COMP 3512
Assignment #1: PHP Pages and API

Version: 1.0 Sept 22
Due Wed October 16, 2024 at midnightish

Overview

This assignment provides an opportunity for you to demonstrate your ability to generate a PHP-based data-generated web site. **You can work in pairs or by yourself (I recommend you do work in pairs if at all possible).** Please don't ask me to be in a group of 3.

The SQLite database you have been provided with contains Formula 1 car racing data. You must implement some Web APIs in PHP, as well as some PHP pages that demonstrate the functionality of the APIs. These APIs (and your pages) will provide the user with the ability to examine races within a single season and examine their results, drivers, and constructors. This database has decades of race data. I want you to focus on just a single season (2022).

Beginning

I feel foolish saying this in a third-year university course, but it is your responsibility to read all the assignment instructions thoroughly and carefully. If you are unclear about something, ask me. But before you do, read the material in question again!

Grading

The grade for this assignment will be broken down as follows:

Visual Design and Styling	20%
Programming Design	10%
Functionality (follows requirements)	70%

Files

You will be able (eventually) to find the most recent version of the database scripts at the GitHub repo for the assignment:

`https://github.com/mru-comp3512-archive/f2024-assign1`

Periodically, I may upload a revised version of the database if errors are found in the data.

GitHub

You need to have "a single source of truth" when it comes to your source code. That is, there must be a "master" location that contains the definitive version of your source code. If you are working as a pair, then one of you will have to decide whose github account has the master location. But even as a pair, each group member will need their own Github account.

Your repo on Github can be either private or public. The free GitHub account doesn't allow private repos; if you sign up for the Student Developer Pack (https://education.github.com/pack) you can have free private repos while a student. The other way is to email me, and I can create a private repo under our department's github organization (https://github.com/MountRoyalCSIS) for your group. You would need to supply the github names or emails for each member. You can also decide to use a public repo.

You will want to push out updates fairly frequently (1-2 times a day when working on it, or more frequently if lots of work is being done). I will be examining the commits when marking. You can push your content to GitHub via the terminal, using the following commands (not the stuff in square brackets though, as those are comments):

```
git init     [only need to do this one command once for your assignment]
git add *
git commit -m "Fixed the rocket launcher"     [alter message and name as appropriate]
git remote add origin https:... your-repo-url.git   [do this just once]
git push -u origin master   [login using your own individual github credentials]
```

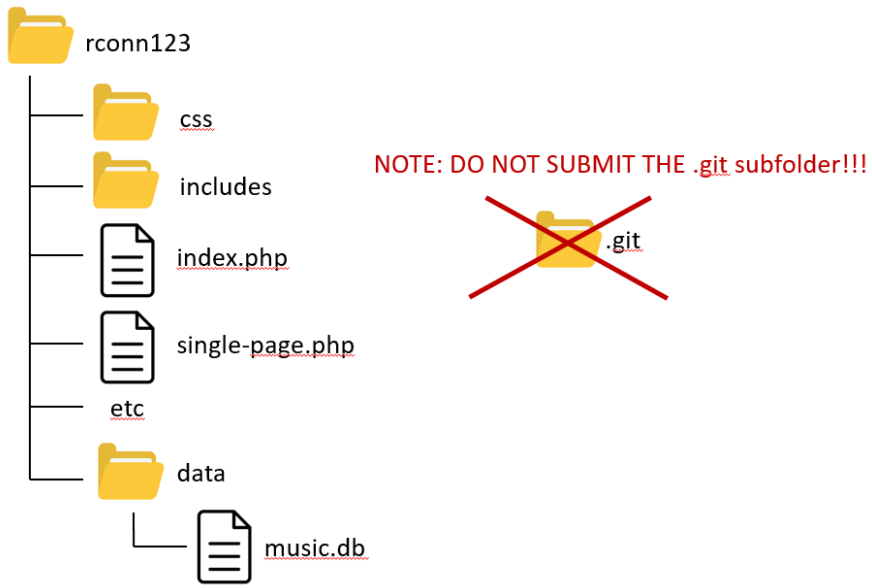
For more information about Git and GitHub, read pages 571-577 of textbook (2nd Edition). There are many online guides to git (for instance, https://guides.github.com/introduction/git-handbook/).

Submitting

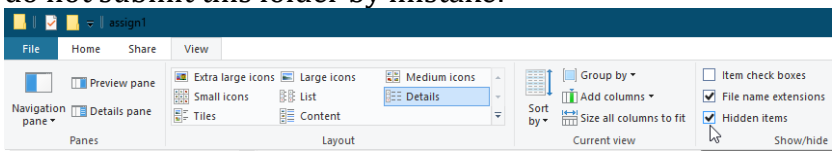
NOTE: I am quite likely to make changes to these submission instructions closer to the due date!

Right now, I am contemplating that you will submit your assignment using the traditional MRU submit drive approach:

1. Your assignment **must** be in a folder whose name is the same as your user id (e.g., rconn123). If you are in a group, pick one your user ids for the folder name. This folder **must** contain your index.php pages and your other files. They must **not** be nested within some other folder! For instance, your folder must look something similar to the following:



2. If you submit a zip file instead (i.e., you use the mymru web portal to upload your assignment), then you are forcing me to do extra steps (unzip then copy), which will make me disappointed and unhappy, which is not how you want me to be when marking your assignment. So, only use the web portal submit approach as a last resort.
3. Make sure you use the SQLite version of the database rather than the MySQL version. That is, make sure your connection string is correct for the SQLite file which must be in a folder named data.
4. Do **NOT** submit the .git subfolder, if you have one!!! This folder contains literally hundreds of extra files, which means copying your submitted folder to my machine for marking takes 10 minutes instead of 30 seconds. If you do this, I will simply give you a mark of zero. You may have system files hidden, so be sure to make them visible so you do not submit this folder by mistake:



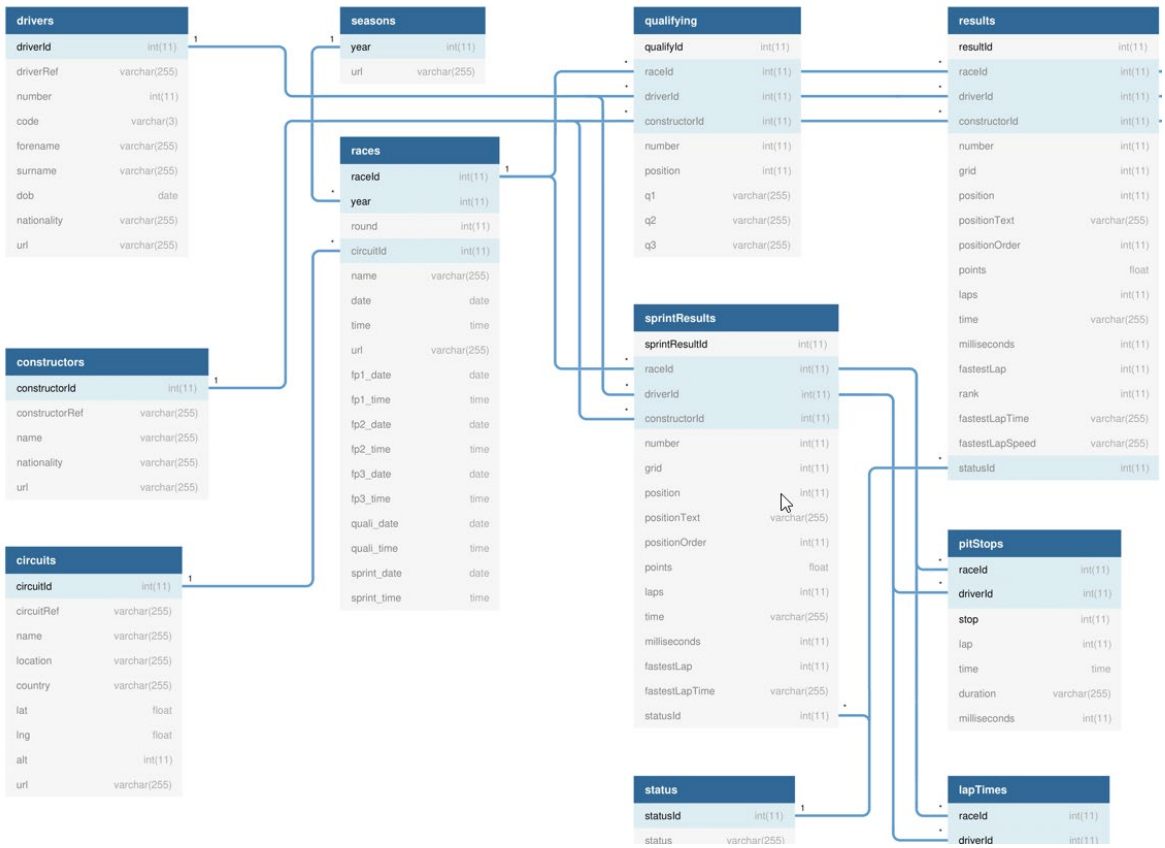
5. When you are ready to submit, I recommend making a fresh folder with the appropriate name in your htdocs folder. Copy the relevant files. Test pages (still work with the SQLite connection?). Check folder structure. Check there is no .git folder. Copy/Submit it.

Data Files

Data has been provided with an SQLite database and a variety of Excel file (for quick examination). The database consists of quite a few tables: you are only interested in this assignment in some of them:

- **circuits**: A race happens on a circuit. A circuit is used 0 or 1 times a season.
- **constructors**: Each constructor usually has two cars (and thus two drivers) in each race. Fans care about the drivers and the constructors.
- **drivers**: Each car has a single driver.
- **races**: In a season there are a certain number of races (in 2022 there were 22 races). For your queries on this table, just hard code the `year` field value to 2022.
- **Qualifying**. Before the race there is qualifying, which consists of three rounds. The five slowest drivers get knocked out of the next round, so only 10 drivers will have q1, q2, and q3 times. The qualifying position determines the race starting positions.
- **results**: A single race has a series of results, one for each driver. The drivers and constructors earn points for their results. Race followers are especially interested in a driver's results, which includes not just their finish position, but how long they took, their fastest lap, and so on.

The tables are organized with primary keys and foreign keys so you will need to use `INNER JOINS` where necessary.



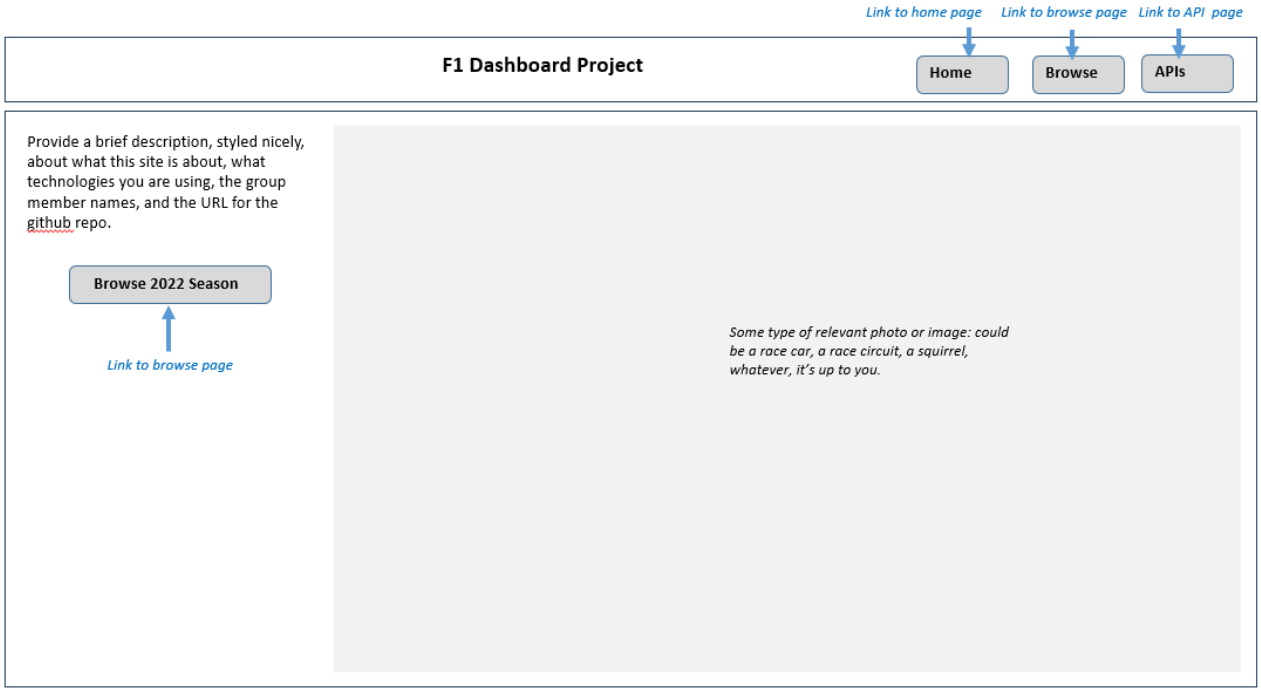
Requirements

1. Your assignment must create the following APIs with the specified routes and functionality. The returned data must be JSON format. You will use these API routes in the second assignment.

`/api/circuits.php`	Returns all the circuits for the season
`/api/circuits.php?ref=?`	Returns just the specified circuit (use the `circuitRef` field), e.g., `/api/circuits.php?ref=monaco`
`/api/constructors.php`	Returns all the constructors for the season
`/api/constructors.php?ref=?`	Returns just the specified constructor (use the `constructorRef` field), e.g., `/api/constructors/mclaren`
`/api/drivers.php`	Returns all the drivers for the season
`/api/drivers.php?ref=?`	Returns just the specified driver (use the `driverRef` field), e.g., `/api/drivers/hamilton`
`/api/drivers.php?race=?`	Returns the drivers within a given race, e.g., `/api/drivers/race/1106`
`/api/races.php?ref=?`	Returns just the specified race. Don't provide the foreign key for the circuit; instead provide the circuit name, location, and country.
`/api/races.php`	Returns the races within the 2022 season ordered by `round`, e.g., `/api/races/season/2022`
`/api/qualifying.php?ref=?`	Returns the qualifying results for the specified race, e.g., `/api/qualifying/1106` Provide the same fields as with `results` for the foreign keys. Sort by the field `position` in ascending order.
`/api/results.php?ref=?`	Returns the results for the specified race, e.g., `/api/results/1106` Don't provide the foreign keys for the race, driver, and constructor; instead provide the following fields: driver (`driverRef`, `code`, `forename`, `surname`), race (`name`, `round`, `year`, `date`), constructor (`name`, `constructorRef`, `nationality`). Sort by the field `grid` in ascending order (1st place first, 2nd place second, etc).
`/api/results.php?driver=?`	Returns all the results for a given driver, e.g., `/api/results/driver/max_verstappen`

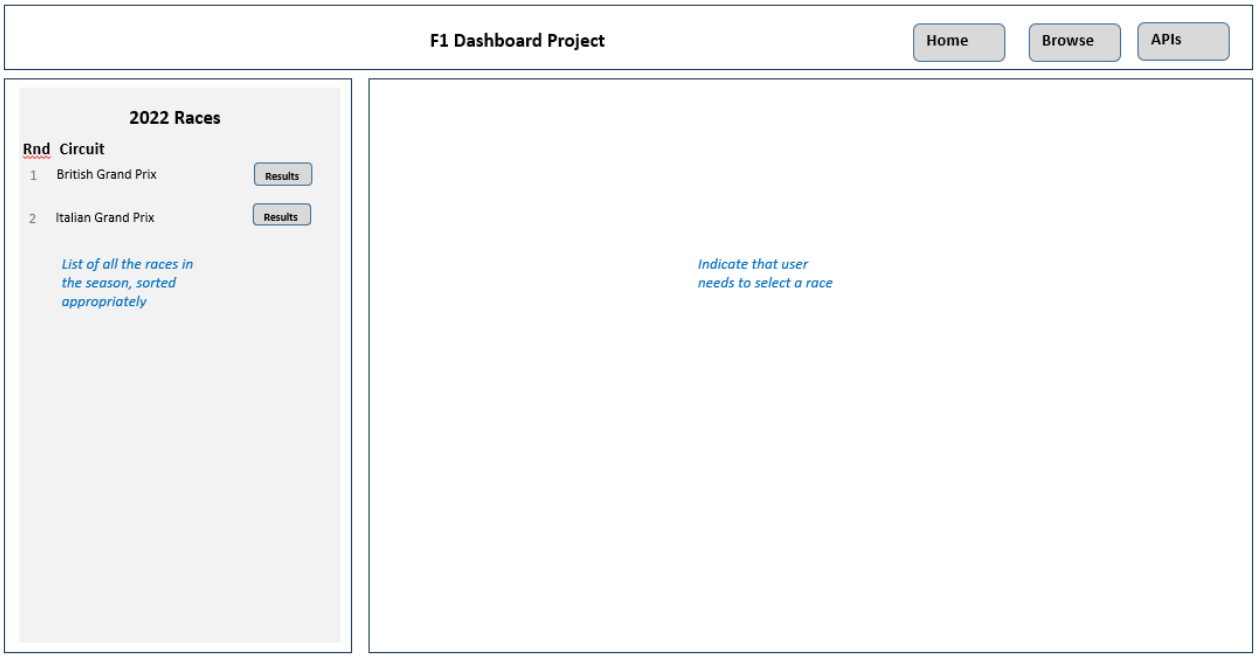
2. You will a PHP page with sample links (described below) that demonstrate your API works properly. This page wouldn't be for a "real" user; they are for me only and don't require any fancy styling.
3. Your assignment will also consist of five PHP pages. I have provided you with simple screen captures to show you roughly the functionality required, but needless to say I would expect your assignment to be a lot more visually styled than that shown in the screen captures.
4. If you make use of CSS recipes you found online, you must provide references (i.e., specify the URL where you found it) via comments within your CSS file. **Failure to properly credit other people's work in your CSS could result in a zero grade.** You can make use of a third-party CSS library; if you do, be sure to indicate this on the home page. Attractive styling with your own CSS will result in a higher style mark than if you use a third-party library; poor styling with your own CSS will likely result in a lower style mark than if you use a third-party library.
5. **No JavaScript on this assignment.** In the second assignment, you will be making a variety of improvements to this assignment using JavaScript.
6. Most of the functionality in the app can be found in the sketches shown on the next few pages. The sketches provided are used to indicate functionality not the layout. You can completely change the layout. Each of these is described in more details below.

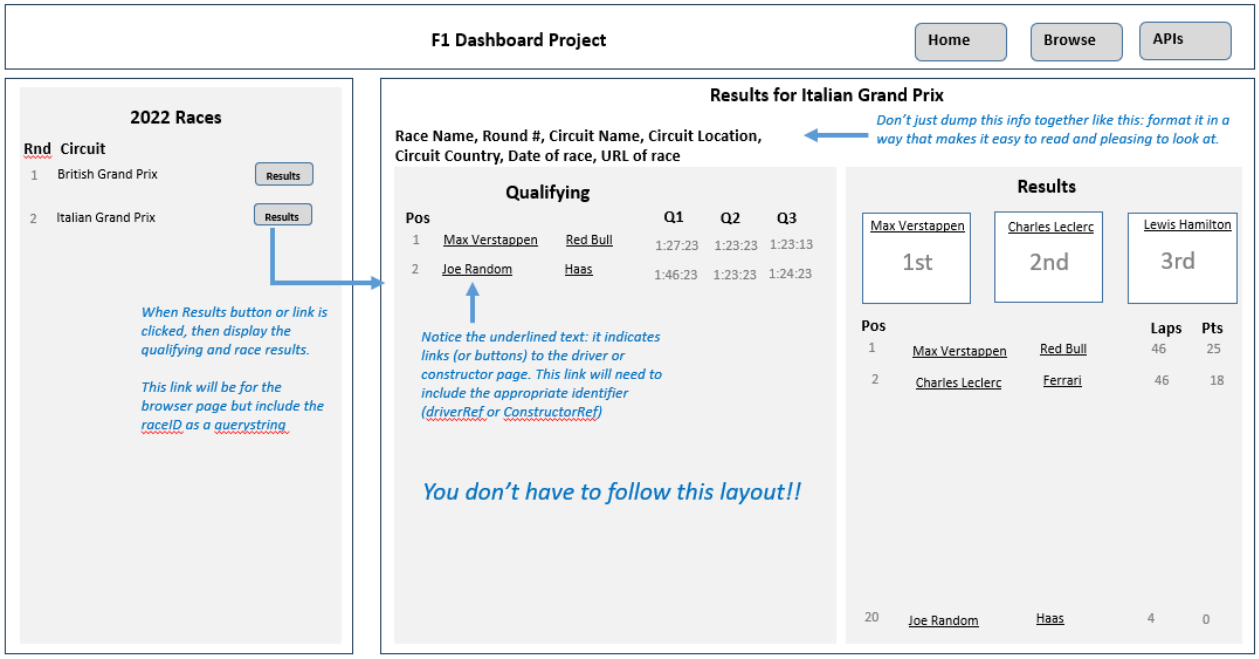
7. The first shown below is the **Home Page**. This will be what the user sees first. It's filename must be `index.php`. The prototype shown below is purposively simple and focuses on functionality. It's up to you to make it look good.



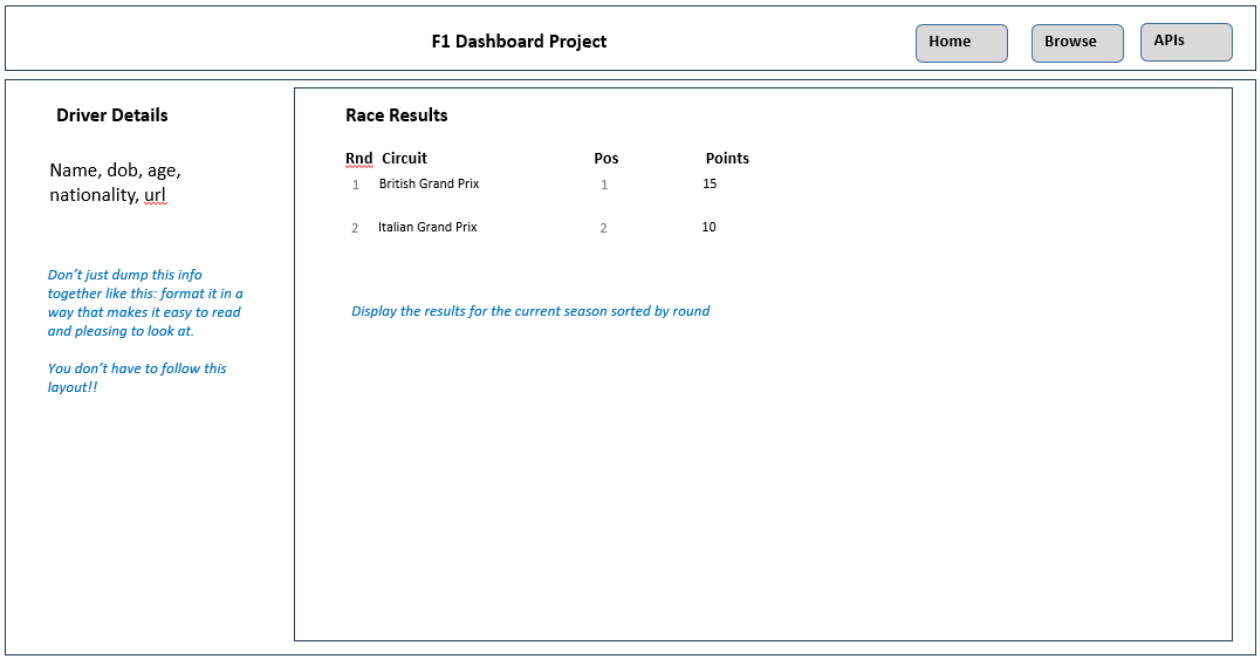
8. **Header**. The page title can be whatever you'd like. You need to provide a brief description (be sure to mention it is assignment #1 for COMP3512 at Mount Royal University), styled nicely, about what this site is about, what technologies you are using, the group member names, and the URL for the github repo. Notice also there are buttons or links that will appear on every PHP page.

9. The next shown is the **Browse Page**. Initially it will display all the races for the season. When the user selects a race, then display more information for the race as well as a list of qualifying results and race results. When we learn JavaScript, we will have a clean way to hide/show data. In PHP, your page will need to "know" whether to display race results or not. If there is a `raceId` passed to the page, then display results.

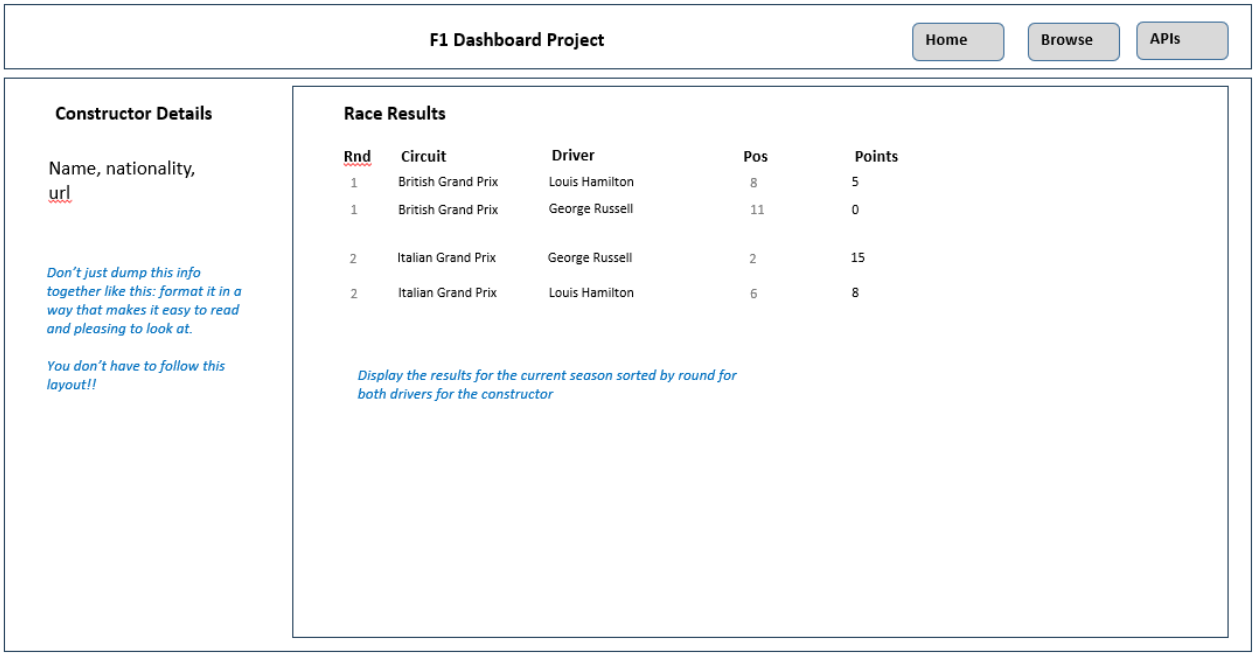




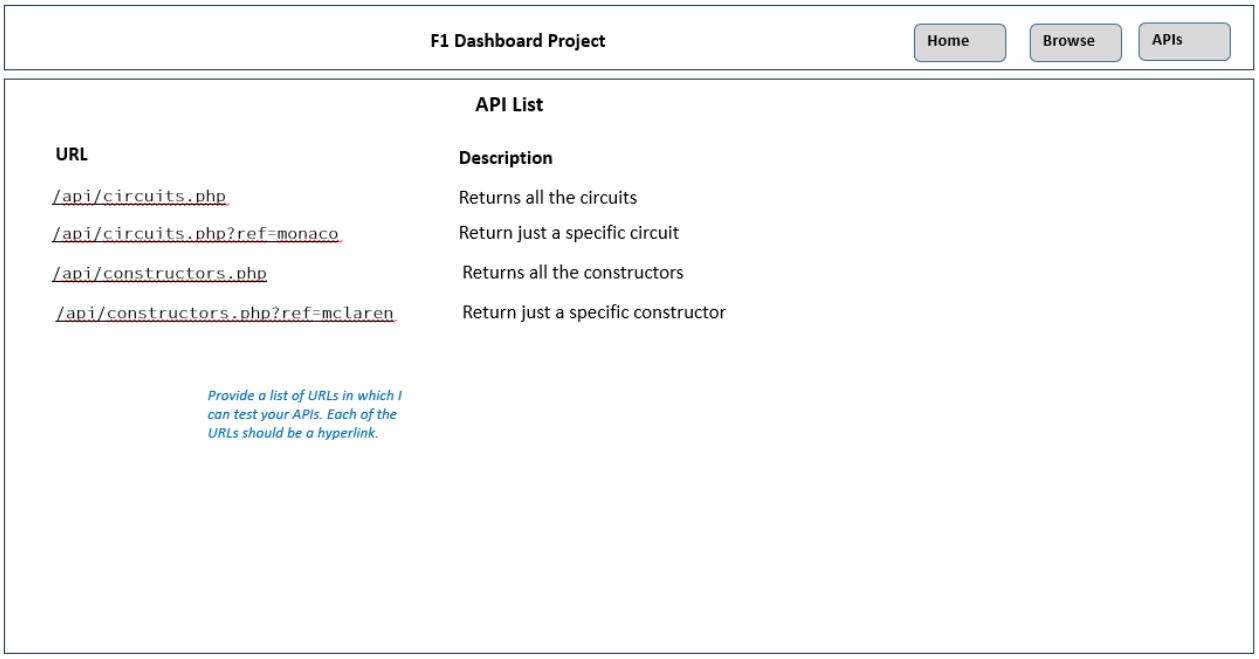
10. The next shown is the **Driver Page**. It includes details about the driver as well as race results for that driver for the season. How will the page "know" which driver to display? It will be passed the `driverRef` value for the driver.



11. The next shown is the **Constructor Page**. It includes details about the driver as well as race results for that driver for the season. How will the page "know" which constructor to display? It will be passed the `constructorRef` value for the constructor.

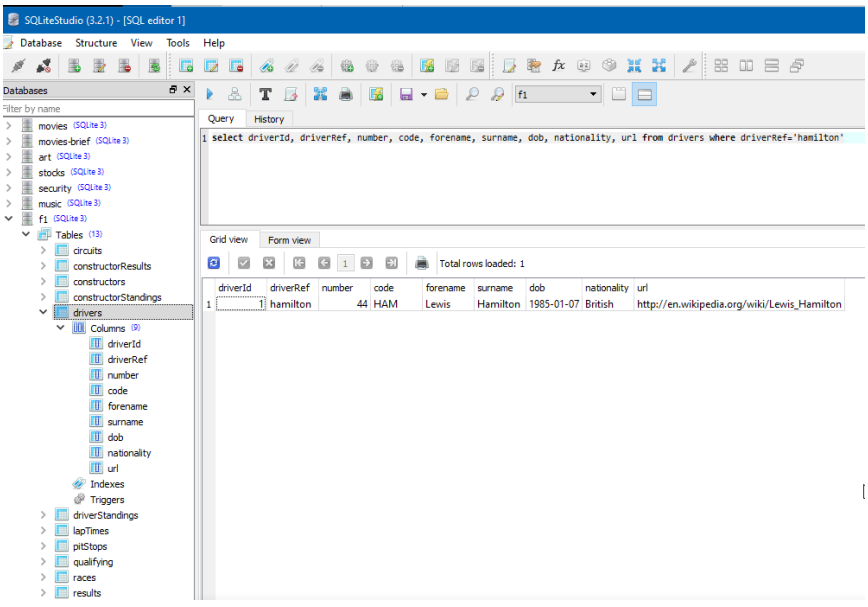


12. The next page is the **API Tester Page**. This page should include sample hyperlinks for each of the routes described on page 5. When one of these links is clicked, the user will see the JSON data returned from the API.



Hints

1. Begin by getting your data access from the database working. Complete Lab14a as soon as possible! You will find that this lab provides a library of functions that will simply your code.
2. I would recommend writing up your SQL for each of the APIs and save this in a separate text file, as you will be using these queries for your pages and for your APIs. You might find SQLiteStudio (shown below) a useful program for testing your queries.



3. I would recommend you complete the **Driver Page** first. It will be passed the `driverRef` as a query string parameter. Don't worry about styling at this point, just get the data display working. Test it with a sample `driverRef`, such as `hamilton`.
4. Complete the **Constructor Page** next.
5. Complete the **Browse Page**. This will be the most complicated page.
6. Begin implementing the different API pages. Look at Exercise 12.10 in Lab12b to see how to create a page that returns JSON instead of HTML.
7. Finish off by implementing the **Home Page**.
8. Most of the visual design mark will be determined by how much effort you took to make the pages look well designed. Simply throwing up the data with basic formatting will earn you minimal marks for this component. At this point, start improving the style of your existing pages.
9. If working in pairs, I would strongly recommend that each member is involved with retrieving and working with data. This will improve your midterm mark significantly.
10. Before submitting, test every single bit of functionality and cross-check with this document to make sure it has all the functionality I've requested. Carefully read the specifications for each bit of functionality. Every year, students lose all sorts of marks because they didn't read this document carefully enough!!