside the owner's information
- 3) Most important SQL queries:
- 4) Retrieving vehicle, owner information for the specific vehicle
 - `SELECT Vehicle_ID, Vehicle_make, Vehicle_model, Vehicle_colour, Vehicle_licence, People_name, People_licence FROM Vehicle LEFT JOIN Ownership USING (Vehicle_ID) LEFT JOIN People USING (People_ID) WHERE Vehicle_licence = {vehicle_licence}`

add_vehicles.php ->

- 1) A form will appear where the user should enter the new vehicle's information such as licence, make, model, colour and owner's licence.
 - i. All the previous attributes are NOT optional because all vehicle's attributes should be well-known. The input form would alert the user to enter all attributes before submitting the form if he/she forgot.
 - ii. Vehicle licence is validated first
- 2) When the form is submitted AND vehicle licence pass validation, there are different scenarios:
 - i. The owner's licence is present in the database
 - The vehicle will be added to the database
 - The Ownership table will have an additional row carrying the person's ID and the vehicle ID
 - If addition is successful, a message stating 'Vehicle added successfully!' would appear.
 - ii. The owner's licence is NOT present in the database

- User is redirected to **add_owner.php** where another form will appear requesting the user to submit details of the new owner. (Licence is validated first)
- A **new vehicle** is added
- A **new owner** is added
- A linkage between the vehicle and the owner is established in **Ownership** table
- iii. A **vehicle** with the **same licence** is **already present** in the database
 - A message stating **Vehicle already registered!** would be prompted.
 - Either an owner licence is present or not, **no additional rows** will be added in any of the tables.
- 3) Most important SQL queries:
 - Checking the existence of the owner in the database
 - **SELECT People_ID FROM People WHERE People_licence = {owner_licence}**

add_owner.php ->

- Most important SQL queries:
 - Inserting the new owner's information collected from form. Update with optional information
 - **INSERT INTO People (People_name, People_licence) VALUES ({name}, {licence})**
 - **UPDATE People SET People_address={address} WHERE People_licence = {licence}**
 - **UPDATE People SET People_YOB={YOB} WHERE People_licence = {licence}**

add_Incident.php ->

- A user could view the available Incidents by clicking on '**View Incidents**' or he/she could **add a new Incident** by filling in the available form
- If a user clicked on '**View Incidents**' he/she will be **redirected** to **view_Incident.php**
 - All the incidents are listed, and the user has got 2 options:
 - Delete an Incident**
 - A message will appear for the confirmation of deletion
 - If pressed **cancel**, nothing will happen
 - If pressed **Ok**, then the **specific incident will be deleted**
 - Edit an Incident**
 - The user will be redirected to **edit_Incident.php** where he/she could:
 - Choose 'Add new Incident' to be redirected back to **add_Incident.php**
 - Edit the specific Incident by **changing the values** of the Incident and pressing Edit which will update the Incident's information and redirect back to **view_Incident.php**
- Form includes **statement, date of incident, vehicle licence, owner's licence, offence type**
 - All the above fields are required to be used in court later
- Addition of a new Incident include the following scenarios:
 - A user **specify whether the owner is the driver or not**, because it will have an effect on **whether to add a link in Ownership table or not**.
 - Vehicle licence and owner licence **are present** and **there IS connection linking the vehicle** with the **owner** in **Ownership table**
 - The **new Incident will be added ONLY** without further additions in any of the tables.

- iii. Vehicle licence and owner licence are present and there is NO connection linking the vehicle with the owner in Ownership table
 - 1. New entry in Ownership table specifying the connection between the owner and the vehicle (If driver is owner).
 - 2. New entry in the Incident table specifying Incident details
- iv. Vehicle licence is NOT present in the database, but Owner IS present
 - 1. Request the new vehicle details from the user by redirecting him/her to `add_vehicle_by_ownerID.php` where on submitting the new vehicle's details:
 - a. There is a Reset link which will reset all the session variables carrying the Incident information submitted from the first Incident form, in order not to forget it, stored and redirect to `add_Incident.php`
 - b. New entry into Vehicle table with the new vehicle information
 - c. New entry into Ownership table linking the new vehicle ID with the owner's ID (If driver is owner)
 - d. New entry in the Incident table specifying Incident details
- v. Vehicle licence IS present in the database, but Owner IS NOT present
 - 1. Request the new owner's details from the user by redirecting him/her to `add_owner_by_licence.php` where on submitting the new owner's details:
 - a. There is a Reset link which will reset all the session variables carrying the Incident information submitted from the first Incident form, in order not to forget it, stored and redirect to `add_Incident.php`
 - b. New entry into Owner table with the new owner's information
 - c. New entry into Ownership table linking the new owner with the new vehicle (If driver is owner)
 - d. New entry in the Incident table specifying Incident details
- vi. BOTH Vehicle licence and Owner licence are NOT present
 - 1. Request the new owner's details from the user by redirecting him/her to `add_owner_by_licence.php` where on submitting the new owner's details:
 - a. New entry into Owner table with the new owner's information
 - 2. Request the new vehicle details from the user by redirecting him/her to `add_vehicle_by_ownerID.php` where on submitting the new vehicle's details:
 - a. New entry into Vehicle table with the new vehicle information
 - b. New entry into Ownership table linking the new owner with the new vehicle
 - c. New entry in the Incident table specifying Incident details
 - d. Redirection back to `add_Incident.php` while showing 'Incident Added Successfully'

- 3) Most important SQL queries:
 - i. Linking owner to vehicle, if not.
 - a. `INSERT INTO Ownership (Vehicle_ID, People_ID) VALUES ({vehicle_ID}, {owner_ID})`
 - ii. Inserting a new Incident
 - a. `INSERT INTO Incident(Vehicle_ID, People_ID, Incdient_Date, Incident_Report, USER_ID, Offence_ID) VALUES ({vehicle_ID}, {owner_ID}, {date}, {Report}, {reporter_ID}, {offence_ID})`

`add_vehicle_by_ownerID.php ->`

- 1) Most important SQL queries:
 - a. Inserting a new vehicle
 - i. `INSERT INTO Vehicle (Vehicle_make, Vehicle_model, Vehicle_colour, Vehicle_licence) VALUES ({make}, {model}, {colour}, {vehicle_licence})`

`add_owner_by_licence.php ->`

- 1) Most important SQL queries -> similar to `add_owner.php`

`add_account.php ->`

- 1) A form will be prompt to ask the `administrator` to insert a new username and password
 - a. Username should be unique, if not, then a message will prompt stating that the 'Username is used before'.
 - b. On successful addition of the new user, a message will prompt stating 'Successfully added new credentials'
 - c. New `username` and `passwords` are stored in `Login table`
- 2) Most important SQL queries:
 - a. Inserting new credentials
 - i. `INSERT INTO LOGIN (Username, Password) VALUES ({username}, {password})`

`add_fine.php ->`

- 1) A form will be prompt asking the user to enter `owner's licence` and `vehicle licence` associated in that Incident
- 2) If there is no Incident linking owner's licence and vehicle licence together, a message will be prompt stating that 'There are no Incidents linking the owner with the vehicle!'.
- 3) If there is `EXACTLY ONE Incident` or `MULTIPLE Incidents present`, the user will have the option to choose which Incident he/she wants to issue fine on.
 - a. Once the Incident is chosen, a `form` will appear to `fill in the fine amount` and `fine points` needed to be added.
 - b. On successful addition, a message will state 'Added fine record successfully!'
 - c. The user could then navigate to `homepage` or fill in another fill in another fine which will be repeated from step 1.
 - d. Adding the fine could fail if the same user or another user added a fine to the same incident before
- 4) Most important SQL queries:
 - a. Listing Incidents related to owner and vehicle to choose from

- `SELECT * FROM Incident WHERE People_ID={owner_ID} AND Vehicle_ID={vehicle_ID}`
- b. Inserting the new fine
 - `INSERT INTO Fines (Fine_Amount, Fine_points, Incident_ID) VALUES ({fine_amount}, {fine_points}, {Incident_ID})`
- c. Retrieve Incident report
 - `SELECT Incident_Report FROM Incident WHERE Incident_ID={Incident_ID}`

`change_password.php` ->

- 1) The user would be required to submit the form including `old password` and `new password`
 - a. If old password `does not match` the user's old password, a message will show stating that 'Old Password is Incorrect!'
 - b. If old password `matches` the user's old password, a message will show stating that password successfully changed
 - The user then will be redirected to the homepage automatically
- 2) Most important SQL queries:
 - a. Retrieving old username, password information to cross-check
 - `SELECT * FROM Login WHERE User_ID={reporter_ID}`
 - b. Updating account with new password
 - `UPDATE Login SET Password={password} WHERE User_ID={reporter_ID}`

`functions.php` ->

- 1) `check_insert_vehicles()`
 - a. Mentioned above.
- 2) `get_licences_from_ID()`
 - a. This function is responsible to retrieve User licence, Vehicle licence and optionally offence description provided that the inputs are user ID, vehicle ID and optionally offence ID.
 - b. This is done instead of joining all the tables on the foreign keys to maintain scalability.
- 3) `reset_session_variables()`
 - a. Resets all stored session variables except username and userID.
 - b. Sometimes when multiple pages are called, session variables stores valuable information related to the first submitted form. When we no longer want those variables, this function is called
- 4) `check_vehicle_presence()`
 - a. Checks if the vehicle is present in the database

AUDIT TABLES OUTLINE for `every` audit table:

- 1) INSERTING:
 - Action: Insert
 - Old_column/s: NULL
 - New_column/s: new values

- **Exception in Auidt_Ownership** table where it only has insert operations, so it doesn't have OLD, NEW. Only the name of the columns
- 2) Updating:
- Action: Update
 - Old_column/s: old values (updated columns only)
 - New_column/s: new values (updated columns only)
- 3) Deleting:
- Action: Delete
 - Old_column/s: old values
 - New_column/s: NULL
- 4) Search:
- Action: Search
 - Old_column/s: old values
 - New_column/s: NULL

Database connection attributes are found in `psxke1-20486945_InstallationFiles/db.inc.php`

CSS files -> `psxke1-20486945_InstallationFiles/style.css`

Javascript files -> `psxke1-20486945_InstallationFiles/script.js`

PHP files -> `psxke1-20486945_InstallationFiles/*.php`

- Table structure is found in the ER diagram.
- Important notes on table structure:
 1. Ownership table is a separate table holding people_ID and vehicle_ID.
 - Relationship between people and ownership is 1:M
 - Relationship between vehicle and ownership is 1:1
 - This is done by **making vehicle_ID**, which is a foreign key, **a unique column in Ownership table** where it forces the relationship between Vehicle and Ownership to be 1:1. **Because a vehicle cannot be owned by multiple owners**
 - We could have put people_ID as a foreign key directly inside Vehicle table because it's a 1:M relationship, however, we made another ownership table to **avoid insert anomaly**. Because if we did so, we couldn't add a vehicle to the database without its owner.
 2. A candidate key in Incident table consisting of ('Vehicle_ID', 'People_ID', 'Offence_ID', 'Incident_Date', 'Incident_Report') is added to force any user NOT to insert the SAME exact incident again. (**UNIQUENESS**)
 3. Fines table has a unique key (Incident_ID) to forces the relationship between Incident and Fines to be 1:1. Because an Incident_ID resembles one specific situation having a specific vehicle, person, offence, etc.. One person could commit another incident with the same vehicle and the same offence but then it would have another Incident_ID due to being present in another date. Again, Fines are separated in another table even though it has a 1:1 relationship, mainly to avoid Insert anomaly

where we would like to insert an incident without adding the fine at the same moment.

