**Implementation Document**

For the implementation of the technical assessment project, we chose the [www.thesportsdb.com/api.php](http://www.thesportsdb.com/api.php) REST API to retrieve data from.

The project consists of three modules and an interface:

1. Pages Module
2. Components Module
3. Services Module
4. Teams interface

**Page Module**

The Pages module consists of the two different pages implemented for the need of the project, the Home and the Details pages.

The **Home Page**, is the page that the user lands on when the applications boots.

On the onInit stage of the page, the getTeamsByLeague method of the sportsDB service is invoked and via this method, the data needed are loaded.

In the Home page, the end user can see all the teams competing in the English Premier League. The teams are displayed by using the list component which will be mentioned below.

The Home page is the host of three functionalities.

1. The user is able to search for a team by typing the name of the team. The filtering of the teams happens when the user clicks outside the input field (onBlur). If the user inputs a string that does not match with any team’s name, then a text appears, indicating that there are no teams found with the name specified.
2. The user is also able to filter the teams displayed, by the location of their stadiums. Here, we made an assumption since the API does not return consistent data for all teams. We assumed, that the second word that can be found in the strStadiumLocation field, is the town that the team is based in.
3. Lastly, the user can click on any of the tiles displayed. This action, will lead to the details page.

The logic behind the filtering of the teams, according to the input keyword and/or the dropdown list, is to render the teams found in both sets. The input creates the inputTeams set while the dropdown list creates the filteredTeams set. The teams found in both sets, are the ones rendered in the home page.

The **Details Page**, is the page that the user is redirected to, when he clicks on any of the tiles in the home page, as mentioned above.

The data are passed from the Home page to the Details page by using the navigation.extras.state provided by the angular/router.

Since we want to re-use the list component that is used in the Home page as well, the data passed in the Details page must be transformed. The reason is that the data passed, is a single object which is not compatible with the data that are defined in the List component. As a result, we need to initialize an array and push this object to this array. Then, this array will be passed to the List component in order to be rendered.

In the Details page, the user will find extra details for the team as described in the project definition.

**Components Module**

The Components Modules, includes the four different components created.

The **Header Component** acts as the header of the app across all pages. The component consists of a page title, that is passed as a property from each page and of a toggle button.

This button is responsible for toggling the theme of the app to dark or light. Upon clicking the theme button, an event is emitted that gets caught by the app.component. Then the toggleTheme function of the theme.service.ts is invoked that toggles the value of the darkMode variable and also invokes the applyTheme function which in turn applies a class to the body element of the DOM. This new class is the one responsible for the change of theme, since inside the styles.less file, the variables defined, for both light and dark theme, are dependent on the class of the body element.

In the Details page, there is also a back button that seems to be part of the header component. Since this button only applies for this specific page, the button was not added to the Header component. In case we had multiple pages that could have used this functionality, the button would have been added to the header component.

The **List Component** is the component responsible for rendering the team tiles. The component takes as an input from each page, an array of the teams to be rendered. Then proceeds by rendering the teams in a list of tiles, each of which contains the team’s badge, name, stadium name and stadium location.

In the home page, the list component has the functionality of redirecting the user to the details list in case the user clicks on any of the tiles. This functionality is disabled in the details page.

Also, the UI behavior of the tiles are different between the home page and the details page. Since the user is not able to click the tile in the details page, we have removed the shadow and the pointer cursor when the user hovers over the tile.

The **Drodpown-list component** hosts the functionality of a default select element.

The options are passed by the home page to the component as an input and the component renders the select list. The home page controller iterates through the teams array and creates an array of unique locations that is passed to the component.

The onOptionSelected function of the home page is passed as a callback function to the dropdown list component. When a selection has been made, the callback is invoked which then proceeds by invoking the filterTeams function according to the option selected.

Last but not least, the **Dialog component** is a modal dialog. It is created by using the angular/material/dialog in order to show to the user that the data are loading. The dialog opens inside the getTeamsByLeagueName function and closes when the API request is completed. In order to open/close the dialog, we use the .open and .closeAll functions of angular/material/dialog.

**Services Module**

In this module, we will find two services files.

The **sportsDB.service** is responsible for the requests made towards the data provider. In this file, we have implemented the getTeamsByLeague function that is responsible for retrieving the initial data. As mentioned above, in this function there is also the implementation for the modal toggle.

The **theme.service** which is responsible for handling the toggle of the theme from light/dark to dark/light. The toggleTheme function, toggles the value of the Boolean variable darkMode and also saves locally the selection of the user, in order to maintain the same theme even when the app refreshes. The applyTheme function adds the dark-mode class to the body element that is responsible for the main colors of the app.

**Teams Interface**

In order to have a better defined and better structured project, the teams interface was created based on the response from the 3rd party API. The interface contains all the fields that are retrieved from the response of the API.