

COMP1004 Coursework

Overview

For this coursework we will use the same database as the one used for the SQL quiz. You will build a front end using HTML/CSS/JavaScript that connects to the database and allows users to query and update the database.

Specification

Front end

The front end must be implemented using HTML, CSS, and JavaScript. No library or framework is allowed such as Bootstrap (CSS) or React (JavaScript). No other language is allowed, either, such as PHP or Python. The front end must be able to communicate with the back end database hosted on [Supabase](#) (more details below).

Back end

The back end database must use [Supabase](#), which provides an online PostgreSQL database (relational).

Access to the database must be through the REST API generated by the Supabase. Connecting directly to the database or creating your own REST API is not allowed.

Access to Supabase must use [its JavaScript client](#). No other library is allowed.

You must use services such as [cron-job.org](#) to stop your database from pausing (Supabase pauses a database after 7 days of inactivity). Please see the lecture on Supabase for details. If Supabase is paused, it will not be possible to mark the database part of the coursework, and you will lose the mark for this part.

Database

This coursework will use a simplified version of the database used for the module quiz with only two tables: [People](#) and [Vehicle](#). The files are in the attached zip file in SQLite ([.db](#)) and CSV format. There are a few ways to import the data into Supabase. Please see [this page](#) for details.

The database tables and records must be kept as they are. You need to restore the database to its original state when submitting the work if changes are made during development. Any change would cause error during marking and lead to loss of marks.

Submission

Only one zip file will be accepted for submission.

The zip file must include all the files for the front end, i.e., HTML, CSS, JavaScript, image, and other files if there is any. Do not include any files for the back end or database.

The zip file must also include a markdown or PDF file (no Word file) that has a description of the additional work for HTML, CSS, JavaScript, and/or database.

- Only the additional work that is described in this file will be considered for mark.
- Please say so in this document if you don't attempt any of the additional work.
- Please see the marking rubrics below for the details about additional work.
- Please describe separately for each aspect, i.e., HTML/CSS/JavaScript/database, what the additional work is, and where they are (file name and line number).
-

For HTML, please include:

A screenshot of 100% accessibility score for the 'add vehicle' page from the browser developer tool. 100% accessibility needs to be achieved on all pages, but only one screenshot is needed.

- A description of the changes you made to improve the accessibility score and where they are in the code file (in which file from which line to which line).
- For CSS, please include:
 - A screenshot of the webpage when the width is less than 500px.
 - A description of the code you add to achieve responsive layout and where they are in the code file (in which file from which line to which line).
- For JavaScript and database, for each feature tests are created for, such as filling out the vehicle information, please include:
 - A list of the conditions tested: always include at least one test that everything is performed correctly and include as many exceptions as you can.
 - For exceptions, these could be one or more of the fields are missing, wrong type of information is entered (e.g., number used for colour), etc.
 - Where the tests are in the code file (in which file from which line to which line).
 - Full mark for this part would require tests covering all major exception types.
- Please also include in this file any issue you are aware of about playwright testing from using the sample test file. Please include the reason and evidence why you think this is a playwright issue and not your code. You can use screenshots or images if useful. Please see the last section about Playwright testing.

Marking

Rubric

The total coursework mark is 100%. Meeting all the 'criteria' (below the rubrics table) will get you 80% for each aspect (html/csss/js/db)/. To achieve full mark, you may need to go beyond what is covered in the lecture. Please see the last row of the rubric table for details.

The marking is based on the end result, e.g., whether a required feature is available and working as prescribed. You are free to choose how it is implemented so long as it meets all the requirements, e.g., not using an external library.

Marks	HTML (20)	CSS (20)	JavaScript (40)	Database (20)
0	No HTML code is provided.	No CSS code is provided.	No JavaScript code is provided.	No database is provided.
Marks	HTML (20)	CSS (20)	JavaScript (40)	Database (20)
5 (10 for JavaScript)	Meet some of the HTML criteria	Meet some of the CSS criteria	Meet some of the JavaScript criteria	A database is provided but there are many errors.
10 (20 for JavaScript)	Meet most of the HTML criteria	Meet most of the CSS criteria	Meet most of the JavaScript criteria	A database is provided but there are a few errors.

15 (30 for JavaScript)	Meet all the HTML criteria	Meet all the CSS criteria	Meet all the JavaScript criteria	A database is provided with no error and database query and update perform as required.
20 (40 for JavaScript)	Meet all the HTML criteria and achieve 100% score in the Lighthouse test for all the pages (more details below).	Meet all the CSS criteria and the layout is responsive, different screen size (more details below).	Meet all the JavaScript criteria and playwright tests are created for all the features and exceptions database criteria ('add vehicle'), such as owner exists/does not exist in the database, missing data, etc.	Meet all the database criteria and requirements are the same as those for JavaScript.

Criteria - HTML

- . There are at least three HTML pages, one for each database query/update.
- . An unordered list `` is used to create the navigation links.
- . All the pages have the same navigation menu and the links works correctly.
- . Each page must have four sections: header, main, sidebar, and footer. These are marked up correctly using semantic elements.
- . There is at least one image or video for each page in the side bar and all the required information is present and correct.
- . HTML forms are used for user to enter and perform database queries.
- . Accessibility: we will use the 'Lighthouse' tool in the browser development tools to meausure the webpage accessibility.
 - o For this requirement you only need to run the 'Accessibility' test.

Generate a Lighthouse report

Mode [Learn more](#)

Navigation (Default)

Timespan

Snapshot

Device

Mobile

Desktop

Categories

Performance

Accessibility

Best practices

SEO

Progressive Web App

Plugins

Publisher Ads

Analyze page load

It returns a score and what can be improved. A score of 100 is needed to get the mark for this criterion. A 100 score is required for all pages, including the dynamic content added during user interaction, such as the search results displayed after user enters a query.

Note: 100% score does not mean there is no accessibility issue. See the 'conducting an accessibility review' link on the report page for details.

13:56:08 - 127.0.0.1:3000

http://127.0.0.1:3000/index.html

93

Accessibility score

Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

What can be improved

NAMES AND LABELS

▲ Form elements do not have associated labels

These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.

There are other accessibility issues that can't be checked automatically.

ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Show

These items address areas which an automated testing tool cannot cover. Learn more in our guide on [conducting an accessibility review](#).

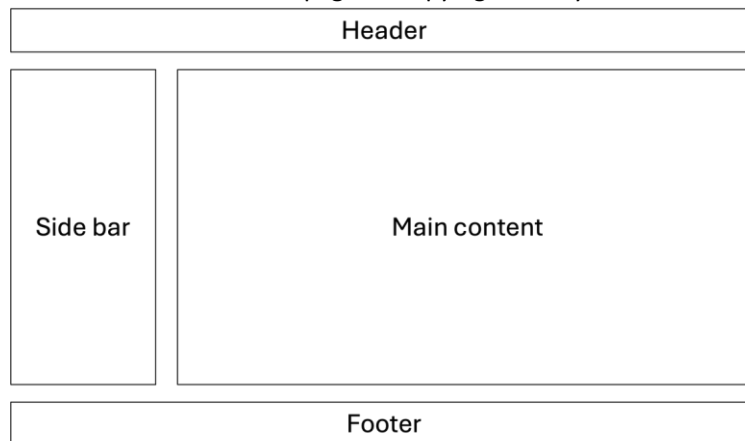
PASSED AUDITS (14)

Show

4 / 9


Criteria - CSS

- . All the pages must share the same external CSS file. Internal or inline CSS will lead to loss of mark unless these are justified. Please provide such justification as code comment.
- . CSS flex is used to place the navigation links horizontally.
 - Class is used to apply CSS flex to the navigation links only and not other `` element. The links
 - must use all the horizontal space.
- . Location selector is used to remove the bullet in front of the navigation links.
 - This must only remove the bullets from the navigation links and not any other unordered list items.
- . Add border, margin, and padding to header, main, sidebar, and footer.
 - A solid black border with width 1px.
 - Margin and padding on all sides with a value of 10px.
- . Use CSS Grid to layout the page (see the figures below):
 - The header must be at the top of a page, occupying the fully width;
 - The side bar must be to the left of the main content;
 - The width ratio between the side bar and the main content must be 1 4.
 - The footer must be at the bottom of a page, occupying the fully width.



People Search

[People search](#)[Vehicle search](#)[Add a vehicle](#)



Search by driver name

Search by driving license number

Submit

Search successful

personid: 1
name: Rachel Smith
address: Wollaton
dob: 1979-06-05
licensenum: SG345PQ
expirydate: 2020-05-05

personid: 2
name: Lewis Thomson
address: Nottingham
dob: 1949-01-15
licensenum: RW765FR
expirydate: 2018-03-25

personid: 3
name: Oliver Reps
address: Nottingham
dob: 1976-10-05
licensenum: JR123DE
expirydate: 2016-01-29

personid: 4
name: Daphne Lai
address: Leicester
dob: 1980-08-13
licensenum: DL890GB
expirydate: 2017-06-24

personid: 5
name: Rachel Johnson
address: London
dob: 2000-01-01
licensenum: JK239GB
expirydate: 2023-08-12

This is a sample for the COMP1004 coursework.

- . Responsive layout: you need to change the page layout to the one shown below when the page width is less than 500px using CSS media queries.
- . The links are now vertically stacked instead of horizontally placed.
- . The side bar is move under the main.
- . The width ratio between the side bar and the footer is 1 4.


6 / 9

People Search

[People search](#)
[Vehicle search](#)
[Add a vehicle](#)

Search by driver name

Search by driving license number



This is a sample for the COMP1004 coursework.

Criteria - JavaScript and database

- . People search: the user must be able to look up people by their names or their driving licence number (by typing either of these in to the system).
 - . If the person is not in the system it must give an appropriate error message.
 - . This search must not be case sensitive and it must work on partial names, e.g., 'Jo', 'Smith' and 'john smith' would all find 'John Smith'.
 - . If there are several people with the same name they must all be listed.
- . Vehicle search: the user must be able to look up vehicle using registration (plate) number.
 - . The system will then show all the details of the car (e.g., type, colour etc.), and the owner's name and license number.
 - . Allow for missing data in the system (e.g., the vehicle might not be in the system, or the vehicle might be in the system but the owner might be unknown).
- . Add vehicle: The user must be able to enter details for a new vehicle.
 - . This will include the registration (plate) number, make, model and colour of the vehicle, as well as its owner.
 - . A search will be performed once the owner name is entered.
 - . This search must allow partial match and be case insensitive just as before.
 - . If one or more matches are found, they will be listed and the user can select a person and assigning them to the new vehicle.
 - . If there is no match or none of the listed person is the intended owner, a new form must be displayed to enter the owner information.
 - . The new owner information must be checked, i.e., no missing information or identical to any existing owner, before saving to the database.

Testing with Playwright

Some of the functions will be tested with the Playwright library. There are some requirements needed to make this work (e.g., the output has a ID called `results`), so Playwright can find the right HTML element for the test. Not meeting these requirements (listed below) will cause test to fail and loss of mark for that feature.

A sample test spec file is attached to the submission page so you can check if your work would work with the Playwright tests. This sample test covers some of the requirements, but not all of them. Therefore, passing all the tests in the sample test file does not guarantee a full mark.

Change the `websiteURL` at the beginning of the test file to the URL of your people search page. This is usually a local URL, something like `http://localhost:3000/`, after you open your website using a local http server. The test will start from this page.

As mentioned earlier, you need to keep the tables and records as they are originally provided. Otherwise some of the tests will fail because the returned results will be different from what is expected. This will lead to lose of mark.

Always include all the information (about a person or vehicle) in the search results. The test will check different fields.

Playwright requirements

All the label text and IDs are case sensitive and must be identical. Otherwise you will lose mark for that part.

Page heading:

- The people search page must have a `h1` heading with text 'People Search'
- The vehicle search page must have a `h1` heading with text 'Vehicle Search'
- The add vehicle page must have a `h1` heading with text 'Add a Vehicle '

Navigation link text :

- The link to the people search page must have text 'People search'
- The link to the vehicle search page must have text 'Vehicle search'
- The link to the add vehicle page must have text 'Add a vehicle'
- The `` used for navigation link must be inside the `<header>`.

The element(s) that CSS grid applies to must have the ID 'container'.

For people and vehicle search:

- The button to start the search must have the label text 'Submit'.
- The HTML element showing the search results must have an ID 'results'.
- Each search result record is shown in a `<div>`, inside the 'results' element.
- There must be an element with ID 'message' (outside the 'results' element) that:
 - Displays a message containing text 'Search successful' if the search is successful.
 - Displays a message containing text 'No result found' if the search returns nothing.
 - Displays a message containing text 'Error' if:
 - Both fields in the people search form are empty.
 - Both fields in the people search form are filled (only one can be used). In the vehicle search form the registration number field is empty.

For people search,

- The text input field for driver name must have the ID 'name'.
- The text input field for license number must have the ID 'license'.

For vehicle search, the text input field for vehicle registration/plate number must have an ID 'rego'.

For the add vehicle page,

- The text input field must have the following IDs:
 - 'rego' for the vehicle registration/plate number
 - 'make' for the vehicle make
 - 'model' for the vehicle model
 - 'colour' for the vehicle colour
 - 'owner' for the vehicle owner
- There must be a button with label text 'Check owner' that is only enabled once the 'owner' field is not empty. Clicking this button would search the database based on the owner information entered.
- If there is one or more matches, these must be displayed in a `<div>` with the ID 'owner-results'.
 - All the information about an owner must be displayed.
 - Each owner must have a button with label text 'Select owner' to select that person as the owner of the new vehicle.
- There must be a button with label 'New owner' if there is no matching owner or none of the person returned is the intended owner.
 - This button must only be available/enabled after 'Check owner' has been performed. After clicking this button, a new form must appear asking the user to enter the owner information, and the input fields in this form must have the following IDs (the system must automatically create a new owner ID):
 - 'name' for the owner name
 - 'address' for the owner address
 - 'dob' for the owner date of birth
 - 'license' for the owner's license number
 - 'expire' for the owner's license expiring date
- The button to add a new owner must have the label text 'Add owner'
- After entering all the information and clicking the 'Add owner' button, there must be an element with ID 'message-owner' that:
 - Displays a message containing text 'Owner added successfully' if there is no error. Displays a message containing text 'Error' if any information is missing. All the information about the new owner is required, i.e., there cannot be any empty field.
 - Displays a message containing text 'Error' if all the information about the new owner is the same as an existing owner. It is OK to have a new owner whose full name is identical to an existing one, but not all the other information as well.
- The button to add a vehicle must have the label text 'Add vehicle'.
- After entering all the information and clicking the 'Add vehicle' button, there must be an element with ID 'message-vehicle' that:
 - Displays a message containing text 'Vehicle added successfully' if there is no error. Displays a message containing text 'Error' if any information is missing. All the information about the vehicle is required, i.e., there cannot be any empty field.