COMP 4513 Assignment #2: React

Due Monday April 8th at midnight
Version 1.0 (March 7, 2024)

Overview

This assignment provides an opportunity for you to demonstrate your ability to generate a single-page application using React. **You can work alone or in groups of two on this assignment. Please don't ask for a group of three or four.**

Beginning

Use the same techniques covered in lab11b, lab11c, and lab14b for this assignment.

I will try to provide you with some additional images that you can use in this assignment. These can be found in the github repo for the assignment (coming soon).

Submit

You must host your site on a working server platform that I can access. Since your React application is a static site, any host that supports static files will work. GitHub Pages, FireBase Hosting, Netlify, Render, and Surge, all provide relatively trouble-free solutions for hosting React apps. If DevOps is your thing, you could use Google Cloud Platform or AWS to create a virtual servers. Or you could use any third-party server that has ftp access.

You must upload the deploy version of Vite using the `vite build` command. You can find instructions for deploying to numerous environments at https://vitejs.dev/guide/build.

I will also need access to the source code, so you will have to make it available to me on GitHub. **If you are hosting on github pages, you will need two Github repos: one with the development version, and one with the production version**! If on a private repo, be sure to invite me.

So, to submit your assignment you must send me an email with three bits of information: 1) the URL of your React app, 2) the development github repo URL, 3) the names of the people in your group.

Grading

This assignment is worth 17% of the course grade. The grade for this assignment will be broken down as follows:

Visual Design, Styling, and Usability	25%
Programming Design + Documentation	5%
Functionality (follows requirements)	70%
Extras	+10%

Requirements

1. You must make use of the `Vite` starting files and structure. The filename for your starting file should be `index.html` (`Vite` does this automatically).
2. You are encouraged to use a third-party CSS library. This can be Tailwind (it would be good to have that on your resume), but you can use any other CSS library (or component library) you'd like.
3. This assignment consists of two main views (remember this is a single-page application, so it is best to think of the assignment consisting of different views). In the remainder of this assignment, I have provided a sketch of the basic layout of each view.

   These sketches do not show the precise styling (or even the required layout); rather they show functionality and content. I will be expecting your pages to look significantly more polished and attractive than these sketches! A lot of the 25% design mark will be based on my assessment of how much effort you made on the styling and design.
4. If you've used JS that you've found online, I expect you to indicate that in your documentation AND in your about dialog. Provide a URL of where you found it in the documentation. Failure to do so might result in a zero mark, so give credit for other people's work!

5. **You have two choices when it comes to how you are going to retrieve the F1 data.** One way, is to simply consume (fetch) the APIs from your first assignment. This will mean you have to ensure your glitch/render server is up and running when you test. If your API isn't providing you with your expected data, you'll need to modify it on your server. The other way is to call the supabase methods directly in your React application (see Exercise 14b.9 which demonstrated how to access supabase in vanilla JS). Essentially, you would copy the relevant route code from your first assignment. The second option *might* be simpler from a hosting standpoint, but I could be wrong.

   Example

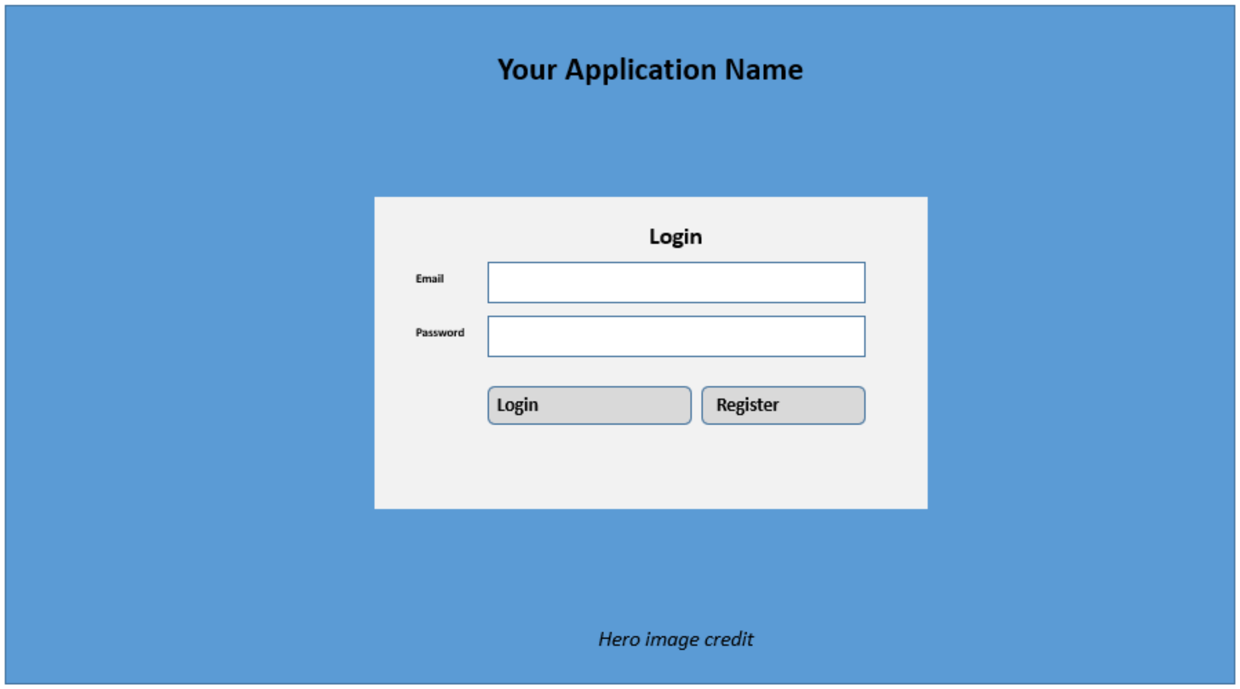
```
// consuming assignment #1 APIs
useEffect( () => {
    url = "http://somecrazy.glichURL.com/f1/api/seasons";
    console.log('fetching … here to check if I've gone infinite');
    fetch (url)
      .then( resp => resp.json() )
      .then( data => { setSeasons(data); }
}, [] );


VERSUS
// accessing supabase directly in react
useEffect( () => {
   selectSeasons();
}, []);

async function selectSeasons() {
   console.log('getting from supabase … here to check if I've gone infinite');
   // uses the same API as your assign 1 solutions
   const { data, error } = await db.from('seasons')
                                   .select('*');
   if (error) {
         console.error('Error fetching seasons:', error);
         return;
   }
   setSeasons(data);
}
```

Be careful you don't get infinite loops with your requests! Use console.log messages and be sure to check them frequently!

Login View

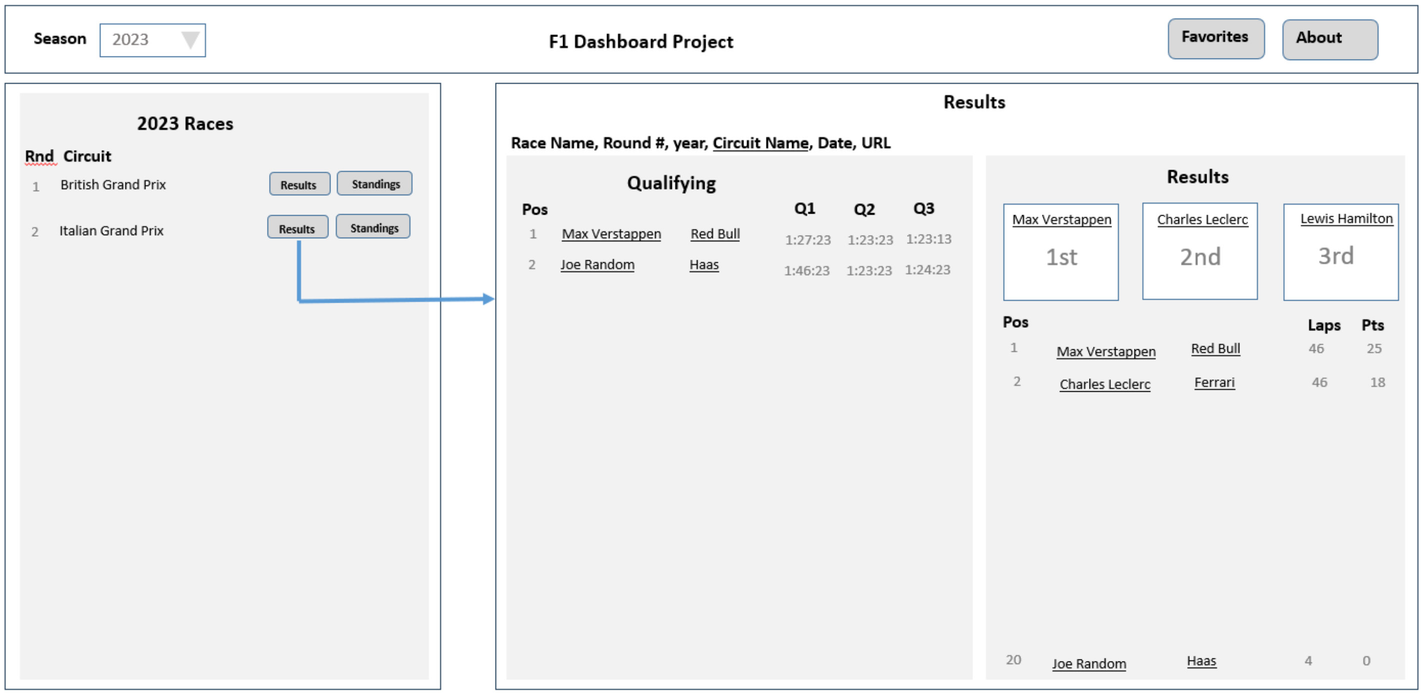


When your assignment is first viewed, it should display the Login View. It consists of a hero image (an image that fills the entire browser width or up to a specific very wide value, say 1800 or 2200 pixels). You could use unsplash.com as a source for your image, but be sure to provide credit information somewhere on this page. In the middle of the page should be another rectangle with a login form.

Depending on whether there is time available, I will ask you to implement the login authentication using supabase auth. It is very likely, however, there won't be enough time, and so you will not need to implement the actual authentication, but will just simulate it by jumping to the Home View when the login button is clicked.

If you decide to expand on this project during the spring or summer, implementing this functionality will make your assignment a fantastic portfolio piece!

Home View 1



Rnd	Circuit		
1	British Grand Prix	Results	Standings
2	Italian Grand Prix	Results	Standings

Pos			Q1	Q2	Q3
1	Max Verstappen	Red Bull	1:27:23	1:23:23	1:23:13
2	Joe Random	Haas	1:46:23	1:23:23	1:24:23

Pos			Laps	Pts
1	Max Verstappen	Red Bull	46	25
2	Charles Leclerc	Ferrari	46	18
20	Joe Random	Haas	4	0

When first displayed, show just the header with the Seasons selector (which should contain the years from 2000 to 2023), the title, and the two buttons. The About button should display a modal dialog (see below) that displays your names, a link to the github repo for the source code, and a brief description of the project and the technology it is using. Ideally, this assignment can act as a portfolio piece to show and impress a potential client/employer.

Until there is something in the favorites collection (see below), the Favorites button should be disabled. If there are favorites, then display the Favorites modal/dialog (see below).

When the user selects a season (or changes it), then display the races for that season/year, sorted by round. Provide some way to view the results and the standings (in the screen shown here, I've done it via simple buttons, but you can use icons, hyperlinks, etc).

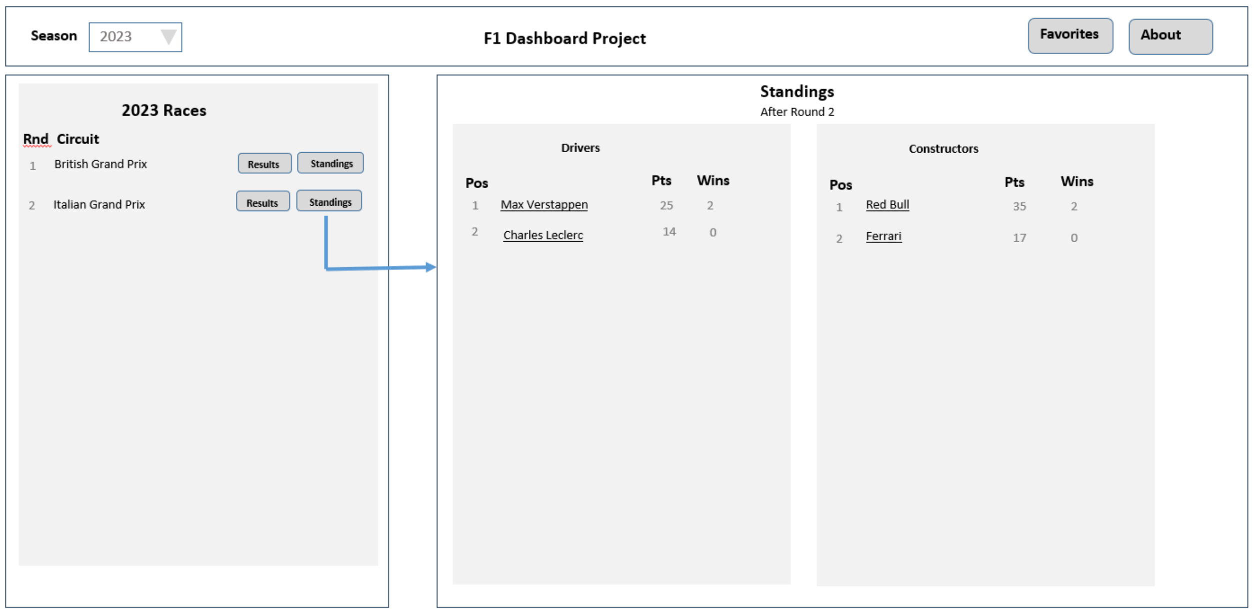
In the image above, you can see what should be displayed if the user selects Results. It should display more information about the race (e.g., race name, round #, etc) from the races table. The circuit name should be a link that will display more information about the circuit in a pop (more on that below). As well, qualifying results (from the qualifying table) and race results (from the results table) should also be displayed. Also find a way to highlight the podium positions (1st, 2nd, 3rd).

You might find that there is not enough space to display both qualifying results and race results next to one another. In that case, make use of some type of show/hide feature so the user can see qualifying or results, but not both.

The driver and the constructor should be links/buttons so when they are clicked, the user will see additional information about the driver or constructor in a modal window (see below).

Extra Marks: add a button/icon to races, to qualifying, and to results that reverses the sort order.

Home View 2



Rnd	Circuit
1	British Grand Prix
2	Italian Grand Prix

Pos		Pts	Wins
1	Max Verstappen	25	2
2	Charles Leclerc	14	0

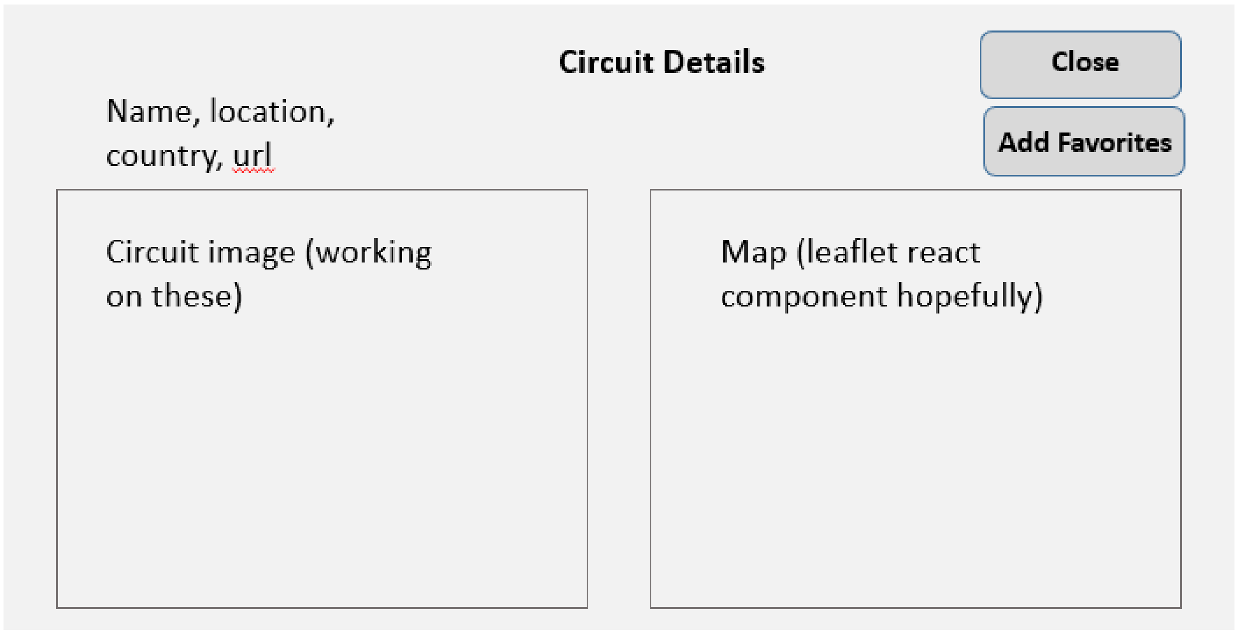
Pos		Pts	Wins
1	Red Bull	35	2
2	Ferrari	17	0

In the image above, you can see what should be displayed if the user selects Standings. It displays the driver standings and the constructor standings up after that race (these were the last two routes in Assignment #1).

The driver and the constructor should be links/buttons so when they are clicked, the user will see additional information about the driver or constructor in a modal window (see below).

Extra Marks: add a button/icon to drivers, and to constructors that reverses the sort order.

Circuit Details Pop-up / Modal Dialog


Circuit Details
Close
Add Favorites
Name, location, country, url
Circuit image (working on these)
Map (leaflet react component hopefully)

When the user clicks on a circuit name, they should see in a modal dialog / pop-up, additional information about the circuit from the circuits table. I will provide some images you can use for *some* of the circuits. Your code has to handle the possibility that an image is unavailable.

You also need to provide the ability to add a circuit to a favorites list. The user will also be able to add drivers and constructors to their favorites.

The Close button will close the dialog.

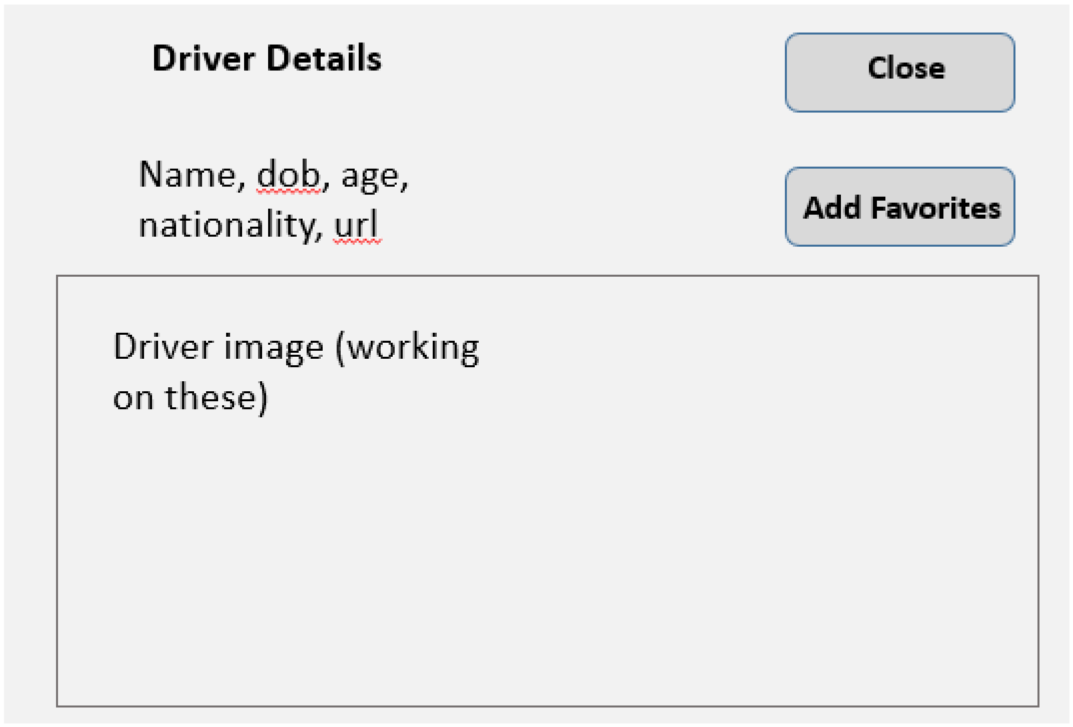
While I used text buttons in the image, feel free to use icons instead.

Sometimes it makes sense to make use of existing components rather than re-invent the wheel. You may decide to use a third-party modal/dialog component (e.g., react-modal or react-modal-dialog) or use one within a CSS library you are using (e.g., such as ant, bootstrap, etc). Alternately, if you look on youtube, you can find lots of examples of how to implement a modal dialog in React and CSS.

Some component libraries provide alternatives to modal dialogs that you can use instead. For instance, the Ant Design component library has a Drawer component, which is like a modal dialog, but is displayed from an edge rather than in the middle (see https://ant.design/components/drawer).

Extra Marks: The database table has latitude and longitude, so I'm hoping that the Leaflet Map React component is relatively easy to use and add to your project. You could instead use the Google Map React component, but you will need to get a Google API key.

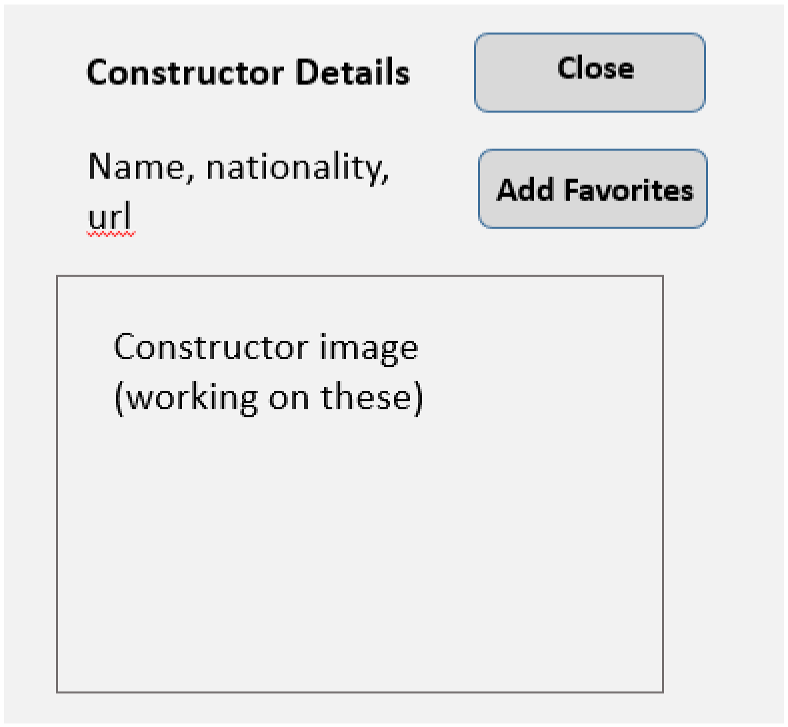
Driver Details Pop-up / Modal Dialog


Driver Details
Close
Name, dob, age, nationality, url
Add Favorites
Driver image (working on these)

When the user clicks on a driver name, they should see in a modal dialog / pop-up, additional information about the driver from the drivers table. I will provide some images you can use for *some* of the drivers. Your code has to handle the possibility that an image is unavailable.

You also need to provide the ability to add a driver to a favorites list. The user will also be able to add circuits and constructors to their favorites.

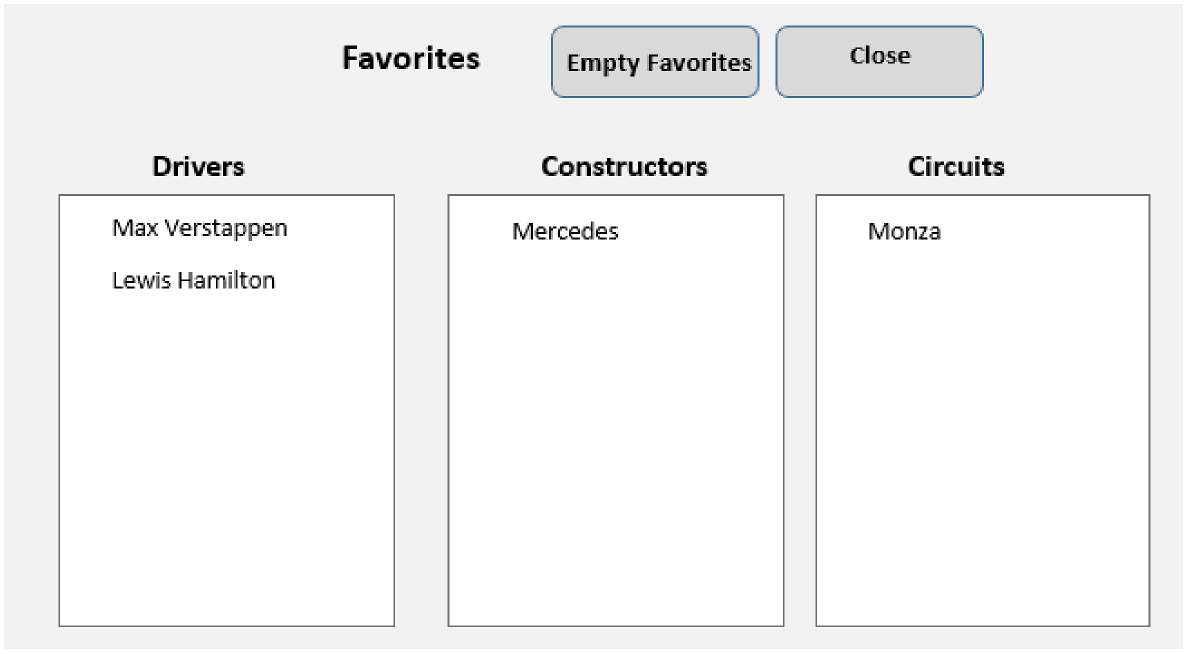
Constructor Details Pop-up / Modal Dialog



When the user clicks on a constructor, they should see in a modal dialog / pop-up , additional information about the constructor from the constructors table. I will provide some images you can use for *some* of the constructors. Your code has to handle the possibility that an image is unavailable.

You also need to provide the ability to add a driver to a favorites list. The user will also be able to add circuits and constructors to their favorites.

Favorites Pop-up / Modal Dialog



Simply display the three lists. The dialog can be closed or the favorites can be emptied.